Abstract

Technology has brought the Internet to mobile users. A wide range of multimedia content and services such as mobile TV, alerts, email and social networking services are being offered at competitive flat rates. Mobile operators and service providers are competing for users' loyalty as multimedia and content services have become a major source for revenues.

Dynamic adaptation is necessary due to the impracticality of static authoring of content versions for the wide range of delivery contexts. Moreover, dynamic user centric adaptation, rather than static device centric adaptation, is crucial for gaining and maintaining users' loyalty.

Content and service adaptation has become a complex problem due to the heterogeneity and dynamicity of the adaptation domain. Therefore, two key requirements are identified and investigated by this research: adaptation management and system extensibility. Adaptation management entails using several adaptation tools to offer personalised adaptation to users' delivery contexts and preferences. Extensibility is vital in all aspects of adaptation systems and ensures interoperability with existing and emerging content adaptation technologies.

This research focuses on two main aspects of adaptation systems: context processing and decision taking. Contributions include mechanisms to ensure efficient context assimilation and modelling that is extensible to new context formats and new adaptation scenarios. Novel approaches are proposed to exploit Semantic Web technologies for adaptation. This includes using Description Logics for rule-based adaptation decision taking, and expressive and formal context modelling.

An Adaptation Manager (AM), a major part in the content and service adaptation test bed, has been implemented to evaluate the contributions of this research. Evaluation results show that the proposed mechanisms succeed in contributing towards building extensible content and service adaptation management systems.

Key words: Adaptation Decision Taking, Adaptation Management, Content/Service Adaptation, Context Modelling, Description Logics, Extensibility, Ontology, Semantic Web, XML.

Email: a.attou@surrey.ac.uk
Acknowledgments

I would like to express my sincere appreciation to Prof Klaus Moessner, my first supervisor, for the valuable help and guidance he provided throughout my PhD.

I would also like to thank my second supervisor, Dr. Francois Carrez, and Dr Ning Li for the helpful discussions. I have enjoyed my stay at the Centre for Communication Systems Research, University of Surrey, and I would like to thank all mobile group members and support staff for their help.

I would like to express my deep gratitude to my family, especially my father Mustapha and my mother Mokhtaria, to whom I owe the success I could achieve in my life.

Very special thanks to my dear wife Sara, to whom I am indebted with unconditional support, care and encouragement, without which I could not get through difficult times.
# Contents

Abstract ................................................................................................................................... i  
Acknowledgments ................................................................................................................... ii
Contents .................................................................................................................................... iii
List of Figures ........................................................................................................................... ix
List of Tables ............................................................................................................................ xii
Glossary of Terms .................................................................................................................... xiii

1 Introduction ....................................................................................................................... 1
  1.1 Prologue .............................................................................................................................. 1
  1.2 Motivations and Objectives .............................................................................................. 3
  1.3 The Mobile Virtual Centre of Excellence Project ............................................................ 5
  1.4 Thesis Layout ...................................................................................................................... 6
  1.5 Original Contributions ..................................................................................................... 9

