The field of software engineering for computer supported cooperative work aims at designing and developing software engineering models, methods and tools that assist individuals and groups to cooperate. However, the cooperation is significantly influenced by participants’ awareness of relevant information. The objective of this thesis is to propose a software engineering methodology to design and develop applications that assist individuals to identify their awareness. The present study is based on three main observations from the literature. First, studies on the utilisation of information technology for cooperation emphasise the role of awareness for identifying the relevance of information in the midst of an abundance of inputs. Second, there are bodies of work that indicate the significant contributions of agents to design and develop software applications to
assist individuals to maintain their awareness. Third, the literature lacks formalised software engineering methods on how to identify and implement awareness. The objective is to provide a mechanism that can be applied with intelligent agents in order to design and develop software applications to identify awareness. This study introduces Policy-based Awareness Management (PAM), a software engineering methodology that proposes the use of existing policy rules as a source to identify awareness. This methodology presents a formalised framework for policy-based awareness and a step-by-step process for awareness identification. The formalism has been built on the logic of general awareness, and its implementation is based on the platform of policy rules in Directory Enabled Networks (DEN). The research also proposes a systematic software engineering methodology for designing and developing architecture of software applications. The present thesis develops a design methodology for software agents to implement the proposed formalism. In this study, PAM using software agents has been evaluated in terms of its efficacy and cost-efficiency by a triangulation of two simulation studies on hypothetical examples and wireless communication procedures in a hospital environment. The original contributions of this study are illustrated through three applications; two in disaster management and one in healthcare. The results of these evaluations show that PAM is effective and cost-efficient subject to the limitations of the study.
PUBLICATIONS

Journal Papers


Conference Papers

Amir Talaei-Khoei, Pradeep Ray, Nandan Parameshwaran, Policy-based Awareness: Implications in Rehabilitation Video Games, 44rd Hawaiian International Conference on System Sciences (HICSS44) (Ranked A by ERA), January 2011.


1 Excellence in Research for Australia (ERA) ranks journals into A*, A, B, and C. For more information, please visit http://www.arc.gov.au/era/.


Book Chapter

ACKNOWLEDGMENTS

This PhD thesis arose in part out of research that has been done since I joined Asia-Pacific ubiquitous Healthcare research Centre (APuHC) where I have worked with people whose contributions have helped and inspired me in my study. It is a pleasure to give them my humble acknowledgements and thank those who made this thesis possible.

My deepest gratitude is to my supervisor, Prof. Pradeep Ray, whose encouragement; supervision and support from the very early stage of this research exceptionally have enriched my progress.

I am heartily thankful to my co-supervisor, Dr. Nandan Parameshwaran, because of the long discussions that helped me develop the technical details of this work and because of his patience and support.

I offer my sincerest regards to all my mates particularly Dr. Amir H. Ghapanchi and Dr. Farhad Daneshgar for friendship, discussions and encouragement. I would like to extend my appreciation to Prof. Lundy Lewis and Terje Solvoll for the collaborations we had in writing papers.

Words fail me to express my heart-felt tribute to my parents whose dedication,
love and persistent confidence in me have considerably contributed in my study. None of this would have been possible without their patience and concerns. I also bless my older brother and younger sister who have been always a source of hope for me.

Finally, I would like to thank all of those who supported me in any respect during this endeavour.
# Table of Contents

## Chapter 1: Introduction

1. **Illustrating the Motivation: Space Shuttle Columbia Disaster** ........................................ 3  
   1.1. **Challenge: Identifying Awareness** ........................................................................ 4  
   1.2. **Research Scope** ................................................................................................ 6  
   1.3. **Research Objectives, Contributions and Assumptions** ...................................... 10  
   1.4. **Research Methodology** ...................................................................................... 13  
   1.5. **Intended Audiences** ............................................................................................ 14  
   1.6. **Thesis Structure** .................................................................................................. 15  

## Chapter 2: Literature Review and Background

2.1. **Literature Review** .................................................................................................. 18  
   2.1.1. **Context, Awareness and Theories for Awareness Maintenance** ....................... 18  
   2.1.2. **Methodology of Systematic Review** .................................................................... 22  
   2.1.2.1. **Identification of Resources** .............................................................................. 22  
   2.1.2.2. **Selection** ............................................................................................................ 24  
   2.1.2.3. **Data Extraction and Synthesis** ........................................................................ 26  
   2.1.2.4. **Data Analysis** .................................................................................................. 28  
   2.1.2.5. **Results** ............................................................................................................. 30  
   2.1.3. **Awareness Maintenance Framework** ................................................................ 31  
   2.1.3.1. **Awareness Obtainment** .................................................................................. 31  
   2.1.3.2. **Context Representation** .................................................................................... 34  
   2.1.3.3. **Context Analysis** .............................................................................................. 37  
   2.1.3.4. **Awareness Utilisation** ...................................................................................... 40  
   2.1.4. **Trend Analysis** .................................................................................................. 42  
   2.1.5. **Gap Analysis: Awareness Identification** .............................................................. 44  
   2.1.6. **Theoretical Support for Awareness Maintenance Framework** ............................. 45  
   2.2. **Background** ........................................................................................................ 47  
   2.2.1. **Awareness in CSCW** ........................................................................................ 47  
   2.2.2. **Awareness in Software Agents** .......................................................................... 52  
   2.2.2.1. **Problem of Logical Omniscience and Perfect Reasoning** ............................. 54  
   2.2.2.2. **Dealing with Logical Omniscience and Perfect Reasoning** ............................. 55  
   2.2.3. **Awareness and Policy Structures** ...................................................................... 58  
   2.3. **Observation and Research Gap** ............................................................................. 61  
   2.4. **Summary and Outlook** ........................................................................................... 67  

## Chapter 3: Research Methodology

3.1. **Research Objectives** ............................................................................................... 70  
   3.2. **Research Framework** ............................................................................................ 71  
   3.2.1. **Choice of Research Paradigm: Design Science** ................................................. 71  
   3.2.2. **Research Methodology** ...................................................................................... 73  
   3.2.3. **Reliability of Research Methodology** ................................................................. 76  
   3.2.3.1. **Guideline 1: Design as a Product** ..................................................................... 77
# Table of Contents

3.2.3.2 Guideline 2: Problem Relevance ................................................................. 77  
3.2.3.3 Guideline 3: Design Evaluation ..................................................................... 78  
3.2.3.4 Guideline 4: Research Contributions .......................................................... 78  
3.2.3.5 Guideline 5: Research Rigor ......................................................................... 79  
3.2.3.6 Guideline 6: Design as a Search Process ....................................................... 79  
3.2.3.7 Communication of Research ......................................................................... 80  
3.3 SUMMARY AND OUTLOOK .............................................................................. 80  

CHAPTER 4: POLICY-BASED AWARENESS MANAGEMENT (PAM) ......................... 83  
4.1 PAM FRAMEWORK ......................................................................................... 85  
4.1.1 Informal Semantics: Intuitions ................................................................. 86  
4.1.1.1 Decision Trees to Branching-time Model of Worlds ................................... 91  
4.1.1.2 Assumed Inputs in PAM ........................................................................... 92  
4.1.2 Formal Semantics: Definitions ................................................................. 93  
4.1.2.1 Commitments: Implicit Knowledge Leading to Actions ......................... 99  
4.1.3 PAM Framework and Space Shuttle Columbia Disaster ......................... 100  
4.2 PAM PROCESS: STEPS TOWARDS IDENTIFYING AWARENESS .................. 104  
4.2.1 Step One: Recognise Relevance of Policy Rules ........................................ 104  
4.2.1.1 Recognise Relevance of Policy Rules in the Space Shuttle Columbia Disaster 108  
4.2.2 Step Two: Recognise Relevance of Information ........................................... 108  
4.2.2.1 Recognizing Relevance of Conditions Referenced to the Relevant Policy Rule in the Space Shuttle Columbia Disaster 110  
4.2.3 Step Three: Change Behaviours .................................................................. 111  
4.2.3.1 Change of Behaviour in the Space Shuttle Columbia Disaster ................. 119  
4.3 SUMMARY AND OUTLOOK ........................................................................... 120  

CHAPTER 5: DEVELOPMENT OF PAM .................................................................. 123  
5.1 DEVELOPMENT GUIDELINES FOR EMPLOYING PAM TO IDENTIFY AWARENESS 125  
5.1.1 Scenario Analysis ......................................................................................... 126  
5.1.1.1 Overall System Study .............................................................................. 126  
5.1.1.2 Process Study .......................................................................................... 127  
5.1.1.3 Logical Components Identification ......................................................... 128  
5.1.2 System Design and Implementation ......................................................... 128  
5.1.2.1 Developing PAM Framework ................................................................. 129  
5.1.2.2 Developing Agents Deliberations ............................................................ 139  
5.2 ANALYSIS VERSUS DESIGN AND IMPLEMENTATION ................................. 154  
5.3 DEVELOPMENT OF THE SPACE SHUTTLE COLUMBIA DISASTER USING PAM 156  
5.3.1 Scenario Analysis of the Space Shuttle Columbia ........................................ 156  
5.3.1.1 Overall System Study .............................................................................. 158  
5.3.1.2 Process Study .......................................................................................... 158  
5.3.1.3 Logical Components Identification ......................................................... 159  
5.3.2 System Design and Implementation in Space Shuttle Columbia ............... 159  
5.4 SUMMARY AND OUTLOOK ........................................................................... 162  

CHAPTER 6: EVALUATION ................................................................................. 165  
6.1 PROOFS OF CONCEPTS ................................................................................. 167  
6.1.1 Rehabilitation Video Games ......................................................................... 167  
6.1.1.1 Problem .................................................................................................. 172  
6.1.1.2 Application of PAM in Rehabilitation Video Games ............................... 173
<table>
<thead>
<tr>
<th>REFERENCES</th>
<th>247</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMPLEMENTATION OF CLASS AGENT</td>
<td>266</td>
</tr>
<tr>
<td>IMPLEMENTATION OF METHOD IMPLICITKNOWLEDGE FOR PATH FORMULAE</td>
<td>268</td>
</tr>
<tr>
<td>IMPLICIT KNOWLEDGE FOR BASIC PATH FORMULAE</td>
<td>268</td>
</tr>
<tr>
<td>IMPLICIT KNOWLEDGE FOR PATH FORMULAE WITH UNTIL COMPOSITION</td>
<td>270</td>
</tr>
<tr>
<td>IMPLEMENTATION OF METHOD POLICYRULEAWARENESS</td>
<td>272</td>
</tr>
<tr>
<td>IMPLEMENTATION OF METHOD INFORMATIONAWARENESS</td>
<td>274</td>
</tr>
<tr>
<td>IMPLEMENTATION OF CHANGEOFBEHAVIORS</td>
<td>275</td>
</tr>
</tbody>
</table>
LIST OF TABLES

TABLE 2-1 COMPARISON OF DIFFERENT AWARENESS TYPES ................................................................. 19
TABLE 2-2 THE ONLINE DATABASES AND RESEARCH FIELDS USED IN THE LITERATURE REVIEW ........... 25
TABLE 2-3 PROBLEMS IDENTIFIED BY EXPLORING THE ARTICLES FOUND IN THE LITERATURE REVIEW .... 29
TABLE 2-4 METHODS FOR AWARENESS CREATION ................................................................................. 33
TABLE 2-5 METHODS FOR AWARENESS SECURITY MANAGEMENT ............................................................ 34
TABLE 2-6 METHODS FOR CONTEXT MODELING ....................................................................................... 35
TABLE 2-7 METHODS FOR CONTEXT DISTRIBUTION .................................................................................. 38
TABLE 2-8 METHODS FOR CONTEXT MANIPULATION ................................................................................ 39
TABLE 2-9 METHODS FOR CONTEXT LABELLING ...................................................................................... 41
TABLE 2-10 METHODS FOR AWARENESS UTILIZATION .............................................................................. 42
TABLE 2-11 COMPARISON OF THE DIFFERENT AWARENESS OBTAINMENT METHODS PROPOSED IN LITERATURE. ............................................................................................................................................ 50
TABLE 2-12 EXAMPLES OF POLICY MODALITY TYPES ............................................................................. 60
TABLE 2-13 CONSISTENCY OF AWARENESS IN CSCW, SOFTWARE AGENTS AND POLICY STRUCTURES ...... 66
TABLE 3-1 RESPONSES TO THE REQUIRED PRODUCTS ............................................................................. 77
TABLE 4-1 VALUE ASSIGNMENT IN DIFFERENT SITUATIONS – SPACE SHUTTLE COLUMBIA DISASTER ........ 102
TABLE 5-1 MAPPING SCENARIO ANALYSIS TO SYSTEM DESIGN AND IMPLEMENTATION .............................. 157
TABLE 6-1 CONFIGURATION PARAMETERS ............................................................................................... 184
TABLE 6-2 THE RESULTS OF THE EXPERIMENTS RELATED TO SIZE – WIRELESS COMMUNICATION PROCEDURES .......................................................................................................................................... 207
TABLE 6-3 THE RESULTS OF THE EXPERIMENTS RELATED TO BRANCHING FACTOR – WIRELESS COMMUNICATION PROCEDURES ............................................................................................................. 210
TABLE 6-4 THE RESULTS OF THE EXPERIMENTS RELATED TO AVAILABILITY OF POLICY RULES - WIRELESS COMMUNICATION PROCEDURES .................................................................................................... 213
TABLE 6-5 SIGNIFICANCE AND GENERALISABILITY OF SIMULATIONS .......................................................... 222
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure Number</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1-1</td>
<td>Illustrative Example in the Space Shuttle Columbia Disaster</td>
<td>4</td>
</tr>
<tr>
<td>Figure 1-2</td>
<td>Research Scope</td>
<td>8</td>
</tr>
<tr>
<td>Figure 2-1</td>
<td>Selection Process of the Literature Review</td>
<td>27</td>
</tr>
<tr>
<td>Figure 2-2</td>
<td>Data Analysis Process in the Literature Review</td>
<td>28</td>
</tr>
<tr>
<td>Figure 2-3</td>
<td>Awareness Maintenance Framework</td>
<td>31</td>
</tr>
<tr>
<td>Figure 2-4</td>
<td>Distribution of Articles by the Year of Publication</td>
<td>43</td>
</tr>
<tr>
<td>Figure 2-5</td>
<td>Distribution of Papers Related to Each Phase of Awareness Maintenance on the Year of Publication</td>
<td>44</td>
</tr>
<tr>
<td>Figure 2-6</td>
<td>Implicit Knowledge, Awareness, and Explicit Knowledge in the Logic of General Awareness</td>
<td>57</td>
</tr>
<tr>
<td>Figure 3-1</td>
<td>Information Systems Research Paradigms - Adopted from (Hevner et al., 2004)</td>
<td>72</td>
</tr>
<tr>
<td>Figure 3-2</td>
<td>Research Framework – Adopted from (Hevner et al., 2004)</td>
<td>74</td>
</tr>
<tr>
<td>Figure 5-1</td>
<td>Concepts in Scenario Analysis Phase for Development of PAM</td>
<td>127</td>
</tr>
<tr>
<td>Figure 5-2</td>
<td>Concepts in System Design Phase for Development of PAM</td>
<td>130</td>
</tr>
<tr>
<td>Figure 5-3</td>
<td>Class PAM_Model</td>
<td>131</td>
</tr>
<tr>
<td>Figure 5-4</td>
<td>Classes Variable, DomainValue and ValuedVariable</td>
<td>132</td>
</tr>
<tr>
<td>Figure 5-5</td>
<td>Classes TrueSet, Interpreter and Proposition</td>
<td>133</td>
</tr>
<tr>
<td>Figure 5-6</td>
<td>Classes Action and DoneAction</td>
<td>134</td>
</tr>
<tr>
<td>Figure 5-7</td>
<td>Classes Event and ReceivedEvent</td>
<td>134</td>
</tr>
<tr>
<td>Figure 5-8</td>
<td>Classes PolicyRule and ForbiddingPolicyRule</td>
<td>135</td>
</tr>
<tr>
<td>Figure 5-9</td>
<td>Class Awareness</td>
<td>136</td>
</tr>
<tr>
<td>Figure 5-10</td>
<td>Class Time</td>
<td>136</td>
</tr>
<tr>
<td>Figure 5-11</td>
<td>Class World</td>
<td>137</td>
</tr>
<tr>
<td>Figure 5-12</td>
<td>Class Situation</td>
<td>137</td>
</tr>
<tr>
<td>Figure 5-13</td>
<td>Class Agent</td>
<td>138</td>
</tr>
<tr>
<td>Figure 5-14</td>
<td>Method Step: Deliberation</td>
<td>140</td>
</tr>
<tr>
<td>Figure 5-15</td>
<td>Method receivedEvents</td>
<td>141</td>
</tr>
<tr>
<td>Figure 5-16</td>
<td>Methods PAM Process for Awareness Identification and Action Commitment</td>
<td>141</td>
</tr>
<tr>
<td>Figure 5-17</td>
<td>Method implicit Knowledge of a Proposition in a Situation</td>
<td>143</td>
</tr>
<tr>
<td>Figure 5-18</td>
<td>Method implicit Knowledge of a Proposition in a Path starting from a situation</td>
<td>145</td>
</tr>
<tr>
<td>Figure 5-19</td>
<td>Method implicit Knowledge of an Until Composition of Two Propositions in a Path starting from a given situation</td>
<td>146</td>
</tr>
<tr>
<td>Figure 5-20</td>
<td>Method implicit Knowledge about a done action or a received event</td>
<td>148</td>
</tr>
<tr>
<td>Figure 5-21</td>
<td>Method Awareness</td>
<td>148</td>
</tr>
<tr>
<td>Figure 5-22</td>
<td>Method explicit Knowledge</td>
<td>148</td>
</tr>
<tr>
<td>Figure 5-23</td>
<td>Method PolicyRuleAwareness</td>
<td>149</td>
</tr>
<tr>
<td>Figure 5-24</td>
<td>Method InformationAwareness</td>
<td>150</td>
</tr>
<tr>
<td>Figure 5-25</td>
<td>Methods changeOfBehaviors</td>
<td>152</td>
</tr>
<tr>
<td>Figure 5-26</td>
<td>Method actionCommitment</td>
<td>153</td>
</tr>
<tr>
<td>Figure 5-27</td>
<td>The Space Shuttle Columbia Disaster – Simplified Object Model in the Scenario Analysis</td>
<td>158</td>
</tr>
<tr>
<td>Figure 5-28</td>
<td>Space Shuttle Columbia Disaster – Simplified Object Model in the Situation S3</td>
<td>160</td>
</tr>
</tbody>
</table>
### Table of Contents

**Figure 6-1** Rehabilitation Process, borrowed from Jester (2007) ........................................... 170
**Figure 6-2** Mobile Monitoring in Rehabilitation Video Games ...................................................... 171
**Figure 6-3** Failure in reaction-time game with problem in lower-extremity muscles .......................... 174
**Figure 6-4** Casa Grande Hazardous-materials Rail Incident .......................................................... 181
**Figure 6-5** Simulation Process ....................................................................................................... 183
**Figure 6-6** Effect of Size on Success Rate – Hypothetical Simulation ............................................. 189
**Figure 6-7** Effect of Size on Cost – Hypothetical Simulation ......................................................... 189
**Figure 6-8** Effect of Size on Efficacy – Hypothetical Simulation .................................................... 190
**Figure 6-9** Effect of Size on Cost-Efficiency – Hypothetical Simulation ........................................ 191
**Figure 6-10** Effect of Branching Factor on Success Rate – Hypothetical Simulation ....................... 193
**Figure 6-11** Effect of Branching Factor on Efficacy – Hypothetical Simulation ............................. 194
**Figure 6-12** Effect of Availability of Policy Rules on Success Rate – Hypothetical Simulation .... 196
**Figure 6-13** Effect of Availability of Policy Rules on Cost – Hypothetical Simulation .................... 196
**Figure 6-14** Effect of Availability of Policy Rules on Cost-Efficiency – Hypothetical Simulation ....... 197
**Figure 6-15** Related Communication Settings at St. Olavs Hospital in Trondheim, Norway ...... 201
**Figure 6-16** Branching-Time Model for the Nurse Agent in the Illustrative Example .................. 204
**Figure 6-17** Deployment of the Sample Procedure .......................................................................... 205
**Figure 6-18** Effect of Size on Success – Wireless Communication Procedures ............................... 207
**Figure 6-19** Effect of Size on Average Cost – Wireless Communication Procedures .................. 208
**Figure 6-20** Effect of Size on Efficacy – Wireless Communication Procedures ............................. 208
**Figure 6-21** Effect of Size on Cost-Efficiency – Wireless Communication Procedures .................. 209
**Figure 6-22** Effect of Branching Factor on Success – Wireless Communication Procedures ...... 211
**Figure 6-23** Effect of Branching Factor on Efficacy – Wireless Communication Procedures ....... 212
**Figure 6-24** Effect of Availability of Policy Rules on Success Rate – Wireless Communication Procedures ...................................................................................................................... 214
**Figure 6-25** Effect of Availability of Policy Rules on Average Cost – Wireless Communication Procedures ...................................................................................................................... 214
**Figure 6-26** Effect of Availability of Policy Rules on Cost-Efficiency – Wireless Communication Procedures ...................................................................................................................... 215
Chapter 1

INTRODUCTION

In a time when information technologies (IT) are increasingly involved in individual’s lives, cooperative environments such as social networks, B2B e-commerce, healthcare and disaster management teams need to adapt themselves to the new approaches in cooperation. The evolution of cooperative environments has been marked by the emphasis given to the methods of how to utilise intelligent IT tools to enhance cooperation among participants (Ray, Shahrestani, et al., 2005). As such, cooperative environments have to deal with information uncertainty (Rennecker, 2005), and one issue that has arisen from the use of IT is that individuals are often overloaded with irrelevant or loosely relevant information (Leinonen et al., 2005). This requires methods to identify the relevance of information as new, certain information comes to the fore.

Research and design practices in computer supported cooperative work (CSCW) emphasise the role of awareness in understanding the relevance of information (Dey, 2001; Ray, Shahrestani, et al., 2005; Omoronyia et al., 2010) cooperation. Daneshgar & Wang (2007) encourage researchers to work on definitive methods to identify such awareness.

CSCW has recently highlighted the significant contribution of complexity-based paradigms (Zacarias et al., 2010) to replace deterministic perspectives of the
internal and external views of systems by agency principles (Magalhães, 2004). Zacarias et al. (2010) define the agency relationship as assistance given by software agents to individuals in cooperation. This has also been proposed by several other studies: (Dommel and Garcia-Luna-Aceves, 2000; van Benthem, 2003; Li et al., 2004; Ray, Shahrestani, et al., 2005; Chan et al., 2008).

The present study proposes Policy-based Awareness Management (PAM) to identify the relevance of information, here called ‘awareness’. For this to happen, PAM employs software agents to aid individuals in order to use existing policy rules as a source for awareness identification.

The motivation of this research is explained by the space shuttle Columbia disaster, US, 2003 (McDanel et al., 2006), which throughout the thesis will demonstrate the proof of concepts.

The rest of this chapter is organised in the following way: Section 1.1 illustrates the motivation of this research by the space shuttle Columbia disaster, US, 2003. Section 1.2 presents the scope of this study. Section 1.3 presents the objectives for this research, the contributions and the assumptions. Section 1.4 presents the way that this study has been conducted to achieve the research goals. Section 1.5 presents the potential audiences that are targeted by the present thesis. Section 1.6 presents how the rest of this thesis is organized.
1.1 Illustrating the Motivation: Space Shuttle Columbia

Disaster

The following scenario took place during the re-entry of the space shuttle Columbia to the earth atmosphere over Texas, USA, on Feb 1, 2003. The disaster was the disintegration of the shuttle that claimed the lives of all seven of its crew.

Very soon after the launch of the space shuttle Columbia flight in 2003, a part of the temporal protection system broke and the shuttle began to shake (McDanel et al., 2006). At the time the NASA engineering team had only very low resolution pictures of the shuttle’s situation. They recognized two possible causes for the vibrations in the shuttle: the shuttle turning around for the re-entry to the atmosphere or the damage to the temporal protection system (TPS). Accordingly, the NASA engineering team requested high-resolution imaging (Wilson, 2003) to determine the effective cause of the shakes. The NASA management, however, declared the shakes to be a turnaround issue1 and ignored the request of the engineering team. In fact, the relation between the possible damage to the TPS and the shakes was not considered by the management at all (Wilson, 2003).

Figure 1-1 shows the mental states of the NASA management. The nodes represent the situations. The arrows going into the nodes represent what event occurred in each of these situations. The lines between different situations

---

1 Re-entering to the atmosphere, the shuttle has a sharp turn. That suddenly slowed the aircraft, which could lead to some shakes. This is known as turnaround effect or issue.
represent actions that the NASA management could have taken to change the situation. The detailed specification of this diagram will be discussed in Chapter 4.

Figure 1-1 Illustrative Example in the Space Shuttle Columbia Disaster

1.1.1 Challenge: Identifying Awareness

Woods (2009) categorized the contributing factors in the Columbia disaster as:

I. Foam events are one of several sources for shaking in shuttles. Due to the several possibilities, finding the source of the shake in the Columbia accident proved difficult. In fact, the guidelines in NASA, giving too much information and possibilities that might happen put the management in a mindset of information overload, while a way to find out the relevant information could be useful. One of the problems was heavily associated
with the pressure that individuals were suffering to select the relevant information from the guidelines, due to the time limitations for the decision. In addition, the tight launch schedule put pressure on the identification and made it hard to identify the relevance of different possible causes.

II. There was no crosscheck in NASA to find and challenge a fault. When the Columbia accident happened, there were three types of risks defined in NASA: in flight anomalies, accepted risks and no-safety issues. However, as I mentioned above there were several possible sources of the shake, which could lead to very different types of risks.

III. There was overconfidence in the NASA management because of their wrong interpretation of past successes.

This study only focuses on the first factor and gives a technical solution to assist the identification of relevant information – that is, awareness – which is vital in such situations. The Columbia disaster is only used as an example to illustrate the proposal of this research. However, I admit that there were several other latent conditions in the incident. For further analyses of lessons learned from the Columbia disaster, refer to Starbuck and Farjoun (2009).

Had NASA recognised the relevance of the damage in the TPS, they would have requested high-resolution imaging from the DoD to find out whether or not
there was damage. There would also have been a procedure by spacewalk for repairing the damage (Wilson, 2003). At the time, there were policy guidelines (NASA, 2003) in NASA stating that when an aircraft experiences unusual shakes, if there is any TPS damage, the spacewalk procedure must be granted. Although the capability and the guidelines were available, the NASA management team did not identify the awareness of the TPS damage, which led them to deny the image request and which resulted in the disaster (McDanel et al., 2006).

John Goodman from NASA discusses the lessons learned from several space shuttle missions including the space shuttle Columbia (Goodman, 2007). He particularly states that safety management in flights requires sophisticated technological assistance between the variously involved members and agencies in order to recognise the relevance of information in a timely fashion. This highlights the significance of employing software agents to help identify awareness.

1.2 Research Scope

There are a vast number of technical methods for the process of awareness maintenance in which one becomes aware of information and further shapes one’s behaviours to know the information. As it is depicted in Figure 1-2, the literature review in this study shows that such methods contribute through four phases. This is also supported by the theory of situation awareness (Endsley,
1995b) and can be explained by the metaphors of pools and streams (Riemer and Haines, 2008):

I. Awareness obtainment is a process in which an individual becomes aware of relevant information.

II. Context representation involves methods for modeling and distributing what an individual is aware of.

III. Context analysis is a process in which one combines and scrutinises the relevance of information.

IV. Awareness utilisation is the process of adapting the behavior of a system in response to changes in an individual’s awareness.

In conclusion, there is a research thrust that favors technology-centric methods to assist individuals in awareness maintenance. A comprehensive awareness maintenance may typically include the above four phases of awareness obtainment, context representation, context analysis and awareness utilisation. The use of software agents at each phase copes with autonomy, social ability, reactivity and pro-activity is appreciated. While such awareness maintenance frameworks have been long targeted by research in software agents and CSCW, the concern of this thesis is within the awareness obtainment phase and, in particular, awareness identification (see Figure 1-2).
Therefore, the scope of this work addresses the employment of software agents in assisting individuals to identify awareness (see Figure 1-2). This scope is designed by the following concerns:
• It is increasingly being recognised from research in CSCW that awareness identification is important to ensure the relevance of information for developing applications that aim to enhance cooperation by removing information uncertainty (Ray, Shahrestani, et al., 2005; Daneshgar and Wang, 2007).

• The growing use of information technology makes the cooperative environments dynamic insofar as an important change can happen at an incredibly fast pace. As such, the dynamic nature of cooperative environments requires that cooperation enhancement addresses uncertainty through the recognition of the relevance of information – that is, awareness in run-time (Ray, Shahrestani, et al., 2005).

• Research in software agents has indicated the significant role of proposals that recognise relevant information when an agent is overloaded with irrelevant or loosely relevant information (Halpern and Pucella, 2010).

The investigations in the space shuttle Columbia disaster in 2003 (McDANels et al., 2006), the Casa Grande hazardous-materials rail incident in 1983 (Yuan and Detlor, 2005) and rehabilitation video games (Smith et al., 2011; Talaei-Khoei, Ray, and Parameshwaran, 2011) illustrate the important contribution of technical
solutions in assisting individuals with run-time awareness of the relevant information.

In order to address this scope, the present research proposes policy-based awareness management (PAM). Although the essential idea of PAM lies in awareness identification, PAM also illustrates the way that identified awareness changes the behavior under scrutiny (see Figure 1-2).

1.3 Research Objectives, Contributions and Assumptions

My literature review shows that research on run-time awareness identification has been encouraged in studies on both CSCW and software agents. Therefore, the present study aims at addressing the following questions:

- How can software agents assist individuals in order to identify awareness of the relevance of information in run-time?

- How can we develop such an agent system equipped with the above method for awareness identification?

In order to achieve the objectives, the present study proposes using existing policy rules. The reason for using policy rules is that they relax the need to directly ask for a desired behaviors, which is discussed here as direct order. Therefore, agents would not need to be ordered to assist individuals for run-time awareness identification.
Hence, the contributions of this research are as follows:

- **PAM framework**: This is a logical framework that can represent policy-based awareness of agents. In brief, the framework provides the definitions required in PAM.

- **Three-step process of PAM**: This is a step-by-step process to identify awareness in run-time from existing policy rules. This process is defined through the foundation provided by the proposed PAM framework.

- **Development guidelines**: In order to develop an agent system equipped by PAM, the Analysis, Design and Implementation phases are proposed, while the details of modeling and implementation are also given.

In order to produce the above contributions, the present study has been designed based on the following assumptions:

- PAM assumes a given set of policy rules. Therefore, it does not refine plain text policies to computable policy rules. Policy refinement, itself, is a well-addressed research scope (Bandara et al., 2004). Including that into the present study may distract the main objective of the work.

- PAM assumes that the given set of policy rules is conflict-free. Identifying, processing and removing conflicts among policy rules complicates the problem space of the study, and it may cause distraction
from the objective. However, the problem has been partially addressed by Moffett and Sloman (1991).

- PAM assumes that all the given policy rules are realistic and possible to follow. Otherwise, it requires proposing a strategy for policy deviation. The initial thoughts for that are given by Lewis et al. (2010).

- PAM does not consider the design, implementation or distribution of policies. The objective of PAM is to identify awareness from policies, which might not necessarily include policy development and deployment.

- PAM assumes given propositional sentences, possible situations, events, actions and paths (in Chapter 4 it will be called the branching-time model). Therefore, PAM does not contribute to the generation or modification of the given actions or paths. The problem targeted in PAM sits in the category of identification of mental attitudes while generating branching-time model can be addressed in plan design. Therefore, avoiding distraction from the objective, the study focuses on only identification of awareness.
• PAM assumes that only given events can occur. This is used, in the study, to fix the branching-time model; otherwise, there could be situations that the agent, not recognizing the event, could be blocked.

• PAM assumes actions that are executable by the system. Otherwise, the agent might not be able to utilize its behaviour based on its awareness.

In the next section, the choice of the research methodology in this study is discussed.

1.4 Research Methodology

This thesis focuses on producing the contributions given above, therefore a design-science approach is chosen for this study. Hevner’s framework (Hevner et al., 2004) is adopted to generate the contributions.

In order to conduct design-science research, there have been several approaches proposed in the literature such as (Nunamaker and Chen, 1990), (Walls et al., 1992), (March and Smith, 1995), (Venable and Travis, 1999), (Markus et al., 2002), (Venable, 2006), (Gregor, 2006), and (Baskerville et al., 2007). Venable (2010) compares different framework for design-science research in information systems and claims that the approach taken by the Hevner’s framework have come to dominate the criteria, standards, guidelines, and expectations of design-science studies. The structured and stepwise nature of the Hevner’s framework
Introduction

has made it as a proper choice for design-science research. However, there are other attempts, some of which are named above.

Following on from Hevner’s framework, in order to illustrate the motivation of the research, three different proofs of concepts are made through the space shuttle Columbia disaster in 2003 (McDanel et al., 2006), the Casa Grande hazardous-materials rail incident in 1983 (Yuan and Detlor, 2005) and rehabilitation video games (Smith et al., 2011; Talaei-Khoei, Ray, and Parameshwaran, 2011). The bodies of knowledge have been studied in CSCW, software agents and policy specification in distributed cooperative systems. The contributions of PAM have been developed through the relevant methods borrowed from the literature. The relevance of study to practice is demonstrated by applying PAM in the identified exemplars. PAM is also evaluated by a triangulation of two kinds of simulations: hypothetical simulations and simulations of wireless communication procedures at St. Olavs Hospital in Trondheim, Norway.

1.5 Intended Audiences

This thesis is intended for software engineers and researchers who are concerned with technology-centred applications in CSCW. In particular, the present study targets audiences that look at developing agents who are able to propose solutions for the problems of uncertainty, thereby helping individuals who are overloaded by an abundance of information. This research is also intended for
researchers in policy-based distributed systems that seek to apply different applications of policies in CSCW.

1.6 Thesis Structure

The rest of this thesis is organised in the following way:

Chapter 2 presents the literature review and background for this study. This chapter defines awareness and provides theoretical supports for the thesis. The chapter systematically reviews articles, published from 1970 to 2010, which are concerned with methodological aspects of awareness maintenance from a technology point of view. The systematic review results in a four-phase framework of awareness maintenance. The chapter analyses the trends and the gaps in knowledge. It also provides the background knowledge on awareness in the fields of CSCW, software agents and policy specification in distributed cooperative systems.

Chapter 3 presents the research methodology chosen for this thesis. The chapter gives the research objectives as well as contributions that it will make to the existing knowledge base. Apart from discussing how the present research has been conducted, this chapter also evaluates reliability of the proposed research methodology against the available guidelines provided in the literature.

Chapter 4 presents the first two contributions of this thesis. The chapter provides formalism on the proposal of this thesis for the use of policy rules in order to
identify awareness. This chapter builds the PAM framework for representing policy-based awareness in software agents on top of the logic of general awareness. It also proposes a three-step process for awareness identification in the PAM framework.

Chapter 5 presents the third contribution of this thesis. It discusses the development aspects of PAM and provides the details of the analysis, design and implementation phases.

Chapter 6 presents the evaluation of PAM conducted by proofs of concepts and simulations. It discusses two specific scenarios, in addition to the space shuttle Columbia disaster, in order to illustrate the relevance and applicability of PAM in real world. This chapter also provides simulation studies on the efficacy and cost-efficiency of PAM. The simulations triangulate the experiments on hypothetical examples generated by computer and experiments on wireless communication procedures at St. Olavs Hospital in Trondheim, Norway.

Chapter 7 considers the implications of this study for academia as well as for practice. This chapter also specifies the unavoidable limitations in the present work, which open new directions for future research. The chapter finally concludes the thesis.
LITERATURE REVIEW AND BACKGROUND\textsuperscript{1}

The purpose of this chapter is to set the present study in the context of the related literature. First, the chapter presents the different definitions of awareness. It then undertakes an analytical summary of awareness maintenance literature in relation to how we can utilise information technology tools and methods to assist individuals to become aware of relevant information and change their behaviour accordingly. The present chapter conducts a systematic literature review on methods that have been proposed for this purpose between 1970–2010. The literature review reveals that the research in this area lacks definitive methods for awareness identification; that is addressed by Policy-based Awareness Management (PAM), which constitutes the main objective of this thesis. In order to explicate the development of PAM, background information on awareness in computer supported cooperative work (CSCW), software agents and policy structures is provided.

The rest of this chapter is organised in the following way: Section 2.1 presents the literature review. This section gives the different definitions of awareness and summarises the methods proposed in literature for awareness maintenance.

\textsuperscript{1} An abridged version of this chapter can be found in \textit{An Awareness Maintenance Framework}, Journal of Network and Computer Applications (Ranked A by The Excellence in Research for Australia), Volume 35, Issue 1, Pages 199-210, 2012.
Section 2.2 presents the background of awareness in CSCW as well as software agents. It also explains DEN-ng policy structure.

2.1 Literature Review

This section presents a systematic literature review to gain insight into the growing area of awareness maintenance. A systematic review of papers published from 1970 to 2010 examines the background and trends of research in this area. The results establish a framework for awareness maintenance and demonstrate trends, gaps and potentially fruitful areas for the research conducted in this thesis. Based on 131 papers, a four-phase framework for awareness maintenance is proposed.