2 State of the Art and Background .................................................................................. 10
  2.1 Introduction ....................................................................................................................... 10
  2.2 Adaptation ........................................................................................................................ 11
    2.2.1 Adaptation Definition: Content vs. Service Adaptation ............................................. 11
    2.2.2 Adaptation Types ........................................................................................................ 12
  2.3 The Semantic Web ............................................................................................................ 13
    2.3.1 The Semantic Web Stack ............................................................................................ 13
  2.4 Reasoning .......................................................................................................................... 16
    2.4.1 Rule-Based Reasoning ................................................................................................ 16
    2.4.2 DL Reasoning (Ontology Reasoning) ........................................................................... 17
    2.4.3 Fuzzy Logic ................................................................................................................ 17
  2.5 Context .............................................................................................................................. 17
    2.5.1 Context Definition ...................................................................................................... 17
    2.5.2 MPEG-7/21 Context Concepts ................................................................................... 18
    2.5.3 Context Modelling ...................................................................................................... 21
  2.6 Content and Service Adaptation Systems ......................................................................... 22
    2.6.1 Content Adaptation Systems (CAS) ................................................................. 22
<table>
<thead>
<tr>
<th>Section Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.6.1.1 The Mobile VCE AMF</td>
<td>26</td>
</tr>
<tr>
<td>2.6.1.2 Categories of Content Adaptation Systems</td>
<td>28</td>
</tr>
<tr>
<td>2.6.2 Context-Aware Ubiquitous Services Adaptation Systems (CAUS)</td>
<td>30</td>
</tr>
<tr>
<td>2.7 Context in Content Adaptation Systems</td>
<td>32</td>
</tr>
<tr>
<td>2.7.1 Multimedia Adaptation Context Technologies</td>
<td>33</td>
</tr>
<tr>
<td>2.7.2 Content Adaptation Oriented Systems Context Modelling</td>
<td>34</td>
</tr>
<tr>
<td>2.7.3 Context Modelling in Context-Aware Ubiquitous Systems</td>
<td>35</td>
</tr>
<tr>
<td>2.7.4 Context in CAUS vs. CAS</td>
<td>36</td>
</tr>
<tr>
<td>2.8 Decision Taking in Adaptation Systems</td>
<td>37</td>
</tr>
<tr>
<td>2.8.1 Rule Based ADT</td>
<td>38</td>
</tr>
<tr>
<td>2.8.2 Search and Optimization Based ADT</td>
<td>38</td>
</tr>
<tr>
<td>2.8.3 Semantic Web Based ADT</td>
<td>39</td>
</tr>
<tr>
<td>2.8.4 Fuzzy Logic Based ADT</td>
<td>39</td>
</tr>
<tr>
<td>2.8.5 Mathematical Modelling Based ADT</td>
<td>39</td>
</tr>
<tr>
<td>2.9 Summary</td>
<td>40</td>
</tr>
<tr>
<td>3 Context-Aware AM</td>
<td>42</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>42</td>
</tr>
<tr>
<td>3.2 Content/Service Adaptation Requirements</td>
<td>43</td>
</tr>
<tr>
<td>3.2.1 Functional Requirements</td>
<td>43</td>
</tr>
<tr>
<td>3.2.1.1 Input Requirements</td>
<td>43</td>
</tr>
<tr>
<td>3.2.1.2 Functionality</td>
<td>45</td>
</tr>
<tr>
<td>3.2.1.3 Output Requirements</td>
<td>47</td>
</tr>
<tr>
<td>3.2.2 Non Functional Requirements</td>
<td>49</td>
</tr>
<tr>
<td>3.2.3 UML Modelling of Requirements</td>
<td>50</td>
</tr>
<tr>
<td>3.2.4 AMF Deployment</td>
<td>52</td>
</tr>
<tr>
<td>3.3 Formal System Specification</td>
<td>53</td>
</tr>
<tr>
<td>3.3.1 Formal Specification Terminology</td>
<td>53</td>
</tr>
<tr>
<td>3.3.2 Types</td>
<td>54</td>
</tr>
<tr>
<td>3.3.3 Functions</td>
<td>54</td>
</tr>
<tr>
<td>3.3.4 Algorithm</td>
<td>54</td>
</tr>
<tr>
<td>3.4 Proposed Solution: The AM</td>
<td>55</td>
</tr>
<tr>
<td>3.5 System Design</td>
<td>56</td>
</tr>
<tr>
<td>3.5.1 High Level Design</td>
<td>56</td>
</tr>
<tr>
<td>3.5.2 Low Level Design</td>
<td>57</td>
</tr>
<tr>
<td>3.5.2.1 ACM</td>
<td>58</td>
</tr>
</tbody>
</table>
6.3.2 Adaptation Context Assimilation ................................................................. 134
  6.3.2.1 Adaptation Operation Context Management ........................................ 134
  6.3.2.2 Context Extraction ................................................................................. 136
  6.3.2.3 Context Formatting with XSLT ................................................................. 137
  6.3.2.4 Context Formatting with Pre-reasoning Parameters Functions ................. 137
  6.3.2.5 Context Refinement using the ADIO ...................................................... 140
  6.3.2.6 Context Refinement using Ontology Reasoning Support Functions ............. 141
6.3.3 ADT ................................................................................................................. 142
  6.3.3.1 Ontology Based Rule Based Reasoning .................................................. 142
6.3.4 Interfacing ...................................................................................................... 144
6.3.5 Handling of Adaptation Requests ................................................................. 145
6.4 Summary ............................................................................................................. 145
7 Evaluation ............................................................................................................ 146
  7.1 Introduction ..................................................................................................... 146
  7.2 Evaluation Methods ......................................................................................... 146
    7.2.1 Performance Modelling and Analysis ....................................................... 146
    7.2.2 Implementation and Demonstration Scenarios .......................................... 147
    7.2.3 Performance Testing .................................................................................. 148
    7.2.4 User Validation ........................................................................................ 148
    7.2.5 Evaluation Methods in State of the Art Adaptation Frameworks ............... 149
    7.2.6 AM Evaluation .......................................................................................... 150
  7.3 Performance Modelling and Evaluation of the Design ....................................... 151
    7.3.1 The Adaptation Working Cycle ................................................................. 152
    7.3.2 Introduction to PEPA ................................................................................ 153
      7.3.2.1 PEPA Syntax ....................................................................................... 154
      7.3.2.2 Semantics and Performance Measures ............................................... 154
    7.3.3 System Analysis ...................................................................................... 154
    7.3.4 Parameter Settings for Performance Modelling and Model Checking ............ 156
    7.3.5 Performance Analysis ............................................................................. 158
      7.3.5.1 Response Time Analysis ................................................................. 158
      7.3.5.2 Throughput and Utilisation Analysis ............................................... 159
      7.3.5.3 Bottleneck Analysis .......................................................................... 161
  7.4 Performance Testing ......................................................................................... 162
  7.5 Non Performance Based Parameters Evaluation ............................................... 164
    7.5.1 System Extensibility Evaluation ............................................................... 165
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.5.1.1 System Extensibility to New Context Standards</td>
<td>165</td>
</tr>
<tr>
<td>7.5.1.2 System Extensibility to New Adaptation Operations</td>
<td>169</td>
</tr>
<tr>
<td>7.5.1.3 System Extensibility to New Adaptation Scenarios</td>
<td>171</td>
</tr>
<tr>
<td>7.5.2 System Maintainability Evaluation</td>
<td>174</td>
</tr>
<tr>
<td>7.5.2.1 Internal Context Model (ADI0) Modification</td>
<td>174</td>
</tr>
<tr>
<td>7.5.2.2 Context Formatting Modification</td>
<td>175</td>
</tr>
<tr>
<td>7.5.2.3 Adaptation Decision Rules Modification</td>
<td>176</td>
</tr>
<tr>
<td>7.5.3 System Interoperability</td>
<td>177</td>
</tr>
<tr>
<td>7.6 Conclusion</td>
<td>178</td>
</tr>
<tr>
<td>8 Conclusions</td>
<td>180</td>
</tr>
<tr>
<td>8.1 Summary</td>
<td>180</td>
</tr>
<tr>
<td>8.1.1 Prototype Implementation</td>
<td>182</td>
</tr>
<tr>
<td>8.2 Future Work and Research Topics</td>
<td>183</td>
</tr>
<tr>
<td>8.3 Research Application</td>
<td>184</td>
</tr>
<tr>
<td>8.4 Epilogue</td>
<td>184</td>
</tr>
<tr>
<td>Appendix A: MPEG Description Standards</td>
<td>185</td>
</tr>
<tr>
<td>References</td>
<td>193</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1: Research issues diagram ................................................................. 4
Figure 2: Thesis structure .............................................................................. 6
Figure 3: The RDF graph ................................................................................ 14
Figure 4: The Semantic Web stack ................................................................. 15
Figure 5: Content Adaptation Systems (CAS) ............................................... 23
Figure 6: PDE, AMF and adaptation .............................................................. 27
Figure 7: Management tasks at different levels of adaptation ...................... 30
Figure 8: Context modelling in Content Adaptation Systems (CAS) .......... 34
Figure 9: Context modelling in Context-Aware Ubiquitous Systems (CAUS) . 35
Figure 10: Adaptation spectrum ................................................................. 47
Figure 11: The AM functional requirements ................................................. 51
Figure 12: Deployment of content Adaptation Frameworks ......................... 52
Figure 13: Adaptation cycle algorithm ......................................................... 55
Figure 14: Content and service AM high level design .................................. 56
Figure 15: UML component diagram of adaptation domain components and interfaces .... 57
Figure 16: AM low level design .................................................................... 58
Figure 17: Main AM interfaces ..................................................................... 58
Figure 18: Adaptation context management by the ACM ......................... 67
Figure 19: Context assimilation and refinement ......................................... 73
Figure 20: Context assimilation algorithm .................................................. 74
Figure 21: Adaptation operations management .......................................... 80
Figure 22: ADIO low level concepts ............................................................ 89
Figure 23: Manual (a) vs. automatic (b) ontology construction .................. 91
Figure 24: MPEG-7 choice tag (1) ............................................................... 92
Figure 25: MPEG-7 choice tag (2) ............................................................... 93
Figure 26: MPEG-21 choice tag ................................................................. 94
Figure 27: Modelling device capabilities with automatic conversion .......... 95
Figure 28: Modelling device capabilities with manual conversion ............. 95
Figure 29: MPEG-21 audible frequency range .......................................... 97
Figure 30: MPEG-21 codec capability ....................................................... 98
Figure 31: AM ADT stages .......................................................................... 103
Figure 32: AM and DIA-ADT ................................................................. 104
Figure 33: ADT management algorithm ................................................................. 106
Figure 34: ADT management .................................................................................. 107
Figure 35: Rules ontology Structure .................................................................... 114
Figure 36: DL-based rule based reasoning steps ...................................................... 119
Figure 37: DL based rule reasoner implementation .................................................. 120
Figure 38: Content and service adaptation management test bed ................................ 125
Figure 39: System start-up ...................................................................................... 129
Figure 40: Types of requests .................................................................................. 130
Figure 41: Demonstrating adaptation stages ............................................................ 131
Figure 42: Inquiry request ..................................................................................... 132
Figure 43: Adapt request ....................................................................................... 132
Figure 44: AM implemented components ............................................................... 134
Figure 45: Adaptation OP XML ............................................................................ 135
Figure 46: Adaptation operations handling .............................................................. 136
Figure 47: Context Extractor interface .................................................................... 136
Figure 48: Adding formatters to the system ............................................................ 137
Figure 49: Pre-reasoning functions ....................................................................... 139
Figure 50: Pre-reasoning negative properties calculations ....................................... 140
Figure 51: Support functions for ontology reasoning .............................................. 141
Figure 52: Support functions for ontology based rule based reasoning ..................... 142
Figure 53: Adaptation rules written with the Protégé ontology editor ....................... 143
Figure 54: Handling of adaptation requests ............................................................ 145
Figure 55: AM working cycle .............................................................................. 153
Figure 56: AG PEPA Definition ........................................................................... 155
Figure 57: CP PEPA definition ............................................................................ 156
Figure 58: Adaptation system PEPA equation ....................................................... 156
Figure 59: Response time vs. calculating parameters and first layer decision taking .... 159
Figure 60: Impact of different functions on system throughput .............................. 160
Figure 61: Proportion of AP$_{set_{1l_{dec}}} vs. calc\_parameters$ and AP$_{set_{1l_{dec}}} vs. get\_types$ rates .................................................. 160
Figure 62: AG working sequentially: Response time vs. the number of APs .......... 161
Figure 63: AP's waiting proportion vs. the number of APs ..................................... 162
Figure 64: Throughput of dec\_msg vs. the number of APs .................................... 162
Figure 65: AM performance analysis .................................................................... 163
Figure 66: UAP prof and ADIO ........................................................................... 166
Figure 67: Adding the UAP prof formatter to the formatters' description ............... 167
Figure 68: UAP prof context formatter successfully loaded .................................... 167
List of Figures

Figure 69: The result of formatting displayed with Protégé ........................................................ 168
Figure 70: Added adaptation operation to the possibleOP.xml file ............................................ 170
Figure 71: Adaptation result shown on the phone emulator ........................................................ 171
Figure 72: Post fix expression described in XML for videoSummaryLengthThreshold .......... 173
Figure 73: Pre-reasoning functions modifications ....................................................................... 175
Figure 74: REL data model ........................................................................................................ 190
Figure 75: MVCO Structure ....................................................................................................... 192
List of Tables

Table 1: Content adaptation systems ................................................................. 29
Table 2: Context in CAUS vs. CAS ................................................................. 37
Table 3: Adaptation request parameters ......................................................... 44
Table 4: Adaptation mechanisms context ....................................................... 45
Table 5: Output of adaptation decisions to the adaptation mechanism management entity .... 48
Table 6: Adaptation response parameters ....................................................... 48
Table 7: Adaptation result parameters ........................................................... 48
Table 8: Interfaces from Figure 17 and Figure 15 .......................................... 61
Table 9: ADIO vs. ADO ............................................................................... 85
Table 10: Demonstration scenarios ............................................................... 128
Table 11: Ontology reasoning support functions .......................................... 142
Table 12: Ontology based rule based reasoning support functions .................. 143
Table 13: The WSDL web service interface: Adaptation requests ............... 144
Table 14: The WSDL web service interface: Adaptation response ............... 144
Table 15: Evaluation in content and service adaptation frameworks .............. 151
Table 16: Parameters setting for PEPA evaluation ........................................ 157
Glossary of Terms