2.1.1 Context, Awareness and Theories for Awareness Maintenance

There exist a number of reviews of information awareness in the literature of CSCW (Sarma, 2005; Storey et al., 2005; Bricon-Souf and Newman, 2007; Omoronyia et al., 2010). They define five types of awareness: (I) Workspace awareness (Gutwin et al., 1995), (II) Common-sense Awareness (Gutwin and Greenberg, 2007), (III) Group Awareness (Carl Gutwin & Saul Greenberg 2007), (IV) Social Awareness (Acquisti and Gross, 2006) and (V) Context Awareness (Gross et al., 2005). Table 2-1 provides the proposed definitions for the different types of awareness and compares them in terms of the types of information they cover.
### Table 2-1 Comparison of different Awareness Types

<table>
<thead>
<tr>
<th>Awareness Type</th>
<th>Definition</th>
<th>About Artifact</th>
<th>About People</th>
<th>About Activity</th>
<th>About Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workspace Awareness</td>
<td>Up-to-the-minute information about the existence of entities in a shared workspace.</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Common-sense Awareness</td>
<td>General sense of who is around and what belong to them.</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group Awareness</td>
<td>Understanding of people in the group, their responsibilities and their status.</td>
<td></td>
<td></td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Social Awareness</td>
<td>Information about presence of people and their activities.</td>
<td></td>
<td></td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Context Awareness</td>
<td>Cognisance of an internal or external entity that causes change in a situation.</td>
<td></td>
<td></td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

Omoronyia et al. (2010) argue that context awareness cuts across the other types of awareness. This awareness considers the changes in states of the other awareness types. People work on different artefacts and activities, in different situations and with different people. Such awareness is highly contextual and cannot be addressed by the other types of awareness. Ray et al. (2005) define context awareness as an understanding of relevant information that is required for an individual. Omoronyia et al. (2010) define the relevant information to an individual as information that fully characterises the desired situations of that individual. The desired situation is the intended situation in which the individual’s behaviour will result. Therefore, awareness is relevance of information that an

---

1 A mediator for any sort of interaction among participants. An artefact can be physical, such as a report, or can be cognitive, such as skills and experiences. For more details about the concept of artefacts, readers must refer to (Omicini et al., 2008).
individual is required to know. By accepting this definition for awareness and considering the above mentioned understanding of relevance, I take the both accounts of awareness introduced by Ray et al. (2005) - emphasizing on individuals, and introduced by Gross et al. (2005) – emphasizing on situation. This relates the two views for context awareness in the literature. Dey et al. (2001) consider context as information about the entity of which the individual is aware. Although the notation of context awareness shows the relevance of the context, it does not refer to the knowledge about the context and does not necessarily infer the validity of the context (Omoronyia et al., 2010). My concept of awareness, as relevance of information, is most closely related to context awareness.

The dominate view in the literature favours the use of information technology to assist individuals in becoming aware of relevant information (Hong, Suh, and Kim, 2009). There are bodies of work (Dommel and Garcia-Luna-Aceves, 2000; van Benthem, 2003; Li et al., 2004; Ray, Shahrestani, et al., 2005; Chan et al., 2008) that propose the use of software agents to assist individuals in achieving context awareness. One of the major questions in CSCW is how to utilise intelligent software agents to maintain contextual awareness (Ray, Shahrestani, et al., 2005).

Endsley (1988, 1995a, 1995b, 2001) propounds the theory of situation awareness, which is concerned with developing information technology solutions for
information overload and uncertainty. His argument lies in understanding how people process information to arrive at a behaviour that removes a perceived uncertainty. In this theory, an individual’s awareness in a situation can be thought of as an internalised mental model that is concerned with the relevance of information to the situation for that individual. Endsley believes that developers should understand how to maintain awareness from the vast amounts of information that is now available from an abundance of information technology tools. In this theory, awareness maintenance involves capturing the relevance of information (Level 0); perception of the relevance of information (Level 1); understanding what this relevance means, particularly in relation to comprehension of the situation (Level 2); and at the highest level, considering the available information awareness, a projection of what will happen with the system in the future (Level 3). A higher level of situation awareness allows people to function in a more timely and effective manner. There are also some examples for application of this theory in (Endsley, 1993).

Riemer and Haines (2008) introduce another theory that uses the metaphor of “pools and streams” to explain the process of awareness maintenance. The metaphor refers to relevance of information that is identifiable, albeit to some extent unknown. In this metaphor, pool refers to awareness and stream refers to flow of knowledge. Therefore, when one identifies the lack of knowledge about relevant information, she or he becomes aware of that information and creates a
pool. Consequently, when one adapts a behavior to gain knowledge about relevant information, one directs streams to the created pool. This results in increased knowledge about what the individual is aware. Riemer and Haines (2008) define awareness maintenance as a process in which one becomes aware of a context and then shapes one’s behaviours to know that context.

Consistent with the above definitions, the following will explain My systematic review of awareness maintenance.

2.1.2 Methodology of Systematic Review

The study was undertaken as a systematic literature review based on the original guidelines proposed by Kitchenham (2004) and Kitchenham et al. (2009). In this case, the goal of the review is to come up with a framework for awareness maintenance by reviewing the work in this field. The steps for this study are as follows: (I) identification of resources, (II) selection, (III) data extraction and synthesis, and (IV) data analysis.

2.1.2.1 Identification of Resources

In order to identify the resources, the first step is to recognise the relevant keywords. For this purpose, I adapted an experimental strategy given by Dieste et al. (2008). In this strategy, an optimum search is performed by keywords that retrieve articles of which 20–25% (that is, the precision rate) are related to the topic. Taking the objective of this review for investigating the literature of
awareness maintenance, the term “awareness” was searched in Google scholar for publications between 1970–2010 in the field of “Engineering, Computer Science, and Mathematics”. The first 100 papers found in the search were considered, and in each of them, terms related to “awareness” were identified. Therefore, “context” by precision rate of 23.7% and “situation” by precision rate of 20.9% were selected to be included. As such, “maintenance” was searched in combination with each of these terms – “awareness maintenance”, “context maintenance” and “situation maintenance”. In this way, “create” with precision rate of 22.6%, “sensitive” with precision rate of 20.6%, “obtain” with precision rate of 23.3%, “identify” with precision rate of 24.1%, and “develop” with precision rate of 24.8% were recognised. The limitation of systematic reviews is that they are heavily dependent on the chosen keywords, as has been observed in the literature of this research method (Kitchenham et al., 2009).

The articles were searched using eight online databases (see Table 2-2). Depending on the search services offered by the databases, the titles, keywords and the abstracts were searched in order to locate papers that have been published between 1970 and 2010, inclusive. In each database, the search was repeated three times by the following phrases:

- \(^1\)AND [(awareness) (OR (create sensitive obtain identify develop))]

\(^1\) Note that the operators come prior to the parameters.
• AND [(context) (OR (create sensitive obtain identify develop))]

• AND [(situation) (OR (create sensitive obtain identify develop))]

For example, the first phrase means all the articles that have the keyword “awareness” and any of the keywords “create”, “sensitive”, “obtain”, “identify” and “develop” in their titles, abstracts or keywords. For each search result, Google scholar and Citeseer were searched to find those papers that have cited the found paper. Therefore, for the papers indexed by these two search engines, those with less than five citing papers were not included in the final list of papers.

The articles were searched in a comprehensive list of subjects. The names of the databases, the subjects, the number of found articles and the number of repeated papers are listed in Table 2-2. 14699 articles were found in total, while 5692 papers were repeated. The selection process excluded the repeated articles from the archive and resulted in 9073 papers.

2.1.2.2 Selection

The objective of this step was to find the articles and exclude the papers that are not relevant to the topic of “awareness maintenance”. Figure 2-1 depicts the selection process. This process had two iterations.
In the first iteration, the aforementioned keywords were searched in the eight databases. Then, steps 1.2, 1.3, and 1.4 excluded 8511 articles based on their titles, keywords and full texts. These steps excluded articles that

- did not focus on maintenance of awareness;

- did not focus on methodological aspects of awareness maintenance;

---

1 This database does not provide any service for searching abstracts. Therefore, I only searched the titles and keywords.
Literature Review and Background

- focused only on applications of awareness maintenance;
- were not in the relevant fields or could not be used in the relevant fields;
- were in languages other than English; and
- were not peer reviewed.

After the exclusion steps, the 91 remaining articles went to an archive in Zotero Research Tool (Zotero Co., 2010) for storage and organisation.

In the second iteration, the references were retrieved from the papers that had come out of the exclusion process in the first iteration. In step 2.1, the keywords were searched in the list of references and those references that are matched with the keywords were chosen. Step 2.2, 2.3, and 2.4 examined the references against the above exclusion criteria, based on their title, keywords and full texts. Then, the remaining references were added to the archive, which makes the archive to consist 131 papers.

2.1.2.3 Data Extraction and Synthesis

The data extraction and synthesis process aims to extract the key details from the 131 papers. Two types of data were extracted from the studies: (I) Methods, where data is synthesised to recognise the different methodological aspects of awareness maintenance, and (II) Demographics, such as year of publication.
Figure 2-1 Selection Process of the Literature Review
2.1.2.4 Data Analysis

In data analysis step, the collected articles were studied in order to discover the phases for awareness maintenance.

Figure 2-2 shows the analysis process for the final list of the papers. The terms and definitions in the articles formed a primary list of categories for the methods. At this stage the different problems that the different methods in the articles were trying to solve were discovered (see Table 2-3).
Having put the methods in the different categories identified by exploring the terminologies used in the articles, the following points were discovered (see Table 2-3):

- Awareness creation can happen in design-time as well as in run-time.
- Context Modeling has different types of methods: tuple and tree-like, mathematical and ontological.
- Context Distribution has different types of methods: Centralised and Distributed.
- Inference has two steps: first, context filtering, abstracting, reasoning and labeling; second, Awareness Utilisation that is concerned with shaping the behaviors by rule-based or machine learning methods.

Table 2-3 Problems Identified by Exploring the Articles found in the Literature Review

<table>
<thead>
<tr>
<th>Categories Identified by Exploring the Terminologies used in the Articles</th>
<th>Categories Identified by Exploring the Methods proposed in the Articles</th>
</tr>
</thead>
</table>
| • Awareness Creation  
• Context Modelling  
• Context Distribution  
• Inference  
• Labelling | • Awareness Creation (in design time, in run-time)  
• Security Management  
• Context Modelling (tuple and tree-like, mathematical, ontological)  
• Context Distribution (Centralized, Distributed)  
• Context Filtering  
• Context Abstracting  
• Context Reasoning  
• Context Labelling  
• Awareness Utilisation (Rule-based, Machine Learning) |
The last stage formed the category-subcategory classifications. In this stage, by looking at the different methods, the following were proposed:

- The first step in awareness maintenance is Awareness Obtainment that involves Awareness Creation and Security Management.
- The second step in awareness maintenance is Context Representation that involves Context Modeling and Context Distribution.
- The third step in awareness maintenance is Context Analysis that involves Context Manipulation and Context Labelling. Context Manipulation includes Filtering, Abstracting and Reasoning. Labelling is related to artifacts, persons, time or location.
- The fourth step in awareness maintenance is Awareness Utilisation.

Finally, taking the above points into consideration, a classification of the methods in awareness maintenance, i.e. *Awareness Maintenance Framework* was derived.

### 2.1.2.5 Results

My results include (I) an Awareness Maintenance Framework, (II) a trend analysis based on demographic data collected from the final list of papers and (III) identification of gaps in this field of research. Subsequently, I first explain the awareness maintenance framework and then I continue with the trend analysis. Finally, I present the gaps in the literature.
2.1.3 Awareness Maintenance Framework

Based on 131 papers, I classified awareness maintenance into four categories; awareness obtainment, context representation, context analysis and awareness utilisation. Figure 2-3 shows this classification framework. The following sections provide details of each category.

2.1.3.1 Awareness Obtainment

Awareness Obtainment is a process in which an individual becomes aware of relevant information. My review shows that there are two types of work in this area: (I) those that focus on creation of awareness and (II) those that highlight the security issues in obtaining awareness.
Awareness Creation is a process whereby one matches actual information with what one is required or desires to be aware of. The methods in this area usually define a dispatcher to match the required awareness with an information driver. The driver is attached to the information source to provide interfaces for the dispatcher. Research in awareness creation can be classified further in regard to their objectives. There are methodologies that look at how to create awareness in design-time, and there are bodies of work that provide technologies for awareness creation in run-time. Table 2-4 summarizes these findings.

Awareness Security Management involves traditional security concerns. Security in awareness obtainment cannot be isolated from awareness creation either in design-time or run-time.

Where security support has appeared in awareness obtainment, it often covers the following topics: Confidentiality, Trust and Identity. Confidentiality secures the relationship between individuals as they become jointly aware of information. Trust illustrates the degree of reliability of the created awareness, and Identity gives access control and authorisation to the created awareness. Table 2-5 summarizes these findings.
### Table 2-4 Methods for Awareness Creation

<table>
<thead>
<tr>
<th>Category</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Daneshgar and Ray (2000a) propose the process of cooperation enhancement based on awareness creation. They define awareness net (tree) where information in each node is attached to drivers. Once an individual is required to be aware of a piece of information, the associated node creates a dispatcher. They argue that if there are dispatchers that can be connected to its appreciated driver then this connection enhances the cooperation by creating awareness in the node dispatcher.</td>
</tr>
<tr>
<td></td>
<td>• (Ray, Shahrestani, et al., 2005) extends the cooperation enhancement process by (Daneshgar and Ray, 2000a) measuring the required awareness as a fuzzy attribute.</td>
</tr>
<tr>
<td></td>
<td>• Riemer and Haines (2008) offer a conceptual model that proposes a theory for the dynamic creation of awareness in mediated settings using metaphor of pools fed by streams of communication. “Pools of awareness are held within users and gradually filled via signals from others. Users desire [require] different pools and direct the streams of interaction to feed those pools first.”</td>
</tr>
<tr>
<td></td>
<td>• (Zacarias et al., 2010) argue an importance of agent perspective to align individual and organisational views. This work proposes an ontological method to conceptualise awareness while designing an agent-based CSCW application for an organisation.</td>
</tr>
<tr>
<td></td>
<td>More: (Frößler et al., 2007), (Daneshgar et al., 2006), (Ray and Chattopadhyay, 2009), (Kaiser et al., 2005), (Lieberman and Selker, 2010), (Ranganathan and Campbell, 2003).</td>
</tr>
<tr>
<td>Design-time</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Context Tailor (Davis et al., 2003) provides a well-defined service called context service. The Context Service has a well-defined dispatcher and driver.</td>
</tr>
<tr>
<td></td>
<td>• SOCAM (Gu et al., 2005), Gaia (Román et al., 2002), (Yiqiang Chen et al., 2009), (Salem and Rauterberg, 2004), (de Freitas Bulcao Neto and da Graca Campos Pimentel, 2005), (Oh and Woo, 2009) and (Borovoy et al., 2010) provide a service called context provider. This service interacts with the available contexts and creates awareness in the run-time.</td>
</tr>
<tr>
<td></td>
<td>• (Dey et al., 2001), (Dey, 2009), and (Eddy and Pei, 2010), in a similar way to SOCAM and Gaia, provide widgets as drivers and aggregators as dispatchers to create awareness in run-time.</td>
</tr>
<tr>
<td></td>
<td>• Merino (Kummerfeld et al., 2003), (Chen et al., 2010), (Little, 2010) and (Gellersen et al., 2002) propose methods for creating awareness at the lowest level by interpreting historical references.</td>
</tr>
<tr>
<td>Run-time</td>
<td></td>
</tr>
<tr>
<td></td>
<td>More: (Keiser and Kriengchhaiyapruk, 2008), (Biegel and Cahill, 2005), (Lalbakhsh et al., 2009), (Taconet et al., 2009), (Müller et al., 2010), (Wustmann et al., 2010), (Baladrón et al., 2010), (Chtcherbina and Franz, 2003).</td>
</tr>
</tbody>
</table>
Table 2-5 Methods for Awareness Security Management

<table>
<thead>
<tr>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>• (Sliman et al., 2009) proposes a framework that collects and generates policy-based security in cross-organisational scenarios. In addition to catering to specifications of security and business policies, the framework integrates contextual information to make the role-based framework flexible and express confidentiality requirements of users.</td>
</tr>
<tr>
<td>• MeCoCo (George and Lekira, 2009), (Khedr and Karmouch, 2005a) and (Gu et al., 2005) propose a fairly generic user's awareness framework for mediated communications. However, this work argues that the awareness cannot be totally generic, as it may harm confidentiality requirements. As such, MeCoCo gives some classes based on the domain ontology which satisfy the confidentiality.</td>
</tr>
<tr>
<td>• CASPER (Chow et al., 2009), (Khungar and Riekki, 2005), (Katsiri and Mycroft, 2006) and (Barbosa et al., 2008) propose a modelling-based method for awareness confidentiality. It models the world by nested containers. The hierarchy of containers provides abstraction in the model.</td>
</tr>
<tr>
<td>• Cosmos (Kim, Lee, Lee, and Ryou, 2008a) provides an integrated awareness framework over the network sensors. The framework consists of a security manager that provides Trust, Authorisation, Authentication and Confidentiality services to the awareness engine.</td>
</tr>
<tr>
<td>• (Sheikh et al., 2008), (Shabtai et al., 2010), (Griswold et al., 2003) provides labelling (explained in Section 2.1.3.3) to the awareness. This method uses generic quality attributes. Two of these attributes are Trust and Identity.</td>
</tr>
<tr>
<td>• (Solt et al., 2009) provides a method to classify awareness (in case of this paper, it is classification of diseases) using security policies covering Trust and Confidentiality. This paper does not look at Identity.</td>
</tr>
<tr>
<td>• (Shand et al., 2004; Biegel and Cahill, 2005; Patrikakis et al., 2009), (Butler, 2001), (Sharifi and Naghavian, 2010) and (Moura et al., 2009) use human notation of Trust for awareness in collaborative environments.</td>
</tr>
<tr>
<td>• (Nugent et al., 2008), (Xirouchaki et al., 2002), (Chang et al., 2008), (Kühn et al., 2010), (Baladrón et al., 2010) and (Hong and Landay, 2004) apply access control and authentication policies in its defined awareness obtainment process.</td>
</tr>
<tr>
<td>• Gaia (Román et al., 2002), COCON (Wang et al., 2004), (Munnelly et al., 2007) and (Şensoy et al., 2009) provide access control for awareness.</td>
</tr>
</tbody>
</table>

More: (Robinson and Beigl, 2004), (Hoffmann, 2005), (Hoffmann and Stotz, 2005), (Bhatti et al., 2005) and (Moore et al., 2010).

2.1.3.2 Context Representation

Context Representation involves techniques for modelling and distributing the awareness. Modelling refers to awareness research that gives context structures while addressing the following problems: distributed versus centralised awareness, validation of awareness, quality of awareness, uncertainty in awareness, formal representation of awareness, and the implementation of awareness. My review
shows that research in awareness supports three categories of context modelling techniques: (I) Tuple and tree-like modelling, (II) Mathematical modelling and (III) Ontological modelling. Table 2-6 summarises these findings.

Table 2-6 Methods for Context Modeling

<table>
<thead>
<tr>
<th>Category</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuple and Tree-like Modelling</td>
<td>• Multi-granularity model (Niu et al., 2010) uses tuples to exploit the relationships among different attributes of awareness, together with the corresponding multi-granularity management approach to strengthen the flexibility and context of dynamic service composition.</td>
</tr>
<tr>
<td></td>
<td>• (Schilit et al., 1994) models the context of application by pairs of tuples expressing the attributes(^1) and their values. This approach emphasises the dynamism of context by changing the value of attributes in the tuples.</td>
</tr>
<tr>
<td></td>
<td>• (Schnaiter et al., 2009) uses tuple-based modelling of context and proposes estimation for giving value to the attributes.</td>
</tr>
<tr>
<td></td>
<td>• (Dey, 2009) provides a tree-like hierarchical context modelling technique for ambient systems</td>
</tr>
<tr>
<td></td>
<td>• CASPER (Chow et al., 2009) builds containment trees to model context. The objective is to provide a model-based confidentiality method. The tree represents nested containers to model the abstraction of entities.</td>
</tr>
<tr>
<td></td>
<td>• Mark-up languages are typically built upon a generic “profile” that represents context. The mark-up languages represent awareness using both tuple and tree-like approaches. The basic schema for these languages is Standard General Mark-up Language. Some of these approaches are as follows:</td>
</tr>
<tr>
<td></td>
<td>• A category of these languages is based on Composite Capabilities/Preferences Profile (CC/PP) such as DELI (Butler, 2001) and CSCP (Held et al., 2002) or User Agent Profile (UAProf) such as (Hinz et al., 2007). These approaches are based on RDF expressiveness structure and XML serialisation. They normally extend the standard CC/PP or UAProf approach to address complexity and dynamism problems as popular challenges in this area. CC/PP Context Extension (Indulska et al., 2003) extends both CC/PP and UAProf by number of component-attribute trees related to some aspect of context awareness.</td>
</tr>
<tr>
<td></td>
<td>• Pervasive Profile Description Language (PPDL) (Chtcherbina and Franz, 2003) that models the context with emphasis on their dependencies while the different contextual aspects and artefacts remain limited. Few parts of this language are available to the public.</td>
</tr>
<tr>
<td></td>
<td>• Another category of mark-up languages is based on Usage Environment Description (UED) such as (Capra et al., 2001) and Digital Item Adaption (Vetro et al., 2006) or (Barbosa et al., 2008). This XML-based language</td>
</tr>
</tbody>
</table>

\(^1\) These attributes sometimes are called “variables”. In some logic-based systems a propositional sentence can be presented by a set of tuples made of variables and values.
Literature Review and Background

<table>
<thead>
<tr>
<th>Category</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>covers context about four categories of information: (1) the user characteristics and her/his preferences, (2) Hardware, (3) Network, and (4) Environment.</td>
</tr>
<tr>
<td></td>
<td>- Awareness Net (Daneshgar and Ray, 2000b; Ray, Shahrestani, et al., 2005) proposes to model context during cooperation with a tree structure. Information in each node is attached to drivers. Once an individual is required to be aware of a piece of information, the associated node creates a dispatcher.</td>
</tr>
<tr>
<td></td>
<td>More: (Bhatti et al., 2005), Citron (Yamabe et al., 2005), Confab (Hong and Landay, 2004), Context Shadow (Jonsson, 2001), (Sharifi and Naghavian, 2010) and (Sharifi et al., 2009).</td>
</tr>
<tr>
<td></td>
<td>Mathematical Modelling</td>
</tr>
<tr>
<td></td>
<td>- McCarthy Model (McCarthy, 1993) avoids giving an explicit definition for context. Instead, the model presents context as an abstract mathematical entity with relevant properties to the situation. The significance of the model is in lifting the truth of a property from one context to another. The McCarthy model supports inheritance.</td>
</tr>
<tr>
<td></td>
<td>- Extended Situation Theory (Akman and Surav, 1997) uses first-class objects of situation theory to represent context related to a particular point of view. They model context by parameter-free expressions supported by situation types that corresponds to the context.</td>
</tr>
<tr>
<td></td>
<td>- (And et al., 1998) uses formal first order predicate logic representation to facilitate the composition of context of individuals into a more complex sensed context.</td>
</tr>
<tr>
<td></td>
<td>- Gaia (Román et al., 2002) implements context using First Order Logic operations such as qualifications, implications, and conjunctions.</td>
</tr>
<tr>
<td></td>
<td>- (Ghidini and Giunchiglia, 2001) is more concerned with reasoning about goals using context; it defines context as individuals’ subjective perspective of the current situation.</td>
</tr>
<tr>
<td></td>
<td>- Logic of General Awareness (Fagin and Halpern, 1988; Sillari, 2008b) models awareness as a set of relevant information to a situation and develops logic to change implicit knowledge to the explicit knowledge using awareness.</td>
</tr>
<tr>
<td></td>
<td>- (Zhang and Li, 2007) describes awareness with dynamic fuzzy logic. In order to handle errors in the sensors, the system models robustness using awards.</td>
</tr>
<tr>
<td></td>
<td>- (Liu, 2010) uses fuzzy logic to model context in image processing. The significance of the model is to improve the accuracy of texture classification.</td>
</tr>
<tr>
<td></td>
<td>- (Sakhanenko and Luger, 2010) models context about change of context using a first-order logic-based probabilistic modeling language called Generalized Loopy Logic (GLL).</td>
</tr>
<tr>
<td></td>
<td>More: (Katsiri and Mycroft, 2006), (Bannon and Hughes, 1993).</td>
</tr>
<tr>
<td></td>
<td>Ontological Modelling</td>
</tr>
<tr>
<td></td>
<td>- (Strang and Linnhoff-Popien, 2003) proposes the Aspect Scale Context (ASC) model to provide a uniform model for model core concepts as well as an arbitrary amount of sub-concepts. This allows us to model context awareness. ASC implements monolithic Context Ontology Languages (CoOL).</td>
</tr>
<tr>
<td></td>
<td>- CONON (Wang et al., 2004) provides upper context ontology for context. This approach models general concepts about basic context, and also provides the ability to add hierarchical domain-specific ontology.</td>
</tr>
<tr>
<td>Category</td>
<td>Methods</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
</tr>
<tr>
<td>• (Kofod-Petersen and Mikalsen, 2005) divides context in the following five categories integrated to form domain ontology: task, social, personal, spatio-temporal and environmental.</td>
<td></td>
</tr>
<tr>
<td>• SOCAM (Gu et al., 2005) provides a two-level ontology to model context: domain independent and domain specific.</td>
<td></td>
</tr>
<tr>
<td>• ACAI (Khedr and Karmouch, 2005b) provides ontology for context-aware applications.</td>
<td></td>
</tr>
<tr>
<td>• KAD (Evangelou et al., 2005) provides context ontology by interweaving concepts from the Knowledge Management, Argumentation Theory, Decision Making, and Multi-Criteria Decision Aid disciplines.</td>
<td></td>
</tr>
<tr>
<td>• PLIB (Pierra, 2008) provides conceptual ontology for awareness in industrial manufacturing.</td>
<td></td>
</tr>
<tr>
<td>• (Segev and Gal, 2007) provides a formal mathematical framework that delineates the relationship between contexts and ontologies. The main purpose of this framework is to manage awareness uncertainty.</td>
<td></td>
</tr>
<tr>
<td>• DEN-ng policy structure (Strassner et al., 2009) provides ontology to model context using policies.</td>
<td></td>
</tr>
<tr>
<td>• (Pereira et al. 2009) provides an ontological model of context to facilitate information retrieval.</td>
<td></td>
</tr>
<tr>
<td>• (Stojanovic et al., 2010) discusses the use of ontologies as a high-level, expressive, conceptual modeling approach for describing awareness.</td>
<td></td>
</tr>
<tr>
<td>• CAUCE (Tesoriero et al., 2010) is a model-driven development approach based on three-layer ontology to implement context-aware applications.</td>
<td></td>
</tr>
</tbody>
</table>

More: (Ejigu et al., 2008), (De Leenheer et al., 2007), (Castano et al., 2006), (Soylu et al., 2009) and (Hervás et al., 2010).

According to what I have adverted to above, awareness is ideally needed to support distributed computing. However, this lacks central control and it creates opportunities to provide highly dynamic structures for context. Distribution refers to methods that represent context in centralised or distributed processing. Table 2-7 summarizes My finding in different techniques for distributing context.

2.1.3.3 Context Analysis

In computer supported cooperative work, an individual becomes aware of multiple types of context from a variety of sources. The relevance of information to the individual can be meaningless, ambiguous or imprecise.
Table 2-7 Methods for Context Distribution

<table>
<thead>
<tr>
<th>Category</th>
<th>Methods</th>
</tr>
</thead>
</table>
| Centralised | • (Ranganathan et al., 2004) uses spatial centralised database to persist context models.  
• SOCAM (T. Gu et al. 2005) includes a context database that stores ontological context models.  
• PACE (Moon et al., 2007) stores context, application, user data, domain knowledge and behavior in a centralised database.  
• LORE (Chen et al., 2010) addresses the aspects of building location-aware centralised services, including positioning, location-dependent query processing, tracking and intelligent location-aware message notification. Three key components of the infrastructure are the location server, a moving object database, and a spatial publish/subscribe engine. |
| Distributed | • (Kaiser et al., 2005) provides a distributed event-based context representation. It considers the quality requirements. COSMIC also supports event channels with different timeliness and reliability classes.  
• (Khungar and Riekki, 2005) is a distributed physical storage that uses distributed servers to store context models.  
• (Malik et al., 2009) estimates individual preferences by distribution of their context models.  
• Smart-Context (Moore et al., 2010) stores personalised context models. It uses OWL for communication between different nodes. |
| More: (Baladrón et al., 2010) and OLLAF (Garcia and Granado, 2009). | More: WBLIS (Moura et al., 2009) and (Liu, 2010). |

Analysis is the process in which one combines and scrutinises the relevance of information in order to interpret the awareness and manage uncertainty of acquired awareness. Analysis includes manipulating contexts to give them meaning while labeling them with metadata.

Manipulation describes methods that involve processing a set of contexts by adding abstract contexts in order to infer different contexts. This includes filtering, abstraction and reasoning. Filtering involves techniques dedicated to addressing the problem of validating the relevant information. These methods
generally address the problem of awareness overload by identifying redundancy and contradictions in a set of contexts. While filtering removes some information from the set of relevant information, abstraction and reasoning can make the context increasingly meaningful, by raising the abstraction level of the context and relating it to the lower level abstraction. Reasoning involves the discovery of relations among the data of which one is aware. Table 2-8 illustrates the methods proposed in the literature for these elements of context manipulation.

Table 2-8 Methods for Context Manipulation

<table>
<thead>
<tr>
<th>Category</th>
<th>Methods</th>
</tr>
</thead>
</table>
| Filtering  | • SOCAM (Gu et al., 2005) filters the context in order to remove conflicts and have higher degrees of consistency.  
• (Pinheiro et al., 2010) filters the context based on preferences.  
• Sentient (Biegel and Cahill, 2005) filters the context in order to ensure certainty. It uses Bayesian networks.  
• I-Gaia (Xirouhaki et al., 2002) validates relevance of information to filter the context. This method can be used by fuzzy logic or First Order Logic.  
• (Hong, Suh, Kim, et al., 2009; Şensoy et al., 2009) use ontology to filter redundant context.  
More: CARISMA (Capra et al., 2003), MoCoA (Senart et al., 2006), (Chang et al., 2008), (Kirsch-Pinheiro et al., 2005). |
| Abstracting| • Gaia (Román et al., 2002) provides logic-based abstraction.  
• (Serral et al., 2010) provides a meta-model to abstract the context.  
• (Kühn et al., 2010) presents a concept for the knowledge-driven opto-acoustic scene analysis based on an object-oriented modeling approach to recognise the required level of abstraction for the available context.  
• (Shabtai et al., 2010) uses temporal concepts to abstract the context.  
• (Hightower et al., 2002) uses a method based on probability to recognise the relations between contextual entities and create the higher level of abstraction.  
• EnviroTrack (Krishnamurthy et al., 2004) abstracts the context based on the network infrastructure.  
More: (Chang et al., 2008), (de Freitas Buleao Neto and da Graca Campos Pimentel, 2005), (Griswold et al., 2003), (Munnelly et al., 2007), (van Kranenburg et al., 2006) and CASPER (Chow et al., 2009). |
Literature Review and Background

<table>
<thead>
<tr>
<th>Category</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reasoning</td>
<td>• (Demetriou and Kazi, 2006) manipulates context by reasoning about tasks.</td>
</tr>
<tr>
<td></td>
<td>• Logic of General Awareness (Fagin and Halpern, 1988; Sillari, 2008b) is a classical model First Order Logic that provides a method to manipulate awareness by reasoning about awareness of others’ knowledge or awareness.</td>
</tr>
<tr>
<td></td>
<td>• (Agostini et al., 2009) reasons about context using ontology techniques.</td>
</tr>
<tr>
<td></td>
<td>• (Cheng and Marsic, 2001) reasons about context using fuzzy logic.</td>
</tr>
<tr>
<td></td>
<td>• (Zimmermann, 2003) reasons about context using case-based reasoning techniques.</td>
</tr>
<tr>
<td></td>
<td>• (Julien et al., 2004) provides a state-based formal reasoning technique about location-aware mobility.</td>
</tr>
<tr>
<td></td>
<td>• CARE (Agostini et al., 2009) uses hybrid reasoning, which supports context awareness in web services.</td>
</tr>
<tr>
<td></td>
<td>• (Halpern and Régo, 2009) provides awareness by reasoning about the knowledge of not-being-aware (that is, unawareness).</td>
</tr>
<tr>
<td></td>
<td>• (Guesgen and Marsland, 2010) provides awareness by reasoning in temporal and special aspects of context.</td>
</tr>
<tr>
<td></td>
<td>• (Ghidini and Giunchiglia, 2001) uses awareness to reason about goals. Similar to the partial theory of Worlds, it takes individuals’ subjective perspective for the current situation to reason about the goal.</td>
</tr>
<tr>
<td></td>
<td>More: (And et al., 2001), (Halpern and Pucella, 2010), (Sillari, 2008a), (Cummins et al., 1991), (Wustmann et al., 2010), (Ma et al., 2009) and (Kofod-Petersen and Mikalsen, 2005).</td>
</tr>
</tbody>
</table>

Labelling refers to tagging relevant information to emphasise the quality of a context. Labelling describes to additional information to the context used for utilisation. There are several dimensions of labelling, such as artefactual, personal, temporal and spatial. Table 2-9 shows the methods proposed in the literature.

2.1.3.4 Awareness Utilisation

When Riemer and Haines (2008) define awareness maintenance, they use the term “shape of behaviors”. This means, from their point of view, the last step for awareness maintenance is when an individual changes his or her behavior based on the obtained awareness to information that has been represented and analysed in previous steps.
Table 2-9 Methods for Context Labelling

<table>
<thead>
<tr>
<th>Category</th>
<th>Methods</th>
</tr>
</thead>
</table>
| Artefactual | • WS-Café (Little, 2010) provides standards for labelling contextual artefacts.  
              • Aura (Ge et al., 2008) supports quality attributes for context.  
              More: (Bailey et al., 2002), COSMIC (Kaiser et al., 2005). |
| Personal | • (Oh and Woo, 2009) is specified for mobile applications and enables users to share their experiences in the format of labels and labels on the context.  
          More: (Sheikh et al., 2008). |
| Temporal | • Time-Frames method (Koen and Bender, 2000) provides labelling based on temporal attributes of context.  
          • MUPE (Salem and Rauterberg, 2004) labels context using temporal and certainty attributes.  
          More: (Cattuto et al., 2010), (Alduncin, 2009). |
| Spatial | • LoSeCo (Yiqiang Chen et al., 2009) proposes a method to tag location context in pervasive computing.  
          More: (Zhang and Li, 2007), (Borovoy et al., 2010), COSMIC (Kaiser et al., 2005). |

Utilisation is the process of adapting the behavior of a system in response to changes in the context of which the system is aware; or in other words, it shows the change in the system behaviours by the change of the relevant contexts in the environment. However, many awareness maintenance methods such as (Dey et al., 2001; Schmidt et al., 2004; Bardram, 2005) do not support utilisation; infrastructures that provide utilisation are usually based on rule-based or machine learning techniques. Table 2-10 shows the different approaches in the literature for utilisation.
Table 2-10 Methods for Awareness Utilization

<table>
<thead>
<tr>
<th>Category</th>
<th>Methods</th>
</tr>
</thead>
</table>
| Rule-based     | • Context Tailor (Davis et al., 2003) provides an API containing contextual, temporal, and statistical components for specifying rules. These APIs are called patterns. The pattern activator triggers an action if the context of the service matches with the context of the pattern.  
• CARISMA (Capra et al., 2003) defines intra- (i.e. external) and inter- (i.e. internal) policy rules. The awareness utilisation happens once inter- and intra-policy rules conflict in a given set of awareness.  
• (Korpipää et al., 2005) the awareness rules are defined in XML. The rule engine triggers a rule by matching the condition clauses in rule with the retrieved context.  
• DEN-ng (Strassner et al., 2009) policy structure used event-condition-action model to utilise awareness by different policy rules.  
• CASPER (Chow et al., 2009) provides a policy model. It uses ponder policy language for specifying its policy rules. The policy editor, policy specification interface and policy manager are components of the framework. CASPER utilises a given set of awareness by a given set of policy rules when the retrieved aware context matches the rule. More: (Ejigu et al., 2008), (Sánchez et al., 2008), (Kim, Lee, Lee, and Ryou, 2008b) and (Eddy and Pei, 2010). |
| Machine Learning | • Gaia (Román et al., 2002) supports Bayesian, neural network and clustering for machine learning for utilisation.  
• Context (Kofod-Petersen and Mikalsen, 2005) uses a case-based method for utilisation. The significant thing is to retrieve a learnt case in order to decide what action to take in the current context.  
• (Lieberman and Selker, 2010) proposes a design methodology to use of machine-learning techniques for utilisation. More: (Ranganathan and Campbell, 2003) and (Korpipaa et al., 2003). |

2.1.4 Trend Analysis

In this section, I present a demographic analysis of the 131 articles. Figure 2-4 demonstrates the distribution of the papers over the years of publication. The figure shows that after the year 2000, researchers have become more interested in the topic of awareness maintenance. Further, the number of articles published between 2005 and 2010 radically increased.
Figure 2-5 demonstrates the distribution of proposed methods in each phase of the awareness maintenance framework in the years of publication. This figure illustrates that awareness obtainment has been the most cited phase of the framework since 2000. However, the number of publications may not be an absolute indicator of the fact that awareness obtainment is more problematic; it may just indicate that it has elicited the greatest attention by researchers in this trending area.
2.1.5 Gap Analysis: Awareness Identification

My awareness maintenance framework can be used to identify the limitations of existing approaches. One of the open questions in the framework that needs further research is how to identify awareness or, in Riemer and Haines’s words (2008), how an individual can create a pool of awareness. Although the existing methods in the literature have addressed the problem of awareness creation in the obtainment phase, they have not answered the fundamental question of what should be aware of. This open question has been pointed out by researchers in CSCW (Ray, Shahrestani, et al., 2005; Daneshgar et al., 2006) as well as in software agents (Halpern and Pucella, 2010). Therefore, my framework lacks the methods for a step prior to awareness creation in the obtainment phase, which
can be called *Awareness Identification*. Awareness identification would be a process in which an individual identifies the relevance of information to his or her situation.

Policy-based Awareness Management (PAM), presented in this work, aims at addressing the problem of awareness identification. PAM proposes the use of policies as a source to identify awareness and employs software agents to assist individuals for this purpose. The contribution of PAM is based on the concepts proposed by the logic of general awareness. The present study applies Directory Enabled Networks – next generation (DEN-ng) as its policy structure.

### 2.1.6 Theoretical Support for Awareness Maintenance Framework

The concept of awareness in the awareness maintenance framework has been borrowed from the field of CSCW. In this section, the implications of this framework in the theory of situation awareness (Endsley 1995) and the theory of pools and streams (Riemer and Haines, 2008) are discussed.

The theory of situation awareness (Endsley 1995) states that once an individual captures the context (in my framework, it is called awareness obtainment), there are three levels of awareness: Level 1 - perception (context representation), Level 2 - Comprehension (context analysis) and Level 3 - Projection of future (awareness utilisation). The Theory emphasises changes in behaviours in respect to the above three levels of mental attitudes. However, as it is discussed in
Section 2.1.5, the theory of situation awareness lacks the step for awareness identification. In the theory of situation awareness, once an individual wants to capture a context, there should be a way to identify which context is relevant to capture, otherwise the individual will suffer from information overload.

In the hypothesis of pools and streams (Riemer and Haines, 2008), when one becomes aware of a relevant context, one creates a pool. Consequently, when one adapts one’s behaviour to gain more knowledge about the relevant context, one directs streams to the created pool. The theory of pools and streams also fails to address the process in which the individual identifies awareness. As discussed above, an individual who wants to remove uncertainty may be overloaded by irrelevant or loosely-relevant information if she or he is unable to identify awareness of relevant information. This would indicate a lack of the awareness maintenance framework for awareness identification.

In the next section, I briefly review the background knowledge about awareness in CSCW, software agents and policy structures. We show the consistency of the literature in these areas. I discuss how identification of awareness is a significant question in both CSCW and software agents.
2.2 Background

In this section, in order to create greater familiarity with the alternative solutions to the problem of awareness identification, I discuss the knowledge in the related work.

Section 2.2.1 presents the awareness concepts in CSCW. Section 2.2.2 presents the awareness of software agents. Section 2.2.3 presents the policy structures and the topic of awareness within them.

2.2.1 Awareness in CSCW

According to my discussion above, in Section 2.1.5, awareness obtainment lacks the methods to identify awareness that can be addressed by the present research. The methods related to awareness obtainment in CSCW are presented in Table 2-4. These methods cover three main perspectives: computer sciences, cognitive sciences, and social sciences.

The studies in computer sciences, especially artificial intelligence, have developed extensive research on awareness obtainment. In this field, awareness obtainment is typically regarded as developing software, often by a software agent, which takes some information relevant. From this perspective, awareness can be seen as a container (or in Riemer and Haines’s words (2008), pools) where the content depends on a set of situational parameters or dimensions. The specific set of parameters varies according to the areas of application.
The research in cognitive sciences develops a dynamic of awareness obtainment to understand how human cognitive processes are influenced by obtaining awareness. In this approach, awareness is a set of all entities relevant to the human behaviours in a particular situation.

Sociological approaches typically discuss the concept of awareness obtainment in a network of interacting entities such as people, actors, artefacts and information technology tools. These approaches focus on how the structure of such networks can be relevant to all roles involved in the network. Whereas some consider the network elements, others focus on the emerging properties of the elements.

Table 2-11 compares the different methods proposed in the literature for awareness obtainment. This table illustrates that Cooperative management Methodology for Enterprise Networks (CoMEN) proposed by Ray et al. (2005) and other related studies, such as (Daneshgar and Ray, 2000a), address the three aforementioned aspects influencing the objective of PAM in computer, cognitive and social sciences.

CoMEN brings together contributions from such diverse areas as computer, social and cognitive sciences. In CoMEN, CSCW techniques are used to analyse practical scenarios based on human interaction concepts in the above areas. CoMEN aims to utilise intelligent Information Technology tools to enhance cooperation among participants. In order to enhance cooperation, I require a
metric to measure cooperation. CoMEN uses the concept of awareness from CSCW, where awareness is defined as understanding the relevance of information.

CoMEN proposes the following two phases to enhance cooperation:

- **Phase one – Scenario Analysis**: this phase provides a method for the definition and analysis of scenarios in a cooperative management environment. This phase encompasses activities relating to requirements analysis. It is based on the holistic analysis of information systems as a foundation to incorporate human factors. The output of this phase is an abstract specification of the scenarios taking place in the system. This phase consists of three main stages: (I) Overall System Study, to produce a rich picture of the system to understand the problem situation; (II) Process Study, to indicate the procedures of each scenario; (III) Logical Component Identification, to look inside of the procedures and the relevant information, that is, awareness in each procedure. CoMEN defines the process of cooperation enhancement stating that if information is indicated as relevant to an individual – that is, required awareness – but he or she is not aware of this – that is, available awareness – then there is an opportunity to enhance cooperation by developing an information technology solution that provide this information to the individual.
Phase two – System Design and Implementation: the objective of this phase is to design and implement of such an application that is specified in the previous phase. CoMEN proposes to employ software agents in this phase in order to aid individuals for awareness maintenance.

Table 2-11 Comparison of the different Awareness Obtainment Methods proposed in Literature.

<table>
<thead>
<tr>
<th>Method</th>
<th>Computer Sciences Approach</th>
<th>Cognitive Sciences Approach</th>
<th>Social Sciences Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Daneshgar and Ray, 2000a), (Ray, Shahrestani, et al., 2005)</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>(Riemer and Haines, 2008)</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>(Zacarias et al., 2010)</td>
<td></td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>(Frößler et al., 2007)</td>
<td></td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>(Kaiser et al., 2005)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Lieberman and Selker, 2010)</td>
<td></td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>(Ranganathan and Campbell, 2003)</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Davis et al., 2003)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Gu et al., 2005)</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>(Román et al., 2002)</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>(Yiqiang Chen et al., 2009)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Salem and Rauterberg, 2004)</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>(de Freitas Bulcao Neto and da Graa Campos Pimentel, 2005)</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>(Oh and Woo, 2009)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Borovoy et al., 2010)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Dey et al., 2001)</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>(Dey, 2009)</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>(Eddy and Pei, 2010)</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>(Kummerfeld et al., 2003)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Chen et al., 2010)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Gellersen et al., 2002)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Keiser and Kriengchaiyapruk, 2008)</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>(Biegel and Cahill, 2005)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Lalbakhsh et al., 2009)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Taconet et al., 2009)</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>(Müller et al., 2010)</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>(Wustmann et al., 2010)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Baladrón et al., 2010)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Chtcherbina and Franz, 2003)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Although CoMEN has been published extensively in the field of cooperative management (Ray et al., 1999, 2005; Daneshgar and Ray, 2000a; Ray and Lewis, 2009), it has the following shortcomings:

- Although the cooperation enhancement process that compares required and available awareness has been widely evaluated by empirical studies, there is no definitive method to identify awareness (Ray, Shahrestani, et al., 2005; Daneshgar et al., 2006).

- The cooperation enhancement process is a static method that lacks dynamic identification of relevant information in run-time (Daneshgar and Ray, 2000a).

- Although CoMEN pays a great attention to scenario analysis, it lacks a well-defined methodology for the system design and implementation phase (Ray and Lewis, 2009).

There are bodies of work in the application of information technology in cooperative environments that propose the use of agent technologies to maintain the awareness of individuals (Dommel and Garcia-Luna-Aceves, 2000; van Benthem, 2003; Li et al., 2004; Ray, Shahrestani, et al., 2005; Chan et al., 2008). Therefore, I will next look at the literature of awareness in software agents.
2.2.2 Awareness in Software Agents

The use of software agents has proved useful in many application areas such as emergency management (Yuan and Detlor, 2005), risk management (Ulieru and Worthington, 2005), mobile health monitoring (Ray et al., 2008) and disaster management (Ray and Chattopadhyay, 2009). CSCW tends to agree that the following characteristics of software agents make them useful in assisting individuals (Woolridge and Wooldridge, 2001):

- **Autonomy**, an agent can operate without the intervention of individuals;
- **Social Ability**, an agent is able to interact with other agents and individuals;
- **Reactivity**, computing power of agents makes it possible for them to react on the changes of the environment in a timely fashion (however, this is heavily dependent on the computing power);
- **Pro-Activity**, an agent tries to achieve the purpose for which it was initiated. Therefore, this study has chosen software agents to assist individuals for awareness identification.

Research in software agents has been interested in the natural semantics for awareness as a mental attitude of agents describing internal features that must be interpreted subjectively from the agents’ point of view. For more about mental attitudes of agents, see (Rao and Georgeff, 1991; Boella and van der Torre, 2003). The classical approach is the possible-worlds model in which a state can be considered possible in addition to true or false (Rao and Georgeff, 1991).

Several formal structures have been proposed for possible-worlds model (Fagin, 2003). One is the Kripkean structure (1963) that starts with a set $U$ of primitive
propositions, which is closed under conjunction ($\land$), negation ($\neg$) and the operator of knowledge ($K$). I consider $\varphi \lor \psi$ to be an abbreviation for $\neg(\neg\varphi \land \neg\psi)$, and $\varphi \Rightarrow \psi$ to be an abbreviation for $\neg\varphi \lor \psi$. The Kripkean structure $M$ is a tuple $(W, W', \pi)$, where $W$ is a non-empty set of possible worlds or worlds for short and $W' \subseteq W$ as worlds that the agent considers possible. $\pi$ is an assignment function that associates a truth of primitive propositions in $U$ to each world in $W$. In this study, in order to simplify the scope of the problem, I assume an agent to be a competent agent (Cohen and Levesque, 1990) who accepts $W = W'$ and presents the structure as $(W, \pi)$. As a consequence, it satisfies true knowledge i.e. $K\varphi \Rightarrow \varphi$. It also follows the knowledge generalisation inference rule (Fagin and Halpern, 1988) i.e. $\varphi \Rightarrow K\varphi$. The satisfiability relationships are as follows:

\begin{align*}
(M, w) \models \varphi & \iff \pi(w, \varphi) = \text{true}, \text{ where } \varphi \in U. \quad \text{Equation 2-1} \\
(M, w) \models \neg \varphi & \iff (M, w) \not\models \varphi. \quad \text{Equation 2-2} \\
(M, w) \models \varphi \land \psi & \iff (M, w) \models \varphi \text{ and } (M, w) \models \psi. \quad \text{Equation 2-3} \\
(M, w) \models K\varphi & \iff (M, w) \models \varphi. \quad \text{Equation 2-4}
\end{align*}

The possible-worlds models provide an intuitive semantics for mental attitudes of agents, but they also commit us to logical omniscience and perfect reasoning. Subsequently, I introduce these two problems and discuss the proposed solutions in the literature.
2.2.2.1 Problem of Logical Omniscience and Perfect Reasoning

Although the Kripkean structure (1963) seemed to be a natural interpretation of such mental attitudes as knowledge representation, it states that (I) agents are reasoners that know all the valid formula, called *logical omniscience*, and (II) agents’ knowledge is closed under logical consequences, called *perfect reasoning*. Therefore, if the agent knows $\varphi$ and knows that $\varphi$ implies $\psi$, it knows $\psi$. This is clearly an idealisation, and in real life people do not know all the truth and neither all the consequences of what they know (Sillari, 2008b).

A case of point is the space shuttle Columbia disaster. Five days into the mission, the NASA management team had been informed that the shuttle was experiencing unusual shakes (situation $s_1$ in Figure 1-1). The NASA management team could have announced the shakes as the turn-around effect or they could have requested high-resolution imaging from DoD. NASA declared the shakes to be due to turnaround, when actually there was damage to the TPS. Therefore, the following two problems appeared in the Columbia disaster:

- *Logical Omniscience*: although the TPS on the left wing of the space shuttle had damage, the NASA management did not know it.

- *Perfect Reasoning*: although the policy guideline was available to initiate a spacewalk procedure, the NASA management did not consider it relevant to the situation.
Not surprisingly, many approaches have been introduced to address logical omniscience and perfect reasoning, which will be briefly overviewed.

2.2.2.2 Dealing with Logical Omniscience and Perfect Reasoning

In the literature, there are three different categories of approaches to the problem of logical omniscience and perfect reasoning. Here, I summarise them as being algorithmic knowledge, synthetic, and awareness approaches. For more details, readers should refer to (Sillari, 2008a; Halpern and Pucella, 2010).

Algorithmic Knowledge Approach: There are a number of computational approaches (Ramanujam, 1999; Artemov and Nogina, 2005) that define knowledge algorithms to return “YES”, “NO” or “?”, where “YES” means that the formula is satisfiable, “NO” means it is not and “?” means nothing is prescribed. Therefore, the agent is not omniscient in the sense that it does not know all the truth. More formally, an algorithmic knowledge structure is a tuple \( M = (W, \pi, \alpha) \), where \( \alpha \) is a knowledge algorithm that returns “YES”, “NO” or “?”. As such I have the following satisfiability relation:

\[(M, w) \models K\varphi \iff \alpha(\varphi) = "YES".\]

Equation 2.5

Syntactic Approach: it is possible to attack the problem of logical omniscience by distinguishing two senses of “knowing”. A weak kind of knowledge, called implicit knowledge, is simply concerned with the truth of a formula. In natural languages, this sense of “knowing \( \varphi \)” \((K\varphi)\) is related to “being informed
about $\varphi$” or “being under the impression of $\varphi$”. The second is a stronger kind of knowledge, called explicit knowledge, which is not only concerned about the truth of a formula, but it is also concerned with the justification of the agent’s knowledge. In this sense, “knowing $\varphi$” ($X\varphi$) may mean that the agent has “all the evidences to assert $\varphi$”, or has “the right to be sure about that $\varphi$”. Therefore, in this approach implicit and explicit types of knowledge are distinguished by the difference of “availability” and “justification”. In this approach, logical omniscience and perfect reasoning are avoided by defining knowledge as an explicit set of sentences that does not necessarily cover all the valid formulae. More formally, there is a function $C$ that associates a set of formulas $C(w)$ to a world $w \in W$. In this approach, I have:

$$(M, w) \models K\varphi \text{ iff } \varphi \in C(w).$$

Equation 2.6

Some examples of this approach are (Hintikka, 1962; Levesque, 1984; Sim, 1997; Rebuschi, 2009).

**Awareness Approach**: in regards to awareness, the literature presents the logic of general awareness (Fagin and Halpern, 1988; Sillari, 2008b). In this logic, the essential idea is relevance of information. Under the Kripkean structure, a valid sentence and its consequences are true in every world that the agent considers possible. However, a known sentence and its known consequences may or may not be relevant. Therefore, in the logic of general awareness, an agent implicitly
knows all the valid sentences ($K\varphi$), but it changes its implicit knowledge to the explicit knowledge ($X\varphi$) if, and only if, the agent is aware of the sentence ($A\varphi$) (see Figure 2-6). Sillari (2008) defines awareness of a propositional sentence as the relevance of that sentence to a situation. Therefore, the notation of awareness does not refer to the validity of a sentence. Regardless of whether a sentence is valid or not, an agent becomes aware of a sentence if, and only if, it identifies the relevance of the sentence to the situation.

$$\models (M,w) \text{ iff } (M,w) \models \varphi \text{ and } \varphi \in A(w).$$

Equation 2-7

Giacomo Sillari (2008) and Halpern & Pucella (2010) analyse the expressive power of the above approaches and claim that all such solutions to logical omniscience and perfect reasoning share a common structure. Therefore, these are equi-expressive in terms of capturing the property of knowledge. In practice, there may be a natural interpretation for each of these approaches, which makes the pragmatic approach as the selective criteria. Looking at the guidelines given
by Halpern and Pucella (2010), I found that the awareness approach corresponds most closely to the problem of identifying relevant information.

Halpern and Pucella (2010) argue that an awareness structure must be understood as a setting heavily dependent on the agent’s view of the situation. Different agents may choose different propositions of which to be aware in a particular situation. Therefore, while obtaining awareness, constructing an awareness set corresponding to a particular situation is critical and I must explain how awareness can be identified. In other words, how agents identify the relevance of propositions to a given situation is a significant question.

2.2.3 Awareness and Policy Structures

The concept of policy has been used in a wide range from high level organisational documents to low level business rules. Wies (2000) defines four different views for policies in cooperative work: enterprise, application, system and infrastructure. The meaning of policy in this research lies in the application view among the aforementioned four views of policies, which helps humans to fulfill their responsibilities individually or in collaboration with other roles. John Strassner (2004) divides the application view into four levels: Business, System, Role and Administration level policies. This work deals with role-level policies as technology-independent mechanisms to control roles’ behaviors in applications of distributed cooperative work.
There are three main role-level policy structures that are widely being used today (Strassner et al., 2009):

- **The TeleManagement Forum (TMF) Shared Information and Data (SID) model**: the Information Framework (SID) provides an information reference model and common vocabulary to develop application policies from a business role perspective. The SID addresses the service providers’ need for a shared understanding of information and data concepts, definitions and models in applications.

- **Distributed Management Task Force (DMTF) Common Information Model (CIM)**: CIM is an approach to the management of applications that apply the basic policies based on the object-oriented paradigm. The approach uses a uniform modelling formalism that supports the cooperative development of object-oriented application schemas across multiple organisations.

- **Directory Enabled Networks next generation (DEN-ng)** (Strassner et al., 2009): in this structure a policy is a set of rules that are used to manage and control applications. In DEN-ng the main idea is that such a policy is a composition of different policy rules, while each rule defines the “event-condition-action” semantics. The given set of policy rules should be loaded


based on the current knowledge and the event that has occurred. Theses semantics are such that the rule is evaluated when an event occurs. When the condition clause is satisfied, the modality of action will be applied which may or may not result in executing the action.

Sloman (1994) defines modality of actions by introducing four types of policy rules: permitting, forbidding, requiring and deterring. This is also confirmed by Twidle et al. (2009). Table 2-12 shows examples for each type of policy rules.

Table 2-12 Examples of policy modality types

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permitting</td>
<td>When an employee leaves a company, if he or she has been working in the customer service unit, it is permitted to remove his or her file from the archive.</td>
</tr>
<tr>
<td>Forbidding</td>
<td>When an employee leaves a company, if he or she has ever worked in the accounting unit, it is forbidden to remove his or her file from the archive.</td>
</tr>
<tr>
<td>Requiring</td>
<td>When an employee leaves a company, if he or she has ever worked in the accounting unit, it is required that his or her file be kept in the archive.</td>
</tr>
<tr>
<td>Deterring</td>
<td>When an employee leaves a company, if he or she has been working in customer service unit, it is deterred to keep his or her file in the archive.</td>
</tr>
</tbody>
</table>

Twidle et al. (2009) and Moffett & Sloman (1991) state that only forbidding and requiring rules, among these four types of policy rules, are in force. This means that when a policy rule prefers me not to do something (that is deterred), I can still do it. In contrast, when a policy rule permits me to do something, I can still avoid doing it. However, requiring and forbidding policy rules come with absolute force.
While TMF SID and DMTF CIM lack awareness view, DEN-ng presents its policy model to support awareness in distributed systems (Strassner, Yan Liu, et al., 2008; Strassner, Liu, and Zhang, 2008; Strassner et al., 2009). Therefore, I borrowed my policy structure from DEN-ng.

DEN-ng proposes that policies can be used to implement the awareness in the design phase of developing cooperative applications. In fact, the assumption in DEN-ng is a given set of awareness. Although, DEN-ng provides a meta-model to implement this awareness with policy rules, the use of the given set of policies as guidelines to which information should be aware has not been addressed.

Policy-based Awareness Management (PAM) uses the existing set of policy rules as a source to identify awareness. This is the main proposal of PAM.

2.3 Observation and Research Gap

Awareness is defined in five categories: Workspace, Common-sense, Group, Social and Context Awareness. Omoronyia et al. (2010) state that context awareness indicating the relevance of information covers the other types of awareness. My definition of awareness is most related to context awareness. In this definition, context refers to the information of which an individual is aware. Awareness of information shows the relevance of the context, but it does not mean that the context is valid.
The Riemer and Haines’s hypothesis of awareness maintenance (2008) and the theory of situation awareness look at how an individual becomes aware of information and how this awareness can change or, in Riemer and Haines’s words, can shape the individual’s behaviours. Although research in this area encourages using information technology tools, models and methodologies to assist individuals in order to maintain their awareness, how to utilise information technology to identify awareness remains as a topic of research in this field.

The systematic literature review presented in this chapter resulted in the following observations:

- The literature in the application of information technology for awareness maintenance favours a framework with four phases: awareness obtainment, context representation, and context analysis and awareness utilisation.

- Awareness obtainment addresses the problem of how one becomes aware of relevant information. Research proposed in awareness obtainment elaborates on (I) awareness creation to match actual information to what is relevant to the situation and (II) awareness security management, which consists of confidentiality, trust and identity.

- Context representation refers to a process which models and distributes the context that the individual is aware of. Context modeling techniques vary
from being tuple and tree-like to mathematical and ontological. The
distribution of context can be centralised or distributed

- Context analysis includes two steps for (I) manipulating contexts to give
  them meaning and (II) labelling them with meta-data. Manipulation
  consists of abstraction, filtering and reasoning, while labelling is tagging the
  context to illustrate the quality attributes of the context.

- Awareness utilisation refers to what Riemer and Haines call as shaping of
  behaviours through awareness of information. Awareness utilisation can be
  done in two categories of methods: rule-based methods and machine
  learning methods.

- Although awareness obtainment has been received great attention by the
  research in this area, it lacks definitive methods to identify awareness. This
  would be a process in which one identifies the relevance of information to
  its situation before one creates awareness to such information.

Next, I show that research in CSCW and software agents encourage researchers
to fill this gap. The present research proposes Policy-based Awareness
Management (PAM) to employ software agents to assist individuals in identifying
awareness of the relevant information based on existing policy rules. The
following observations are made by studying the body of knowledge for
There are bodies of research in CSCW that propose methodologies to address computer, cognitive and social perspectives in awareness obtainment. According to my comparison presented in Table 2-11, Cooperative management Methodology for Enterprise Networks (CoMEN) addresses all these three aspects of awareness obtainment.

CoMEN emphasises the role of awareness obtainment in cooperation enhancement. In order to enhance cooperation, CoMEN proposes the following two phases: (I) Scenario analysis and (II) The Design and Implementation phase.

The following drawbacks have been indicated in CoMEN, which are going to be addressed by PAM. First, there is no method to identify awareness. This is also pointed out in (Ray, Shahrestani, et al., 2005; Daneshgar et al., 2006). Second, CoMEN is not able to enhance cooperation in run-time. Third, although CoMEN gives the general objectives of the phase for design and implementation and it proposes to use software agents, it lacks detailed explanation of the steps.
• Research in the area of awareness of software agents proposes the logic of
general awareness as an alternative response to logical omniscience and
perfect reasoning. The intuition behind awareness definition in this logic is
the relevance of propositions to a given situation. However, the relevance
does not mean that the proposition is valid. Although the logic of general
awareness appeared to be a natural interpretation of awareness, it does not
cover the process in which an agent becomes aware of a proposition. This
is emphasised as a significant open question in the field (Halpern and
Pucella, 2010).

• PAM, as a proposed alternative for awareness identification, uses existing
policy rules. Therefore, I also studied the policy structures in cooperative
distributed systems. Directory Enabled Networks – next generation (DEN-
ng) define policies as a set of policy rules in an event-condition-action
structure. DEN-ng assumes a given set of relevant information as
awareness and implements them into system by policy rules. However, the
use of existing DEN-ng policy rules to identify awareness is an open
question, which is addressed by PAM.

Looking at this overview of awareness in CSCW, software agents and policy
structures, I argue that the literature in these fields are consistent in terms of their
definitions and the questions they pose in related to awareness identification.
Table 2-13 presents the comparison of awareness in these fields and demonstrates the consistency.

Table 2-13 argues that the awareness obtainment process in CSCW, software agents and policy structures lacks a method to identify awareness, which is characterised by the following features:

<table>
<thead>
<tr>
<th>Area of Research</th>
<th>Who</th>
<th>What</th>
<th>Definition</th>
<th>Open Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>Awareness in CSCW (CoMEN)</td>
<td>Individuals</td>
<td>Context</td>
<td>Relevance</td>
<td>Awareness identification process</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Run-time cooperation enhancement</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Definitive stages in the design and implementation phase</td>
</tr>
<tr>
<td>Awareness in Agents (Logic of General Awareness)</td>
<td>Agents assisting individuals</td>
<td>Propositions representing context</td>
<td>Relevance</td>
<td>Awareness identification process</td>
</tr>
<tr>
<td>Awareness in Policy (DEN-ng)</td>
<td>Agents assisting individuals</td>
<td>Context</td>
<td>Relevance</td>
<td>Use of policies in awareness identification process</td>
</tr>
</tbody>
</table>

- The proposal should identify the awareness of a set of information that is relevant to a given situation.

- The proposal should employ software agents to assist individuals identifying their awareness in run-time.

- The proposal should specify the details of the design and implementation phase for developing an agent system equipped with such a method.
Policy-based Awareness Management (PAM), built on the logic of general awareness, employs software agents and proposes a step-by-step process of using policies as a source for identifying awareness.

2.4 Summary and Outlook

The objective of this chapter was to present the body of knowledge in awareness maintenance. The chapter also aimed at providing background information for the present study.

In the first part of the chapter, a systematic review of methods proposed in the literature of awareness maintenance was presented. The systematic review of articles published during 1970–2010 resulted in four phases in awareness maintenance including awareness obtainment, context representation, context analysis and awareness utilisation. Analysis of the trends indicated that awareness obtainment has drawn the most research attention in this field. The review of papers illustrated the drawback of the awareness obtainment phase for a definitive method to identify awareness. This is addressed by Policy-based Awareness Management (PAM) proposed in this research.

In the second part of the chapter, the background information required to present PAM was given. This covered awareness in CSCW, software agents and policy structures. In CSCW, CoMEN was discussed as a cooperative management methodology that uses awareness to enhance cooperation. The
drawbacks for CoMEN were noted in the lack of a run-time method, the lack of definitive methodology for design and implementation phase and the lack of a method for awareness identification. In regard to awareness of software agents, the Kripkean structure for mental attitudes of agents was presented in response to logical omniscience and perfect reasoning. Three alternative approaches, considered in this account, were algorithmic knowledge, syntactic and awareness structures. The equi-expressiveness of these methods was discussed, which makes the pragmatic justification more significant than the theoretical reasons. Therefore, based on the nature of the awareness identification problem, the awareness approach to overcome logical omniscience and perfect reasoning was chosen. On the awareness approach, the logic of general awareness was presented as a logical foundation on which to build PAM. DEN-ng as a policy structure used in PAM, was also discussed.

The third part of the chapter showed that the literatures of awareness in the areas of CSCW, software agents and policy structures are consistent in terms of definitions as well as the omission of awareness identification. The requirements of any solution for the problem of awareness identification are given as (I) defining awareness as relevance of information, (II) employing software agents to assist individuals for awareness identification and (III) specifying the details of design and implementation of such systems.
Having delineated the lack of the methods on awareness identification, in the next chapter I discuss the research methodology taken in the present study to address this gap. I also evaluate the reliability of the methodology.
Chapter 3

RESEARCH METHODOLOGY

Well-planned research is characterised by a clear methodology. A research methodology is a guide that shows how to conduct research and how to come up with findings, in order to meet the specified objectives (Carlsson, 2010). This chapter aims at clarifying the objectives for this research, demonstrating the main contributions and showing the way that I am going to obtain these contributions. The chapter adapts a design-science approach from Hevner et al. (2004). It also evaluates the presented research methodology against the guidelines given by Hevner et al. (2004).

The rest of this chapter is organised in the following way: Section 3.1 presents the research objectives. Section 3.2 adopts Hevner's framework (2004) and presents the research methodology employed in this study. Section 3.3 concludes the chapter and briefly discusses Chapter 4.

3.1 Research Objectives

Chapter 2 has discussed the consistency of the research on awareness in CSCW, software agents and policy structures. The required features for employing software agents to assist individuals in maintaining awareness were specified. The chapter argued that the literature has fallen short in addressing the following questions:
Research Methodology

- How can software agents assist individuals in order to identify awareness of the relevance of information in run-time?

- How can I develop such an agent system equipped with the above method for awareness identification?

I next present the research framework that I followed in order to obtain the above objectives.

3.2 Research Framework

In this section, I explain the way that I carried out the results of the above objectives. Section 3.2.1 discusses the choice of research paradigm. Section 3.2.2 adopts Hevner's framework (2004) for the objectives given in the previous section. Section 3.2.3 evaluates the reliability of the research methodology against the guidelines given by Hevner et al. (2004).

3.2.1 Choice of Research Paradigm: Design Science

Hevner et al. (2004) characterise most of the research in information systems as being within two paradigms: behavioral science and design-science. Behavioral science research develops and verifies theories to explain human and organisational behaviors. These theories inform researchers and practitioners of the interactions among people, technologies and organisations that are important to achieve the goal of information systems. Hevner et al. (2004) defines the
design-science research as a research paradigm to “create innovations that define the ideas, practices, technical capabilities, and products through which the analysis, design, implementation, management, and use of information systems can be effectively and efficiently accomplished”. In comparison, behavioral science seeks to predict or explain phenomena and truths that occur with respect to the use of artefacts that have been created by design-science while utilising truths. (see Figure 3-1).

![Figure 3-1 Information Systems Research Paradigms - Adopted from (Hevner et al., 2004)](image)

This research aims at utilising the body of knowledge in the creation of a framework and a method to identify awareness using policy rules. My goal is not to predict or explain the truths, instead it is to develop and create a framework, a process and a development methodology for identifying awareness using policy rules. Therefore, in this study, the design-science paradigm was adopted.
3.2.2 Research Methodology

The research methodology in this study has been adapted from the framework proposed by Hevner et al. (2004) for design-science research practices. Figure 3-2 depicts the details of the steps taken to achieve the above objectives given in Section 3.1. Conducting research in this study, like any other design-science exercise, has four steps: awareness of problems, suggestions, contributions and evaluation.

The present work begins with the study of exemplars in their environments; that is real-world applications. The space shuttle Columbia disaster is given in Chapter 1 as illustration of this motivation and purpose. There are two more exemplars explained in Chapter 6: the Casa Grande hazardous materials rail incident, AZ, in 1983 and rehabilitation video games. These exemplars demonstrate two objectives for the present research: run-time awareness identification and development guidelines of the proposal for awareness identification. For example, in the space shuttle Columbia disaster, if the NASA management had benefited from a system that could recognise the relevance of TPS damage, they would have asked the Department of Defense for high-resolution images. To reach this conclusion, Chapter 2 illustrates that the literature encourage researchers to contribute to the problem of awareness identification.
Based on the above objectives and according to literature, Policy-based Awareness Management (PAM) suggests employing software agents to assist
individuals for using policy rules as a source to identify awareness. The main contributions of PAM are listed below:

- **PAM Framework**: this is a logical framework built on top of the logic of general awareness that can represent policy-based awareness of agents. In this framework, the policy rules should be represented based on DEN-ng policy structure. In brief, the framework provides the definitions required for PAM.

- **Three-step Process of PAM**: this is a step-by-step process to identify awareness in run-time from the existing policy rules. This process is built on the foundation provided in the proposed PAM framework. The process follows three steps:
  
  - Recognise the relevance of policy rules
  - Recognise the relevance of conditions given in the relevant policy rules
  - Change behaviours

- **Development Aspects in PAM**: in order to develop an agent system equipped by PAM, the detailed stages for analysis, design and implementation of PAM-supported systems are given.
The next step is to evaluate the contributions of the research. In this study evaluation is conducted by proofs of concepts as well as simulations. By applying PAM to the exemplars given in the environment of the research framework, I prove the applicability of the concept for the use of policies in awareness identification. Then, in order to make sure that PAM is applicable in more complex situations, I use simulation techniques. The simulations allow me to compare results of PAM against an awareness-free method; that is, the basic Kripkean structure (1963) introduced in Section 2.2.2. This is beneficial to understand in which situations the use of PAM is effective and cost-efficient. The definitions for efficacy and cost-efficiency are given in Chapter 6. I conduct two simulation studies: (I) simulations by populating the study with hypothetical examples and (II) simulations by populating the study with wireless communication procedures at St. Olavs Hospital in Trondheim, Norway. Comparing these two simulations is helpful to see the consistency of results of using PAM in hypothetical complex situations and in real situations. As a final step, I add PAM to the body of knowledge by publications listed in the section for publications at the beginning of this document. This also adds external evaluation for the validity of PAM.

3.2.3 Reliability of Research Methodology

The validity of final products in design-science research is heavily dependent on the reliability of the methodology that I use to conduct the study (Carlsson,
Therefore, in this section, I evaluate my research methodology against the guidelines given by Hevner et al. (2004).

3.2.3.1 Guideline 1: Design as a Product

Design-science research produces constructs, models, methods and instantiations (Hevner et al., 2004). Table 3-1 presents how the present study responds to the required products that a design science research has to generate.

Table 3-1 Responses to the Required Products

<table>
<thead>
<tr>
<th>Product</th>
<th>How is being addressed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construct</td>
<td>The intuitions behind the entities involved in the PAM framework are the constructs. These constructs are explained in Section 4.1.1 as informal semantics.</td>
</tr>
<tr>
<td>Model</td>
<td>A model is defined by the definitions of the entities and the correspondence between the entities that can be used as a basis to propose the method (Pfeiffer and Niehaves, 2005). The present study proposes the PAM framework as a foundation in which the process for awareness identification is grounded. The model is explained in Section 4.1.2.</td>
</tr>
<tr>
<td>Method</td>
<td>PAM proposes a step-wise process for awareness identification. This is explained in Section 4.2. Based on the model proposed, The present study also gives development guidelines for analysis, design and implementation of agent systems equipped by PAM. This is explained in Chapter 5.</td>
</tr>
<tr>
<td>Instantiation</td>
<td>Based on PAM, I develop (I) the generated hypothetical inputs for PAM and (II) wireless communication procedures at St. Olavs Hospital. In fact, simulations explained in Section 6.2 are instances of the contributions to this study.</td>
</tr>
</tbody>
</table>

3.2.3.2 Guideline 2: Problem Relevance

A design-science research must aim at solving a relevant problem that contributes in real life (Hevner et al., 2004). The relevance of the contributions in the present study is proved by three exemplars. The given exemplars in Section 6.1 illustrate the motivation of this research. These exemplars also prove the applicability of the concepts by running the proposed contributions into these given real life examples. However, PAM has not been yet involved in real production
environments, the applicability of the concepts has been proved by exemplars, which is strongly recommended in software engineering research and practices (Cysneiros et al., 2004). I have also simulated PAM by real inputs from the wireless communication system at St. Olavs Hospital.

3.2.3.3 Guideline 3: Design Evaluation

The quality of the proposals in a design-science research must be rigorously demonstrated via well-executed evaluation methods (Hevner et al., 2004). We present the evaluation of PAM in Chapter 6, where we design two simulation studies: simulations on hypothetical inputs and simulations on inputs from the wireless communication system of St. Olavs Hospital. We opt to change the number of situations, the branching factor and the number of policy rules to evaluate PAM in different conditions. Then, we choose to measure the success rate and cost of PAM. From these two outputs, we calculate efficacy and cost-efficiency. I repeat this for each of these two simulation studies. For more details, readers should refer to Section 6.2.1.

3.2.3.4 Guideline 4: Research Contributions

A design-science research project must be clear in terms of the outputs and the contributions (Hevner et al., 2004). The present study offers a clear definition of its contributions. The contributions of the research are given in Section 3.2.2 and are demonstrated via a running example on the space shuttle Columbia disaster. The contributions briefly address the PAM framework, the PAM process of
awareness identification and development aspects of agent systems that support PAM.

3.2.3.5 Guideline 5: Research Rigor

Rigor of research addresses the way in which research is conducted. Design-science studies often rely on mathematical formalism to describe the specified and constructed outputs (Hevner et al., 2004). PAM is heavily based on formalism built on the logic of general awareness. In addition, the research outputs must be exercised in the appreciate environment (Hevner et al., 2004). Therefore, PAM proves the claims in three real life exemplars and it also practices the contributions in the wireless communication procedures of St. Olavs Hospital.

3.2.3.6 Guideline 6: Design as a Search Process

Hevner et al. (2004) emphasise the iterative nature of design-science research. The present study develops the constructs, models, methods and instantiations through incremental and iterative processes. In each of these iterations, the constructs were built, and evaluated based on the proofs of concepts. Then, in the next iteration the constructs were modified to improve and also some other required constructs were added. The use of exemplars to ensure the relevance of the proposed solutions to real-life solutions, in iterations, is extremely helpful.
3.2.3.7 Communication of Research

I have published in technical-oriented conferences and journals as well as in application-oriented publications. The complete list of publications from this study is given in the section for publications at the beginning of this document.

Design-science research must be presented both to technology-oriented and application-oriented audiences (Hevner et al., 2004). The technology-oriented audiences require sufficient level of details. From this concern, I articulate the formalism of PAM (Talaei-Khoei, Solvoll, Ray, et al., 2011a) as well as development guidelines (Talaei-Khoei, Solvoll, Ray, et al., 2011b). Application-oriented audiences expect to take advantage of the benefits offered by the research. In this regard, I offer exemplars in the space shuttle Columbia disaster (Talaei-Khoei, Solvoll, Ray, et al., 2011a; Talaei-Khoei et al., 2011; Talaei-Khoei, Ray, et al., 2010), the Casa Grande rail incident (Talaei-Khoei, Bleistein, et al., 2010), rehabilitation video games (Smith et al., 2009, 2011; Talaei-Khoei, Ray, and Parameswaran, 2011). I have also provided some simulation studies on the wireless communication procedures at St. Olav’s Hospital (Talaei-Khoei et al., 2011).

3.3 Summary and Outlook

This chapter proposed to take a design-science approach in order to address objectives of this research on (I) run-time awareness identification and (II) development aspects to support analysis, design and implementation of the
proposal for awareness identification. In order to do so, this chapter adapted the Hevner's framework (2004). The chapter also explained the steps that are necessary to generate the outputs of this study. The first step is awareness of problems with the objectives of the research. The use of exemplars for illustration of problems is a very helpful and direct way of understanding the problems and research objectives. The second step is the formulation of the initial ideas that will address the objectives. PAM suggests using software agents to assist individuals for run-time identification of awareness. PAM also proposes the use of existing policy rules for awareness identification. The third step is to actually produce the contributions of the research. In this step, the main contributions of the work must be generated. The contributions are the PAM framework, a step-by-step process for awareness identification and development aspects for analysis, design and implementation of such agents systems that apply PAM. The last step is to evaluate the products. The evaluation is conducted through (I) proofs of concepts in the space shuttle Columbia disaster, Texas, 2003, Casa Grande hazardous-materials rail incident, AZ, 1983 and rehabilitation video games. It is also conduct by (II) the simulation of hypothetical examples and simulation of wireless communication procedures at St. Olavs Hospital in Trondheim, Norway.

The next chapter develops the formalism of PAM. The chapter proposes a logical framework extended from the logic of general awareness, which defines the
constructs and their correspondences in PAM. This is a foundation for the PAM process of awareness identification. The process is defined in three steps: recognise the relevance of policy rules, recognise the relevance of conditions in the relevant policy rules and change behavior based on the recognised awareness. The contributions of the chapter are demonstrated through application to the space shuttle Columbia disaster.
Chapter 4

POLICY-BASED AWARENESS MANAGEMENT (PAM)\(^1\)

In this chapter, I propose a framework and a step-by-step process called policy-based awareness management (PAM), which uses existing policy rules as a source to identify the awareness of agents. The framework sets a foundation that consists of the different definitions in which the process of PAM is grounded. In this chapter, I present the intuitions as well as the formalism for the PAM framework, before introducing the process of PAM itself. The objective of this process is to propose a policy-based mechanism that can be used with intelligent software agents to identify awareness of relevant information.