3GPP 3rd Generation Partnership Project
ACM Adaptation context Manager
AdaptationQoS Adaptation Quality of Service
ADIO Adaptation Decision Interface Ontology
ADME Adaptation Mechanism
ADO Adaptation Domain Ontology
ADT Adaptation Decision Taking
ADTE Adaptation Decision Taking Engine
AF Adaptation Framework
AM Adaptation Manager
AM-ADE Adaptation Manager Adaptation Decision Engine
AMF Adaptation Management Framework
API Application Programming Interface
BAE Bitstream Adaptation Engine
BBL Bit-stream Binding Language
CA Content Adaptor
CAS Content Adaptation Systems
CAUS Context Aware Ubiquitous Systems
CC/PP Composite Capabilities Preferences Profiles
CMF Context Management Framework
CoBrA Context Broker Architecture
CS Content and Service
CSCP Composite Structure Context Profiles
CSS Cascading Style Sheets
DA Dispatcher Agent
DAML DARPA Agent Mark-up Language
DARPA Defence Advanced Research Projects Agency
DCC-MPR Maximum Profit Replacement with Dynamic Cash Categories
DCMI Dublin Core Metadata Initiative
DDL Description Definition Language
DECAFE Distributed Content Adaptation Framework
DIA Digital Item Adaptation
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DID</td>
<td>Digital Item Declaration</td>
</tr>
<tr>
<td>DIDL</td>
<td>Digital Item Declaration Language</td>
</tr>
<tr>
<td>DII</td>
<td>Digital Item Identification</td>
</tr>
<tr>
<td>DIP</td>
<td>Digital Item Processing</td>
</tr>
<tr>
<td>DL</td>
<td>Description Logics</td>
</tr>
<tr>
<td>DL-KB</td>
<td>Description Logics Knowledge Base</td>
</tr>
<tr>
<td>DME</td>
<td>Device Management Entity</td>
</tr>
<tr>
<td>DRM</td>
<td>Digital Rights Management</td>
</tr>
<tr>
<td>DS</td>
<td>Description Scheme</td>
</tr>
<tr>
<td>DTD</td>
<td>Data Type Definitions</td>
</tr>
<tr>
<td>EventsML</td>
<td>Events Mark-up Language</td>
</tr>
<tr>
<td>EXIF</td>
<td>Exchangeable Image File Format</td>
</tr>
<tr>
<td>FACAD</td>
<td>Framework for Context Aware Content Adaptation and Delivery</td>
</tr>
<tr>
<td>FL</td>
<td>Fuzzy Logic</td>
</tr>
<tr>
<td>GIF</td>
<td>Graphics Interchange Format</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>GSM</td>
<td>Global System for Mobile communications</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Mark-up Language</td>
</tr>
<tr>
<td>IBM</td>
<td>International Business Machine</td>
</tr>
<tr>
<td>IPMP</td>
<td>Intellectual Property Management and Protection</td>
</tr>
<tr>
<td>IPTC</td>
<td>The International Press Telecommunication Council</td>
</tr>
<tr>
<td>JESS</td>
<td>Java Expert System Shell</td>
</tr>
<tr>
<td>JPEG(JPG)</td>
<td>Joint Photographic Experts Group</td>
</tr>
<tr>
<td>JVM</td>
<td>Java Virtual Machine</td>
</tr>
<tr>
<td>KB</td>
<td>Knowledge Base</td>
</tr>
<tr>
<td>LP</td>
<td>Logic Program</td>
</tr>
<tr>
<td>MDS</td>
<td>Multimedia Description Schemes</td>
</tr>
<tr>
<td>MOV</td>
<td>QuickTime Movie</td>
</tr>
<tr>
<td>MP3</td>
<td>MPEG-1 Audio Layer 3</td>
</tr>
<tr>
<td>MPEG</td>
<td>Moving Picture Expert Group</td>
</tr>
<tr>
<td>MVCE</td>
<td>Mobile Virtual Centre of Excellence</td>
</tr>
<tr>
<td>MVCO</td>
<td>Mobile Value Chain Ontology</td>
</tr>
<tr>
<td>NewsML</td>
<td>News Mark-up Language</td>
</tr>
<tr>
<td>NF</td>
<td>Negation as Failure</td>
</tr>
<tr>
<td>NORST</td>
<td>Non Ordered Relation Score Tree</td>
</tr>
<tr>
<td>OIL</td>
<td>Ontology Inference Layer</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>OMA</td>
<td>Open Mobile Alliance</td>
</tr>
<tr>
<td>OP</td>
<td>Operation</td>
</tr>
<tr>
<td>ORM</td>
<td>Object Role Modelling</td>
</tr>
<tr>
<td>OWA</td>
<td>Open World Assumption</td>
</tr>
<tr>
<td>OWL</td>
<td>Ontology Web Language</td>
</tr>
<tr>
<td>OWL-S</td>
<td>Semantic Mark-up for Web Services</td>
</tr>
<tr>
<td>PAA</td>
<td>Personal Assistant Agent</td>
</tr>
<tr>
<td>PCM</td>
<td>Personal Content Manager</td>
</tr>
<tr>
<td>PDE</td>
<td>Personal Distributed Environment</td>
</tr>
<tr>
<td>PEPA</td>
<td>Performance Evaluation Process Algebra</td>
</tr>
<tr>
<td>PNG</td>
<td>Portable Network Graphics</td>
</tr>
<tr>
<td>Prolog</td>
<td>Programming in Logic</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>RDD</td>
<td>Rights Data Dictionary</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>RDFS</td>
<td>Resource Description Framework Schema</td>
</tr>
<tr>
<td>REL</td>
<td>Rights Expression Language</td>
</tr>
<tr>
<td>RM</td>
<td>Reasoning Manager</td>
</tr>
<tr>
<td>RMI</td>
<td>Remote Method Invocation</td>
</tr>
<tr>
<td>SLL</td>
<td>Scored Linked List</td>
</tr>
<tr>
<td>SOAP</td>
<td>Simple Object Access Protocol</td>
</tr>
<tr>
<td>SOCAM</td>
<td>Service Oriented Context-Aware Middleware</td>
</tr>
<tr>
<td>SportsML</td>
<td>Sports Mark-up Language</td>
</tr>
<tr>
<td>SVC</td>
<td>Scalable Video Coding</td>
</tr>
<tr>
<td>SWRL</td>
<td>Semantic Web Rule Language</td>
</tr>
<tr>
<td>UAPof</td>
<td>User Agent Profile</td>
</tr>
<tr>
<td>UCD</td>
<td>Universal Constraints Description</td>
</tr>
<tr>
<td>UED</td>
<td>Usage Environment Description</td>
</tr>
<tr>
<td>UML</td>
<td>Uniform Modelling Language</td>
</tr>
<tr>
<td>UMTS</td>
<td>Universal Mobile Telecommunications System</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>VDM</td>
<td>Vienna Development Method</td>
</tr>
<tr>
<td>VTP</td>
<td>Versatile Transcoding Proxy</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium</td>
</tr>
<tr>
<td>WAR</td>
<td>Web Application Archive</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>WBMP</td>
<td>Wireless Bitmap</td>
</tr>
<tr>
<td>WLAN</td>
<td>Wireless Local Area Network</td>
</tr>
<tr>
<td>WMAX</td>
<td>Worldwide Interoperability for Microwave Access</td>
</tr>
<tr>
<td>WML</td>
<td>Wireless Mark-up Language</td>
</tr>
<tr>
<td>WSDL</td>
<td>Web Service Description Language</td>
</tr>
<tr>
<td>WWW</td>
<td>World Wide Web</td>
</tr>
<tr>
<td>XHTML</td>
<td>Extensible HTML</td>
</tr>
<tr>
<td>XM</td>
<td>Experimental Model</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Mark-up Language</td>
</tr>
<tr>
<td>XMP</td>
<td>Extensible Metadata Platform</td>
</tr>
<tr>
<td>XSD</td>
<td>XML Schema Definition</td>
</tr>
<tr>
<td>XSL</td>
<td>Extensible Style Sheet Language</td>
</tr>
<tr>
<td>XSLT</td>
<td>Extensible Style Sheet Language Transformation</td>
</tr>
</tbody>
</table>
List of Publications