First, I informally indicate the characteristics that are required for an agent system to identify awareness. As an example of such a system, I consider the context of the space shuttle Columbia disaster. The relevant characteristics are given below:

I. At any instance of time, there can be potentially more than one situation that a system can evolve from. In the space shuttle Columbia disaster, the situation of the system could change in various ways. For example, at time \( t_2 \), situation \( s_2 \) or \( s_3 \) (see Figure 1-1) could evolve.

\(^1\) An abridged version of this chapter can be found in *Maintaining Awareness Using Policies: Enabling Agents to Identify Relevance of Information*, Journal of Computer and System Sciences (Ranked A* by The Excellence in Research for Australia), Volume 78, Issue 1, Pages 370-391, 2011.
II. In any situation, there can be more than one action that an agent can execute. In the space shuttle Columbia disaster, the NASA management, in situation $s_1$, could simply announce the shake as a turnaround effect, or could request high-resolution images from Department of Defense (DoD).

III. In any situation, executing an action changes the situation to one, and only one, situation. As a case of point, in the space shuttle Columbia disaster, announcing the shake as a turnaround effect changes $s_1$ to $s_2$.

IV. In any situation, the events representing the change of environment may (or may not, depending on the cognition posed by the receiver) change the current situation. In the space shuttle Columbia disaster, when the shake was reported, the NASA management became cognisant of the shake in the shuttle and changed the situation to $s_1$.

V. In any situation, the relevance of information can be an indicator in the selection of an action. For example, in the space shuttle Columbia disaster, if the NASA management, in situation $s_1$, had identified the relevance of TPS damage, it would have requested imaging from DoD to find out whether the TPS was damaged.

VI. In any situation, policy rules can be an indicator for the identification of the relevance of information. In the example of the space shuttle Columbia
disaster, there was a guideline stating that when an aircraft experiences unusual shakes, if there is any TPS damage, the spacewalk procedure must be granted (NASA, 2003). Therefore, the NASA management, in $s_1$, should have identified the relevance of TPS damage to the situation.

Considering the above characteristics, in this chapter, I develop a theoretical framework for PAM. I also describe a step-by-step process for awareness identification.

The rest of this chapter is organised in the following way: Section 4.1 presents the PAM framework. Section 4.2 presents the process of identifying awareness through the PAM framework. Section 4.3 discusses the importance of PAM and concludes the contributions of this chapter.

4.1 PAM Framework

In this section, I present a set of definitions for policy-based awareness in agent systems. The objective of this section is to form a foundation that can be used to introduce the process of awareness identification in the next section. I first present the intuitions and then formalise the concepts. The PAM framework is supported by literature in the field of software agents (Fagin and Halpern, 1988; Allen and Fergusson, 1994; Sloman, 1994; Rao and Georgeff, 1995, 1998; Sillari, 2008a; Strassner, Liu, and Zhang, 2008; Sillari, 2008b; Strassner et al., 2009).
4.1.1 Informal Semantics: Intuitions

I consider a system involved with different agents. These agents are being run in the same system, although they have their own models of the world. One way of modeling the world’s given characteristics (I) and (II) is as a branching tree structure with a single past and multiple futures (Emerson, 1990; Surhane et al., 2010). In the tree, each branch represents an alternative execution path. Each node in the structure represents a certain situation of the world. This structure is called the branching-time model of the world. An example of the branching-time model is given in Figure 1-1, where the world describes the space shuttle Columbia disaster. A specific time-point in a specific world (a specific branching-time model) is called a situation. In Figure 1-1, situations are shown by circles. Situations present the different circumstances in the world. Once, an agent perceives a change in the circumstances, it shifts from entertaining one situation to a new situation.

Actions transform one situation into another. Given characteristic (III), actions here are primitive. This means that they are performable directly by the agents. A primitive action uniquely determines the next situation in the branching-time model. Therefore, an action is defined as a connection between two different situations. Having been transformed from one situation to another, the agent takes the action as a done action in the destination situation. The branches in the world – that is, actions emanating from a situation – can be viewed as the choices
available to the agent at each situation. For example, if there are two possible actions for an agent, then there are two different situations for this agent to consider. In fact, an action equals to another action if and only if it connects the same situations.

*Events* are also a way by which I classify changes in the system as it is required in characteristic (IV). However, these changes do not necessarily change the agent’s situation in its world. In fact, an agent might change its situation in response to an event, while another one might be apathetic about the same event. Therefore, the event itself is cognitively determined. That is, the agents involved in the system treat the events in different ways depending on their mental attitudes in the situation in which they receive the event.

The difference between the ability of events and situations to express a change becomes clear when I say an event occurs, or is received, but I say a situation is held. That is, in events the agent seeks a way of reacting to the circumstances, while situations are more descriptive about the different true or false propositions as it will be described, shortly, in the next paragraph. In a world, there are a set of events that they have been received, and an agent has also a queue of received events. I also differentiate the actions being executed and the events taking place in the system. Actions represent available choices for the agent, and events represent uncertainty in the environment.
The system consists of a set of variables and a defined set of domain values that can be assigned to the variables. Propositional sentences are also defined in a way that can be true or false in each situation. Each proposition consists of a set of variables, while in each situation there is a defined assignment of values to the variables. Intuitively, valued variables represent data, and propositions represent information that can be either valid or invalid. The interpreter calculates the truth or falsity of a proposition in a situation considering the values that are assigned to the variables of the proposition. In order to validate a proposition, each proposition has a set of true-sets. A true-set is a set of tuples that represent valued variables that satisfy the validity of the proposition. This set is independent from the situation. If an interpreter finds that the actual assignment of variables to the domain values belongs to the true-set of the proposition in the situation, it validates the proposition as a true one. Note that in this work I do not differentiate between propositions, propositional sentences and formulae.

I use formalism similar to Computation Tree Logic (CTL) (Surhone et al., 2010). I evaluate a proposition in two different ways: (I) in a specific situation in the branching-time model; (II) in a path, in the branching-time model. I use the operators of CTL. The operator inevitable is to be true of a proposition at a particular situation if and only if the proposition is true of all paths starting from that situation. The operator optional is true for a proposition at a particular situation in a branching-time model if and only if it is true of at least one path
starting from that situation. I also apply the standard temporal operators $\circ$ (next), $\Diamond$ (eventually) and until. They are defined respectively as the proposition is true in the next situation, finally in a situation, or it is true until another proposition becomes true. The modal and the standard operators can be combined in various ways to describe the available choices to the agent.

An agent implicitly knows a proposition if and only if the truth or falsity of the proposition is available for the agent. In other words, as I discussed above, in a situation, the interpreter can validate a proposition based on the values assigned to the variables referenced in the proposition. The agent implicitly knows all the valid propositions and, accordingly, does not implicitly know all the invalid propositions. An agent is aware of a proposition if and only if it considers that as relevant. An agent explicitly knows a proposition if and only if it is aware of the proposition and has implicit knowledge to that (Fagin and Halpern, 1988; Sillari, 2008a, 2008b).

Since the system has to act, the agent needs to select the appropriate action to execute from the various choices available. The design of such a selection function should enable the agents to effectively achieve the value of what they have already identified as relevant information – the hypothesis of Riemer and Haines (2008). Given characteristic (V), the agent selects a path that brings it to a situation with implicit knowledge about the truth or falsity of a proposition that the agent is aware. The question of how I calculate awareness (Halpern and
Policy-based Awareness Management (PAM)

Pucella, 2010) still remains. This is addressed in the PAM process by using policy rules as specified in characteristic (VI).

PAM has borrowed its *policy* structure from Directory Enabled Networks - next generation (DEN-ng) (Strassner et al., 2009), which provides a meta-model for the implementation of awareness with policies (Strassner et al., 2008). However, DEN-ng does not address the use of policies to identify awareness, such a use of policies is proposed by PAM. DEN-ng specifies that a policy is a composition of different policy rules, where each rule is defined as an inherence of a propositional sentence with “event-condition-action” semantics. The semantics are such that a rule is triggered when an event occurs. When the condition clause is satisfied, then the modality of the action will be applied, which may or may not result in the execution of the action. Sloman (1994) defines modalities of actions applied in policy rules as including permitting, forbidding, requiring and deterring. Therefore, I characterise policy rules as being one of these four types. Defining policy rules as a proposition provides the opportunity to define the applicability of each policy rule in a situation. A policy rule is applicable in a situation if and only if the rule in that particular situation is a true proposition. Therefore, if a policy rule is applicable in a situation, the agent implicitly knows that rule. This idea of defining the applicability of policy rules by giving them true values has been borrowed from Fallis (2004).
4.1.1.1 Decision Trees to Branching-time Model of Worlds

As I discussed above, in the branching-time model of worlds, there may be more than one choice available to execute. Here, we begin with the classical decision tree and show how I can view this concept in a way that is closer to traditional epistemic models of awareness and agency.

Informally, a decision tree consists of decisions and chances. Decisions represent points where the agent has to choose one alternative from a number of choices available. Chances represent points where probability plays a dominant role and reflects alternatives over which the agent has no control. Decision trees also include probability functions that map chances to real-value probabilities and a pay-off function that maps decisions to a real number. I transform decision trees to an equivalent model (Rao and Georgeff, 1995) that represent decisions with optional actions and chances with events. This transformation provides an alternative basis for cases in which I am not able to form pay-off or probability functions.

I begin by considering a decision tree, in which every possible path, including those with zero pay-off, is represented. I start from the root node of the tree and traverse each arc. For each unique arc emanating from a chance node, I create a new decision tree that is identical to a tuple made of (I) the chance node, called event and (II) the decision node located before the chance node. The difference between the result tree and the original one is that (I) the chance node is removed
and (II) the arc coming to the chance node is connected to the arc emanating from the chance node. This process is carried out recursively until there is no chance node left, which results in a set of trees with no chance node, while each corresponds to a different possible state of the environment; that is, from the traditional possible worlds perspective. Indeed, each of these trees represents a possible world.

The resulting possible worlds model, called the branching-time model of worlds, contains two types of information: probability across the worlds and the pay-off assigned to the paths. The probability is represented by the different value of variables in the situations in each world, constituting the agent’s implicit knowledge of the propositional sentences in each situation. In this study, the algorithm on which probability assigns values to variables is assumed to be a given input. The pay-off is represented by the relevance of information to each situation, called awareness. How I identify awareness is addressed in the PAM process.

4.1.1.2 Assumed Inputs in PAM

Under the above-mentioned characteristics, there are at least two types of inputs required for PAM: First, given characteristic (IV), it is essential that the system has information about the environment, which is represented by events. Second, given characteristics (I), (II), (III) and (V), it is required to have system information, which is represented by branching-time models of worlds that
Policy-based Awareness Management (PAM)

include situations, actions between situations, domain values assigned to the variables in each situation and propositional sentences including policy rules.

In the following section, I present a formal framework based on the above intuitions.

4.1.2 Formal Semantics: Definitions

In this section, I form a set of formal definitions that build a logical framework for policy-based awareness in PAM. The basic definitions for the PAM framework are given below.

**Definition 1:** A model $M$ of a system with agents $0, 1, 2, \ldots, n$ is a tuple $(M_0, M_1, \ldots, M_n)$.

**Definition 2:** A model $M_i$ of agent $i$ from the system is defined to be a tuple $(W, T, S, E, V, D, U, \pi, <, I, A_i, <^{\text{done}}_i, E^{\text{received}}_i, P)$. $W$ is a set of worlds. $T$ is a set of time-points. Therefore, situations in a world $w \in W$ at time $t \in T$ is represented by $S_t^w \subseteq W \times T$ which is $S_t^w = \{s_{t,0}^w, s_{t,1}^w \ldots\}$. $E$ is a set of events. $V$ is a set of variables, and $D$ is a set of domain values that can be assigned to the variables. $U$ is a set of propositions, which is closed under conjunction ( $\land$ ), negation ( $\neg$ ) and the operator of knowledge ( $K$ ). I consider $\varphi \lor \psi$ to be an abbreviation for $\neg(\neg \varphi \land \neg \psi)$, and $\varphi \Rightarrow \psi$ to be an abbreviation for $\neg \varphi \lor \psi$. $<$ is a binary relationship in situations, which represents a set of actions. As such,
\( s_{t+1} \) describes one of the situations next after \( s_t \), if and only if there exists an action \( a = (s_t, s_{t+1}) \in < \). A sequence of actions \( ((s_t, s_{t+1}), \ldots, (s_{t+e-1}, s_{t+e})) \) is called a path with length of \( e \), which is represented as \((s_t, s_{t+e})\). The assignment function \( \pi \) is a relation that maps a variable in a situation to a domain value. More formally, \( \pi \subseteq V \times S \times D \). In situation \( s_t \), for \( v \in V \) and \( d \in D \), \( \pi^v_{s_t} = d \) expresses that the domain value \( d \) is assigned to the variable \( v \) in situation \( s_t \). Interpreter \( I \) maps a proposition from \( U \) to a set of true-sets. A true-set for a proposition is a set of valued variables that, regardless of the situation, makes the proposition true. Therefore a proposition can be written as \( \varphi(v_0, v_1, \ldots, v_r) \in U \), while \( I_\varphi \) shows the true-set for \( \varphi \). The relation awareness of agent \( i \) maps the situation to a sub-set of \( U \). Therefore, \( A_i \in S \times \{ \varphi: \varphi \in U \} \).

The relation done-action of agent \( i \) maps the situation to a sub-set of \( < \). As such, \( <_{\text{done}}^i \in W \times T \times \{ a \in < \} \). In a situation, the relation received-event of agent \( i \) maps the situation to a sub-set of \( E \), which means \( E^\text{received}_i \in S \times \{ e \in E \} \). \( P \) is a set of policy rules that will be defined later in Definition 5.

**Definition 3:** A world \( w \in W \) is a tuple \( (T_w, S_w, <_w, E_w) \). In world \( w \), \( T_w \subseteq T \) is a set of time-points, \( S_w \subseteq S \) is a set of situations, \( <_w \subseteq < \) is a set of actions and \( E_w \subseteq E \) is a set of events.
Definition 4: Consider the model $M_i$ for the agent $i$. The satisfiability$^1$ relations are defined below:

\[(M_i, s^w_i) \quad \varphi(v_0, \ldots, v_r) \iff (\pi_{v_0}, \ldots, \pi_{v_r}) \in I_{\varphi}.\] Equation 4-1

\[(M_i, s^w_i) \quad \neg \varphi \iff \text{Not } [(M_i, s^w_i) \quad \varphi].\] Equation 4-2

\[(M_i, s^w_i) \quad \varphi \land \psi \iff (M_i, s^w_i) \quad \varphi \text{ and } (M_i, s^w_i) \quad \psi.\] Equation 4-3

\[(M_i, s^w_i) \quad K_i \varphi \iff (M_i, s^w_i) \quad \varphi.\] Equation 4-4

Therefore, from Equation 4-4:

\[K_i \varphi \iff \varphi.\] Equation 4-5

Provided Equation 4-3 and Equation 4-4, I conclude:

\[K_i (\varphi \land \psi) \iff K_i \varphi \land K_i \psi.\] Equation 4-6

\[(M_i, s^w_i) \quad A_i \varphi \iff \varphi \in A_i.\] Equation 4-7

\[(M_i, s^w_i) \quad X_i \varphi \iff (M_i, s^w_i) \quad K_i \varphi \land A_i \varphi.\] Equation 4-8

\[(M_i, s^w_i) \quad <_i \varphi \iff \alpha \in <_{i}^{\text{done}^w_i}.\] Equation 4-9

---

$^1$ Satisfiability relations are true sentences that are defined in general conditions related to the prior definitions.

$^2$ This is correspondence to the assumption explained in Section 2.2.2 for competency of agents i.e. $W' = W$, where $W$ is a nonempty set of possible worlds or worlds for short and and $W' \subseteq W$ as worlds the agent considers possible.
(M_i, s^w_t) \quad E_i \varphi \text{ iff } a \in E^\text{received}_r^w. \quad \text{Equation 4-10}

(M_i, s^w_t) \quad \circ \varphi \text{ iff there exists a } s^w_{t'} \text{ such that}

(s^w_t, s^w_{t'}) \in \prec_i \text{ and } (M_i, s^w_{t'}) \varphi. \quad \text{Equation 4-11}

(M_i, s^w_t) \quad \odot \varphi \text{ iff there exists a path } (s^w_t, ..., s^w_{t+m})

\text{while } (M_i, s^w_{t+m}) \quad \varphi \text{ and for each } (s^w_{t+s}, s^w_{t+s+1}) \text{ of this path}

(s^w_{t+s}, s^w_{t+s+1}) \in \prec_i. \quad \text{Equation 4-12}

(M_i, s^w_t) \quad \odot_{\leq m} \varphi \text{ iff there exists a path } (s^w_t, ..., s^w_{t+m})

\text{associated with a set of actions } \prec^m \text{ such that}

|\prec^m| = m \text{ while } (M_i, s^w_{t+m}) \quad \varphi \text{ and for each } (s^w_{t+s}, s^w_{t+s+1}) \text{ of this path}

(s^w_{t+s}, s^w_{t+s+1}) \in \prec_i. \quad \text{Equation 4-13}

(M_i, s^w_t) \quad \varphi \text{ Until } \psi \text{ iff there exists a path } (s^w_t, ..., s^w_{t+m})

\text{that for each } (s^w_{t+s}, s^w_{t+s+1}) \text{ of this path}

(s^w_{t+s}, s^w_{t+s+1}) \in \prec_i \quad \text{and there is an action } (s^w_{t+m-1}, s^w_{t+m}) \text{ associated with the}

\text{path such that } (M_i, s^w_{t+m-1}) \quad \varphi \text{ and } (M_i, s^w_{t+m}) \quad \neg \varphi \land \psi. \quad \text{Equation 4-14}

(M_i, s^w_t) \quad \varphi \text{ Until } \leq m \psi \text{ iff there exists a path } (s^w_t, ..., s^w_{t+m})

\text{associated with a set of actions } \prec^m \text{ such that } |\prec^m| = m \quad \text{Equation 4-15}
and for each \((s_{t+s}^w, s_{t+s+1}^w) \in \prec^m\) of this path

\((s_{t+s}^w, s_{t+s+1}^w) \in \prec_i\) while there should also exists an action

\((s_{t+m-1}^w, s_{t+m}^w) \in \prec^m\) in this path such that

\((M_i, s_{t+m-1}^w) \quad \varphi\) and \((M_i, s_{t+m}^w) = \neg \varphi \land \psi\).

\((M_i, s_t^w)\) inevitable \(\varphi\) iff for each path \((s_t^w, ..., s_{t+m}^w)\)

starting from \(s_t^w\) that for each \((s_{t+s}^w, s_{t+s+1}^w)\) of this path

\((s_{t+s}^w, s_{t+s+1}^w) \in \prec_i\) and in this path

there also exists an action \((s_{t+m-1}^w, s_{t+m}^w) \prec_i\) such that

\((M_i, s_{t+m}^w) \quad \varphi\).

\((M_i, s_t^w)\) optional \(\varphi\) iff there exists at least one path

\((s_t^w, ..., s_{t+m}^w)\) starting from \(s_t^w\) that for each \((s_{t+s}^w, s_{t+s+1}^w)\)

of this path \((s_{t+s}^w, s_{t+s+1}^w) \in \prec_i\) in this path

there also exists an action \((s_{t+m-1}^w, s_{t+m}^w) \prec_i\) such that

\((M_i, s_{t+m}^w) = \neg \varphi\).

Considering the awareness satisfiability relation in Equation 4-7, some properties of the logic are given below:

\[A_i(\neg \varphi) \Leftrightarrow A_i \varphi.\]
\[ A_i(\varphi \land \psi) \Leftrightarrow A_i\varphi \land A_i\psi. \]  
Equation 4-19

\[ A_i(\varphi \Rightarrow \psi) \Rightarrow A_i\varphi. \]  
Equation 4-20

\[ A_iK_i\varphi \Rightarrow A_i\varphi. \]  
Equation 4-21

\[ A_iA_i\varphi \Rightarrow A_i\varphi. \]  
Equation 4-22

Therefore, from Equation 4-21 and Equation 4-22, I generate:

\[ A_iX_i\varphi \Rightarrow A_i\varphi. \]  
Equation 4-23

**Definition 5:** A policy rule \( \rho \in P \) is defined as a propositional sentence that belongs to \( U \), while the forbidding, permitting, deterring and requiring policy rules are defined as follows:

- **Forbidding policy rule:** A policy rule \( \rho \in P \) is forbidding, i.e. \( \rho \in P_{Forbidding} \), if and only if there exists a proposition \( \varphi \in U \), an action \( a \in \prec \), and an event \( e \in E \) such that \( \rho = (\varphi \land E_i e \Rightarrow \text{inevitable} \circ (\neg \prec_i a)) \).

- **Permitting Policy Rule:** A policy rule \( \rho \in P \) is permitting, i.e. \( \rho \in P_{Permitting} \), if and only if there exists a proposition \( \varphi \in U \), an action \( a \in \prec \), and an event \( e \in E \) such that \( \rho = (\varphi \land E_i e \Rightarrow \text{optional} \circ \prec_i a) \).
• **Deterring Policy Rule.** A policy rule $\rho \in P$ is deterring, i.e. $\rho \in P_{\text{deterring}}$, if and only if there exists a proposition $\varphi \in U$, an action $a \in \ll$, and an event $e \in E$ such that $\rho = (\varphi \land E_i e \Rightarrow \text{optional} \circ (\neg \ll_i a))$.

• **Requiring Policy Rule.** A policy rule $\rho \in P$ is requiring, i.e. $\rho \in P_{\text{requiring}}$, if and only if there exists a proposition $\varphi \in U$, an action $a \in \ll$, and an event $e \in E$ such that $\rho = (\varphi \land E_i e \Rightarrow \text{inevitable} \circ \ll_i a)$.

Note that defining a policy rule as a propositional sentence allows the agent to define a variable, ensuring the validity constraint for the policy rules, such as the period of applicability (Fallis, 2004). In fact, the agent’s implicit knowledge to a policy rule can be inferred once the policy rule becomes valid.

### 4.1.2.1 Commitments: Implicit Knowledge Leading to Actions

In this section, I discuss how implicit knowledge of an agent guides the agent to select an action. The following definition for the agent’s commitment to an action demonstrates that an agent selects an action to achieve its current implicit knowledge about the future.

Note that, according to Equation 4-5, $K_i(\circ \text{inevitable} \ll_i a)$ provides that $\circ \text{inevitable} \ll_i a$, which means that the agent is going to do action $a$. Therefore, $K_i(\circ \text{inevitable} \ll_i a)$ is defined as “commitment axiom”. In what
follows, Definition 6 shows how an agent updates its implicit knowledge to commit an action.

**Definition 6:** An agent $i$ commits to an action $a \in \prec_i$ by updating its implicit knowledge, following the axiom below:

$$K_i \left( \text{inevitable} \vdash K_i \varphi \lor \text{inevitable} \vdash (\neg K_i \varphi) \right) \land$$

$$K_i \left[ \text{optional} \circ \left( \prec_i a \land K_i \left( \text{inevitable} \vdash \varphi \right) \right) \right] \land (\neg K_i \varphi) \Rightarrow$$

$$K_i (\circ \text{inevitable} \prec_i a).$$

Definition 6 states that an agent commits to the execution of an action if that action brings the agent to a situation that the current implicit knowledge of the agent about the future is achievable, whether now or in the future, from that situation. However, for this to happen, the agent should not currently have that implicit knowledge. By this definition, the agent’s implicit knowledge about its future implicit knowledge derives the agent’s actions.

Note that the problem with this definition is that there might be no action that takes the agent to $K_i \left( \text{inevitable} \vdash \varphi \right)$, or there might also be more than one action available. I address these problems later in Section 4.2.3.

**4.1.3 PAM Framework and Space Shuttle Columbia Disaster**

In this section, I show how the PAM framework can model the awareness in the space shuttle Columbia disaster.
Let me assume three agents in the system: (I) the NASA management, (II) the engineering team, and (III) the Department of Defense (DoD). I define a set of events that consists of $e_0$: shakes in the shuttle. In this system, I define a set of variables as $\{v_0, v_1, v_2, v_3, v_4, v_5\}$ such that $v_0$: status of the shuttle, $v_1$: physical status for the TPS on the right wing, $v_2$: physical status for the TPS on the left wing, $v_3$: image request, $v_4$: image approval to the engineering team and $v_5$: turn around effect. $\{d_0, d_1, d_2, d_3, d_4, d_5\}$ presents the set of domain variables such that $d_0$: no damage, $d_1$: damage, $d_2$: shake, $d_3$: fine, $d_4$: sent and $d_5$: pending.

The propositional sentences are given in $\{\varphi_0, \varphi_1, \varphi_2, \varphi_3, \varphi_4\}$, where $\varphi_0$: the shuttle is shaking, $\varphi_1$: an imaging request has been sent to DoD, $\varphi_2$: DoD has approved the images to the engineering team, $\varphi_3$: the shuttle is suffering of turnaround effect and $\varphi_4$: the shuttle is suffering from damage to its TPS. The interpreter is also defined as $I = \{l_{\varphi_0}, l_{\varphi_1}, l_{\varphi_2}, l_{\varphi_3}, l_{\varphi_4}\}$ where $l_{\varphi_0} = \{(v_0, d_2)\}$, $l_{\varphi_1} = \{(v_3, d_4)\}$, $l_{\varphi_2} = \{(v_4, d_4)\}$, $l_{\varphi_3} = \{(v_5, d_2)\}$, $l_{\varphi_4} = \{(v_1, d_1)\}$, $\{(v_2, d_1)\}$}. Table 4-1 shows how domain values are assigned to the variables in this scenario.

In the situation $s_2$ (see Figure 1-1), given $\pi_{s_2}^{v_5} = d_2$ and provided $l_{\varphi_3} = \{(v_5, d_2)\}$, I can say that $K_{Mn_d} \varphi_3$. Therefore, the NASA management, in this situation, implicitly knows that the shake is caused by the turnaround effect.
### Table 4-1 Value Assignment in different Situations – Space Shuttle Columbia Disaster

<table>
<thead>
<tr>
<th>Situation</th>
<th>Value Assignment to Variables</th>
<th>Situation</th>
<th>Value Assignment to Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_0$</td>
<td>$\pi_{s_0}^v = d_3.$</td>
<td>$s_4$</td>
<td>$\pi_{s_4}^v = d_2$, $\pi_{s_4}^v = d_4$ and $\pi_{s_4}^v = d_4$.</td>
</tr>
<tr>
<td>$s_1$</td>
<td>$\pi_{s_1}^v = d_2.$</td>
<td>$s_5$</td>
<td>$\pi_{s_5}^v = d_2$, $\pi_{s_5}^v = d_4$, $\pi_{s_5}^v = d_4$, $\pi_{s_5}^v = d_0$, $\pi_{s_5}^v = d_0$.</td>
</tr>
<tr>
<td>$s_2$</td>
<td>$\pi_{s_2}^v = d_2$ and</td>
<td>$s_6$</td>
<td>$\pi_{s_6}^v = d_2$, $\pi_{s_6}^v = d_4$, $\pi_{s_6}^v = d_4$, $\pi_{s_6}^v = d_0$, $\pi_{s_6}^v = d_0$.</td>
</tr>
<tr>
<td>$s_3$</td>
<td>$\pi_{s_3}^v = d_2$, $\pi_{s_3}^v = d_4$ and $\pi_{s_3}^v = d_4$.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In the situation $s_5$, given $\pi_{s_5}^v = d_0$ and provided $l_{\varphi_3} = \{(v_5, d_2)\}$, similar to $s_2$, I can say $K_{Mn\varphi_3}$. Furthermore, in the situation $s_5$, given $\pi_{s_5}^v = d_0$ and $\pi_{s_5}^v = d_0$, I have $K_{Mn\varphi_4}$. Therefore, the NASA management, in this situation, implicitly knows that the shake is because of turnaround effect and not because of TPS damage. The difference between the situation $s_5$ with $s_2$ is that the management in $s_5$ implicitly knows the shake is not because of TPS damage.

In the situation $s_6$, given $\pi_{s_6}^v = d_1$, and provided $l_{\varphi_4} = \{(v_1, d_1), (v_2, d_1)\}$, I have $K_{Mn\varphi_4}$. In addition to that, from $\pi_{s_5}^v = d_3$, I conclude $K_{Mn\varphi_3}$. Therefore, the NASA management, in this situation, implicitly knows that the shake is because of TPS damage and not because of turnaround effect.
If I come back to situation $s_1$, looking at Figure 1-1, in this situation the NASA management has two choices: (1) $a_1$: announcing the shake as a turnaround effect; (2) $a_2$: requesting imaging from DoD. In this situation, I have 

$$K_{Mng} \left( \text{inevitable} \uparrow K_{Mng} \varphi_3 \lor \text{inevitable} \uparrow (\neg K_{Mng} \varphi_3) \right).$$

In addition to that, I can also infer $K_{Mng}(\neg \varphi_3)$. Furthermore, I have 

$$K_i \left[ \text{optional} \circ \left( \prec_i a_1 \land K_{Mng} \left( \text{inevitable} \uparrow \varphi_3 \right) \right) \right].$$

Taking the action commitment definition into account, I can conclude that $K_{Mng}(\circ \text{inevitable} \prec_{Mng} a_1)$, which means that the NASA management team commits to announce the shake as a turnaround effect.

Here, I also analyse the commitment of the NASA management to request imaging from DoD in situation $s_1$. Although $K_{Mng}(\neg \varphi_4)$, the first and the second promise of the action commitment in Equation 4-24 is not satisfied for $\varphi_4$. Therefore, the management agent does not commit to $a_2$ and does not request imaging from DoD.

According to the discussion above, when the management agent is in situation $s_1$, it announces the shakes as a turnaround effect. In what follows, I introduce a process for awareness identification and show how the management agent requests imaging, if it identifies its awareness of $\varphi_4$. The process uses policy rules to identify the relevance of propositions to a situation.
4.2 PAM Process: Steps towards Identifying Awareness

PAM defines the notion of a *policy-aware agent* and proposes a three-step process to (I) recognise the relevance of policy rules, (II) recognise the relevance of information required for invoking the relevant policy rules and (III) change behaviours based on the recognised relevant information. The main proposal of PAM is the use of policy rules as a source by which to identify awareness.

4.2.1 Step One: Recognise Relevance of Policy Rules

The objective of this step is to show how agents identify the relevance of a policy rule in a situation. I define *policy-aware agent* as an agent that identifies awareness of a policy rule when it recognises that there is a possibility sometime now or in future that it may violate the rule.

Given a branching-time model of a world to an agent, if there is no possibility of violating the policy rule, according to the action commitment axiom, it simply follows its current implicit knowledge. Therefore, based on Definition 6, the agent commits to an action that brings it to a situation in which the current implicit knowledge about the future is satisfied. However, in step 3, I show how identifying awareness updates the agent’s implicit knowledge and leads to an action commitment.
Policy-based Awareness Management (PAM)

Note that since permitting and deterring policy rules are not in force, violating the policy rules can only happen in forbidding and requiring rules, which is shown, more formally, in Theorem 1.

**Theorem 1 - Satisfiability of Action Commitment:**

a. \( \forall \rho \in P_{Forbidding} : K_i \rho \land K_i \varphi \land K_i \mathcal{E}_i e \land K_i (optional \circ <_i a) \) is not satisfiable.

b. \( \forall \rho \in P_{permitting} : K_i \rho \land K_i \varphi \land K_i \mathcal{E}_i e \land K_i (optional \circ \neg(<_i a)) \) is satisfiable.

c. \( \forall \rho \in P_{deterring} : K_i \rho \land K_i \varphi \land K_i \mathcal{E}_i e \land K_i (optional \circ <_i a) \) is satisfiable.

d. \( \forall \rho \in P_{requiring} : K_i \rho \land K_i \varphi \land K_i \mathcal{E}_i e \land K_i (optional \circ \neg(<_i a)) \) is not satisfiable.

**Proof:**

(a) Provided \( K_i(\rho \in P_{Forbidding}) \), I can say \( K_i(\varphi \land \mathcal{E}_i e \Rightarrow inevitable \circ (\neg<_i a)) \). Given \( K_i \rho \land K_i \varphi \land K_i \mathcal{E}_i e \), I have \( K_i(inevitable \circ (\neg<_i a)) \), which has a clear contradiction with \( K_i (optional \circ <_i a) \). Therefore, the given phrase is not satisfiable.

Cases (b), (c) and (d) can be proved in a similar way.
As I noted above, only forbidding and requiring policy rules are in force, which makes it possible for them to be violated. Therefore, I now provide the formal definition of the policy-aware agent based on these two types of policy rules.

**Definition 7**: A *policy-aware agent* is an agent who identifies its awareness of a policy rule if and only if the agent recognises a possibility, sometime now or in future, of violating that policy rule. There are two situations in which an agent violates a policy rule and accordingly identifies its awareness of the rule:

I. Given a forbidding policy rule, if there is a possibility sometime now or in future for the agent to implicitely know the following items, the agent becomes aware of the policy rule:
   a. The policy rule is true, which implies the applicability of the policy rule.
   b. The event has been received.
   c. The conditions referenced in the policy rule are satisfied.
   d. There is a possibility in the branching-time model that the agent in the next time instant executes the forbidden action referenced in the policy rule.

Therefore, $\forall \rho \in P_{Forbidding}$:

\[
A_i\rho \text{ Until } \left[\neg \text{optional} \bigvee (K_i\rho \land K_i e \land K_i(\text{optional} \circ_i a))\right].
\]

Equation 4-25
II. Given a requiring policy rule, if there is a possibility sometime now or in future, for the agent, to implicitly know the following items, the agent becomes aware of the policy rule:

a. The policy rule is true, which implies the applicability of the rule.

b. The event has been received.

c. The conditions referenced in the policy rule are satisfied.

d. There is a possibility in the branching-time model that the agent in the next time instant does not execute the required action referenced in the policy rule.

Therefore, $\forall \rho \in P_{Requiring}$:

$$A_i \rho \text{ Until } [\neg \text{optional} \land K_i e \land (optional \land (\neg (\leq \land a)))].$$

Equation 4-26

In the above definition, I consider the situations in which the agent has a possibility, sometime now or in future, in the branching-time model that its commitment to an action is not satisfiable (see Theorem 1). For example, in forbidding policy rules, $K_i(\text{optional } \land \leq \land a)$ is not satisfiable in the presence of a forbidding policy rule that refers to action $a$, if the conditions are satisfied. Therefore, such a situation creates the possibility, sometime now or in future, of violating the policy rule. This requires the agent to regard the policy rule as relevant and to identify its awareness of the rule. Similarly, for requiring policy rules, $K_i(\text{optional } \land \neg (\leq \land a))$ may not be satisfiable in the existence of a
4.2.1.1 Recognise Relevance of Policy Rules in the Space Shuttle Columbia Disaster

In the space shuttle Columbia disaster, I mentioned that there was a relevant policy rule available in NASA. The policy rule stated that “when an aircraft experiences unusual shakes, if there is any TPS damage, the spacewalk procedure must be granted.” The formalisation of this policy rule in the PAM framework is

$$\rho = (\varphi_4 \land E_{Mng}e_0 \Rightarrow \text{inevitable} \circ_{Mng} a_5).$$

This is a requiring policy rule. Assume that the policy rule $\rho$ is valid in all the situations, so the NASA management implicitly knows that the policy rule in all these situations, i.e.

$$K_{Mng}\rho.$$ By theorem 1, the NASA management becomes aware of \(\rho\), all the way, in both paths \((s_1, s_3, s_4, s_5)\) and \((s_1, s_3, s_4, s_6)\) in Figure 1-1.

4.2.2 Step Two: Recognise Relevance of Information in the Relevant Policy Rules

The objective of this step is to show how agents identify their awareness of a relevant proposition that is required to invoke the relevant policy rule. A policy-aware agent identifies the conditions referenced in a policy rule until it drops its awareness of that rule. Theorem 2 shows how policy awareness in agents causes

requiring rule referring to action \(a\), if the conditions are satisfied. Therefore, the agent regards a policy rule as relevant by looking at the possibility of violating the rule.
them to identify their awareness of the relevant information that is required to 
invoke the relevant policy rule.

**Theorem 2: Awareness of the Policy Rule’s Condition**

A policy-aware agent identifies its awareness of the conditions referenced in a 
forbidding or requiring policy rule, until it becomes aware of that rule:

\[ \forall \rho \in P_{\text{Permitting}} \cup P_{\text{Requiring}}: A_i \varphi \text{ Until } \neg A_i \rho. \]

**Proof:**

First, I prove the theorem for forbidding policy rules. Taking the definition of 
*Until* into account (Surhone et al., 2010), I can say \( A_i \rho \text{ Until } \neg A_i \rho \). By 
replacing the forbidding policy rule \( \rho \) with its definition, I have \( A_i (\varphi \land E_i e \Rightarrow \text{inevitable } \circ (\neg \prec_i a)) \text{ Until } \neg A_i \rho \). By Equation 4-20, I can generate 
\( A_i \varphi \text{ Until } \neg A_i \rho \).

For requiring policy rules, I can prove the theorem in a similar way.

According to the properties of the logic given in Equation 4-21, Equation 4-22 
and Equation 4-23, if an agent is aware of other agents’ awareness, or aware of 
others’ implicit knowledge or explicit knowledge about a propositional sentence, 
the agent will be aware of that sentence. Therefore, if an agent is aware that 
another agent in the system identifies a policy rule as a relevant rule – that is, 
awareness of awareness – then the agent will become aware of the condition for
that particular policy rule. This awareness can be useful when the system involves different agents. Theorem 3 formalises this claim.

**Theorem 3: Awareness Under Others’ Awareness and Knowledge:**

A policy-aware agent identifies its awareness of the conditions referenced in a policy rule when the agent is aware of the others’ awareness – their implicit or explicit knowledge about that rule.

\[ \forall \rho \in P_{permitting} \cup P_{requiring} : \]

\[ a. A_i \varphi \text{ Until } \neg A_i A_j \rho. \]

\[ b. A_i \varphi \text{ Until } \neg A_i K_j \rho. \]

\[ c. A_i \varphi \text{ Until } \neg A_i X_j \rho. \]

**Proof:**

(a) In the theorem 3, replacing \( A_i A_j \rho \) with \( A_i \rho \) (Equation 4-23) results in (a).

Cases (b) and (c) can be proved in a similar way.

**4.2.2.1 Recognizing Relevance of Conditions Referenced to the Relevant Policy Rule in the Space Shuttle Columbia Disaster**

In Section 4.2.1.1, I showed that the NASA management identifies its awareness of policy rule \( \rho \) all the way in both paths \((s_1, s_3, s_4, s_5)\) and \((s_1, s_3, s_4, s_6)\). Having \( \rho = (\varphi_4 \land E_{Mng} e_0 \Rightarrow \text{inevitable} \circ <_{Mng} a_5) \) and by theorem 2, I can
conclude that the management agent identifies its awareness of \( \varphi_4 \) in the situations of the aforementioned paths. In the next step, I show how this awareness results in committing the management agent to \( a_2 \).

### 4.2.3 Step Three: Change Behaviours based on Recognised Relevant Information

The logic of PAM so far has treated policy-based awareness as a mental attitude of agents, which indicates the relevance of a proposition to a situation. However, so far I have not formalised how awareness guides or determines the agent’s future knowledge. In other words, I have not discussed how the agent’s current awareness leads to its future implicit knowledge and how it results in selecting an action.

An alternative is to look at the relationship between current awareness and future implicit knowledge involving different strategies for updating knowledge. Different types of agents will have different types of strategies. In answer to the aforementioned problem of agents’ commitments in Section 4.1.2.1, I propose three different strategies: Volitional Agent, Cautious Volitional Agent and Hasty Volitional Agent. I will call the last two ones as Cautious and Hasty agents.

When a volitional agent is aware of a proposition, it updates its implicit knowledge of the proposition, stating that it will, eventually and inevitably, have
implicit knowledge about the proposition or about the negation of that proposition. There are two problems with this strategy:

I. There might be no path in the branching-time model that takes the agent to such implicit knowledge. In other words, it might not be possible to have implicit knowledge about the relevant proposition or about the negation of that proposition, any time now or in future. For example in Figure 1-1, being in \( s_1 \), there is no path for the agent to have implicit knowledge to \( \neg \phi_0 \), or in another word the agent cannot impliedly know that there is no shake. Accordingly, I define the Cautious Volitional Agent, or in short, Cautious Agent. In this strategy, when a cautious agent is aware of a proposition, it checks the possibility of achieving implicit knowledge about the proposition or about the negation of the proposition. Then, if it is possible, the agent updates its implicit knowledge of the proposition stating that it will, eventually and inevitably, have implicit knowledge about that proposition or about the negation of that proposition.

II. There might be several paths in the branching-time model that take the agent to such implicit knowledge. In other words, there is more than one possibility of having such implicit knowledge in the branching-time model. For example, in Figure 1-1, being in \( s_1 \), there are more than one paths that end me with situations in which the agent implicitly believes
that there is no damage on TPS i.e. any path which ends to either $s_2$ or $s_5$. Therefore, I define the Hasty Volitional Agent, or in short, *Hasty Agent* in short. In this strategy, when a hasty agent is aware of a proposition, in the shortest path possible, it updates its implicit knowledge to the fact that it will eventually arrive at the implicit knowledge of the proposition or the negation of that proposition.

Hence, in what follows, I give the formal definitions for each of these proposed strategies:

**Definition 8:** I define three types of agents:

a. Volitional Agent:

$$A_i \varphi \Rightarrow \left[ \text{inevitable} \left( K_i \left( \text{inevitable} \odot K_i \varphi \right) \text{Until } K_i \varphi \right) \right] \lor$$

$$\left[ \text{inevitable} \left( K_i \left( \text{inevitable} \odot K_i (\neg \varphi) \right) \text{Until } K_i (\neg \varphi) \right) \right].$$

Equation 4-27

b. Cautious Agent:

$$A_i \varphi \Rightarrow \left[ \text{inevitable} \left( K_i \left( \text{inevitable} \odot K_i \varphi \right) \text{Until } \left( \neg K_i \left( \text{optional} \odot K_i \varphi \right) \lor K_i \varphi \right) \right) \right] \lor$$

$$\left[ \text{inevitable} \left( K_i \left( \text{inevitable} \odot (K_i (\neg \varphi)) \right) \text{Until } \left( \neg K_i \left( \text{optional} \odot K_i (\neg \varphi) \right) \lor K_i (\neg \varphi) \right) \right) \right].$$

Equation 4-28

c. Hasty Agent:
∀d_{min} \in \mathbb{N} \text{ such that:}

\[
A_i \varphi \Rightarrow \left[ \text{inevitable} \left( K_i(\text{inevitable} \Diamond K_i \varphi) \text{Until } \leq d_{\text{min}} \right) \right] \lor \left[ \text{inevitable} \left( K_i(\text{inevitable} \Diamond (K_i(\neg \varphi))) \text{Until } \leq d_{\text{min}} \right) \right].
\]

There exists no \( d \leq d_{\text{min}} \) such that:

\[
A_i \varphi \Rightarrow \left[ \text{inevitable} \left( K_i(\text{inevitable} \Diamond K_i \varphi) \text{Until } \leq d \right) \right] \lor \left[ \text{inevitable} \left( K_i(\text{inevitable} \Diamond (K_i(\neg \varphi))) \text{Until } \leq d \right) \right].
\]

Equation 4-29

A volitional agent reaches an identical conclusion to a proposition if and only if it can recognise the relevance of that proposition to its situation and becomes aware of it. Then, the agent inevitably, sometime now or in future, implicitly knows the validity (falsity or truth) of the proposition that the agent is aware. As such, being aware of a proposition updates the agent’s implicit knowledge that it will eventually and inevitably have implicit knowledge of the proposition or its negation. Similarly, a cautious agent has the same knowledge, if it is possible. A hasty agent also has the knowledge that it is implicitly, inevitably and in the shortest path aware of the proposition. This knowledge-update solves the problems mentioned in Section 4.1.2.1. More formally, I have the following
theorem for different types of agents, namely, volitional, cautious and hasty. The theorem shows how awareness leads to updates in the agent’s implicit knowledge.

**Theorem 4: Awareness Leading to Knowledge Update**

a. Volitional Agent:

\[ A_i \varphi \Rightarrow K_i \left( \text{inevitable} \Diamond K_i \varphi \lor \text{inevitable} \Diamond K_i (\neg \varphi) \right). \]

b. Cautious Agent:

\[ A_i \varphi \land \left[ \left( \text{inevitable} \left( K_i \text{optional} \Diamond K_i \varphi \text{ Until } K_i \varphi \right) \right) \lor \left( \text{inevitable} \left( K_i \text{optional} \Diamond K_i (\neg \varphi) \text{ Until } K_i (\neg \varphi) \right) \right) \right] \Rightarrow K_i \left( \text{inevitable} \Diamond K_i \varphi \lor \text{inevitable} \Diamond K_i (\neg \varphi) \right). \]

c. Hasty Agent:

\[ \forall d_{min} \in \mathbb{N} \text{ such that:} \]

\[ A_i \varphi \land \left[ \left( \text{inevitable} \left( K_i \text{optional} \Diamond K_i \varphi \text{ Until } \leq d_{min} K_i \varphi \right) \right) \lor \left( \text{inevitable} \left( K_i \text{optional} \Diamond K_i (\neg \varphi) \text{ Until } \leq d_{min} K_i (\neg \varphi) \right) \right) \right] \Rightarrow K_i \left( \text{inevitable} \Diamond \leq d_{min} K_i \varphi \lor \text{inevitable} \Diamond \leq d_{min} K_i (\neg \varphi) \right). \]

*there exists no* \( d \leq d_{min} \text{ such that:} \)
Policy-based Awareness Management (PAM)

\[
A_i \varphi \land \left[ \left( \mathsf{inevitable} \left( K_i \mathsf{optional} \mathsf{\downarrow} K_i \varphi \mathsf{ Until} \, \preceq_d K_i \varphi \right) \right) \lor \left( \mathsf{inevitable} \left( K_i \mathsf{optional} \mathsf{\downarrow} K_i (\neg \varphi) \mathsf{ Until} \, \preceq_d K_i (\neg \varphi) \right) \right) \right]
\implies K_i \left( \mathsf{inevitable} \mathsf{\downarrow} \preceq_d K_i \varphi \lor \mathsf{inevitable} \mathsf{\downarrow} \preceq_d K_i (\neg \varphi) \right).
\]

Proof:

a. Assume the promise \( A_i \varphi \). By Definition 8, I can conclude to the following axiom:

\[
\left( \mathsf{inevitable} \left( K_i \mathsf{\downarrow} K_i \varphi \right) \mathsf{Until} K_i \varphi \right) \lor \\
\left( \mathsf{inevitable} \left( K_i \mathsf{\downarrow} K_i (\neg \varphi) \right) \mathsf{Until} K_i (\neg \varphi) \right).
\]

By the definition of \( \mathsf{Until} \), the axiom \( \varphi \mathsf{Until} \psi \) interferes \( \mathsf{inevitable} \mathsf{\downarrow} \varphi \).

Therefore, by Equation 4-5, I have \( \mathsf{inevitable} \mathsf{\downarrow} K_i \varphi \lor \mathsf{inevitable} \mathsf{\downarrow} K_i (\neg \varphi) \)
which can infer \( K_i \left( \mathsf{inevitable} \mathsf{\downarrow} K_i \varphi \lor \mathsf{inevitable} \mathsf{\downarrow} K_i (\neg \varphi) \right) \).

Cases (b) and (c) follow a similar proof.

So far I have shown how an agent, being aware of a proposition, updates its implicit knowledge. The objective of this step is to change behaviour through changes in awareness. Here, in theorem 5, I take Definition 6 for action commitment into consideration to see how awareness of a proposition leads in committing to an action.
Theorem 5: Awareness Leading to Action Commitment

a. Volitional Agent:
When a volitional agent is aware of a proposition but it does not implicitly know the proposition, the agent selects the action that brings the agent to a situation in which the implicit knowledge about that proposition will be ensured either in that situation or some future time after that situation.

$$A_i \varphi \land K_i \left[ \text{optional } \circ \left( \prec_i a \land K_i (\text{inevitable } \diamond \varphi) \right) \right] \land (\neg K_i \varphi)$$

$$\Rightarrow K_i (\circ \text{inevitable } \prec_i a).$$

b. Cautious Agent:
When a cautious agent is aware of a proposition but it does not implicitly know the proposition, the agent selects the action that brings the agent to a situation in which the implicit knowledge about that proposition will be ensured either in that situation or some future time after that situation, unless this implicit knowledge is impossible to achieve in the branching-time model.

$$A_i \varphi \land \left[ \left( \text{inevitable } \left( K_i \text{optional } \diamond K_i \varphi \text{ Until } K_i \varphi \right) \right) \lor \left( \text{inevitable } \left( K_i \text{optional } \diamond K_i (\neg \varphi) \text{ Until } K_i (\neg \varphi) \right) \right) \right]$$

$$\land K_i \left[ \text{optional } \circ \left( \prec_i a \land K_i (\text{inevitable } \diamond \varphi) \right) \right] \land (\neg K_i \varphi)$$

$$\Rightarrow K_i (\circ \text{inevitable } \prec_i a).$$

c. Hasty Agent:
When a hasty agent is aware of a proposition but it does not implicitly know the proposition, the agent selects the action that brings the agent to a situation in which the implicit knowledge about that proposition will be ensured in the shortest path possible either in that situation or some future time after that situation.

\[ \forall d_{\text{min}} \in \mathbb{N} \text{ such that:} \]

\[ A_i \phi \land \left[ \left( \text{inevitable} \left( K_i \text{optional} \odot K_i \phi \text{ Until} \leq d_{\text{min}} K_i \phi \right) \right) \lor \left( \text{inevitable} \left( K_i \text{optional} \odot K_i (\neg \phi) \text{ Until} \leq d_{\text{min}} K_i (\neg \phi) \right) \right) \right] \land \neg K_i (\neg \phi) \]

\[ \Rightarrow K_i (\text{optional} \prec_i a). \]

there exists no \( d \leq d_{\text{min}} \) such that:

\[ A_i \phi \land \left[ \left( \text{inevitable} \left( K_i \text{optional} \odot K_i \phi \text{ Until} d K_i \phi \right) \right) \lor \left( \text{inevitable} \left( K_i \text{optional} \odot K_i (\neg \phi) \text{ Until} d K_i (\neg \phi) \right) \right) \right] \land \neg K_i (\neg \phi) \]

\[ \Rightarrow K_i (\text{optional} \prec_i a). \]
Proof:

a. Assume the promise $A_i \phi$. By theorem 4, I can say $K_i (\text{inevitable} \circ K_i \phi \lor \text{inevitable} \circ K_i (\neg \phi))$. Provided the promise $K_i \left(\text{optional} \circ (\preceq_i a \land K_i (\text{inevitable} \circ \phi))\right) \land (\neg K_i \phi)$ and according to Definition 6, I can conclude $K_i (\circ \text{inevitable} \preceq_i a)$.

Cases (b) and (c) follow a similar proof.

In steps 1 and 2 of the process, I have identified the relevant policy rules and information required to trigger those policy rules. In this step, Theorem 5 shows that awareness can result in a change of agents’ commitment to their actions. In fact, when an agent identifies the relevance of a proposition, it commits to an action that somehow brings the agent to the implicit knowledge of that proposition in that situation or afterwards. For this to happen, it requires some conditions dependent on the type of the agent’s strategy, namely, volitional, cautious or hasty.

4.2.3.1 Change of Behaviour Based on Awareness in the Space Shuttle Columbia Disaster

In order to simplify the problem, I posit the NASA management to be a volitional agent.

$$A_{Mng} \phi_0 \land K_{Mng} \left[\text{optional} \circ \left(\preceq_{Mng} a_2 \land K_{Mng} (\text{inevitable} \circ \phi_0)\right)\right] \land (\neg K_{Mng} \phi_0),$$

by Theorem 5, concludes the action commitment axiom for $a_2$. 
Therefore, the management agent commits to $a_2$ and requests imaging from DoD.

All in all, when the unusual shakes happened in the shuttle, the TPS damage as the condition of the policy rule was relevant information that NASA overlooked. This oversight resulted in the failure to ask DoD for the high-resolution images.

4.3 Summary and Outlook

As I discussed in chapter 2, awareness in the field of computer-supported cooperative work and in the area of software agents lacks a definitive method to identify relevant information. I also indicated that although several policy frameworks such as Directory Enabled Networks - next generation (DEN-ng) are currently being used to implement awareness in agent systems, the use of policies as guidelines to identify required information for awareness has not been forthcoming. In this chapter, I have shown that in different situations policy rules can be a source of awareness that change the behaviour of roles without the need for direct orders. I proposed a method called Policy-based Awareness Management (PAM) that refines policy rules to awareness in different situations. I suggested using policy rules as an alternative to identify awareness and to recognise the relevance of information. I demonstrated PAM via its application to the space shuttle Columbia disaster in US in 2003.

This chapter had two main contributions: (I) a framework for policy-based awareness of agents and (II) a three-step process to identify awareness using
policy rules. The framework is provided in order to define different concepts being used in the process, and it is supported by the literature of software agents.

The PAM framework is based on a tree-like structure with a single past and multiple futures, called branching-time model of world. At any change of environment – that is, the occurrence of an event – PAM defines a possible world represented by a branching-time model. In each model, an arrangement of situations identifies the actions which of each change one situation to another. The PAM framework follows the logic of general awareness. In this logic, implicit knowledge about a proposition can be inferred in a situation where the proposition is valid. Awareness is defined as a relevant proposition of the situation. The PAM framework also defines explicit knowledge where both implicit knowledge and awareness are gained.

The PAM process is built on the above framework and uses policy rules as a source to identify awareness in three steps. The first step is to identify the relevant policy rules. In this step, the agent becomes aware of a policy rule if and only if in the branching-time model of the world there is a possibility, sometime now or in future, that the agent violates the rule. The second step is to identify the relevance of information. When an agent becomes aware of a policy rule, in order to trigger the rule, it needs to explicitly know the condition of the rule. Therefore, the agent becomes aware of the policy rule and considers the conditions as relevant propositions. In the third step, the agent changes its
behaviour to gain implicit knowledge of the conditions of the policy rules. The agent selects an action that brings the agent into a situation from which the relevant proposition can be achieved.

I have so far discussed intuitions and formalisms of PAM. However, I have not yet discussed how PAM can be developed. To this end, in the next chapter I discuss the development aspects related to analysis, design and implementation of agent systems that are equipped by PAM.
The objective of this chapter is to provide guidelines for developing applications that use PAM to identify awareness. The initial ideas have been borrowed from Cooperative management Methodology for Enterprise Networks (CoMEN). The contribution of this chapter is based on four main observations. First, it is regarded as uncontroversial that the dynamic and uncertain nature of everyday life overloads individuals with irrelevant or loosely relevant information (Rennecker, 2005). Once new, certain information comes to the fore, a method to identify awareness is essential. This is missing in CoMEN. Second, computer supported cooperative work (CSCW) has recently evolved to embrace a complexity-based paradigm (Zacarias et al., 2010). This paradigm replaces deterministic perspectives of the internal and external views of systems by agency principles (Magalhães, 2004). The agency principles emphasise the role of individuals in a system. Zacarias et al. (2010) define the agency relationship as interactions between individuals and software agents, in order to perform tasks on the individuals’ behalf. Much research proposes the use of agents to aid individuals maintaining their awareness (Ray, Shahrestani, et al., 2005; Ulieru and Worthington, 2006). Although CoMEN proposes the use of software agents to
implement CSCW-based applications, it lacks authoritative stages to design and implement such applications to identify awareness in run-time. By taking the initial ideas from CoMEN, PAM employs software agents to assist individuals. Third, the recent discussions in CSCW have been marked by the emphasis given to models and modeling activities as a means of facilitating development (Macías et al., 2009), which is not given by CoMEN. Note that the modeling concept, discussed in this chapter, refers to the model of agent systems to identify awareness. Fourth, in order to implement PAM, I require a methodological roadmap, which can be given by CoMEN. However, it needs improvement to support PAM.

Here, I do not model context awareness; instead, I develop agent systems that assist individuals to identify awareness using policy rules. I pick up one of the methodologies for awareness obtainment and extend it for awareness identification. The methodologies related to this study and presented in my literature review, address three main perspectives: computer sciences, cognitive sciences and social sciences. Based on the discussion in Chapter 2, CoMEN, proposed by Pradeep Ray et al. (2005), and its related work, such as that by Daneshgar and Ray (2000a), address these three aforementioned aspects, influencing the objective of this chapter. As a result, in this chapter I opt to borrowed some ideas from CoMEN for awareness identification using PAM.
In this chapter, I present development guidelines in order to build agent-based systems that apply PAM. The guidelines extend CoMEN and employ agents to identify awareness. In this chapter, I present modeling techniques as well as implementation aspects of development based on the PAM framework. Although the implementation is given in a Java toolkit called Recursive Porous Agent Simulation Toolkit, the main concepts can be applied in most agent development toolkits that support the object-oriented perspective.

The rest of this chapter is organised in the following way: Section 5.1 presents the main contribution of this chapter. This section gives the implementation guidelines. Section 5.2 presents a mapping of scenario analysis concepts to design and implementation classes proposed in the previous section. Section 5.3 demonstrates the application of the contributions of this chapter in the space shuttle Columbia disaster. Section 5.4 summarises the chapter and explains the objectives of the next chapter.

5.1 Development Guidelines for Employing PAM to Identify Awareness

Recalling CoMEN, I define two phases: (I) Scenario Analysis and (II) System Design and Implementation. One of the most common ways to differentiate analysis from design is to say that analysis asks “what” and design asks “how”. This sounds compelling at first. Clearly, if I can first define what I want the
System to do, then it will be easier to define how the system should do it. As I discussed in Chapter 2, although CoMEN pays great attention to scenario analysis, it lacks a well-defined methodology for the system design and implementation phase. However CoMEN briefly gives the objectives for this phase. Following these objectives, this chapter will customise the analysis phase of CoMEN and propose a structure for the design and implementation of agent systems that apply PAM for awareness identification. I also relate the scenario analysis phase to the proposal for the design and implementation phase.

5.1.1 Scenario Analysis

Scenario Analysis is an abstract specification that studies the scenarios taking place in the system. This phase involves a top-down analysis under the following stages integrated with PAM concepts: Overall System Study, Process Study and Logical Component Identification. Figure 5-1 shows the concepts involved in this phase.

5.1.1.1 Overall System Study

This stage produces a rich picture of the system used to gain an appreciation of the problem situation. In this stage, I begin by finding the different scenarios of the system.
Therefore, it is important to identify the scenarios that are in the scope of the system. In this stage, I also study the environment and recognise the particular changes in the environment that initiate the scenarios. Finally, I identify the roles involved in each scenario. The structure of entities in this analysis stage is depicted in Figure 5-1.

5.1.1.2 Process Study

This stage is the study of scenarios in terms of their procedures. In Chapter 4, I presented the idea of transforming decision trees to branching-time models of worlds. Besides the various definitions for procedures, I adopt the definition of Rao & Georgeff (1995), who define procedures as paths with an accurate starting and ending point in the decision tree.
5.1.1.3 Logical Components Identification

The objective of this stage is to examine the content of the procedures and indicate the information involved in each of them. I also identify the inputs and outputs for the procedures, as well as the policies that the roles should consider when performing the procedures. As it is depicted in Figure 5-1, each procedure takes some information as inputs and changes information as outputs. The policies are applicable in the presence of some information. This information demonstrates the policy applicability parameters.

5.1.2 System Design and Implementation

This phase encompasses building an agent-based application that facilitates the human behaviours by identifying the relevance of information, namely, awareness. In this phase, I implement the application considering the concepts and the formalism given in Chapter 4 for the PAM framework and process. This gives an architecture that involves developing the PAM Framework as well as Agent Deliberation. Figure 5-2 shows the concepts related to the system design and implementation phase.

Although the presented concepts, here, are generic and can be applied in any object-oriented and agent-based development toolkits, in this section I develop the concepts in a Java toolkit, called Recursive Porous Agent Simulation Toolkit.
(Repast). This toolkit provides a core collection of classes for the building and running of agent-based systems. I used version 3.1 of the Java support for Repast and the development framework was eclipse-SDK-3.4.1-win32.

5.1.2.1 Developing PAM Framework

In this stage, I develop the PAM framework as it is explained in Section 4.1. The basic components of this stage are depicted in Figure 5-2. This figure shows the actual classes and their associations. This class diagram helps to explain the implementation dimension of PAM. The implementation of each class in Repast is explained briefly, and due to space limitations, some classes are provided in the Appendices.

I begin with the class PAM_Model (see Figure 5-3). In run-time it creates instances of the components given in Figure 5-2. The instances of this class will be actual running systems. In order to run a system in Repast, the associated class should be inherited from SimpleModel. When I run the model, the method setup configures the environments. Therefore, any property related to the environment, such as the number of agents, should be specified here. The method buildModel configures and creates the model in run-time. Therefore, here, the instances of the entities involved in the system should be created from the class types shown in Figure 5-2. This implements the design model of the

---

system. preStep and postStep are two optional methods of this class that are run before and after setup.
import uchicago.src.sim.engine.SimInit;
import uchicago.src.sim.engine.SimpleModel;

public class PAM_Model extends SimpleModel

//PUBLIC PROPERTIES

//MODEL COMPONENTS

//PRIVATE PROPERTIES

//CONSTRUCTOR
public PAM_Model ()
name = "PAM_Model";

// METHODS
public void setup()
super.setup();
autoStep = true;
shuffle = true;

public void buildModel()

//optional
protected void preStep()
System.out.println("Initiating step " + getTickCount());

//optional
protected void postStep()
System.out.println("Done step " + getTickCount());

public static void main(String[] args)
SimInit init = new SimInit();
PAM_Model model = new PAM_Model();
initloadModel(model, ",", false);

Figure 5-3 Class PAM_Model

Figure 5-4 shows the implementation of classes Variable, DomainValue and ValuedVariable. In the design model of the system, I define certain variables and the values that can be assigned to these variables. This assignment is actually based on the class ValuedVariable that implements \( \pi \) in Definition 2 of Chapter 4. In fact, each instance of the class ValuedVariable
shows the value of a variable in a particular situation. I discuss the class Situation later in this section.

```java
public class Variable
    //PUBLIC PROPERTIES
    String name;
    //CONSTRUCTOR
    // to initiate the properties

    //**************************************************************************

public class DomainValue
    //PUBLIC PROPERTIES
    String name;
    //CONSTRUCTOR
    // To Do: initiate the properties

    //**************************************************************************

public class ValuedVariable
    //PUBLIC PROPERTIES
    Situation situation;
    Variable variable;
    DomainValue value;
    //CONSTRUCTOR
    // To Do: initiate the properties
```

Figure 5-4 Classes Variable, DomainValue and ValuedVariable

Figure 5-5 shows the implementation of the classes TrueSet, Interpreter and Proposition. The class TrueSet shows all the possibilities that a proposition can become true. Therefore, an instance of TrueSet assigned to a proposition is a set of different sets that shows the values that should be assigned to the variables for validity of the proposition. In Figure 5-5, this is implemented using a Hashtable with variables as its keys. Therefore, each instance of the class TrueSet is assigned to a proposition. It has also a hashtable to store
the different values that each variable can get to make the proposition valid. The variables are chosen to be the keys of the hashtable. The class Interpreter stores the different instances of TrueSet for the different propositions.

```java
import java.util.Hashtable;

public class TrueSet
//PUBLIC PROPERTIES
Proposition proposition;
Hashtable [] tuples;
//CONSTRUCTOR
// To Do: initiate the properties

/***********************

public class Interpreter
//PUBLIC PROPERTIES
TrueSet [] trueSet;
//CONSTRUCTOR
// To Do: initiate the properties

***********************

public class Proposition
///PUBLIC PROPERTIES
String name;
//CONSTRUCTOR
// To Do: initiate the properties
```

Figure 5-5 Classes TrueSet, Interpreter and Proposition

Figure 5-6 shows the implementation of the classes Action and DoneAction. An action is defined by the situation in which the agent executes the action, i.e. source, and the situation to which the agent moves after executing the action, i.e. destination. In this situation, the agent holds the executed action as an instance of the class DoneAction. Therefore, this refers to both the destination of the action as well as to the executed action.
public class Action
    //PUBLIC PROPERTIES
    public String name;
    public Situation source;
    public Situation destination;

    //CONSTRUCTOR
    // To Do: initiate the properties

    //*******************************************

public class DoneAction
    //PUBLIC PROPERTIES
    Situation situation;
    Action action;

    //CONSTRUCTOR
    // To Do: initiate the properties

Figure 5-6 Classes Action and DoneAction

Figure 5-7 shows the implementation of the classes Event and ReceivedEvent. For each instance of the class Event, there exist a number of situations in which the event can happen. Therefore, this class refers to an array of situations. When an agent receives an event in a situation, it refers to an instance of the class ReceivedEvent.

public class Event
    //PUBLIC PROPERTIES
    public String name;
    Situation [] situations;
    boolean received;
    //CONSTRUCTOR
    // To Do: initiate the properties

    //*******************************************

public class ReceivedEvent
    //PUBLIC PROPERTIES
    Situation situation;
    Event event;
    //CONSTRUCTOR
    // To Do: initiate the properties

Figure 5-7 Classes Event and ReceivedEvent
I define each policy rule as a type of Proposition, while the class PolicyRule has also different types such as permitting, deterring, forbidding and requiring. Therefore, the class PolicyRule extends the class Proposition, and at the same time, each type of policy rules extends the class PolicyRule. Figure 5-8 is the implementation of the classes policyRule and forbiddingPolicyRule. The classes deterringPolicyRule, permittingPolicyRule and requiringPolicyRule are implemented in a similar way to forbiddingPolicyRule.

```java
class PolicyRule extends Proposition {
    //PUBLIC PROPERTIES
    Action action;
    Event event;
    Proposition condition;

    //CONSTRUCTOR
    To Do: initiate the properties

    //CONSTRUCTOR
    public ForbiddingPolicyRule(Action act, Event e, Proposition cond, String name)
        super(e, cond, name);
}
```

Figure 5-8 Classes PolicyRule and ForbiddingPolicyRule

Figure 5-9 shows the implementation of the class Awareness. Awareness is defined as a set of propositions where each is relevant in a given situation. Therefore, each instance of the class Awareness refers to a situation as well as a proposition.
The class `Time` is implemented in Figure 5-10. In Repast, current time changes with the CPU tick and can be counted by `getTickCount` (see the methods `preStep` and `postSetup` in Figure 5-3). Therefore, I can identify each time point by a given integer named `tickIndex`. Based on the branching-time model of worlds and the first characteristic mentioned in Chapter 4, at each time point, the system can be in different situations. As such, each time point, or to put it in another way, each instance of the class `Time`, refers to an array of situations.

Figure 5-11 shows the implementation of the class `World`. According to Definition 2 in Chapter 4, a world \( w \in W \) is a tuple \( (T_w, S_w, <_w, E_w) \). The essence of this definition is that each world only includes a sub-set of all the time
points, situations, actions and events that are involved in a particular world. For example, a time point can exist but may not be involved in the branching-time model of a specific world. The class World refers to the arrays of times, situations, events and actions.

```java
public class World

    // PUBLIC PROPERTIES
    Time[] times;
    Situation[] situations;
    Event[] events;
    Action[] actions;

    // CONSTRUCTOR
    // To Do: initiate the properties

Figure 5-11 Class World
```

Figure 5-12 shows the implementation of the class Situation. In Definition 2 in Chapter 4, I defined \( S \subseteq W \times T \). Accordingly, each instance of the class Situation refers a particular point of time and in a specific world. The class Situation also has an identification index.

```java
public class Situation

    // PUBLIC PROPERTIES
    Time time;
    World world;
    int situationIndex;

    // CONSTRUCTOR
    // To Do: initiate the properties

Figure 5-12 Class Situation
```
Figure 5-13 shows the implementation of the class Agent. Definition 1 in Chapter 4 defines the model of a system as a tuple of the different models associated with the different agents involved in the system. This highlights the role of the class Agent in the design and implementation of the system. Figure 5-2 depicts the relationship of Agent to the different classes in the model. In this stage of development, I implement the agent, while the development of agents’ deliberations belongs to the next stage. This should be noted that although the class Agent has an array of the class Situation with access to the class Time, the time ticks in situations imply those associated with situations while the time points related to the agent consider in which time tick the agent can be defined. For complete implementation of this class, readers should refer to Appendix 1.

```java
import java.util.ArrayList;
import uchicago.src.sim.engine.Stepable;
public class Agent

//PUBLIC PROPERTIES
Situation currentSituation;
World [] worlds;
Time [] times;
Situation [] situations;
Variable [] variables;
DomainValue [] values;
Event [] events;
Action [] actions;
Proposition [] propositions;
ValuedVariable [] valuedVariables;
ForbiddingPolicyRule [] forbiddingPolicyRules;
RequiringPolicyRule [] requiringPolicyRules;
PermittingPolicyRule [] permittingPolicyRules;
DeterringPolicyRule [] deterringPolicyRules;
Interpreter interpreter;
ArrayList<Awareness> awareness;
ArrayList<ReceivedEvent> receivedEvents;
ArrayList<DoneAction> doneActions;

//CONSTRUCTOR
```

Figure 5-13 Class Agent
5.1.2.2 Developing Agents Deliberations

When designing any agent-based system, it is important to determine how sophisticated the agents’ reasoning will be. There are two levels of sophistication for agents’ reasoning: Reactive and Deliberative. A reactive agent defines a set of actions that will be triggered in certain conditions. Once the conditions are satisfied, the reactive agent retrieves a pre-defined action. A deliberative agent, unlike a reactive agent, maintains its mental attitudes and commits to an action by predicting the effect of the action (Balch and Parker, 2002). In the definition of action commitment in Chapter 4 and later in the proposed process of awareness identification, PAM introduces a deliberative mechanism for agents’ reasoning. Therefore, here, I look at the inside of the agents designed in the previous stage and develop the deliberation process for the agents. Hence, the agents will be able to maintain their mental attitudes and, thereafter, depending on these attitudes, agents will execute an action.

The objective of this stage is to develop a method of the class `Agent` that runs by each tick of time. This requires the implementation of the inbuilt Repast interface, `Stapable`. This interface provides a public method called `step`. The method `step` is given in Figure 5-14.

The instance of the class `PAM_Model`, by each tick of time, iterates through all the agents and calls the `step` method that is defined in the class `Agent`. Therefore, the method `step` adverted to the class `Agent` identifies what the
agent needs to do in each tick of time. As such, the signature of the class `Agent` will change to `public class Agent implements Stepable`.

```java
// DELIBERATION
public void step()

receiveEvents();
Action act = PAM_Process ();
if (act!=null){
    /* there is an action to find the validity of relevant information. */
    currentSituation = act.destination;
    doneActions.add(new DoneAction (currentSituation, act));
    else
        act = actionCommitment();
    if (act != null){
        /* there is an action based on existing implicit knowledge. */
        currentSituation = act.destination;
        doneActions.add(new DoneAction (currentSituation, act));
        // else: nothing to do.

Figure 5-14 Method step: Deliberation

With each tick of time, the agent first receives the events. In order to do so, the agent checks all the events associated with the current situation. The agent considers those events that are also related to the world associated with the current situation. The agent checks the occurrence of the events. It receives those events that have already occurred. Then, the agent pushes them into the list of received events (see Figure 5-15).

In receiving an event, the agent looks at the events that have occurred in the environment. It filters the ones that do not belong to the agent’s current situation or do not belong to the world associated with the current situation.
private void receiveEvents()
    
    for (int i = 0 ; i < events.length; i++)
        for (int j = 0 ; j < events[i].situations.length; j++)
            for (int k = 0 ; k <currentSituation.world.events.length;
                 k++)
                if (currentSituation == events[i].situations[j] &&
                    currentSituation.world.events[k].name == events[i].name
                                      && events[i].occurred)
                    receivedEvents.add(new ReceivedEvent
                                        (currentSituation, events[i]));

Figure 5-15 Method receivedEvents

Figure 5-14 shows that after receiving events, the agent executes an action. In
order to find which action to execute, the agent first runs PAM_Process. Then,
if there is no action required to find the validity of the relevant information, as
proposed by PAM, the agent runs the action commitment based on Equation
4-24. Once the agent finds the action to execute, it changes the current situation
to the destination of the action. It then adds the action to the list of done actions.
Figure 5-16 shows the implementation of the three-step process of PAM. The
method for action commitment will be discussed later in this section.

private Action PAM_Process ()
    
    // step 1
    policyRuleAwareness ();
    // step 2
    informationAwareness ();
    // step 3
    return changeOfBehaviors();

Figure 5-16 Methods PAM Process for Awareness Identification and Action Commitment
The process of awareness identification, as it has been explained, involves three steps. These steps are implemented by the following methods: the method `policyRuleAwareness`, the method `informationAwareness` and the method `changeOfBehaviors`. In order to implement these methods, I need to implement the notation of implicit knowledge and awareness.

Implicit knowledge is defined by the satisfiability relation of the PAM framework. An agent implicitly knows anything that is valid. The method `implicitKnowledge` returns a boolean with different input arguments, which implement the agent’s implicit knowledge or validity of different propositions. Figure 5-17 depicts the implementation of a basic signature for the method `implicitKnowledge`. This method implements Equation 4-1 for satisfiability of a proposition in a particular situation. The method checks the agent’s interpreter and finds a true set associated with the proposition that is given in the input arguments. In the situation, if the agent’s set of valued variables is a subset of one of the true sets for the proposition, then the method returns true. This means that the agent in the given situation implicitly knows the proposition. Another similar signature for the method `implicitKnowledge` is `private boolean implicitKnowledge (Proposition prop)`, which shows the agent’s implicit knowledge of the given proposition in the current situation. This can be implemented by replacing the current situation with the input argument s, in Figure 5-17. In a similar implementation, the agent can implicitly know about
the negation of a proposition, i.e. \texttt{private boolean implicitKnowledge (String negation, Proposition prop, String modal, String temp)}.

Figure 5-17 shows the implementation of such a method for implicit knowledge of agent.

```java
/* IMPLICIT KNOWLEDGE ABOUT A PROPOSITION IN A SPECIFIC SITUATION.*/
private boolean implicitKnowledge (Proposition prop , Situation s)
  for (int i = 0 ; i < interpreter.trueSet.length; i++){
    if (interpreter.trueSet[i].proposition == prop)
      for (int j=0; j<interpreter.trueSet[i].tuples.length; j++)
        Enumeration e = interpreter.trueSet[i].tuples[j].keys();
        while (e.hasMoreElements())
          Variable var = (Variable) e.nextElement();
          DomainValue val = (DomainValue)
            interpreter.trueSet[i].tuples[j].get(var);
          //finding the valued variable in the situation
          boolean varSituation = false;
          for (int k = 0 ; k <valuedVariables.length ; k++)
            if (valuedVariables[k].situation == s &&
                valuedVariables[k].variable == var)
              varSituation = true;
          /* if I cannot find the valued variable in the situation or the assigned valued does not match the value of the proposition is false.*/
          if (!varSituation)
            return false;
        else
          for (int k = 0 ; k <valuedVariables.length ; k++)
            if (valuedVariables[k].situation == s &&
                valuedVariables[k].variable == var &&
                valuedVariables[k].value != val)
              return false;
  return true;
```

Figure 5-17 Method implicitKnowledge of a Proposition in a Situation
The method implicitKnowledge should also be able to validate a path formula. In this signature, I add two more input arguments to this method: modal and temp. The input modal can be "optional" or "inevitable". The input temp can be "next" or "eventually". This generates the four different types of path formulae explained in Section 4.1.2.

Figure 5-18 implements this signature. The implementation goes through the available paths; starting from the given situation and considering the modal and temp input arguments, it indicates the validity of the proposition. Due to space limitations, some parts of the implementation are removed from Figure 5-18. For the complete implementation of this method, readers should refer to Appendix 2.

Another similar signature of the method implicitKnowledge for considering path formulae is private boolean implicitKnowledge (Proposition prop1, Proposition prop2, String modal, String temp), which returns the validity of the proposition in a path starting from the current situation. This can be implemented by replacing the current situation with the input argument situation in Figure 5-18. Equation 4-13 defines the validity of a proposition in a path in a specific deadline. This can be implemented by a similar signature to Figure 5-18 with an extra check on the number of actions: i.e. private boolean implicitKnowledge (Proposition prop, String modal, int deadline, Situation
Development of PAM

situation) or private boolean implicitKnowledge

(Proposition prop, String modal, int deadline).

/* IMPLICIT KNOWLEDGE ABOUT A PROPOSITION IN A PATH STARTING FROM A SPECIFIC SITUATION.*/
private boolean implicitKnowledge (Proposition prop,
String modal, String temp, Situation situation)

if (temp == "next"){
/* TO DO: evaluating the proposition in the next time point from the current situation.*/
ArrayList <Situation> nextSituations =
  new ArrayList <Situation> ();
for (int i = 0; i <times.length ; i ++)
// TODO: finding the next time: times[i].tickIndex+1
// TODO: finding the next situations
/* they have to be in the situation list of agent as well as the current world.*/
/* there must be an action connecting the currentSituation to these situations.*/

if (modal == "optional")
/* TO DO: evaluating the proposition to see if it is valid in at least one situation from all the possible next situations */

if (modal == "inevitable")
/* TO DO: evaluating the proposition to see if it is valid in all the possible next situations */

if (temp == "eventually")
ArrayList<Situation> evenSituations = new
ArrayList<Situation>();
/*To Do: finding all the situations that may happen now or sometime in future. */
/* they have to be in the situation list of agent as well as the current world.*/
if (modal == "optional")
/* TO DO: if the proposition is valid at least in one of the possible situations, it returns true.*/
if (modal == "inevitable")
/* TO DO: if the proposition is valid in all the possible situations, it returns true.*/

// if temp is none of the defined choices, above.
return false;

Figure 5-18 Method implicitKnowledge of a proposition in a path starting from a given situation.
In addition to the validation of the above path formulae, Equation 4-14 defines *until*. Therefore, Figure 5-19 shows the implementation for validity of an *until*-composition of two propositions in a path starting from a given situation. The signature is validated by an "optional" or "inevitable" modal. Considering the modal, the method checks the validity of the first proposition until the situation in which the second proposition becomes valid. Due to space limitations, some parts of the implementation are removed from Figure 5-19. For the complete implementation of this method, readers should refer to Appendix 2.