Patents

Journals

Conferences
1 Introduction

1.1 Prologue

Multimedia content and services have witnessed a significant proliferation in the past few years. The amount of available multimedia content on the web is increasing due to several factors, including the improved connectivity in terms of ubiquity of networking access, the availability of advanced multiple connectivity enabled mobile devices and the emergence of social networking multimedia platforms for video and image sharing such as YouTube [1] and Flickr [2]. Moreover, a variety of multimedia content and services are offered to mobile users at affordable prices, in some cases flat monthly subscriptions. Unlimited mobile Internet browsing has become an affordable add-on rather than a special feature.

In an environment characterised by the rapid emergence of trends relating to content and device technologies, content and service providers in particular are competing fiercely to gain a good customer base and hence a strong position in the multimedia content and services provisioning arena. For example, in the area of mobile browsing, Opera Software [3] has released its Opera mini [4] browser for mobile devices which is capable of rendering Extensible Hypertext Mark-up Language (XHTML) [5] to mobile devices and provides a zooming capability to view different sections of web pages. However, flash [6] content such as YouTube and BBC iPlayer [7] videos is not supported. SkyFire [8] has recently released the Skyfire browser [9] which is capable of playing any content on the web including flash videos with good performance. Moreover, cellular service providers are now competing in providing a range of services to users at competitive prices including mobile TV, news services, emails, multimedia content alerts services, such as sport alerts, adapted popular web site such as YouTube, etc. The provision of such services is now possible due to the advanced features available in mobile devices and the enhanced cellular network bandwidth. Furthermore, users are no longer restricted to using the operator's network as the latest devices are Wireless Local Area Network (WLAN) enabled. This opens opportunities to third party service providers to provide services without network operators' restrictions.

As the range of multimedia content and services targeted to mobile users has become more varied, and as mobile Internet access via powerful mobile devices has become an ordinary rather than a special feature, the following has become apparent:
• Statically adapted content/services do not make full use of users’ devices, and cannot provide a personalised user experience. This is due to the fact that the number of statically adapted content versions to be generated for the available devices is large.

• With regard to content and service adaptation, it is important to shift the focus to users rather than to their devices [10], i.e. provide user centric adaptation to enhance the user experience. Device centric adaptation to merely fit content and services to users’ devices is certainly not the decisive factor to win and maintain their loyalty.

• Apart from price differences, a personalized service to user needs/communication capabilities is the mean by which service providers attract customers, gain and maintain their loyalty, hence strengthen their position in the market.

Therefore, to endure the competition, service providers need to be able to reach a wider range of users by adapting their services to a wide range of delivery contexts. Ideally, a multimedia service is dynamically and uniquely customised and personalised to each user. Such a service would need to be adapted at different stages, starting from content selection to presentation adaptation for the delivery device.