```java
/* IMPLICIT KNOWLEDGE ABOUT AN UNTIL-COMPOSITION OF TWO
PROPOSITIONS IN A PATH STARTING FROM A GIVEN SITUATION.*/
private boolean implicitKnowledge (Proposition prop1,
Proposition prop2, String modal, String temp,
Situation situation)
if (temp == "until")
// possible situations
ArrayList<Situation> evenSituations = new
    ArrayList<Situation>();
    /* To DO: finding all the situations that may happen now or
    Some time in future.*/
    /* they have to be in the situation list of agent as well
    As the current world.*/
if (modal == "optional")
/* To DO: if the proposition2 is valid in one of the
possible situations and proposition 1 has been true from
the current situation until there, then the agent returns
true.*/
if (modal == "inevitable")
/* To DO: if for all the possible situations the
proposition 2 is valid some time in future and the
proposition one has been valid from the current
situation till there, then the agent returns true.*/
    // if temp is none of the defined choices, above.
return false;
```

Figure 5-19 Method implicitKnowledge of an until composition of two propositions in a path starting from a given situation.
Another signature of this method refers to validation of the *until*-formulae in a path starting from the current situation, i.e. `private boolean implicitKnowledge (Proposition prop1, Proposition prop2, String modal, String temp)`. This can be implemented by replacing the current situation instead of the input argument `situation` in Figure 5-19. I can also validate an *until*-formula with a deadline, i.e. `private boolean implicitKnowledge (Proposition prop1, Proposition prop2, String modal, String temp, int deadline)`. The definition of `until` with a deadline is given in Equation 4-15, which checks the number of actions in the branching-time model. Therefore, this only adds an additional check on the number of actions so that they are not greater than the deadline.

Another type of implicit knowledge is an agent’s validation of the execution of an action or occurrence of an event. Figure 5-20 shows the implementation of an agent’s implicit knowledge of a done action or a received event. These implementations are simple searches in the `doneActions` or `receivedEvents`. I can also replace situation `s` with the current situation, to validate the execution of an action or occurrence of an event in the current situation.

```java
/* IMPLICIT KNOWLEDGE ABOUT A DONE ACTION IN A SPECIFIC SITUATION.*/
private boolean implicitKnowledge (Action act, Situation s)
for (int i = 0 ; i < doneActions.size(); i++)
    if (doneActions.get(i).action.name == act.name)
        return true;
return false;
```
/* implicit knowledge about a received event in a specific situation.*/
public boolean implicitKnowledge (Event e, Situation s)
    for (int i = 0 ; i < receivedEvents.size(); i++)
    if (receivedEvents.get(i).event.name == e.name)
        return true;
    return false;

Figure 5-20 Method implicitKnowledge about a done action or a received event.

In addition to implicit knowledge, Equation 4-7 defines awareness. Figure 5-21 depicts implementation of the method awareness in the agent’s model. This implementation involves a basic search in the array list awareness. Replacing the current situation with s, the method also has a signature for awareness in the current situation.

// AWARENESS IN A SPECIFIC SITUATION.
private boolean awareness (Proposition prop, Situation s)
    for (int i = 0 ; i < awareness.size(); i++)
        if (awareness.get(i).proposition.name == prop.name)
            return true;
    return false;

Figure 5-21 Method awareness

Equation 4-8 specifies that an agent explicitly knows a proposition if it is aware of that and it has implicit knowledge about the proposition. Figure 5-22 shows the implementation of the method explicitKnowledge. This method can also have a signature for the agent’s explicit knowledge in the current situation by replacing the current situation with s in the input arguments of the method.

// EXPLICIT KNOWLEDGE IN A SPECIFIC SITUATION.
private boolean explicitKnowledge (Proposition prop, Situation s)
    if (implicitKnowledge(prop, s) && awareness (prop, s))
        return true;
    else
        return false;

Figure 5-22 Method explicitKnowledge
At this point of implementation, I am able to implement the PAM process. The first step is to identify the relevant policy rules or policy rule awareness. In Figure 5-23, the method `policyRuleAwareness` implements the first step, which includes the implementation of Equation 4-26 and Equation 4-25. This method searches in all the forbidding and requiring policy rules to find such forbidding and requiring policy rules that satisfy the conditions of the definition for policy-aware agents in Chapter 4. Therefore, if for a forbidding or requiring policy rule there exists a possible situation in which the policy rule condition is satisfied, the event has been received and the modality of the action referenced in the rule can be violated, then the agent becomes aware of that policy rule. Due to space limitations, some parts of the implementation are removed from Figure 5-23. For the complete implementation of this method, readers should refer to Appendix 3.

```java
//STEP1: POLICY RULE AWARENESS
private void policyRuleAwareness ()
// possible situations
ArrayList<Situation> evenSituations = new ArrayList<Situation>();
/* TO DO: finding all the situations that may happen now or Sometime in future.*/
/* they have to be in the situation list of agent as well As the current world.*/
for (int i = 0 ; i < forbiddingPolicyRules.length; i++)
/* TO DO: for all the forbidding policy rules, I should check the conditions given in the definition of policy-aware agent and become aware of the eligible ones*/
for (int i = 0 ; i < requiringPolicyRules.length; i++)
/* TO DO: for all the requiring policy rules, I should check the conditions given in the definition of policy-aware agent and become aware of the eligible ones.*/
```

Figure 5-23 Method policyRuleAwareness
According to the second step of the PAM process, in each situation the agent looks for those forbidding and requiring policy rules that are already in the awareness set of the agent. Then, the agent becomes aware of the condition referenced in the policy rule. Figure 5-24 shows that the agent in a similar implementation searches the `forbiddingPolicyRules` and `requiringPolicyRules`. Then, the agent simply adds the conditions referenced into the rules in the `awareness` array list and becomes aware of them. Due to space limitations, some parts of the implementation are removed from Figure 5-24. For the complete implementation of this method, readers should refer to Appendix 4.

```java
// STEP2: INFORMATION AWARENESS
private void informationAwareness ()
// possible situations
ArrayList<Situation> evenSituations = new ArrayList<Situation>();
/* TO DO: finding all the situations that may happen now or Sometime in future.*/
/* they have to be in the situation list of agent as well As the current world.*/
for (int i = 0 ; i < forbiddingPolicyRules.length; i++)
/* TO DO: for all the forbidding policy rules, in each possible situation the agent becomes aware of the conditions referenced to those forbidding policy rules that the agent is already aware of. */

for (int i = 0 ; i < requiringPolicyRules.length; i++)
/* TO DO: for all the requiring policy rules, in each possible situation the agent becomes aware of the conditions referenced to those requiring policy rules that the agent is already aware of. */
```

Figure 5-24 Method `informationAwareness`

According to the third step of the PAM process proposed in Chapter 4, the agent, being aware of a proposition, selects an action for a situation in which the
implicit knowledge about that proposition can be inferred in that situation or sometime in the future. In fact, it selects an action that makes such an implicit knowledge possible to achieve. Figure 5-25 shows the implementation of the methods changeOfBehaviors. In the PAM process, I start with the Volitional strategy. In this strategy, the agent looks for the first proposition in the awareness set and, if it is not possible to achieve such implicit knowledge of that proposition, it recursively calls the cautious strategy. In the cautious strategy, the agent finds all the actions that bring the agent to the implicit knowledge about propositions in the awareness set. If there is no action, it will return null; this sends the method step to the actionCommitment. If there is more than one action, then it recursively calls the hasty strategy, which choses the shortest path. Due to space limitations, some parts of the implementation are removed from Figure 5-25. For the complete implementation of this method, readers should refer to Appendix 5.

```java
// STEP3: CHANGE OF BEHAVIORS
private Action changeOfBehaviors() {
  return changeOfBehaviors("Volitional");
}

// CHANGE OF BEHAVIORS BASED ON THE AGENT'S STRATEGIES.
private Action changeOfBehaviors(String agentStrategy) {
  Action committedAction = null;
  // possible next situations
  ArrayList<Action> nextActions = new ArrayList<Action> ();

  //ToDo: finding the next possible actions.
  /* they have to be in the current situation as well as the world associated with the current situation. */
  if (agentStrategy == "Volitional") {
    /* with Volitional Strategy, the agent looks for the first
proposition in the awareness set and if it cannot find any suitable action to the future implicit knowledge about that proposition, it recursively calls the cautious strategy. */

```java
if (committedAction != null)
    return committedAction;
else
    return changeOfBehaviors("Cautious");
```

```java
if (agentStrategy == "Cautious")
/* with Cautious Strategy, the agent looks for all the propositions in the awareness set to find an action suitable for the future implicit knowledge about that proposition. If it cannot find an action for any of those propositions, it will return null. But, if it can find more than one action, it will recursively call the hasty strategy. */

if (num > 1)
    return changeOfBehaviors("Hasty");
return committedAction;
```

```java
//with Hasty Strategy
if (agentStrategy == "Hasty")
/* with Hasty Strategy, the agent looks for all the propositions in the awareness set to find an action suitable for bringing the agent into future implicit knowledge about that proposition in the shortest path possible. If it cannot find an action for any of them, it will return null. But, if it can find more than one action it will recursively call the hasty strategy. */

Figure 5-25 Methods changeOfBehaviors
```

According to Figure 5-14, if the PAM process does not return any action related to the relevant information, the agent calls actioncommitment. In Figure 5-26, the method actionCommitment is implemented according to Definition 6 of Chapter 4. The agent commits to an action that brings it to a situation in which the agent’s implicit knowledge or its negation will be inferable in that situation or sometime after that situation. Therefore, the phrase (implicitKnowledge(propositions[i],"inevitable","eventual" etc.)
(implcitKnowledge ("not", propositions[i], "inevitable", "eventually")) checks the agent’s implicit knowledge about the present or sometime in the future. Then, the agent selects an action among all the possible options available to the agent, which brings the agent to a situation where implicitKnowledge (propositions[i] "inevitable", "eventually", actions[j].destination) is inferable. It also checks that the agent currently does not have that implicit knowledge of the proposition, i.e. !implicitKnowledge (propositions[i]).

// ACTION COMMITMENT
private Action actionCommitment ()

for (int i = 0 ; i<propositions.length; i++)
if ((implicitKnowledge (propositions[i], "inevitable", "eventually") ||
     (implicitKnowledge ("not", propositions[i], "inevitable", "eventually")))

// finding the next possible actions
for (int j = 0 ; j< actions.length; j++)
if (actions[j].source == currentSituation)

/*the next possible situation has to infer the same implicit knowledge. */
if (implicitKnowledge(propositions[i], "inevitable", "eventually", actions[j].destination))
if (!implicitKnowledge(propositions[i]))
    return actions[j];
return null;

Figure 5-26 Method actionCommitment

Going back to the class PAM_Model in Figure 5-3, based on the design, if I create instances of the above classes in the method buildModel, in each tick of time, the model executes the method step for every agent. This brings the system into run-time.
One merit of PAM is the dynamic nature of the process for becoming aware of information. Agents, considering the policy rules, become aware of the conditions of the policy rules; however, in case the set of rules change for any reason, the run-time process of policy awareness will be still able to calculate the awareness from the new set of rules, by only taking the steps given for policy awareness. As an alternative the policy rules can be described in a file, which can be changed easily. The format and syntax of this file it depends on the implementation issues and is out of the scope of PAM. Then, for each agent, the method step of the class Agent runs by every time tick and goes through the three steps for policy awareness. This enables the agent to be able to refresh its awareness at each time tick based on the new policy rules, if they changed.

5.2 Analysis versus Design and Implementation

Despite the strong endorsement of the analysis and design concepts, there are many variations on what belongs to analysis and what should be done in design. In the development guidelines, there is a tremendous difference between the content of analysis and design entities. Similar to CoMEN, the application development begins from scenario analysis. The scenario analysis provides an abstract specification of the system to the design and implementation phase where I can create a running application. Here, following PAM development guidelines, I show how analysis and design concepts are related.
In the analysis phase, the entities given in Figure 5-1 are generally viewed as conceptual tools, while in the design and implementation phase, the classes given in Figure 5-2 are concrete classes in the programming language used for implementation. I propose using Table 5-1 to map concepts in the scenario analysis of the system to the classes in the design and implementation phase.

Table 5-1 depicts the mapping between the analysis phase and the system design and implementation phase in PAM. The objective of the scenario analysis phase is to study the different scenarios in the system and find out what needs to be developed. The output of this phase is a set of abstract specifications. In the system design and implementation phase the goal is to indicate how the system fulfils the specifications coming out of the analysis. In the second phase, I design and implement a system that runs over that time and renders the abstract specifications. Therefore, I need to characterise the system over that time. Hence, the key to transform analysis to design and implementation is incorporating time into the analysis model and to characterise the system over time, based on the abstract specification of the analysis phase. This shows “how” to develop “what” needs to be developed. In order to incorporate time into the design and implementation, Figure 5-2 proposes to use the class Time.

While specifying the system in different time points, the class Situation can be instantiated. One can also say that the class Time has a one-to-many association with the class Situation. This means that each situation belongs to only one
time point, while each time point can evolve more than one situation. This is what is explained in the first characteristic discussed in Chapter 4. Hence, the two classes Time and Situation are critical in transforming the analysis model to the design and implementation of the running system. In other words, the design and implementation phase characterises the system in different situations over time. For example, inputs and outputs in the scenario analysis phase are defined as valid information. Therefore, they relate to a particular situation where they are implemented by the valued variables. The valued variables assign the domain values to the variables; therefore, the interpreter can validate the related proposition.

5.3 Development of the Space Shuttle Columbia Disaster using PAM

In this section, I demonstrate the above aspects of the development in PAM by the case of the space shuttle Columbia disaster. Section 5.3.1 presents the scenario analysis phase, and Section 5.3.2 presents the system design and implementation phase.

5.3.1 Scenario Analysis of the Space Shuttle Columbia

In this phase, I study the space shuttle Columbia disaster. This study is based on the presented concepts in Figure 5-1, for analysis of scenarios using PAM. Figure 5-27 depicts the instances of these concepts for the space shuttle Columbia
disaster. Due to space limitations and in order to increase readability of the figure, I have simplified the model to deliver the main objective of the section.

Table 5-1 Mapping Scenario Analysis to System Design and Implementation

<table>
<thead>
<tr>
<th>Scenario Analysis</th>
<th>Role</th>
<th>Procedure</th>
<th>Environment change</th>
<th>Scenario</th>
<th>Information</th>
<th>Policy</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td></td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DomainValue</td>
<td></td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ValuedVariable</td>
<td></td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interpreter</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TrueSet</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proposition</td>
<td></td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PolicyRule</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PermittingPolicyRule</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RequiringPolicyRule</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ForbiddenPolicyRule</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeterringPolicyRule</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agent</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Situation</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Time</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>World</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReceivedEvent</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoneAction</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Awareness</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5.3.1.1 Overall System Study

Figure 5-27 presents the scenario analysis for the space shuttle Columbia disaster. This scenario is initiated by the report coming from the shuttle that reveals it is experiencing some shakes. There are three roles involved in the scenario: the NASA management, the Engineering team and DoD.

5.3.1.2 Process Study

Looking at what I discussed in Chapter 1, I have found three different possible procedures in this scenario: (I) the NASA management team simply considers the shakes as caused by the turnaround effect; (II) the NASA management team regards the TPS damage to be relevant information and finds out there is no damage in the TPS, and therefore considers the shakes as merely caused by the
Development of PAM

turnaround effect; or (III) the NASA management team regards the TPS damage to be relevant information and finds that there is damage to the TPS, and therefore operates the spacewalk rescue. Figure 5-27 presents these procedures.

5.3.1.3 Logical Components Identification

Figure 5-2 presents information involved in the space shuttle Columbia disaster. In this stage, I relate the identified information to the inputs and the outputs of the procedures from the previous stage. In this stage, I also identify the available policies. Although there are various policies available, due to the limitation of the Figure 5-2, I only present a particular policy applicable to my discussion. This policy states that if an aircraft experiences some shakes and there is damage to the TPS, the spacewalk procedure must be operated.

5.3.2 System Design and Implementation in Space Shuttle Columbia

In this section, I describe the design and implementation of a system for the application of PAM in the space shuttle Columbia disaster. Using Table 5-1, I map the scenario analysis to the system design and implementation phase.

Figure 5-28 depicts the object model of the system based on the design and implementation concepts given in Figure 5-2. Due to space limitations, Figure 5-28 has been simplified.
As I explained in the previous section, I should model the system according to each agent’s point of view and in all possible situations. However, in order to increase the readability of the model, Figure 5-28 only presents the NASA management agent’s model of the system in the situation $s_1$, i.e. $(M_{NASA \text{ Management}}, s_1)$. 

Figure 5-28 Space Shuttle Columbia Disaster – Simplified Object Model in the Situation $s_1$. 

160 | Development of PAM
In different situations, the domain values shown in Figure 5-28 can be assigned differently to the variables. The model shows that in the situation s1, the valued variable object, valuedVariable1, assigns d2 to the variable v0, which reports the shakes in the shuttle. This is explained in Table 4-1, where I am applying the formalism of the PAM framework to this scenario, i.e. \( \pi^{s_1} = d_2 \). Having said this, the agent, while running the method step, validates the prop0 and implicitly knows the proposition. This is because the agent has an interpreter with the true set i0.

As I explained above, in Figure 5-28 the current situation of the NASA management agent is s1. The agent also includes a world that presents the scenario of the space shuttle Columbia disaster in the time point t1, however, there more time points that can been seen as time ticks in Figure 1-1. The time point t1 associates with the decision of the NASA management for announcing the shakes as turnaround effect, or finding out whether if there is any damage on TPS which causes the shakes (see Figure 1-1). Figure 5-28 avoiding complexity presents the system in only time point t1, where I can demonstrate the contribution of PAM. In Figure 5-28, the agent has different action objects, each of which is based on a source situation as well as a destination situation. In s1, according to Figure 1-1, the action for realising the shake is completed. The agent also includes the event that reports the shakes in the shuttle, which is already
received in the situation s1. Therefore, taking the policy rule into account, the
management agent first, gets aware of the rule, second, it becomes aware of the
TPS damage as the condition or the rule, and third, it requests the imaging from
DoD.

5.4 Summary and Outlook

In brief, CoMEN provides methodological guidelines to develop applications in
CSCW. CoMEN proposes two main phases to facilitate development of
applications for incorporating awareness obtainment: (I) Scenario Analysis and
(II) System Design and Implementation.

Although CoMEN pays great attention to the scenario analysis phase, it lacks
both (I) a definitive method for identifying awareness and (II) definitive stages
for the design and implementation phase. On the other hand, PAM addresses the
problem of awareness identification by requiring a methodological basis for
development. In this chapter, I borrowed the main ideas from CoMEN for the
development of applications based on PAM, and I then provided the details of
each development phase.

The objective of the scenario analysis is to study the scenarios occurring in the
system. The chapter explains this phase in three stages: overall system study,
process study and logical components identification. Figure 5-1 presents the
concepts involved in scenario analysis and their associations. Then, the chapter
Development of PAM proposes the concepts required in the design and implementation of systems. This phase has two stages: development of the PAM framework and the development of agents' deliberations. The objective of the first stage is to design and implement the formalism given in the PAM framework in Chapter 4. The goal of the second stage is to design and implement how agents, by each time tick, change their mental attitudes and execute an action. This includes the three-step process of PAM for awareness identification and the concept of action commitment.

In order to avoid discontinuities between the two proposed phases for “what” to develop and “how” to develop them, Table 5-1 gives a mapping structure. This transforms the concepts of scenario analysis to the classes involved in the system design and implementation phase. The main idea in this mapping is the characterising of “what” comes out of the scenario analysis in order to develop a way to show “how” it will be run over the specified time. Therefore, Time and Situation are two key classes to specify the system in each time point.

Finally, the development guidelines are demonstrated via its application in the space shuttle Columbia disaster. The chapter has gone through the proposed phases and has given the object model used for implementation.

In the next chapter, I evaluate the use of PAM for identifying awareness based on existing policy rules. First, I prove the concept by studying the other real
scenarios in addition to the space shuttle Columbia disaster. Then, I analyse the efficacy and cost-efficiency of PAM by triangulation of two simulations: (I) simulation of hypothetical examples and (II) simulation of procedures found in a wireless communication system at a hospital.
This chapter aims at providing ex ant and ex post evaluation of PAM (Pries-Heje et al., 2008). Ex ant evaluation aims at Declining whether or not the proposed concepts are relatively applicable in real life. In this thesis, for the ex ant perspective, proofs of concepts are presented by application of PAM in real life exemplars. Ex post evaluation involves operating the outputs of the research. In the ex post perspective a certain set of outputs are evaluated by executing PAM to see if PAM is an effective and cost-efficient method. Le Coze (2008) emphasises the potential of bias during evaluation and recommends the use of both of ex ant and ex post evaluation. Therefore, this chapter presents the evaluation of PAM in ex ant and ex post perspectives.

In ex ant evaluation, in order to demonstrate the applicability of PAM in real life, I chose to prove the concepts via three exemplars: (I) the space shuttle Columbia disaster, (II) the Casa Grande hazardous-materials rail incident, AZ, and (III) Abridged versions of this chapter can be found in:

- Policy-Based Awareness: Implications in Rehabilitation Video Games, In the Proceedings of 44th Hawaiian International Conference on System Sciences (HICSS44) (Ranked A by The Excellence in Research for Australia), 2011.
rehabilitation video games. These three exemplars demonstrate the significance of motivation as well as the applicability of the proposed solution. The first one has been already discussed. The other two exemplars are presented in this chapter.

In ex post evaluation, I am concerned with the execution of PAM. Yang & Padmanabhan (2005) distinguish approaches in ex post evaluation by methodologies that use real settings and those that do not. However, Carlsson (2010) encourages triangulating both of these methods in order to increase the reliability of the outcomes. Therefore, in the ex post evaluation of PAM, I use a simulation of hypothetical examples as well as a simulation of wireless communication procedures in St. Olavs Hospital, Trondheim University Hospital, Norway. In order to design these two simulation studies, I benefited from guidelines given by Akhtman & Hanzo (2010). I chose size, branching factor and number of policy rules as configuration parameters. Success rate and cost were output criteria of these studies. By varying the configuration parameters in the inputs of the simulations, I measured output criteria. By having success rate and cost as outputs, I calculate efficacy and cost-efficiency of using PAM.

The rest of this chapter is organised in the following way: Section 6.1 presents the proofs of concepts as ex ant evaluation. Section 6.2 presents the hypothetical simulations as well as simulation of wireless communication procedures at St. Olavs Hospital as instances of ex post evaluation. Section 6.3 concludes the chapter and presents a brief overview of the next chapter.
6.1 Proofs of Concepts

Proofs of concepts through exemplars are used to illustrate to which kinds of problem genres of research can be applied. In this study, three exemplars are selected to demonstrate the applicability of PAM in real life. The first exemplar, the space shuttle Columbia disaster, has already demonstrated the motivation and the contributions of this research. In this section, I present two more exemplars: rehabilitation video games (Talaei-Khoei, Ray, and Parameshwaran, 2011; Smith et al., 2011, 2009) and the Casa Grande hazardous-materials rail incident in 1983 (Talaei-Khoei, Bleistein, et al., 2010; Talaei-Khoei, Ray, et al., 2010).

6.1.1 Rehabilitation Video Games

Falls and fall-related injuries are common and often devastating problems associated with ageing, which cause a tremendous amount of morbidity and mortality. Most of these falls are related to the muscle-weakness of the elderly while moving in steps. Researchers in rehabilitation have been investigating different rehabilitation exercises in step movements to prevent falls. The Physiological Profile Approach (PPA) (Lord et al., 2003) deals with impairments, irrespective of physiological causes, due to falls in aged people. The key idea in PPA is that the maintenance of balance depends on the interaction of multiple sensory, motor and integrative systems. One of the major concerns in PPA is step training treatments involving muscle force and reaction time. Muscle force practices force production of three lower-extremity muscle groups, namely, knee
flexors, extensors, and ankle dorsiflexors. These muscles are important when performing daily tasks such as rising from a chair or walking. Reaction time relates to the required minimum response time to move muscles.

Rehabilitation by its nature is repetitive, and repetition makes the treatments dull and reduces the patients’ motivation. Traditional rehabilitation is done one-by-one; that is, one activity at any time. Thus, the cost of treatments tends to be high, and therefore the rehabilitation treatments are difficult in remote areas. Rehabilitation video games have been recently considered as a promising approach in rehabilitation treatments.

In parallel with advances in functional recovery, engineering professions have witnessed recent developments in mobile monitoring and video gaming technologies. In recent years, interactive video games, in which an individual interacts with the game by moving her limbs or whole body, have found applications in the field of home-based rehabilitation medicine. The engaging nature of games increases adherence to rehabilitation training exercises. The dull and repetitive nature of rehabilitation exercises is transformed into entertaining and interactive games that motivate the elderly (Tinetti et al., 1999). Home-based rehabilitation, however, presents supervision problems. Smith et al (Smith et al., 2009, 2011) propose an SMS-based monitoring system for rehabilitation practices using video games. In rehabilitation practices, an Occupational Therapist (OT), focusing on the abilities of a patient observed from her previous activities,
identifies the relevant, purposeful tests. This requires cooperation between the OT and the patient. However, by their nature, home-based rehabilitation video games engender situations in which cooperation may require unknown information to decide which treatment should be granted to a patient. The monitoring system designed for this kind of games automates the process of sending the rehabilitation game results to the OT. In practice, the OT is rarely available to answer a SMS\(^1\), and sporadic contacts between the OT and the patient result in inadequate cooperation. Enhancing cooperation requires the OT identifying the awareness of patients’ conditions when choosing an appropriate game.

Figure 6-1 depicts the rehabilitation process. It often takes place in four steps (I) Referral: The GP Doctor refers the patient to the Occupation Therapist and refers the patient’s problem to the OT. (II) Planning: The OT gives a rehabilitation plan to the patient. (III) Progress Report: The patient reports her/his progress to the OT. (IV) Discharging: If the OT is sure that the patient will be able to do her/his daily tasks, the OT discharges the patient (Otherwise the rehabilitation process goes back to Step 2). This rehabilitation process will work, as described above, provided that the patient and the OT are located physically at the same place. However, in home-based rehabilitation games, where the patient is typically at

\(^1\) Short Message Service
home, it is unlikely that the communication between the patient and the OT will take place regularly and promptly.

Smith et al. (2009, 2011) propose the application of mobile technologies for monitoring the patient when she or he plays the rehabilitation game at home. This is an agent-based system meant for efficient and effective cooperation between the patient and the OT by communicating the patient's progress promptly. It also provides the OT's feedback to the patient. Figure 6-2 presents the mobile monitoring system for home-based rehabilitation using video games. It uses two types of software agents: (I) Agents that are implemented on the game set running at the patient's home or on the OT's computer at clinics; (II) Agents that are implemented on the patient's mobile phone.
The interaction sequences are as follows: (I) the doctor refers the patient to the OT, informing them of the patient’s problem. (II) The OT recommends a game, taking the patient’s condition into account. (III) The OT agent sends an SMS containing the game details to the patient agent. (IV) The patient agent displays the game details to the patient. (V) The patient selects the game that the OT wants her/him to play and starts playing it. (VI) The game agent sends the score, via bluetooth, to the patient agent; that is, a software agent assisting the patient. The score is calculated based on the z-Score system, as explained in PPA (Lord et al., 2003). (VII) The patient agent sends an SMS containing the z-score to the OT agent; that is, a software agent assisting the OT. (VIII) The OT queries the OT agent for the patient’s z-score. (IX) The OT agent displays the z-score to the OT. This score shows the fall risk of the patient. (X) If the z-score is satisfactory, and the OT can make sure that the patient is not at fall risk, then the OT discharges the patient and sends a STOP command to the OT’s agent; otherwise, the OT repeats Step 2. (XI) The OT agent sends the STOP SMS to the patient agent.
(XII) The patient agent displays the STOP SMS to the patient, and the patient terminates her/his game.

In practice, since the OT is rarely available to participate in the dialogue above, automating the process – minimising the dependency on the OT – will improve the overall rehabilitation process. However, automating the process is faced with the following problem.

6.1.1.1 Problem

Suppose that, in step 2 above the OT recommends that the patient plays a game that is related to reaction time rehabilitation exercises, and in step 7, the score shows that the patient is at the fall risk. In such a situation, there often exist two different possibilities (Lord et al., 2003): (I) The problem of reaction time (that is, how the reaction time values from a given game are satisfactory) has not been solved satisfactorily, so the patient still needs more practice and must continue to play the game. (II) The problem of lower-extremity muscles, which requires granting a new game for muscle force treatments.

If the problem is because of lower-extremity muscles and the doctor recommends continuing the reaction time treatment, then the patient will be injured by too much stress on her lower-extremity muscles (Lord et al., 2003). Therefore, although the OT agent; that is a software agent assisting the OT, does not know whether the patient has a problem in her/his lower-extremity muscles,
the OT agent must recognise the relevance of this information, i.e. awareness. Once the OT agent recognises the relevance of the information (that the patient has pain in her/his lower-extremity muscles), it grants *force muscle game* to see if there is actually a problem in these muscles. If the result is satisfactory, it then asks the patient to continue the game for reaction time treatments. This procedure has been borrowed from Lord et al. (2003).

6.1.1.2 Application of PAM in Rehabilitation Video Games

Here, I enhance the cooperation between the OT and the patient by implementing a policy-aware intelligent agent (the OT agent) to assist the OT to identify her/his awareness about the patient’s rehabilitation problem.

Guidelines are available in PPA (Lord et al., 2003) that can be used to provide policy awareness in software agents. A policy rule states that if the patient fails in reaction-time treatments, while she has problems with lower-extremity muscles, continuing response time treatments is forbidden. Thus, when the OT observes the failure of patient reaction-time, she or he must realise whether the information (that the patient has a problem with lower-extremity muscles or not) is relevant to the situation. The OT should grant a game appropriate to force muscle treatment and find out if the patient has a problem with her lower-extremity muscles. If the patient has a problem with these groups of muscles, then the OT is forbidden to grant reaction-time treatments. As such, I need a method to identify the relevance of information in each step of the treatment. I
propose to apply PAM in order to identify awareness. I show that PAM can identify the relevance of the problem of lower-extremity muscles in the treatment procedure that relies on rehabilitation video games.

![Diagram of failure in reaction-time game with problem in lower-extremity muscles](image)

**Figure 6-3** Failure in reaction-time game with problem in lower-extremity muscles

By applying PAM, the OT agent takes the following steps to overcome the problem:

1. **Recognise the relevance of policy rules:** In Figure 6-3, at situation $s_1$, the OT agent has already received an event for the failure of the patient in
the reaction-time game. The OT agent is suffering from a problem of omniscience, in the sense that it does not know whether the patient has a problem in her lower-extremity muscles or not. If there is a problem with these muscles, then recommending the reaction-time game would violate the forbidden PPA policy rule given above. There is a possibility of violating the policy rule, i.e. The OT agent takes this policy rule as a relevant one and identifies the awareness for the policy rule. Therefore, the OT agent takes this policy rule as a relevant one and identifies the awareness for the policy rule.

II. **Recognise relevant information in the relevant policy rule:** In Figure 6-3, at situation $s_1$, the OT agent has received an event for the failure of the patient in the reaction-time game. The agent is already aware of the policy rule saying that if the OT receives an SMS showing a problem with the patient in the reaction time (z-score is less than 0 – that is, the patient is suffering from a problem in her lower-extremity muscles), then recommending the reaction-time game is forbidden. Therefore, the OT agent should first find out the condition of the policy rule as the relevant proposition, which is the patient is suffering from a problem in her lower-extremity muscles. Therefore, the OT agent requires finding out whether the patient is suffering from a problem in her lower-extremity muscles.
muscles or not, as it is relevant information of which the OT agent is required to be aware.

III. Change of Behaviours based on Recognised Relevant Information: The OT agent, being aware of the patient’s problem in lower-extremity muscles, should select an action which effects the situation in which the agent knows whether the patient is actually suffering from this problem or not. As a result, the agent’s selected action would be recommending the muscle force game to see if there is any problem in these muscles (see Figure 6-3).

Hence, this exemplar shows that in rehabilitation video games, supportive software agents can contribute to an effective choice of treatment. The dynamic nature of rehabilitation treatments, in which situations can change dramatically, makes recognition of relevance of information difficult, particularly when cooperation between roles is inadequate. In such situations, policy rules can be used as an alternative to identify awareness and recognise the relevance of information.

6.1.2 Casa Grande Hazardous-Materials Rail Incident

In 1983, a train arrived in Casa Grande, Arizona, with burning hazardous materials in its cargo. The train engineer did not perceive any fire until it became
serious. Shortly afterwards, with smoke coming from one of its cars, the train stopped near the city centre of the town (Charles, 1988).

Although the reaction of firefighters was fast, no information was immediately available about the threat of burning hazardous materials in the cargo. The train company was not informed until 20 minutes after the fire was discovered, and even then the cargo information was incomplete.

Firefighters began to put out the fire according to the normal procedures for local firefighting. After 40 minutes, several firefighters became incapacitated. It was then that the emergency team realised the full danger, and the fire department advised the police to evacuate the area. Since only limited information on toxicity of the smoke was available, the emergency team did not know to what extent the hazardous material would threaten human health. The discussion of how to evacuate the area took place 30 minutes after the original discovery of the fire. As a result, local residents were not asked to evacuate until the situation became dangerous and the area was inundated with smoke. At the time there were six hazardous-materials handling teams in Arizona, none of whom were contacted. The hazardous-materials coordinator of the state also did not arrive on the scene on time.
6.1.2.1 Problem

In the Casa Grande incident, the emergency response operator did not know that the materials in the cargo were hazardous yet asked fire firefighters to follow the normal procedure for fighting local fires. Being bombarded with irrelevant or loosely relevant information about different possible situations, the operator could not realise that she or he should be aware of the cargo information to decide on the correct fire suppressing procedure. However, at the time there was a policy guideline available for the operator, stating that in transportation fire incidents, if there are hazardous materials involved, the jurisdictional hazardous-materials coordinator should be contacted (Yuan and Detlor, 2005).

6.1.2.2 Application of PAM in Casa Grande Rail Incident

Yuan and Detlor (2005) quote the Casa Grande incident and state that identifying the relevant information is a key to an efficient incident response. In the Casa Grande incident, of course, the emergency response operator asked firefighters to follow the normal procedure for local fires because she or he did not know that the materials in the cargo were hazardous. If the operator had contacted the rail company, she or he would have realised the threat and would have contacted the jurisdictional hazardous-materials coordinator. The coordinator would have sent one of the hazardous-materials handling teams to the scene to control the incident appropriately. Yuan and Detlor encourage researchers to work on the
methods that help for filtering irrelevant information. This is the main contribution of PAM.

Here, I propose the use of PAM to identify awareness based on the existing policy rules. For this to happen, I employ software agents that assist the emergency operator to identify awareness. By applying PAM to the Casa Grande rail incident, the emergency operator takes the following steps to identify the relevance of the fact of whether the cargo has hazardous materials or not:

I. **Recognise relevance of policy rules:** In Figure 6-4, at situation $s_1$, the emergency response operator agent has received a fire call from the train engineer. In this situation the agent has two choices: (I) asking for the cargo information from the rail company to make sure there is no hazardous material involved or (II) directly notifies the fire department to send the fire fighters into the scene. Looking at the policy stating in transportation fire incidents, if there are hazardous materials involved, the jurisdictional hazardous-materials coordinator should be contacted, and taking the branching-time model of the world given in Figure 6-4 into consideration, the emergency response operator agent can see the possibility for violation of the policy rule in the future. The world shows that if the agent simply notifies the fire department, it is possible that some hazardous materials may be involved. That would violate the policy
rule above. As a result, the agent takes the policy rule as a relevant one and is made aware of its content.

II. Recognise relevance of information in the relevant policy rules: In Figure 6-4, at situation $s_1$, the emergency response operator agent, being aware of the policy rule, takes the condition of the rule i.e. presentence of hazardous materials in the cargo relevant information and becomes aware of that.

III. Change of Behaviours Based on Recognised Relevant Information: As I mentioned above, the emergency response operator agent has two choices. The agent, while it is aware of information about the existence of hazardous materials in the cargo, asks for cargo information from the rail company i.e. selected action. This action brings the agent into the situation that has a future with implicit knowledge about the existence of hazardous materials in the cargo. Therefore, by choosing this action, the agent will acquire the implicit knowledge about what it is aware of.
In the Casa Grande rail incident, the emergency operator did not have sufficient awareness to identify the relevance of hazardous materials to the situation. Therefore, she or he did not ask about the cargo information from the rail company. This led to incapacitation of several fire fighters. I argue that the use of PAM to identify the relevance of information in such situations is beneficial.

6.2 Simulations

The term “simulation” can be defined in numerous ways depending on the area of research. Here, I subscribe to Raczynski’s (2006) definition, stating that a simulation is performing computer experiments on a proposed phenomenon to
evaluate its abstraction. The simulations presented in this section aim at evaluating PAM in terms of efficacy and cost-efficiency. In this section, I evaluate PAM by two types of simulations: (I) hypothetical simulation and (II) simulation based on the wireless communication procedures at St. Olavs Hospital, Trondheim University Hospital, Norway. I also discuss the simulation outcomes based on the theoretical support.

6.2.1 Methodology

As I discussed above, the objective of the simulations in this study is to evaluate PAM. For this to happen, I should transform the proposed phenomenon of PAM into a running computer program that can be measured in terms of pre-defined criteria. Sometimes, I am interested to see how these pre-defined criteria vary by changing the particular inputs of the program, called configuration parameters. One of the methods to change the configuration parameters is generating hypothetical examples (Guizzardi et al., 2002). A drawback of this method is suspicious applicability of the generated configuration parameters, which can be fixed by checking the results based on another simulation that takes its configuration parameters from a real environment. Although, the second simulation, due to input limitations, may lack clarity, it can be used as corroboration of the results for hypothetical simulations (see Figure 6-5).
Hence, in this study, in order to ensure the reliability of the results, I triangulate the results of hypothetical simulations with the results of simulations based on the case study of the wireless communication procedures in St. Olavs Hospital. Each of these two simulations includes two experiments: (I) an experiment in applying PAM and (II) an experiment on an awareness-free method, which is implemented according to the basic Kripkean structure (1963) explained in Section 2.2.2. Then, I compare the results to conclude the outcomes (see Figure 6-5). In both of these simulations, I measure the same output criteria and I vary the same configuration parameters. In what follows, I explain the criteria that I
measure in the outputs of the experiments and I discuss configuration parameters that I vary to see the change of the outputs (see Figure 6-5).

6.2.1.1 Configuration Parameters

By varying each of the parameters in Table 6-1, the conditions of the branching-time models can be simulated. As it is shown in Figure 6-5, this is a comparative study. I compare the application of PAM against an awareness-free method. The objective is to observe the efficacy and cost-efficiency by varying each of the configuration parameters. The first two parameters are borrowed from Taibi (2008), and due to the policy related nature of PAM, I also added “availability of policy rules” to the configuration parameters.

Table 6-1 Configuration Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>The number of situations.</td>
</tr>
<tr>
<td>Branching Factor</td>
<td>The arrangement of the situations in the branching-time model, meaning how flat the branching-time model of a world looks. I present the factor by the ratio of size to the number of time points. Therefore, the more horizontal branching-time models have bigger branching factors.</td>
</tr>
<tr>
<td>Availability of policy Rules</td>
<td>The number of the policy rules.</td>
</tr>
</tbody>
</table>

6.2.1.2 Output Criteria

In my simulations, I chose to measure success rate and cost as output criteria. Then, I calculated efficacy and cost-efficiency using the obtained success rate and cost. These factors are borrowed from (Akhtman and Hanzo, 2010). In what follows, I define success rate, cost, efficacy and cost-efficiency.
Success rate \( (S) \) is defined as the percentage amount that agents achieve a specific pre-defined situation in the given branching-time model. Cost \( (C) \) is defined as the number of situations that the agent navigates to arrive at the final situation. According to the definition of actions in Chapter 4, obviously the number of navigated situations is directly related to the number of actions. These two parameters were being measured as outputs of the system under each experiment. Based on these two parameters, I calculated efficacy and cost-efficiency.

Efficacy is a comparative measure that characterises the correspondence between the success rates obtained by using PAM against those of an awareness-free method. Efficacy \( (\zeta) \) is defined as the ratio between attained success rate in PAM and the success rate in the awareness-free method (see Equation 6-1).

\[
\zeta = \frac{S_{PAM}}{S_{Awareness-free}} \tag{Equation 6-1}
\]

In order to compare PAM and the awareness-free method in terms of how much success rate increases according to a particular amount of spending cost, I define cost-efficiency. Cost-efficiency \( (\varepsilon) \) is defined as an extra success rate that I obtain by comparing differences in cost (see Equation 6-2). By increasing the success rate and decreasing the cost, the cost-efficiency of the system will increase.
Evaluation

\[ \varepsilon = \frac{S_{PAM} - S_{Awareness-free}}{C_{PAM} - C_{Awareness-free}} \]  

Equation 6-2

Readers wishing more details about efficacy and cost-efficiency should see Akhtman and Hanzo (2010).

In the following sections, I present the results of the success rate, cost, efficacy and cost-efficiency for varying the configuration parameters in two simulations: a hypothetical simulation and a simulation of the wireless communication procedures (See Figure 6-5).

6.2.2 Hypothetical Simulations

In this simulation, the objective is to see the changes of output criteria by varying the configuration parameters. Therefore, In order to change the configuration parameters in a desired way, I ran a java program to randomly generate 1000 different branching-time models that include situations and actions. Attached to each branching-time model, the Java program generated a randomized event. The program also generated 1000 propositions, which were selected randomly for each branching-time model and distributed randomly as valid propositions to each situation in the model. In each branching-time model, the Java program selected a situation as a successful one where the agent is desired to reach there. These selections were also randomized. Then, the program generated 200 policy rules based on the set of all situations, event, actions and propositions. For each random generation, I used Java random function with a different random seed.
In experiments on size, branching factor and availability of policy rules, I gave these inputs to both PAM and the awareness-free methods. Then, I compared the results. I ran the experiments on a computer with 2.33 GHZ Intel dual-core CPU and 2 GB of RAM, which had a professional version of Windows XP. In what follows, I explain the results of these experiments.

6.2.2.1 Results

In this section, I present the results of my hypothetical simulations. In each experiment, I chose a configuration parameter and changed it to see the effect on success rate, cost, efficacy and cost-efficiency of PAM. This gives me a picture on how and when to use PAM. In the following, I present the results in terms of the aforementioned configuration parameters.

Size: The objective of this experiment was to see how the size of the branching-time model influences the efficacy and cost-efficiency of PAM. I measured the size of the branching-time models by the number of their situations. In my experiment, the size was in the range of 3 to 250. I categorised the branching-time models based on their size and put the models with the same size in the same group. I repeated the experiment for each group 100 times. In each time, I randomly selected one of the branching-time models involved in the related group. I ran the system once with PAM and once with the awareness-free method. After 100 times, I counted the number of times that the agent was
successful, that is, had reached the pre-defined successful situation, which was recorded as success rate.

Figure 6-6 shows the change of success rate by varying the size of the branching-time models. The experiment shows that increasing the size of the model drops the success rate in both PAM and the awareness-free method. Figure 6-6 shows that the PAM curve stays above the curve of the awareness-free method, which illustrates the positive effect of using awareness on success rate. On increasing the size of branching-time models, the curves diverge from each other. This means that although increasing the number of the situations decreases the success rate, the success of PAM, compared to the awareness-free method, drops less.

Figure 6-7 shows the growth of cost by increasing the size of the branching-time model. The experiment reveals that increasing the number of situations in branching-time models increases the average number of the situations that an agent passes in each experiment. On increasing the size of branching-time models, cost, by using PAM, grows in a higher rate than the awareness-free method.
Figure 6-6 Effect of Size on Success Rate – Hypothetical Simulation

Figure 6-7 Effect of Size on Cost – Hypothetical Simulation

Figure 6-8 shows the efficacy of updating the awareness-free method to PAM. The figure depicts the growth of efficacy by increasing the number of
situations involved in the branching-time models. This means that PAM becomes more effective in larger branching-time models. Figure 6-8 also reveals that the correspondence between the efficacy of PAM and the size of the branching-time models is not a linear relationship. The efficacy of PAM grows at a higher rate in larger models.

Figure 6-8 Effect of Size on Efficacy – Hypothetical Simulation

Figure 6-9 shows that by increasing the size of the branching-time model, the cost-efficiency increases. This means that updating the awareness-free method to PAM is more cost-efficient in bigger branching-time models. Figure 6-9 exhibits that using PAM increases the cost, which is to be expected, but in fact increasing the size of the branching-time model increases the average cost in both approaches. However, in bigger branching-time models the success rate grows
more sharply than the average cost. As a result, the ratio of success rate subtraction to the cost subtraction, called cost-efficiency, rises by increasing the size. Therefore, the size of branching-time models has a positive impact on the cost-efficiency of upgrading the awareness-free method to PAM. It can be also argued that in bigger models employing PAM instead of the awareness-free method is more cost-efficient compared with smaller branching-time models.

**Figure 6-9 Effect of Size on Cost-Efficiency – Hypothetical Simulation**

**Branching Factor:** The branching factor of a time-branching model is defined as the ratio of size to the number of time points in the model. This illustrates how flat the branching-time model looks. In this experiment, I categorised the branching-time models by rounding branching-time factors to the nearest
decimal number. This is borrowed from Taibi (2008). The reason for rounding the branching factor is to categorise the branching-time models. In fact, the whole point of branching factor is to show the flatness of the models. However, I can fairly estimate how flat the models would be by the rounded numbers. I could find branching-time models that were flat; that is, branching-time factor = 1, while there were also models with branching factor equals to 5.1 (see Figure 6-10 and Figure 6-11). I repeated the experiment for each specific branching factor 100 times. Each time, I randomly selected branching-time models from the related category. I ran the system with the awareness approach and with the awareness-free method. The results are presented below.

I found that the cost does not have a recognised pattern corresponding to branching factor. This is because it is possible to have a branching-time model that is very horizontal but small in terms of size. Therefore, by increasing the branching factor, cost does not follow an identified pattern. As such, I did not study the cost-efficiency by varying the branching factor.

In the flat branching-time models, which do not include any option for taking different actions, the success rate is equal in both PAM and the awareness-free method. In fact, in such models, PAM does not contribute. The experiment shows that by increasing the branching factor, the success rate of PAM falls down. Figure 6-10 illustrates that the PAM curve stays above the curve for the
awareness-free method, which indicates the positive effect of using awareness on
the success of the system.

In Figure 6-11, the curves, on increasing the branching factor, diverge from each
other. This means that the success drop is more radical in the awareness-free
method compared with PAM. This shows that by increasing the branching factor,
despite the fact that the success rate decreases, updating the awareness-free
method to PAM is still effective. This is shown in Figure 6-11. The non-smooth
characteristic of the curve, particularly in branching factor 3 and 4, is due to the
random selection of scenarios. In some cases, this randomization is to the favour of matching with the available policy rules. Therefore, the efficacy experiences jumps.

Figure 6-11 Effect of Branching Factor on Efficacy – Hypothetical Simulation

**Availability of Policy Rules:** Associated with the available situations, events, actions and propositions, the Java program generated 200 policy rules. I ran the system developed by PAM with zero policy rules; That is, equivalent to the awareness-free method with one policy rule, with two policy rule and so on up to 200 policy rules. Each time, I ran 100 selected branching-time models from the 1000 generated models and recorded the success rate and the average number of navigated worlds as cost. These experiments evaluate the main idea of PAM. That is, the use of policy rules as a source to identify awareness. I operated these
experiments only for PAM, and I can consider PAM with zero policy rules to be the same as the awareness-free method\textsuperscript{1}.

Figure 6-12 exhibits the growth of success rates by increasing the number of policy rules. This is because there would be no policy rule to identify awareness. Since PAM with no policy rule is similar to awareness-free method, the efficacy of updating the awareness-free method to PAM can be calculated by dividing the success rate of PAM in each point of time with the success rate of PAM-with-no-policy-rule, which is a constant number. Therefore, the efficacy curve, in terms of growth by increasing the number of policy rules, has the same shape as the curve plotted for the success rate. As a result, looking at Figure 6-12, I can say that updating the awareness-free method to PAM becomes more effective with more policy rules. Figure 6-13 shows that by increasing the number of policy rules, the cost of the system increases.

Figure 6-14 displays the growth of cost-efficiency by increasing the number of policy rules when using PAM instead of the awareness-free method. The growth of curve indicates that despite the fact that increasing the number of policy rules lifts up the average cost (see Figure 6-13), this update makes the system more efficient in terms of the spent cost and received success rate. This conclusion is because of the tremendous success rate growth compared to the increasing cost rate when the system involves more policy rules.

\textsuperscript{1} When I run PAM with no policy rule, it becomes similar to the awareness-free method, because PAM identifies awareness from policy rules.
196 Evaluation

Figure 6-12 Effect of Availability of Policy Rules on Success Rate – Hypothetical Simulation

Figure 6-13 Effect of Availability of Policy Rules on Cost – Hypothetical Simulation
6.2.3 Simulation of Wireless Communication Procedures at Olavs Hospital

In this section, I present a simulation study on the application of PAM in wireless communication procedures at St. Olavs Hospital, Trondheim University Hospital, Norway.

An interpretive case study (Stockdale and Standing, 2006) has been conducted by Norwegian Centre for Integrated Care and Telemedicine (NST) at St. Olavs Hospital, Trondheim University Hospital, Norway (Botsis et al., 2007; Solvoll and Scholl, 2008).¹ They found that physicians had too many interruptions by IP-

---

¹ A part of this research at NST is supported by the Research Council of Norway, grant no. 176852/S10. I would also like to thank all the physicians at St. Olavs Hospital for all their help and collaboration.
phones when they were busy with other activities such as examinations. In the following, I give an example of such situations in Section 6.2.3.3. I briefly explain how NST collected data in the hospital and then I simulate the collected procedures. The simulation study presented in this section aims at evaluating PAM in the context of the wireless communication procedures in St. Olavs Hospital (see Figure 6-5).

6.2.3.1 Data Collection

In order to collect procedures, a researcher from NST moderated an interpretive data collection (Stockdale and Standing, 2006) at St. Olavs Hospital. This consisted of participatory observations, non-structured and mostly ad-hoc interviews and discussions. The study was conducted among a selected group of physicians at various levels of hierarchy and roles. Regarding sensitive information gathered during the study, a non-discloser agreement had been signed before data collection. For more details, readers should refer to (Botsis et al., 2007; Solvoll and Scholl, 2008).

Observations: The moderator followed the independent work of 11 physicians at the clinics for a total of 135 hours. In order to have a better understanding of the wireless communication procedures at the hospital, the moderator took the role of a first year medical student, dressed and acted like a physician to blend in as much as possible. He followed each physician in his or her everyday work for at least two workdays/​nights/​duties. The moderator contacted each physician to
make an appointment in the morning. He recorded every call/page/message, type of device, reaction and physicians’ situation.

**Interviews and Discussions:** During the observation time, the moderator had an open office with other assistant physicians. This made opportunities for several discussions on communication procedures. The moderator did not use any pre-defined interview guidelines. However, he asked questions related to what he had observed. The moderator asked two types of questions from interviewees: (I) specific questions in respect of the role of interviewees, (II) general questions asked from everyone; and then the answers were compared.

Here, the focus is on wireless communication among medical personnel. The moderator, during the data collection, mainly on his own observations, found 43 different procedures and 24 policy guidelines related to interruption via the wireless communication at the hospital. In the next section, I explain how I deployed PAM into these procedures in the simulation study. The deployment will be, next, discussed and then I present a sample procedure in the hospital.

### 6.2.3.2 Deployment

PAM can be deployed using intelligent communication systems (Terashima, 2001) such as Cisco IP SoftPhone. Cisco IP SoftPhone is a Windows-based voice-over-IP (VoIP) communication application for the PC used as a stand-alone end station in Cisco Systems line of IP telephones. The implementation in
the evaluation used Cisco IP Phone Simulator installed on PC. The objective of intelligent communication systems is to provide communication processing that includes (I) a Service Logical Program (SLP) to receive users’ requests and (II) a Service Logical Interpreter (SLI) to execute the users’ requests and return the results to the users. The SLP and SLI can be implemented by software agents. This implementation can benefit from using PAM to recognise the relevance of information and request it from users.

In the next section, I present a sample wireless communication procedure at St. Olavs Hospital to demonstrate the nature of the procedures in this simulation.

6.2.3.3 Sample Procedure

St. Olavs Hospital in Trondheim, Norway has implemented a single converged Internet Protocol (IP) network for communication among medical staff. The medical personnel of this hospital communicate via this IP network. Nurses and physicians carry Cisco 7921G IP-phones. The standard call service from the nurse station to medical staff is integrated over these IP-phones. The personnel must log into the system from the station personnel phones. In this way, the system is informed about the presence of staff. The nurse at the nurse station calls personnel by clicking the call button on the PC (see Figure 6-15).

Assume a situation where a physician is in a sterile dress and gloves. When the physician receives a call about a patient’s medication from a nurse, he has to stop
the examination, take off his gloves and answer the phone. Then, after finishing the call, he washes his hands and treats them in antibacterial liquid. He also has to put new gloves on and start the examination all over again.

Figure 6-15 Related Communication Settings at St. Olavs Hospital in Trondheim, Norway

Physicians believe that most of the phone calls that they receive during the examinations can be answered by another available colleague. There is a guideline in the hospital for operating the call from the nurse station, which is concerned with such situations. The guideline says that once an issue appears that requires a physician’s consultation, if the physician is not available, the nurse must not call the physician. Therefore, calling the physician when she or he is not available is forbidden. In the procedure, the physician was not available and the call should have been diverted to another physician. However, the nurse failed to identify the relevance of information about the availability of the physician. Accordingly, the
nurse did not ask for the availability of the physician from the communication center before calling, while communication centre connects directs the calls in the hospital and it also stores the availability of the physicians.

The problem, here, is that the nurse failed to identify the relevance of availability of the physician. In the following, I present the steps proposed by PAM to identify the relevance of physicians’ availabilities; that is, awareness. Figure 6-16 shows the branching-time model of the world for the nurse when an issue comes up, which requires the calling of a physician. Figure 6-17 presents the deployment of procedure. The three-step process of PAM is given, here:

I. **Step 1: Recognise Relevance of Policy Rules:** The nurse agent implicitly knows that the physician is required. However, the nurse agent does not know whether the physician is available or not. Following the nurse’s branching-time model, if she simply calls the physician, she goes to s₈. If the nurse finds out that they are optionally and eventually at s₄, she will implicitly know about the physician’s unavailability. Therefore, by calling the physician and going to s₈, the nurse optionally will eventually do the forbidden action involved in the mentioned policy guideline. As such, calling the physician when he or she is not available is forbidden and violates the rule. The agent thus takes this rule as relevant and becomes aware of it.
II. **Step 2: Recognise Relevance of Information**: Being aware of the rule, the nurse agent needs to implicitly know the conditions for the rule, which makes the availability of the physician relevant. Therefore, the nurse agent needs to become aware of the physician’s availability.

III. **Step 3: Change of Behaviour Based on the Recognised Relevant Information**: While the nurse is aware that the physician is not available, the nurse agent needs to implicitly know whether “the physician is not available” is a true or false sentence. Therefore, it takes an action that eventually brings it to this knowledge. The nurse agent asks the communication centre for availability of physicians and goes to $s_2$. This is because that the truth or falsity of the sentence promotes going to $s_4$ or $s_5$, which are accessible from the path that begins from $s_2$.

Figure 6-17 depicts the mentioned procedure as an example of how a PAM-supported system can be deployed. Every day, the physicians give their available schedule to the communication centre via their station phones. Then, once a nurse dials the physician, the wireless phone checks the availability of the physician with the SLI agent sitting on the server at the communication centre. If the physician is available, the SLP agent at the nurse station will continue dialling. If the physician is not available, then the agent refers to the uploaded delegator list and calls the proposed delegator.
The setting followed Figure 6-15. I ran my experiments on a computer with four types of agents: station phone agent, nurse station agent, communication center agent and medical personnel agent. The processing power for the computer was characterised as a 2.00 GHZ Dual Core Pentium 4 processor with 2.00 GB memory. The nurse station agent was given the policy guideline rules to apply PAM. In each experiment, the agent ran three steps of PAM. The communication center agent had access to the schedules.

Figure 6-16 Branching-Time Model for the Nurse Agent in the Illustrative Example
The sample procedure presented above is simple, which might seem to impair the relevance and applicability of PAM in solving this type of problem. However, the objective of this example is to illustrate how PAM can identify the awareness from a given set of policies in such procedures. The procedure demonstrates the run-time process of PAM for awareness identification. The significance of PAM becomes manifest when the set of policy rules changes, yet PAM is still able to identify awareness. In fact, the dynamism offered by the run-time process of PAM makes it possible for the agents to behave effectively and cost-efficiently even when the policies or procedures change. This is what is not possible with normal workflow approaches.
I have already discussed the methodology for the experiments in Section 6.2.1. In what follows, I describe the results obtained from the simulation of the wireless communication procedures.

6.2.3.4 Results

In this section, I present the results of the experiments. In a way similar to the hypothetical simulations, in each experiment, I chose one of the configuration parameters and changed it to see its influence on the success rate, cost, efficacy and cost-efficiency of PAM.

Size: The size of the procedures was in the range of 4 to 26 situations. I categorised the procedures according to their size and put the procedures of the same size in the same group. I repeated the experiment for each group 100 times. Each time, I randomly selected one of the procedures involved in the related group. I ran the system once with PAM and once with the awareness-free method. Table 6-2 presents the results of these experiments based on the size in the wireless communication procedures in the hospital.

Figure 6-18, Figure 6-19, Figure 6-20, and Figure 6-21 depict the results obtained from simulating the wireless communication procedures of the hospital by varying the size. Comparing the obtained results with the results of hypothetical simulations, I can see that simulating the procedures of a real case corroborates the trends identified in the hypothetical simulation.
### Table 6-2: The Results of the Experiments Related to Size – Wireless Communication Procedures

<table>
<thead>
<tr>
<th>Size</th>
<th>Awareness-free Method</th>
<th>PAM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average Cost</td>
<td>Success Rate (%)</td>
</tr>
<tr>
<td>4</td>
<td>3.22</td>
<td>86.61</td>
</tr>
<tr>
<td>8</td>
<td>5.17</td>
<td>79.92</td>
</tr>
<tr>
<td>12</td>
<td>6.2</td>
<td>72.33</td>
</tr>
<tr>
<td>14</td>
<td>7.33</td>
<td>59.28</td>
</tr>
<tr>
<td>15</td>
<td>7.51</td>
<td>48.74</td>
</tr>
<tr>
<td>16</td>
<td>7.83</td>
<td>38.93</td>
</tr>
<tr>
<td>17</td>
<td>8.11</td>
<td>35.86</td>
</tr>
<tr>
<td>19</td>
<td>8.84</td>
<td>24.41</td>
</tr>
<tr>
<td>20</td>
<td>9.4</td>
<td>16.9</td>
</tr>
<tr>
<td>22</td>
<td>10.17</td>
<td>9.36</td>
</tr>
<tr>
<td>23</td>
<td>10.43</td>
<td>5.96</td>
</tr>
<tr>
<td>24</td>
<td>11.27</td>
<td>4.23</td>
</tr>
<tr>
<td>25</td>
<td>11.82</td>
<td>4.01</td>
</tr>
<tr>
<td>26</td>
<td>12.26</td>
<td>3.5</td>
</tr>
</tbody>
</table>

**Figure 6-18:** Effect of Size on Success – Wireless Communication Procedures
Figure 6-19 Effect of Size on Average Cost – Wireless Communication Procedures

Figure 6-20 Effect of Size on Efficacy – Wireless Communication Procedures
Figure 6-21 contributes to the analysis of the following trends. These trends will be supported by theory in Section 6.2.4.1:

- By increasing the size of branching-time models, the success rate for both PAM and the awareness-free method decreases. However, in bigger branching-time models, PAM has a smaller reduction in its success rate than the awareness-free method. As a result, by increasing the number of situations, the efficacy of updating the awareness-free method to that of PAM increases.
Evaluation

- By increasing the size of branching-time models, the average cost for both PAM and the awareness-free method increases. However, in bigger branching-time models, PAM has a higher growth in cost.

By increasing the size, the cost-efficiency of updating the awareness-free method to PAM increases.

**Branching Factor:** The branching factor in the wireless communication procedures was in the range of 1 to 2.9. I categorised the procedures based on their branching factor and put the procedures with the same branching factor in the same group. I repeated the experiment for each group 100 times. Each time, I randomly selected one of the procedures involved in the related group. I ran the system once with PAM and once with the awareness-free method. Table 6-3 presents the results of these experiments based on the branching factor of the procedures in the wireless communication procedures in the hospital.

Table 6-3 The Results of the Experiments Related to Branching Factor – Wireless Communication Procedures

<table>
<thead>
<tr>
<th>Branching Factor</th>
<th>Awareness-free Method</th>
<th>PAM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Success Rate (%)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>1.2</td>
<td>69.92</td>
<td>82.66</td>
</tr>
<tr>
<td>1.4</td>
<td>60.56</td>
<td>77.31</td>
</tr>
<tr>
<td>1.8</td>
<td>41.12</td>
<td>64.32</td>
</tr>
<tr>
<td>2.1</td>
<td>22.92</td>
<td>50.34</td>
</tr>
<tr>
<td>2.4</td>
<td>10.96</td>
<td>38.2</td>
</tr>
<tr>
<td>2.9</td>
<td>4.45</td>
<td>23.45</td>
</tr>
</tbody>
</table>
Figure 6-22 and Figure 6-23 depict the results obtained from simulating the wireless communication procedures of the hospital by varying the branching factor. Comparing the obtained results with the results of hypothetical simulations, I can see that simulating the procedures of a real case corroborates the trends identified in the hypothetical simulation. By increasing the branching factor, the success rate for both PAM and the awareness-free method decreases. However, in more horizontal branching-time models, PAM has a smaller reduction in its success rate than the awareness-free method. As a result, by increasing the branching factor, the efficacy of updating the awareness-free method to PAM increases.
Availability of Policy Rules: The moderator during his observation, interviews and discussions at St. Olavs Hospital found 24 policy guidelines that medical personnel are supposed to follow in their communications. I ran the system developed by PAM with zero policy rules – which is equivalent to the awareness-free method, with one policy rule, with two policy rules and so on up to 24 policy rules. In each time, I ran all the 43 identified procedures and recorded the success of the system and the number of navigated situations as cost. Then, I calculated the success rate and average cost. Table 6-4 shows the results of this experiment.
Table 6-4 The Results of the Experiments Related to Availability of Policy Rules - Wireless Communication Procedures

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>21.32</td>
<td>11.41</td>
<td>7</td>
<td>32.93</td>
<td>14.21</td>
<td>14</td>
<td>65.4</td>
<td>17.73</td>
<td>21</td>
</tr>
<tr>
<td>1</td>
<td>22.11</td>
<td>12</td>
<td>8</td>
<td>35.1</td>
<td>14.56</td>
<td>15</td>
<td>68.43</td>
<td>18.07</td>
<td>22</td>
</tr>
<tr>
<td>2</td>
<td>23.43</td>
<td>12.2</td>
<td>9</td>
<td>41.12</td>
<td>15.37</td>
<td>16</td>
<td>70.2</td>
<td>18.14</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>24.52</td>
<td>12.46</td>
<td>10</td>
<td>48.91</td>
<td>16.2</td>
<td>17</td>
<td>72.6</td>
<td>18.26</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>25.92</td>
<td>12.83</td>
<td>11</td>
<td>54.72</td>
<td>16.93</td>
<td>18</td>
<td>75.9</td>
<td>18.53</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>27.64</td>
<td>13.21</td>
<td>12</td>
<td>59.92</td>
<td>17.28</td>
<td>19</td>
<td>79.6</td>
<td>18.69</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>29.14</td>
<td>13.45</td>
<td>13</td>
<td>62.11</td>
<td>17.44</td>
<td>20</td>
<td>83.1</td>
<td>18.91</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6-24, Figure 6-25, and Figure 6-26 depict the results obtained from simulating the wireless communication procedures of the hospital by varying the number of policy rules. Comparing the obtained results with the results of hypothetical simulations, I can see that simulating the procedures of a real case corroborates the trends identified in the hypothetical simulation. By increasing the number of policy rules, the success rate of PAM, the average cost and the cost-efficiency increase. Since PAM with zero policy rules is similar to the awareness-free method, efficacy has the same trend as the success rate in PAM.
Figure 6-24 Effect of Availability of Policy Rules on Success Rate – Wireless Communication Procedures

Figure 6-25 Effect of Availability of Policy Rules on Average Cost – Wireless Communication Procedures
6.2.4 Simulation Outcomes and Theoretical Support

In the first experiment, I found that bigger procedures have less success rate and higher average cost, which are both undesirable but to be expected. In fact, there is a chance that a branching-time model has many situations and very few optional paths, which makes the shape of the branching-time model very flat. However, because of the randomised process of branching-time model selection, when I repeated each experiment 100 times and each time chose the branching-time model randomly, the chance of getting such a flat branching-time model was very low. The randomised selection process, gives a normal distribution on the branching-time model. The analysis on the branching-time models shows that increasing the size often increases the length of the path as well as the horizontal situations in a time point. In fact, by increasing the number of situations, the
number of the branches as well as the number of situations in each branch increases. This is because, in real procedures taken from St Olvas hospital there are always options for different paths and in each of those the agent needs to take few situations. This increases the number of decision points where the agent has more than one optional action to execute. Having more options increases the number of the mistakes and decreases the success rate. Longer paths clearly require the taking of more actions, which increases the average cost of the procedure. The data analysis in the first experiment shows that updating the system to a PAM-supported system is more effective and cost-efficient in bigger procedures. In terms of efficacy, since PAM identifies awareness and selects an action that brings it to the knowledge of relevant information, PAM works more effectively than the awareness-free method. In fact, if in a system cost is a critical issue, then PAM is not a proper choice, because by increasing the size of the procedures the average cost of using PAM, in comparison with the awareness-free method, grows more sharply. This is because a PAM-supported agent has to go and find out the validity of relevant information. This increases the success rate but it requires the agent to navigate more situations, which results in a higher average cost. However, the study shows that in bigger procedures, PAM also becomes more cost-efficient. This is because in bigger procedures, the growth of the success rate by using PAM is considerably more than the growth of the cost. This means that if in a system, such as the interruptions in the wireless communication of St. Olavs Hospital, success is the main issue, then PAM is a
Evaluation

good choice for bigger procedures. PAM is also a good method if I am concerned about cost and success together, i.e. cost-efficiency. Since the study shows that by increasing the size of the procedures, the cost-efficiency of applying PAM increases, the proposed method is also a proper choice for considering cost-efficiency especially in bigger procedures.

In the second experiment, I found that the procedures with a bigger branching factor involve more action selections, which decrease the success of the system. Comparing the success rate in PAM and the awareness-free method, I argue that when the branching factor increases, PAM is effective. This is because PAM identifies awareness as a method by which to select a proper action that equips an agent with cognisance of the relevant information.

In the third experiment, I examined the idea of using policy rules as a source of awareness to obtain a higher success rate. The results show that by increasing the number of policy rules, PAM becomes more expensive but also more effective and cost-efficient. Since PAM uses policy rules to identify awareness, more policy rules provide more relevant propositions. Therefore, at points of decision the agent accrues more choice. However, to obtain knowledge about these relevant propositions, the agent has to navigate more situations. Therefore, the cost increases. In other words, if the cost is an issue in a system, then using PAM with a large number of policy rules is not a wise choice. However, when the success is
a concern and becomes an important factor for the system, then using PAM with a large number of policy rules is effective and cost-efficient.

6.2.4.1 Theoretical Support

The objective of this section is to corroborate the theory of Pools and Streams (Riemer and Haines, 2008) in order to support the results of the experiments. As I explained in Section 2.1.1, this theory introduces a metaphor of “pools and streams”. Once one identifies the relevance of information, it creates a pool. Having become aware of the relevant information, one directs streams of knowledge about the created pool, or in other words, one adapts his or her behaviours to attend a situation in which the implicit knowledge about that information is inferable. In the following, I justify the results of the simulations based on this theory.

In the first experiment, by increasing the size of the branching-time models, the number of situations increases. Consequently, the average number for the streams of actions increases. This increases the average cost. In addition to that, increasing the number of streams makes the choice of action harder. Therefore, the average success rate decreases. However, the use of pools can be beneficial as a criterion to remove some of the streams from the set of available choices. Therefore, the drop in the curve of PAM simulation is less than the drop in the awareness-free method for bigger procedures.
In the second experiment, a higher branching factor increases the number of horizontal situations in the models. This increases the number of available streams. Therefore, the choice among the different streams becomes harder and the average success drops. As such, I need to provide selection criteria to choose the best stream. These can be pools of awareness. Therefore, only those streams can be chosen that flow the knowledge to the pools. This explains the divergence of the curves for PAM and the awareness-free method.

In the third experiment, by using PAM, increasing the number of policy rules increases the number of pools. However, the number of available streams remains constant. Therefore, by having more criteria, it is easier to select among streams. Therefore, the success rate increases. However, I need to make sure that all the created pools get filled up by streams. As such, this takes longer streams of actions, which increases the cost.

6.2.5 Validity of Simulations

In this section, I address the following three questions: (I) how far can I generalise the simulations? (II) how significantly does PAM influence the output criteria? (III) how impartially were the simulations conducted?

6.2.5.1 Significance and Generalisability

The experiments presented in this paper have so far shown that PAM is effective and cost-efficient. However, I have not explained whether the finding is
significant enough to call it a contribution. I should also provide how limited these experiments are to generalise the outcomes. In order to do so, I use a statistical method called t-test. This method assesses whether the data collected from the two groups are statistically different from each other. This analysis is appropriated whenever you want to compare the two groups. Readers wanting more details on t-tests should refer to Montgomery (2008). Here, I test the efficacy and the cost-efficiency gleaned from each experiment.

For example, in the t-test for efficacy, I first set the statistical hypothesis such as “the set of data \( \mu_{PAM}^{efficacy} \) is greater than the other set of data \( \mu_{Awareness-free}^{efficacy} \)” I want to show that this statement is true, and, if so in which degree of probability I can generalise the finding. I calculate the population means by Equation 6-3 for these two sets of data where \( \xi_{PAM}^{efficacy} \) is the population means for \( \mu_{PAM}^{efficacy} \), \( \xi_{PAM,j}^{efficacy} \) is the \( j \)th efficacy and \( n_{PAM}^{efficacy} \) is the size of the data for the set \( \mu_{PAM}^{efficacy} \). Similar definitions can be given for \( \mu_{Awareness-free}^{efficacy} \).

\[
\xi_{PAM}^{efficacy} = \frac{\sum_{j=1}^{n_{PAM}^{efficacy}} \xi_{PAM,j}^{efficacy}}{n_{PAM}^{efficacy}} \quad \text{Equation 6-3}
\]

Then, Equation 6-4 estimates the variance for \( \mu_{PAM}^{efficacy} \).

\[
\theta^2 = \frac{\sum_{j=1}^{n_{PAM}^{efficacy}} (\xi_{PAM,j}^{efficacy} - \xi_{PAM}^{efficacy})}{n_{PAM}^{efficacy} - 1} \quad \text{Equation 6-4}
\]
The formula of Equation 6-5 pools the individual sample variances.

\[ \theta_P^2 = \frac{(n_{\text{efficacy}} - 1)\theta_{PAM}^2 + (n_{\text{Awareness-free}} - 1)\theta_{\text{Awareness-free}}^2}{n_{\text{efficacy}} + n_{\text{Awareness-free}} - 2} \]

Equation 6-5

Hence, the degree of freedom for the sampling is calculated by

\[ n_{\text{PAM}} + n_{\text{efficacy}} + n_{\text{Awareness-free}} - 2. \]

Once I calculate the t-value from Equation 6-6, I look at the table of t-values (Montgomery, 2008). Considering the degree of freedom, if I can find the minimum probability in which the calculated t-value is still greater than the number provided by the table, then the hypothesis is significant.

\[ t_0 = \frac{\bar{z}_{\text{PAM}} - \bar{z}_{\text{Awareness-free}}}{\theta_P \sqrt{\frac{1}{n_{\text{efficacy}}} + \frac{1}{n_{\text{Awareness-free}}}}} \]

Equation 6-6

I name this probability as \( \alpha \) and define the confidence interval as \( 100 \times (1 - \alpha) \), which shows the generalisability of the findings.

Table 6-5 shows the results for the experiments done in the simulations of PAM in regard to their significance and generalisability.
Despite the fact that the experiments are limited to confidence interval, the rates for generalisability are acceptable.

### Table 6-5 Significance and Generalisability of Simulations

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Hypothetical Simulation</th>
<th>Simulation on Wireless Communication Procedures</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Significant?</td>
<td>Confidence Interval</td>
</tr>
<tr>
<td>Size</td>
<td>yes</td>
<td>99.8%</td>
</tr>
<tr>
<td>Branching Factor</td>
<td>yes</td>
<td>83.1%</td>
</tr>
<tr>
<td>Availability of Policy Rules</td>
<td>yes</td>
<td>84.6%</td>
</tr>
</tbody>
</table>

**Cost-Efficiency**

<table>
<thead>
<tr>
<th>Experiment</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>yes</td>
<td>87.3%</td>
</tr>
<tr>
<td>Branching Factor</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Availability of Policy Rules</td>
<td>Yes</td>
<td>84.2%</td>
</tr>
</tbody>
</table>

### 6.2.5.2 Randomised Inputs to Avoid Biasing

It is essential to avoid biasing the input data, which can harm the results. Two major reasons for biases are selection biases and observation biases. In this study, as recommended by the literature (Bailey, 2008), I use random inputs for avoiding selection biasing. In order to avoid observation biases, during the collection of procedures in the hospital, the moderator discussed with physicians what had been found in the observations.

### 6.3 Summary and Outlook

The objective of this chapter was to show the applicability of PAM in real life as well as evaluating PAM by simulation results.
Evaluation

In order to demonstrate the application of PAM in real life, this chapter gave three proofs of concepts in (I) the space shuttle Columbia disaster, (II) the Casa Grande hazardous materials incident, and (III) rehabilitation video games. While PAM has not been used in real production environments, these proofs of concepts provided exemplars to illustrate the significance of the motivation for awareness identification and to demonstrate the applicability of PAM in such situations.

In this chapter, simulations were conducted to evaluate the efficacy and cost-efficiency of using PAM in comparison with an awareness-free method. In order to do so, the simulation designed three configuration parameters as size, branching factor and number of policy rules. These parameters can vary the inputs of simulations to compare the change of success rate and cost. By changing each of the configuration parameters in PAM and the awareness-free method, I was able to calculate efficacy and cost-efficiency for each of these changes. In order to increase the reliability of the simulation, I triangulated the simulations with (I) the hypothetical simulation of generated branching-time models and (II) simulation of wireless communication procedures in St. Olavs Hospital, Norway. Finally, I concluded that PAM is an effective and cost-efficient method subject to the limitations of the simulations.

In the next chapter, I discuss the implications and limitations of PAM as well as the conclusions and directions for future research.
Chapter 7

DISCUSSION, FUTURE WORK AND CONCLUSION

The present chapter provides a closure to the thesis – that is, awareness identification, aimed in this study, has now been addressed by Policy-based Awareness Management (PAM). However, the chapter makes it clear that there are certain limitations to this study that open new directions for future research. This chapter sums up the arguments made in this thesis. It briefly summarises the main points, implications and limitations as well as future work for the present study.

The rest of this chapter is organised in the following way: Section 7.1 presents the implications of policy-based awareness management. This section discusses how this work relates to research in academia as well as practices in information technology development. Section 7.2.2 presents the limitations of this study, which encourage future researchers to contribute to the policy-based awareness identification. Section 7.2 concisely concludes the thesis. This section states the research objectives, motivations of the study and presents the essence of PAM.