Upon receiving requests from (or pushing a service to) users, certain content and functionality are selected. Content selection is performed according to users’ preferences, browsing history and their current situation. Content selection is particularly important to push-type services, for example registering for specific alerts such as sport news. Once content and functionality are selected, the necessary adaptations that need to be performed on them are decided. These adaptations are necessary to fit services to delivery environments capabilities and personalise them to users’ preferences. Adaptation decision taking (ADT) involves input and output modalities (audio, video, images and text), coding formats and parameters (file size, resolution, bit rate, code size, etc). This process also needs to be aware of available adaptation operations. In accordance with this process, the presentation and the structure of the service is defined to maximise users’ experience based on their browsing preferences.

Selection, ADT and presentation adaptation are all important stages in the lifecycle of a service that is uniquely personalised to users and that maximises their experiences. The three stages are subjects of continuing research. Adaptive content selection and presentation including navigation have been investigated by the adaptive hypermedia and web systems community [11], whereas content ADT has been of interest to the multimedia communications communities [12].
1.2 Motivations and Objectives

Current research efforts focus on developing adaptation systems to achieve certain adaptation tasks, for example to adapt video content to fit small displays by removing less important regions from video frames. However, it is evident that content and service adaptation is becoming a more complex problem due to the availability of different content types and formats, the wide range of delivery environments and the availability of several adaptation techniques. Therefore, it is important to investigate management mechanisms for adaptation systems to manage the complexity and ensure extensibility, interoperability and ease of maintenance.

Moreover, existing research in this field focuses on ADT and content adaptation for constrained environments such as mobile environments. However, the processing and modelling of context was either treated superficially or not treated at all. Context is a core component in adaptation systems and is vital to all stages of the adaptation. The amount of information in context profiles defines the extent to which the multimedia service can be tailored to users. The efficiency of context processing and modelling greatly affects the quality of adaptation, the efficiency of the adaptation system and the extensibility of the ADT logic to new adaptation scenarios. The topic of content and service adaptation context, including how different standards and technologies can be employed for efficient modelling and processing of contextual information, needs to be investigated.

Adaptation context includes any information that describes entities involved in the delivery of services to users including users and services. Thus, context encompasses the description of entities such as users, devices, access networks, usage environments, adaptation operations, content and services.

Semantic Web [13] technologies are enjoying wide support from the research community, several applications such as reasoners and development environments are available, and possible enhancements and optimizations to core technologies such as Description Logics (DL) [14] are being investigated. Semantic Web technologies will play a vital role in tomorrow’s web; therefore, it is important that an adaptation system is Semantic Web compatible.

Figure 1 depicts a summary of the research issues targeted in this thesis. Adaptation systems need to provide mechanisms to manage different adaptation tools in order to adapt for the wide range of delivery contexts. The heterogeneity of adaptation tools, delivery environment capabilities and user requirements introduces complexity in the adaptation domain. Paramount to solving this complexity is adopting mechanisms to ensure interoperability and extensibility with existing and new adaptation tools including context formats, ADT techniques, adaptation operations, etc. Extensibility is more important as it insures interoperability. Technologies that are used in this
thesis to ensure extensibility and interoperability include the Moving Pictures Expert Group (MPEG) 21 [15] and MPEG-7 [16] standards, the User Agent Profile (UAProf) [17] standard, the Extensible Mark-up Language (XML) [18], DL, rule based ADT, Extensible Style Sheet Language Transformation (XSLT) [19], etc. Defining clear interfaces between adaptation systems components and adopting a modular design are also crucial.

![Research issues diagram](image)

**Figure 1: Research issues diagram**

This research aims to study the issues identified above, and investigate technologies and tools that can be used to solve them. Therefore, the objectives of this research are as follows:

- Survey content and service adaptation systems and classify them according to their functionality and purpose.
• Investigate content and service adaptation management mechanisms in state of the art adaptation systems.

• Determine key stages where adaptation management is important in the content and service adaptation process.

• Identify key system features that are important to a content Adaptation Management Framework (AMF) and propose mechanisms to ensure these features are satisfied.

• Examine the different technologies used for content and service adaptation management.

• Propose mechanisms for adaptation management for key parts of the content and service adaptation process, namely, ADT and context management.

• Investigate the use of Semantic Web technologies for content and service adaptation management.

It is important to note that it is not the aim of this research to introduce yet another content adaptation system, for example for improved performance or extended adaptation functionality. Rather, the aim is to study adaptation systems and identify sources of complexity and introduce mechanisms to manage the complexity in order to ensure important system features such as extensibility and maintainability. The author believes that research in this particular area is lacking.

1.3 The Mobile Virtual Centre of Excellence Project

This research contributes to the Mobile Virtual Centre of Excellence (Mobile VCE) project [20] named “Removing the Barriers to Ubiquitous Services”, which aims at defining novel mechanisms to overcome the barriers to the successful deployment and adoption of ubiquitous services. Such mechanisms can be considered at three levels: the user level [21], the network level [22] and the content and service level [23]. At the user level, the project aims to provide mechanisms to hide the complexity of the delivery environments from the user, for example, by proposing intelligent entities that act as agents on their behalf. At the network level, proposed mechanisms aim to combine mobility, security and quality of service mechanisms for better content and service delivery. This research relates to the content and service level, which aims to provide mechanisms to enable content and service personalisation to the user preferences and delivery environment capabilities. Detail on proposed MVCE mechanisms are presented in Section 2.6.1.1.
1.4 Thesis Layout

This thesis consists of 8 chapters described below. The relationship between the chapters is depicted by Figure 2.

- Chapter 1: Introduction

This chapter serves as an introduction to the thesis; it introduces the area of research and identifies the motivations and objectives. The research methodology is explained and the structure of the thesis is presented by outlining the content of each chapter. The chapter concludes with a list of author’s contributions to the research domain discussed in this thesis.

- Chapter 2: Content and Service Adaptation State of the Art

Chapter 2 surveys the state of the art in content and service adaptation. The domain is rather vast, and several concepts and technologies are explained in the chapter such as adaptation, context, content and services, Semantic Web technologies, reasoning and decision taking.