7.1 Discussion

The objective of this section is to demonstrate the rigor of the present study and show how PAM is associated with current theories and practices. This section presents the implications of PAM for academia as well as practitioners.
7.1.1 Academic Perspective

The theoretical implications of this work for academia can be categorised in Computer Supported Cooperative Work (CSCW), Software Agents, and Policy Structures in Distributed Cooperative Systems. In what follows, I explain these implications.

7.1.1.1 Implications for the Field of CSCW

We borrowed the concept of awareness from the field of CSCW. I identify the implications of PAM in Theory of Situation Awareness (Endsley, 1995b), Theory of Pools and Streams (Riemer and Haines, 2008) and CoMEN (Ray, Shahrestani, et al., 2005).

The theory of situation awareness (Endsley, 1995b) provides a conceptual model for the process of awareness maintenance. This theory states that once an individual captures the context (in the framework presented in 2.1.3, it is called awareness obtainment), there are three levels of awareness: Level 1 - perception (context representation), Level 2 – Comprehension (context analysis) and Level 3 – Projection of future (awareness utilisation). The Theory emphasises the change of behaviours in respect of the above levels of mental attitudes. In this sense, the theory of situation awareness corroborates the claim in PAM, correspondent with the action selection based on the identified awareness. However, as I discussed in Section 2.1.5, the theory of situation awareness lacks the step for awareness identification. PAM as a technical approach, rather than a conceptual model, aims
at assisting individuals to identify awareness using software agents. In the theory of situation awareness, once an individual wants to capture a context, there should be a way to identify which context is relevant to capture, otherwise the individual suffers from the problem of information overload. PAM addresses this, while the individual is assisted by an agent-based system that uses the existing policy rules to identify awareness.

Riemer and Haines (2008) provide metaphors of pools and streams to explain the process of awareness maintenance. In this metaphor, the pool refers to awareness and the stream refers to the flow of knowledge. Once an individual becomes aware of information, she or he pools that information. Then, individuals shape the behaviours or direct the streams in a way that makes it possible to fill the pool. In other words, the individual selects an action that brings him or her to a situation in which the knowledge about that information is achievable. The theory of pools and streams also fails to address the process in which the individual identifies awareness. As I discussed above, an individual who wants to remove some uncertainty can be overloaded by irrelevant or loosely relevant information, if she or he is not able to identify awareness of relevant information. PAM, as a technical alternative, addresses this issue by supporting the awareness maintenance process with an agent system that extracts awareness from policy rules.
As I discussed in Section 2.2.1, CoMEN enhances cooperation by using the conceptual model of awareness. The essence is that if an individual is required to become aware of information that she or he is not currently aware of, there is an opportunity to enhance cooperation by developing an information technology solution that provides this information to the individual. However, there is currently no definitive method for recognising the required awareness (Ray, Shahrestani, et al., 2005; Daneshgar et al., 2006). Given policy-aware agents defined in PAM, participants in cooperative endeavors can identify the relevance of policy rules and the information that they require to enhance cooperation in run-time. As such, the process of cooperation enhancement becomes a run-time process, which allows individuals to dynamically identify the relevant information and enhance the cooperation rather expending extra analysis, design and implementation efforts.

7.1.1.2 Implications for the Field of Software Agents

As I discussed in Section 2.2.2.1, the possible-worlds model commits us to logical omniscience and perfect reasoning. An alternative solution would be the theory of general awareness (Fagin and Halpern, 1988; Sillari, 2008b). In this theory, the agent implicitly knows all the valid propositions, but in order to have explicit knowledge of that proposition, the agent has to become aware of the proposition. While this logic defines awareness as relevance of information, it is clear that being aware of a proposition does not necessarily mean that proposition is
explicitly known. Therefore, an agent does not explicitly know all the valid propositions nor all the implications of his or her explicit knowledge. The agent only has explicit knowledge of a proposition when it is aware of that proposition and it has implicit knowledge of that. The logic clearly defines the way that an agent obtains the implicit knowledge. As it is defined above, the agent implicitly knows the proposition if and only if the proposition is valid. The concept of awareness, in this logic, is defined as relevance of propositions. However, how this set can be created, or in other words, how an agent can identify awareness of a proposition remains an open question (Halpern and Pucella, 2010). The present study builds the PAM framework on the foundation grounded from the logic of general awareness and proposes to use the existing policy rules as an alternative to identify awareness. This enables an agent to recognise the relevance of policy rules and then identifies its awareness of the conditions required for triggering the rules. Finally, the agent selects an action that brings it to a situation in which the path to implicit knowledge about the policy rule conditions is accessible.

The idea of using existing policy rules to identify awareness increases the opportunity for the logic of general awareness to relax its assumption indicating a given set of relevant propositions. This assumption commits agents to be dependent on a pre-defined awareness set, which disables agents to adapt them dynamically to them in run-time. Instead, PAM assumes a given set of policy
rules that change much less frequently than awareness. Alternatively, the agents can read the policy rules from an updated file.

7.1.1.3 Implications for the Field of Policy Specification in Distributed Cooperative Systems

Among the widely used policy structures, Directory Enabled Networks - next generation (DEN-ng) (Strassner et al., 2009) provides a meta-model to implement a given set of awareness using policy rules. The contribution in this meta-model focuses on generating policy rules that enables the system to support a given set of relevant information, awareness. In fact, the meta-model enables designers to generate such policy rules that provide knowledge about the relevant information to the agents. Therefore, the assumption in this meta-model is a given set of awareness. However, PAM assumes a set of policy rules and extracts the awareness.

In order to provide adaptive services, DEN-ng, similar to the logic of general awareness, requires pre-defined awareness associated with the situations. In such systems, policy rules are designed in a way that they can control the system behaviours based on perceived changes. However, the pre-definition of the awareness set decreases the adaptability of the system. Alternatively, PAM, rather than assuming a set of awareness, gets a set of policy rules as an input. According to these rules, PAM identifies the awareness set in run-time.
7.1.2 Practice Perspective

This section provides the potential use of PAM for practitioners. Although, PAM can be taken in wider area of applications, here I give examples on the potential use of PAM in particular domains.

7.1.2.1 Bridging the Gap between Management Issues and Software Agent Development

Practices in developing software agents typically address awareness in terms of programming agents. Such software agents should reason and make suggestions to assist individuals. However, the current approaches for the implementation of awareness suffer from two weaknesses which can be overcome by considering the proposal of this study: first, the suggestions made by software agents are dependent on their understanding of the situation and limited by their implementation. As such, agents are not able to identify the relevance of information to a situation. Policy rules reflect managerial issues and they are not limited by the agents’ individual understandings, which are highly dependent on the agents’ implementation. Second, the standard approaches to software agents are technology dependent, but distributed agencies often use different technologies simultaneously. Therefore, integration of the system can be difficult while ignoring policies. Taking these two reasons into consideration, PAM proposes the use of policy rules to identify awareness.

The development guidelines provided in Chapter 5 bring this idea to the practice of software agent development in cooperative applications. The guidelines
provide the details of the phases required for developing such applications that use policies to identify awareness.

7.1.2.2 Potential Solution for Information Focus: Protecting Individuals from Irrelevant or Loosely Relevant Information

Information technologies are being incorporated into the people’s lives to remove information uncertainty. However, identifying the relevance of information in the midst of an abundance of information can be extremely difficult. This influences the former objective to remove uncertainty. Individuals bombarded by a large amount of irrelevant or loosely relevant information are not able to identify the relevance of information – i.e. awareness – therefore achieving the right knowledge to remove uncertainty can be very difficult. PAM aims at awareness identification and takes the policy rules as an alternative by which to recognise the relevance of information.

7.1.2.3 Potential Solution for Learning from Experiences

Learning from an experience is the process by which people give meaning to the experience itself. Experiences and their educational implications influence the ability of individuals to appropriately judge information as relevant and to reach a realistic understanding of a given situation. Policy rules have long been one of the alternatives by which I can implement a procedure-oriented way to learn from experiences in CSCW.
PAM provides a technical solution to assist individuals in order to transfer what is learnt from experiences in different situations for the purpose of awareness identification. In fact, PAM makes it possible for individuals to take the experiences into consideration while identifying awareness of relevant information.

7.1.2.4 Potential Solution for Improving Intelligent Communication Systems

Intelligent communication systems provide communication processing by (I) a service logical program (SLP) that receives callers’ requests and (II) a service logical interpreter (SLI) that processes callers’ requests and returns proper results to the callers.

In call centres, the calls are not necessarily sequential and there might also multiple topics involved, therefore the volume and pace of information can become overwhelming. This can result in information overload when trying to select the best answer for the callers. As such, the use of an intelligent communication system would be apt in call centres. Therefore, run-time awareness identification can be appreciated for SLI in such situations.

Software agents are a good alternative to implement SLIs. As I discussed in Section 6.2.3.2, in order to make SLI agents able to identify awareness, PAM provides the relevance of information for SLI agents. In fact, policy rules and
guidelines are often available in communication centres. Therefore, PAM can be promisingly deployed for recognizing the relevance of information.

7.1.2.5 Potential Solution for Applying Dynamic Model of Disaster Management in Practice

Plotnick et al. (2009) take the various factors derived in prior empirical studies and propose a conceptual model for practices of disaster management, called a dynamic model of disaster management. As a part of this model, they indicate the significant role of “learning from experiences” in “recognising relevance of information” and “improving awareness” in disaster situations. However, this model remains at the stage of recommendations and guidelines; PAM provides a technical solution for this purpose.

As I discussed in Section 7.1.2.3, PAM offers a technical approach to bring the lessons learnt from prior experiences into different situations in order to identify awareness. Therefore, for implementation of the dynamic model of disaster management, an alternative would be an agent-based system that applies PAM to use existing policy rules reflecting the lessons learnt in previous experiences and which continually identifies the relevance of information.

7.1.2.6 Potential Solution for the Supervision Problem in Home-based Medical Treatments

Aged care and chronic diseases, such as cancer, asthma, diabetes and so forth, demand a multidisciplinary team of healthcare professionals. Homecare for
chronic patients is often important, due to the length of treatments. Home-based medical practices, providing treatments at home in familiar surroundings, offer comfort for patients while it is not as costly as hospital beds. However, because of sporadic contacts between the healthcare team and the patients whilst the patient is under treatments at home, supervised hospital-based practices are often more effective than unsupervised home-based treatments (Regensteiner et al., 1997).

The use of mobile computing, in order to track patients, can be an alternative solution for the supervision problem in home-based treatments. In recent years, there has been an increasing trend towards this approach in the mobile health industry. However, one problem for such systems is that automating the treatments to be able to provide such services anywhere and at any time is difficult. The problem can be revealed when I look at the complexity of practices with several relevant factors that affect treatments.

Therefore, in such situations recognising the relevance of information is an issue that is addressed by PAM. Fortunately, medical treatments are often benefited by available guidelines that help to implement the policy rules for awareness identifications. However, due to the complexity of the domain, this approach requires further exploration.
7.2 Limitations and Future Work

The current study suffers from some limitations that open new directions for future research. The limitations of the present work can be categorised between those related to PAM and those related to the evaluation.

7.2.1 Limitations Related to PAM

In this section, I present the limitations in theoretical aspects of PAM. These limitations encourage researchers to contribute to the field.

7.2.1.1 Assumptions Related to Policy Rules

PAM is not a process to design, generate or distribute policies; rather, it assumes a given set of policy rules. Therefore, PAM relies on the quality of the policy rules, which indicates the following limitations:

- **Interactions and conflicts between policy rules**: Policy rules may interact with each other, and a new policy rule may conflict with existing rules. Conflicts can happen as a result of some differences in policy rules such as space modality, applicability period, role, entities and so forth (Syukur et al., 2005).

  Policy conflict research has been long a topic of interest in network management. The primary goal of detecting a conflict is to investigate several possible sources of conflicts that may occur within the system. The following steps have been proposed by different methods in literature for policy conflicts (Syukur et al., 2005).
Discussion, Future Work and Conclusion

I. Analysing the possibility of occurrence for each conflict.

II. Indicating the source of conflict.

III. Investigating the techniques for conflict detection based on the indicated sources.

IV. Investigating the techniques for resolving the conflicts.

Refining high-level policies to DEN-ng policy rules: PAM assumes a given set of policy rules in DEN-ng structure. Therefore, high-level policies in ordinary texts need to be refined into DEN-ng form, before applying PAM.

Research in the area of policy for network management has been looking at this problem to make the refinement process automated. For this purpose, the following steps have been indicated by the different methods proposed in the literature of policy refinement (Bandara et al., 2004):

I. Determining the resources that are needed to satisfy the requirements of the policy.

II. Transforming high-level policies into desired structure.

III. Verifying that the generated policy rules actually meet the requirements specified by the high-level policies.
This encourages future research that aims at methods for refining high-level policies to DEN-ng structure for PAM. Although the literature in this area proposes several methods (Bandara et al., 2004; Rubio-Loyola et al., 2006; Loyola, 2007; Rochaeli and Eckert, 2007; Udupi et al., 2007; Kevin Carey et al., 2008), they admit the semi-manual nature of the policy refinement (Bandara et al., 2004).

7.2.1.2 Policy Deviation

PAM is based on the assumption of agents’ ability to comply with policy rules. However, policies do not necessarily always take all situations into consideration; particularly, it often happens in situations like disasters. In such cases, any action may violate the policy. It thus becomes important to investigate that if policies have to be violated, how we can optimize the policy deviation to have minimum incompliancy.

One basic idea can be the notation of distance measures between the concepts. Distance between concepts means how concepts are similar. One way to measure the similarity of concepts is ontology mapping that has different techniques based on heuristics, semantics, and statistics (Lewis et al., 2010).

7.2.1.3 Generating and Modifying Branching-Time Models

I assume that branching-time models are given to PAM. The objective of this study is to identify awareness in a given branching-time model using a given set of policy rules. This illustrates the limitation of PAM in generating and modifying
the provided branching-time models. However, this can be helpful and increase the adaptability of the agents in run-time.

One approach for future research in this matter can be considering deterring or permitting policy rules, which are not considered in PAM. These two types of policy rules can be beneficial in formulating better branching-time models. In fact, permitting and deterring policy rules give agents additional knowledge to assist generating or modifying branching-time models. For example, a permitting policy rule may indicate that there is a path that definitely leads to the fulfillment of the knowledge about the future (knowledge commitment axiom) and may be better than other paths that possibly lead to dead-end or higher costs. In PAM, agents only change their behaviour when there is a policy rule violation. However, the agents can also change their behaviours, if they become aware of a better branching-time model, which can be achieved by considering permitting and deterring policy rules. This is perceived as being a valuable contribution for future research in the field.

7.2.1.4 Primitive Actions and Events

PAM only deals with primitive actions and events. Primitive actions, as I defined in Section 4.1, are those actions that are directly executable by the agents. Primitive events are those events that they can be received directly by the agents. Primitive actions and events can be assigned to nodes, i.e. situations in the branching-time model of the world.
Future research can explore methods that take non-primitive actions and events. Such methods should map the non-primitive actions or events to the composition of situations in the branching time model. Another approach could be the decomposition of non-primitive actions or events to primitive ones. The initiative for this decomposition would be the structure of the branching-time model.

7.2.1.5 Competency of Agents

As I discussed in Section 4.1.2, PAM assumes policy-aware agents as competent agents (Cohen and Levesque, 1990) who accept $W = W'$, where $W$ is a nonempty set of possible worlds or worlds for short, and $W' \subseteq W$ is worlds that the agent considers possible.

This commits agents to a single view of worlds. However, in some domains, it can be an idealisation. In fact, when an event happens, agents, reflecting more realistic representation of individuals, consider different views of a world and assign a probability value to them. This value shows the chance for occurrence of the worlds.

One approach for future study can be the Neighborhood Model and the application of the work of Arló-Costa (2002) and Arló-Costa & Pacuit (2005) to PAM. The idea of the neighborhood model is that, for each agent, each particular world is associated with a set of worlds, called neighborhoods, and the agent
thinks about them when it is in that particular world. In fact, the agent views a world that can be any of the neighborhood worlds of the world that the agent is situated at the time. This would relax the competency assumption and makes it possible to have $W'$ such that $W \neq W'$.

7.2.2 Limitations Related to Evaluation of PAM

This section discusses the limitations related to the proofs of concepts as well as limitations correspondent to the conducted simulations.

7.2.2.1 Swiss Cheese Effect, Hindsight Bias and Simplicity of Proofs of Concepts

PAM is untried in real production environments. However, three proofs of concepts made in the space shuttle Columbia disaster, the Casa Grande hazardous-materials rail incident and rehabilitation video games prove the relevance of the proposed concepts, which correspond to the following limitations:

- **Swiss Cheese Effect**: The identical characteristic of Swiss cheese is its ampleness of holes created during the curing process when gases in the cheese are unable to escape and so produce holes. This is a metaphor for the issue that attacks the consciousness of the research about different factors affecting a problem. Swiss cheese is also well known in safety management where failures are seen as a result of accumulated errors and are misleading Reason (1997). For example, in the space shuttle Columbia
Discussion, Future Work and Conclusion

disaster, Woods (2009) categorises three different categories for contributing factors, which are mentioned in Section 1.1. However, in this study, I have only focused on the first category.

Although I admit the Swiss cheese effect in the exemplars, particularly in the space shuttle Columbia disaster and the Casa Grande rail incident, I emphasised the lack of a definitive method to identify awareness. However, the discussed exemplars were actually affected by several other factors.

- **Hindsight Bias**: This bias tends to occur when I believe (after an incident) that the onset of the incident was predictable. However, this is an overestimation since I benefit from the feedback given about the outcome of the incident (Hertwig et al., 2003). For example, the given propositional sentences in Section 4.1.3 were identified in hindsight. Other propositions could have been identified before the accident and considered relevant.

Le Coze (2008) acknowledges this issue in incident investigation research and recommends the confirmation of results by other evaluation methods. This way, hindsight can actually be seen as a method of human cognition (Hoffrage et al., 2000) that demonstrates the concept in real experiences. Therefore, I also evaluated PAM by two simulation studies.
- **Simplicity**: Generalising from exemplars is an essential aspect of everyday mental life. In order to appreciate the core concept of PAM, I decrease the complexity by studying exemplars, while bolstering accessibility. Feldman (2003) reviews some evidence that complexity-minimisation does indeed play a significant role in the illustration of relevance and applicability of a concept in real life. Therefore, I simplified the exemplars made in the space shuttle Columbia disaster, the Casa Grande rail incident and the rehabilitation video games.

Researchers are encouraged to discuss the scalability of PAM when the number of situations and actions increases, as well as the increase of policy providing parties.

The objective of the exemplars, in the proofs of concepts, was only to demonstrate the application of PAM to propose a technical solution assisting situations in which I need to identify awareness. While exemplars are a common way in software engineering research and practices to offer a trial proof of the relevance of a new proposal (Cysneiros et al., 2004), I provide an initial validation to PAM by triangulation (Johnson et al., 2007) of hypothetical simulation and simulation of the procedures in wireless communication at a hospital in Norway.

Hence, I recommend further analysis, productions and real-world experimentations for PAM.
7.2.2.2 Measuring Processing Power in Simulations

The objective of the simulations was to conduct a comparative study of efficacy and cost-efficiency between PAM and an awareness-free method. Therefore, there was no direct evaluation of processing power, although the cost, as a related concept to the number of the actions, was measured.

In order to evaluate the processing power required to run PAM, I recommend evaluating the system by implementation in production environments, which allow us to measure the processing power in a period of time considering different workloads.

7.2.2.3 Measuring the Implementation and Design Cost of Distributing Policy Rules in Simulations

Distributing policy rules are often associated with implementation and design costs. The proposal of PAM is a run-time process of identifying awareness from a given set of policy rules. Therefore, the simulation studies lack consideration of design and implementation costs, which can be added in the final results.

Such investigation to measure the implementation and design cost for distributing policies requires production of PAM in real world environments. In fact, simulation studies are not sufficient for this purpose. Therefore, this must be conducted as future research.
7.3 Conclusion

The rise in the use of information technologies has led to an exponential proliferation of information, which overloads individuals with irrelevant or loosely relevant information. This results in the failure of humans to remove uncertainty from their knowledge. Therefore, a widespread demand for techniques that can identify the relevance of information, called awareness, exists.

Policy-based awareness management aims at identifying awareness. For this purpose, PAM employs software agents to assist individuals while the employed agents use existing policy rules for identifying awareness. Policy rules, in the way that they are defined in PAM, indicate the individuals’ behaviours without the need for direct orders. However, correct behaviour, as driven by policy rules, is required for the validation of basic conditions. In PAM, software agents identify the relevant policy rules and become aware of the conditions referenced in the rules. The agents, thus, propose an action that enables individuals to validate the conditions required to trigger the policy rules.

Theoretical work in CSCW, software agents and policy structures will benefit from the proposal of this study, in that they shift their assumptions about a given set of awareness instead of identifying awareness. PAM provides a technical solution for this purpose and aids individuals to alternatively recognise the relevance of information from the existing policy rules. The practices of software
agent development can also apply PAM to different domains where individuals require assistance in order to identify awareness.

Such development will help to simplify the seemingly inexorable information uncertainty and overload that has come to characterise the today’s world.
REFERENCES


References


References


Verification, Linear Temporal Logic, Computation Tree Logic, Model Checking, Logical Connective. Betascript Publishing.


264 | References


import java.util.ArrayList;
import uchicago.src.sim.engine.Stepable;

public class Agent {
    //PUBLIC PROPERTIES
    Situation currentSituation;
    World [] worlds;
    Time [] times;
    Situation [] situations;
    Variable [] variables;
    DomainValue [] values;
    Event [] events;
    Action [] actions;
    Proposition [] propositions;
    ValuedVariable [] valuedVariables;
    ForbiddingPolicyRule [] forbiddingPolicyRules;
    RequiringPolicyRule [] requiringPolicyRules;
    PermittingPolicyRule [] permittingPolicyRules;
    DeterringPolicyRule [] deterringPolicyRules;
    Interpreter interpreter;
    ArrayList<Awareness> awareness;
    ArrayList<ReceivedEvent> receivedEvents;
    ArrayList<DoneAction> doneActions;

    //CONSTRUCTOR
    public Agent (World [] wlds, Time [] t, Situation [] s,
                  Variable [] vars, DomainValue [] vals,
                  Event [] e, Action [] acts,
                  Proposition [] props,
                  ValuedVariable [] valvars,
                  ForbiddingPolicyRule [] fpr, 
                  RequiringPolicyRule [] rpr,
                  PermittingPolicyRule [] ppr ,
                  DeterringPolicyRule [] dpr, Interpreter intrp){
        // memory allocation
        worlds = new World [wlds.length];
        times = new Time [t.length];
        situations = new Situation [s.length];
        variables = new Variable [vars.length];
        values = new DomainValue [vals.length];
        events = new Event [e.length];
        actions = new Action [acts.length];
        propositions = new Proposition [props.length];
valuedVariables = new ValuedVariable [valvars.length];
forbiddingPolicyRules = new ForbiddingPolicyRule [fpr.length];
requiringPolicyRules = new RequiringPolicyRule [rpr.length];
deterringPolicyRules = new DeterringPolicyRule [dpr.length];
permittingPolicyRules = new PermittingPolicyRule [ppr.length];

// initialization
for (int i = 0 ; i <wlds.length ; i++)
    worlds[i]= wlds[i];
for (int i = 0 ; i <t.length ; i++)
    times[i]= t[i];
for (int i = 0 ; i <s.length ; i++)
    situations[i]= s[i];
for (int i = 0 ; i <vars.length ; i++)
    variables[i]= vars[i];
for (int i = 0 ; i <vals.length ; i++)
    values[i]= vals[i];
for (int i = 0 ; i <e.length ; i++)
    events[i]= e[i];
for (int i = 0 ; i <acts.length ; i++)
    actions[i]= acts[i];
for (int i = 0 ; i <props.length ; i++)
    propositions[i]= props[i];
for (int i = 0 ; i <valvars.length ; i++)
    valuedVariables[i]= valvars[i];
for (int i = 0 ; i <fpr.length ; i++)
    forbiddingPolicyRules[i]= fpr[i];
for (int i = 0 ; i <rpr.length ; i++)
    requiringPolicyRules[i]= rpr[i];
for (int i = 0 ; i <dpr.length ; i++)
    deterringPolicyRules[i]= dpr[i];
for (int i = 0 ; i <ppr.length ; i++)
    permittingPolicyRules[i]= ppr[i];
interpreter = intrp;
awareness = new ArrayList<Awareness> ();
receivedEvents = new ArrayList<ReceivedEvent> ();
doneActions = new ArrayList<DoneAction> ();}
Implicit Knowledge for Basic Path Formulae

/* implicit knowledge about a proposition in a path starting
from a specific situation.*/
private boolean implicitKnowledge (Proposition prop,
String modal, String temp, Situation situation){
    if (temp == "next"){
        ArrayList <Situation> nextSituations =
new ArrayList <Situation> ();
        for (int i = 0; i < times.length ; i ++){
            // finding the next time: times[i].tickIndex+1
            // -1: agent is currently is at the final time point
            if (currentSituation.time.tickIndex ==
            times[i].tickIndex){
                if (i== times.length)
                    return false;
                else{
                    // finding the next situations
                    /* they have to be in the situation list of agent as
                    well as the current world.*/
                    /* there must be an action connecting the
                    currentSituation to these situations.*/
                    for (int j=0; j< situations.length; j++){
                        if (situations [j].time.tickIndex ==
                        times[i].tickIndex+1){
                            for (int k = 0; k <
currentSituation.world.situations.length; k++){
                                if (situations [j].situationIndex ==
currentSituation.world.situations[k].situationIndex)
                                { if (isThereAction(currentSituation,
situations [j]))
                                    nextSituations.add(situations [j]);
                }}}))}}}
        if (modal == "optional"){
            /* if the proposition is valid in one of the
            nextSituations, agent returns true.*/
            for (int i = 0; i < nextSituations.size(); i++){
Implementation of Method implicitKnowledge for Path Formulae

```java
if (implicitKnowledge (prop, (Situation)nextSituations.get(i))
    return true;
} return false;

if (modal == "inevitable") {
    /* if the proposition is false in one of the
    next situations, agents return false. */
    for (int i = 0; i < nextSituations.size(); i++) {
        if (implicitKnowledge (prop, (Situation)nextSituations.get(i)) == false)
            return false;
    }
    if (nextSituations.size() < 0)
        return false;
    else
        return true;
}

if (temp == "eventually") {
    ArrayList<Situation> evenSituations = new ArrayList<Situation>();
    /* finding all the situations that may happen now or
    sometime in future. */
    /* they have to be in the situation list of agent as well
    as the current world. */
    for (int i = 0; i < times.length; i++) {
        if (times[i].tickIndex >= currentSituation.time.tickIndex) {
            for (int j = 0; j < situations.length; j++) {
                if (situations[j].time.tickIndex == times[i].tickIndex) {
                    for (int k = 0; k < currentSituation.world.situations.length; k++) {
                        if (currentSituation.world.situations[k].situationIndex
                            == situations[j].situationIndex) {
                            evenSituationsPath (currentSituation, situations[j])
                            evenSituations.add(situations[j]);
                    }
                }
            }
        }
    }
    if (modal == "optional") {
        /* if the proposition is valid in one of the
        next situations, agent returns true. */
        for (int i = 0; i < evenSituations.size(); i++) {
            if (implicitKnowledge (prop, (Situation)evenSituations.get(i))
                return true;
        }
        return false;
    }
```
if (modal == "inevitable"){
    /* if the proposition is false in one of the
    next situations, agents return false.*/
    for (int i = 0; i < evenSituations.size(); i++){
        if (implicitKnowledge (prop,
            (Situation)evenSituations.get(i)) == false)
            return false;
    }
    if (evenSituations.size()<0)
        return false;
    else
        return true;
}
// if none of the defined choices, above.
return false;
}

**Implicit Knowledge for Path Formulae with Until Composition**

public boolean implicitKnowledge (Proposition prop1,
Proposition prop2, String modal, String temp, Situation situation){
    if (temp == "until"){
        ArrayList<Situation> evenSituations = new
        ArrayList<Situation>();
        // finding all the situations that may happen now or some
time in future.
        // they have to be in the situation list of agent as well
        as the current world.
        for (int i = 0 ; i < times.length ; i++){
            if (times[i].tickIndex>= currentSituation.time.tickIndex){
                for (int j = 0 ; j < situations.length; j++){
                    if (situations[j].time.tickIndex == times[i].tickIndex){
                        for (int k = 0 ;
k<currentSituation.world.situations.length; k++){
                            if (currentSituation.world.situations[k].situationIndex ==
situations[j].situationIndex){
                                if (isTherePath (currentSituation, situations[j])){
                                    evenSituations.add(situations[j]);
                                }
                            }
                        }
                    }
                }
            }
        }
        if (modal == "optional"){
            // if the proposition2 is valid in one of the
            evenSituations and proposition 1 has been true from the
            current situation until there, agent returns true.
Implementation of Method implicitKnowledge for Path Formulae

```java
for (int i = 0; i < evenSituations.size(); i++){
    if (implicitKnowledge (prop2,
    (Situation)evenSituations.get(i))){
        if (isValidBetween (currentSituation,
        (Situation)evenSituations.get(i)))
            return true;
    }
}
return false;

if (modal == "inevitable"){
    // if always the proposition 2 is valid some time in future and the proposition one has been valid from the current situation till there, then the agent return true. // the proposition 2 has to become valid.
    for (int i = 0; i < evenSituations.size(); i++){
        if (implicitKnowledge (prop2,
        (Situation)evenSituations.get(i)))
            if (isValidBetween (currentSituation,
            (Situation)evenSituations.get(i))== false)
                return false;
    }
    boolean prop2BecomeValid = false;
    for (int i = 0; i < evenSituations.size(); i++){
        if (implicitKnowledge (prop2,
        (Situation)evenSituations.get(i)))
            prop2BecomeValid = true;
    }
    if (!prop2BecomeValid)
        return false;
    if (evenSituations.size()<0)
        return false;
    else
        return true; }
// if none of the defined choices, above.
return false;
```
IMPLEMENTATION OF METHOD POLICYRULEAWARENESS

//Step1: Policy Rule Awareness
private void policyRuleAwareness (){
    ArrayList<Situation> evenSituations = new ArrayList<Situation>();
    /*finding all the situations that may happen now or sometime in future.*/
    /* they have to be in the situation list of agent as well as the current world.*/
    for (int i = 0 ; i < times.length ; i++){
        if (times[i].tickIndex>= currentSituation.time.tickIndex){
            for (int j = 0 ; j < situations.length; j++){
                if (situations[j].time.tickIndex == times[i].tickIndex){
                    for (int k = 0 ; k<currentSituation.world.situations.length; k++){
                        if (currentSituation.world.situations[k].situationIndex == situations[j].situationIndex){
                            if (isTherePath (currentSituation, situations[j]))
                                evenSituations.add(situations[j]);
                    }}}}
            // for all the forbidding policy rules
            for (int i = 0 ; i < forbiddingPolicyRules.length; i++){
                for (int j=0; j <evenSituations.size(); j++){
                    if (implicitKnowledge (forbiddingPolicyRules[i], "optional", "eventually", evenSituations.get(j)) &&
                        implicitKnowledge (forbiddingPolicyRules[i].condition, "optional", "eventually", evenSituations.get(j)) &&
                        implicitKnowledge (forbiddingPolicyRules[i].event, "optional", "eventually", evenSituations.get(j))){
                        for (int k = 0 ; k< actions.length; k++){
                            if (actions[k].source == evenSituations.get(j) &&
                                actions[k] == forbiddingPolicyRules[i].action){
                                if (!awareness (forbiddingPolicyRules[i], evenSituations.get(j)))
                                    awareness.add(new Awareness
                                        (evenSituations.get(j),forbiddingPolicyRules[i]));
                    }}}}
            // for all the requiring policy rules
            for (int i = 0 ; i < requiringPolicyRules.length; i++){
                for (int j=0; j <evenSituations.size(); j++){
                    if (implicitKnowledge (requiringPolicyRules[i], "optional", "eventually", evenSituations.get(j)) &&
                        implicitKnowledge (requiringPolicyRules[i].condition, "optional", "eventually", evenSituations.get(j)) &&
                        implicitKnowledge (requiringPolicyRules[i].event, "optional", "eventually", evenSituations.get(j))){
                        for (int k = 0 ; k< actions.length; k++){
                            if (actions[k].source == evenSituations.get(j) &&
                                actions[k].condition == requiringPolicyRules[i].condition &&
                                actions[k].event == requiringPolicyRules[i].event){
                                if (!awareness (requiringPolicyRules[i], evenSituations.get(j)))
                                    awareness.add(new Awareness
                                        (evenSituations.get(j),requiringPolicyRules[i]));
                    }}}}
            // for all the requiring policy rules
            for (int i = 0 ; i < requiringPolicyRules.length; i++){
                for (int j=0; j <evenSituations.size(); j++){
                    if (implicitKnowledge (requiringPolicyRules[i], "optional", "eventually", evenSituations.get(j)) &&
                        implicitKnowledge (requiringPolicyRules[i].condition, "optional", "eventually", evenSituations.get(j)) &&
implementation of Method policyRuleAwareness

implicitKnowledge (requiringPolicyRules[i].event, "optional", "eventually", evenSituations.get(j)){
    int numOfNotRequiredActions = 0;
    int numOfActionsFromTheSituation = 0;
    for (int k = 0 ; k< actions.length; k++){
        if (actions[k].source == evenSituations.get(j)){
            numOfActionsFromTheSituation++;
            if (actions[k] != requiringPolicyRules[i].action)
                numOfNotRequiredActions++;
        }
    }
    // if the required action is not possible
    if (numOfNotRequiredActions == numOfActionsFromTheSituation){
        if (!awareness (requiringPolicyRules[i], evenSituations.get(j))){
            awareness.add(new Awareness (evenSituations.get(j), requiringPolicyRules[i]));
        }
    }
}
IMPLEMENTATION OF METHOD INFORMATION AWARENESS

// Step2: Information Awareness
private void informationAwareness (){
    ArrayList<Situation> evenSituations = new ArrayList<Situation>();
    // finding all the situations that may happen now or some
time in future.
    // they have to be in the situation list of agent as well as
the current world.
    for (int i = 0 ; i < times.length ; i++) {
        if (times[i].tickIndex >= currentSituation.time.tickIndex) {
            for (int j = 0 ; j < situations.length; j++) {
                if (situations[j].time.tickIndex == times[i].tickIndex) {
                    for (int k = 0; k < currentSituation.world.situations.length; k++) {
                        if (currentSituation.world.situations[k].situationIndex == situations[j].situationIndex) {
                            if (isTherePath (currentSituation, situations[j])) {
                                evenSituations.add(situations[j]);
                            }
                        }
                    }
                }
            }
        }
    }
    // for all the forbidding policy rules
    for (int i = 0 ; i < forbiddingPolicyRules.length; i++) {
        for (int j = 0; j < evenSituations.size(); j++) {
            if (awareness (forbiddingPolicyRules[i],
evenSituations.get(j))) {
                if (!awareness (forbiddingPolicyRules[i].condition,
evenSituations.get(j))) {
                    awareness.add(new Awareness (evenSituations.get(j),
forbiddingPolicyRules[i].condition));
                }
            }
        }
    }
    // for all the requiring policy rules
    for (int i = 0 ; i < requiringPolicyRules.length; i++) {
        for (int j = 0; j < evenSituations.size(); j++) {
            if (awareness (requiringPolicyRules[i],
evenSituations.get(j))) {
                if (!awareness (requiringPolicyRules[i].condition,
evenSituations.get(j))) {
                    awareness.add(new Awareness (evenSituations.get(j),
requiringPolicyRules[i].condition));
                }
            }
        }
    }
}
// Step3: Change of Behaviours
private Action changeOfBehaviors (String agentStrategy){
    Action commitedAction = null;

    // finding the next possible actions.
    // they have to be in the current situation as well as the
    // world associated with the current situation.
    ArrayList<Action> nextActions = new ArrayList<Action>();
    for (int i = 0 ; i <actions.length; i++){
        for (int j = 0 ; j <currentSituation.world.actions.length; j++){
            if (actions[i].name ==
                currentSituation.world.actions[j].name){
                nextActions.add(actions[i]);}}}
        // with Volitional Strategy
        if (agentStrategy == "Volitional"){for (int i = 0 ; i< awareness.size(); i++){
            if (!implicitKnowledge (awareness.get(i).proposition)){
                for (int j = 0; j < nextActions.size(); j++){if (implicitKnowledge (awareness.get(i).proposition,
                    "inevitable", "optional", nextActions.get(j).destination)){
                commitedAction = nextActions.get(j);}}}
            if (commitedAction !=
                null)
                return commitedAction;
            else return changeOfBehaviors ("Cautious");}}}
        // with Cautious Strategy
        if (agentStrategy == "Cautious"){
            int num = 0;
            for (int i = 0 ; i< awareness.size(); i++){
                if (!implicitKnowledge (awareness.get(i).proposition)){
                    for (int j = 0; j < nextActions.size(); j++){
                        if (implicitKnowledge (awareness.get(i).proposition,
                            "inevitable", "optional", nextActions.get(j).destination)){
                            num++;
                            commitedAction = nextActions.get(j);}}}
                    if (num > 1)
                        return changeOfBehaviors ("Hasty");
                    return commitedAction;}
            }
        //with Hasty Strategy
        if (agentStrategy == "Hasty"){
            for (int i = 0 ; i< awareness.size(); i++){
                if (!implicitKnowledge (awareness.get(i).proposition)){
                    for (int j = 0; j < nextActions.size(); j++){
                        if (implicitKnowledge (awareness.get(i).proposition,
                            "inevitable", "optional", nextActions.get(j).destination)){
                            commitedAction = nextActions.get(j);}}}
                    return changeOfBehaviors ("Hasty");
                    return commitedAction;}
            }
```java
int length = -1;
for (int j = 0; j < nextActions.size(); j++) {
    if (implicitKnowledge (awareness.get(i).proposition, "optional", length, nextActions.get(j).destination)) {
        commitedAction = nextActions.get(j);
        // the deadline (length) here would be less or equal to the previous deadline
        length =
        lengthofOptionalEventualKnowledge (awareness.get(i).proposition, nextActions.get(j).destination);
    } else if (length == -1 && implicitKnowledge (awareness.get(i).proposition, "optional", "eventually", nextActions.get(j).destination)) {
        // the first time that the agent can implicitly know what it is aware of.
        commitedAction = nextActions.get(j);
        length =
        lengthofOptionalEventualKnowledge (awareness.get(i).proposition, nextActions.get(j).destination); }
}
return commitedAction;
return commitedAction;
```