There exist a large number of systems and frameworks developed for content adaptation. This chapter surveys the most important ones and classifies them into categories to ensure that all types of systems are included.

In addition to content adaptation frameworks, the chapter surveys context aware ubiquitous services frameworks in order to study context-awareness aspects, because context awareness is
not well developed in content and service adaptation systems despite the fact that it is crucial in today's content and service adaptation domain as explained in Section 1.1.

Context modelling and ADT in surveyed systems are discussed and relevant context and ADT technologies are listed and explained.

- **Chapter 3: Adaptation Manager (AM)**

  This chapter introduces the AM, a central entity in a context aware content and service AMF. The aim of the AM is to manage the adaptation process to achieve relevant and accurate adaptation according to the delivery context. This research focuses on management aspects of adaptation rather than on how the adaptation is performed. Adaptation management is crucial because of the increasing complexity of the adaptation domain. This aspect has not been adequately treated in state of the art adaptation systems.

  The chapter introduces the formal specification methodology that is used in Chapters 3, 4 and 5. The formal methodology used is largely based on the Vienna Development Method (VDM) [24]. The chapter specifies the requirements, architecture and design of the AM entity. Two main components that are specified are the Adaptation Context Manager (ACM) and the Adaptation Decision Engine; these will be discussed in detail in Chapters 4 and 5 respectively where the main contributions of this thesis are presented.

- **Chapter 4: Adaptation Context Management**

  This part of the thesis discusses adaptation context management. Certain aspects of adaptation context processing such as extensible context formatting have not been adequately treated in state of the art content adaptation systems. Several features of an adaptation system depend on context processing and modelling because adaptation depends on context input. The main problems with regard to content adaptation context are heterogeneity and dynamicity of context parameters due to the changing user requirements and the volatility of content and delivery environment technologies. In order to cope with these problems, adaptation systems require extensibility to new technologies/adaptation scenarios and interoperability with existing technologies. This chapter presents an adaptation context management framework to ensure these two vital requirements are met. The framework constitutes two major parts. The first part is a context assimilation interface to assimilate context from different heterogeneous sources. The second part is a higher level context interface to absorb changes to the context model and reduce the impact on the rest of the adaptation system, specifically, to ADT.

  Finally, the chapter investigates how Semantic Web technologies can be applied to context modelling for content and service adaptation. Specifically, the chapter examines efforts to convert
Chapter 1. Introduction

XML based standards to ontology. The author compares the available approaches and discusses the advantages and disadvantages of each.

Chapter 5: Adaptation Decision Taking

This chapter discusses the second major component in adaptation systems, i.e. ADT. For an effective and accurate adaptation, several adaptation decision technologies have to be employed. Two key decision technologies that are necessary are rule based ADT and the MPEG-21 Digital Item Adaptation (DIA) ADT Engine (ADTE). The first can be used to implement adaptation rules and to manage several specialised ADT techniques while the second is used to accurately select adaptation parameters.

One of the objectives of this thesis is to investigate the use of Semantic Web technologies, for example DL, in adaptation systems. This chapter presents a novel mechanism to use DL for rule based reasoning. The approach can be applied to scenarios other than content adaptation where reasoning revolves around a central entity; the central entity in this example is the user. This approach can be extended to scenarios that do not have a central entity, for example reasoning about relationships between family members. This requires implementing a variables management layer to associate rules with the corresponding variables. In the family reasoning example, the variables represent family members.

Chapter 6: Implementation

The implementation chapter provides a proof of concept demonstration to prove the feasibility of the contributions presented in this thesis. The proof of concept demonstration is run as a demonstrator application. A prototype of the AM with the context management and the ADT subsystems is implemented. The AM is a central part in the content and service adaptation test bed described in the chapter. The chapter presents key implementation technologies and tools and explains key AM implementation aspects.

Chapter 7: Evaluation

Evaluation can be considered as the most important part of research. Evaluation can help in assessing the validity of the author’s contributions and determines the extent to which the aims and objectives have been achieved.

The chapter starts with presenting different evaluation techniques used for similar systems in the state of the art, and identifies which evaluation techniques are used for the AM. It is noted that the aim of this research is related to non functional requirements and hence evaluation takes this into account. Therefore, the main method of evaluation is based on demonstration scenarios, for example, an extension to the context model scenario to evaluate the extensibility of the adaptation
context management system, a very important feature targeted by this research. However, other
types of evaluation are also used, for example performance testing to evaluate the viability of the
implementation in terms of performance.

- Chapter 8: Conclusions

The conclusion chapter concludes the thesis by providing a summary, conclusions and directions
to future work and research topics.

1.5 Original Contributions

The contributions of this thesis are as follows:

- Adaptation Context

  - An adaptation context management framework to manage the heterogeneity of
    context profiles. The framework consists of an adaptation context assimilation
    interface and a high level adaptation context interface. The high level adaptation
    context interface defines the Adaptation Decision Interface Ontology (ADIO) using
    the latest expressiveness extensions of the Ontology Web Language (OWL) 2.0

  - A comparison between manual and automatic conversion of XML based context
    description standards to ontologies, including illustrating the inefficiency of
    automatic conversion and the necessity of manual conversion.

- Adaptation Decision Taking

  - A definition of an adaptation decision framework consisting of two main layers of
    ADT where the first layer is based on rule based reasoning and the second layer is
    based on the MPEG-21 DIA-ADTE.

  - A novel approach to use a Descriptions Logics based language such as OWL 2.0 to
    perform rule based ADT.

The contributions are described in detail in Chapters 3, 4, 5 and 6. Chapter 2 provides a
background to the content and service adaptation domain and an overview of the main related
work.
2 State of the Art and Background

2.1 Introduction

Numerous contributions related to context aware content and service adaptation have been proposed. This chapter aims to survey main systems and related technologies to content and service adaptation management.

The chapter starts with the definition of core concepts for this thesis such as adaptation, content vs. service and context in Sections 2.2 and 2.5. Relevant technologies and systems are surveyed in the rest of the chapter.

Sections 2.3 and 2.4 present Semantic Web technologies, which are a very important set of technologies for the web in general and for content adaptation in specific. Semantic Web has gained momentum lately and its technologies such as ontologies have been applied to a number of domains.

Section 2.6 presents the state of the art in context-aware adaptation systems. Two types of adaptation systems are presented, content adaptation systems (CAS) and context-aware ubiquitous systems (CAUS). The author’s contribution is related to content adaptation systems (CAS) which are classified in different categories in Section 2.6.1.2. However, context modelling is more developed and enhanced in context-aware ubiquitous systems (CAUS). Therefore CAUS are surveyed to learn lessons with regard to context modelling aspects.

Context modelling and ADT are the main components of adaptation systems. Section 2.7 and Section 2.8 present the state of the art of context modelling and ADT respectively for both CAS and CAUS. The different technologies that are used are presented and explained. Most of the surveyed context modelling approaches are from CAUS; this is because context awareness in CAS is not as developed as in CAUS. This chapter only presents context modelling approaches in the latest CAUS, all of which are based on Semantic Web technologies.

Decision taking in content adaptation systems can occur at different stages, and different technologies can be used. This is discussed in Section 2.8. Finally, Section 2.9 concludes the chapter.
2.2 Adaptation

Adaptation is a core concept to this research and is used in different contexts in the state of the art. Section 2.2.1 defines what is meant by content and service adaptation in this research. Section 2.2.2 presents the different types of adaptation according to adaptation location, time of occurrence and purpose.

2.2.1 Adaptation Definition: Content vs. Service Adaptation

Content adaptation [25] can be defined as “a process of selection, generation or modification that produces one or more perceivable units in response to a requested Uniform Resource Identifier [26] in a given delivery context”. A perceivable unit is defined in [26] as a: “set of material which, when rendered by a user agent, may be perceived by a user and with which interaction may be possible”. Adaptation has become important due to the diversity in both delivery contexts and requested content. The delivery context is characterised by the user environment, their preferences and their device and access technology. To distinguish between content and service adaptation, content and services need to be defined.

- Content vs. Service

Content can be defined as the object requested by the user and delivered by the requested service. The object may be text, audio, speech, image, video or a multimedia document. On the other hand, a service can be defined as the actions taken upon the object such as delivering and presenting. For example a service may deliver news to mobile users, the news service contains content such as multimedia or single media objects.

- Content Adaptation vs. Service Adaptation

Content adaptation can be defined as the changes performed upon the information contained in content whereas service adaptation can be defined as the changes to the way the content is delivered. For example, if the user requests a news service, content adaptation would be removing, summarising or translating parts of the news content elements while service adaptation would be changing the resolution of image items or changing the coding format for other items to meet the bandwidth or display requirements.
2.2.2 Adaptation Types

Adaptation can be divided into types according to its time of occurrence [27], location [27] and purpose [12].

- **Time of Occurrence**

Time wise, adaptation can be either static or dynamic. Static adaptation produces a fixed set of adapted versions prior to user requests targeted at different delivery contexts. At request time, the appropriate version is selected and rendered to the user device. Dynamic adaptation on the other hand generates a content and service version at request time (dynamically) to suite the delivery context. The latter approach is favourable over the former because the static approach is expensive in terms of storage and maintenance and unable to cover all delivery contexts especially in today’s volatile content and service delivery domain.

- **Adaptation Location**

Adaptation can be classified in three types according to where it is controlled [28]: server-side adaptation, intermediate adaptation and client-side adaptation if the adaptation process takes place at the server, an intermediate proxy or at the client device respectively. Client side adaptation is controlled by the end terminal. The advantage of this method is that it has direct access to the client characteristics. However adaptation is very limited due to the limitations of mobile devices. An example of adaptation is styling web documents using Cascading Style Sheets (CSS) [29] and XSLT [19]. Proxy adaptation occurs at an intermediary proxy and has the advantage of reducing the complexity at the client and content/service provider sides. Neither the server nor the client needs to be changed. However, the disadvantage is that content authors have less control over their content. Examples of proxy adaptation systems are [30-32]. The third type of adaptation is content/service provider based such as in [33, 34]. This type of adaptation requires more resources at content servers, and used to be preferred where service providers needed to control how their content is adapted. However, Digital Rights Management (DRM) mechanisms [35] are now available to enable service providers to control adaptation outside their servers [36-38]. MPEG-21 includes tools for Right Expression Language (REL) in Part 5 of the standard [39] and tools for Right Data Dictionary (RDD) in Part 6 of the standard [40]. REL can be used to specify rights and permissions on content using terms defined in RDD. For example, the system in [37] uses DRM information specified in REL to restrict content adaptation of multimedia content in a virtual classroom collaboration environment in order to manage digital rights and protect intellectual property.
• Adaptation Purpose

Purpose wise, adaptation can be divided into three categories [12]: content selection, content adaptation and presentation adaptation. Content selection aims to deliver only content that is of interest to users, this can be applied to both push or pull services, however, it is more important for push services. The selection can be based on a number of parameters including user preferences, location, and usage history. Content Adaptation deals with calculating parameters of an adapted version that fits to the user delivery environment. This includes transcoding between formats to save resources or cross modal adaptation such as video to audio adaptation. Presentation adaptation aims at presenting the content in the most suitable way for ease of browsing and to meet user presentation preferences. This is important for example for accessing web page content on mobile devices because of screen size limitations.

2.3 The Semantic Web

The Semantic Web is an effort to extend the current web with semantics and enable data to be shared and processed among different players. It is defined by the World Wide Web (WWW) inventor Tim Berners-Lee in [13] as “An extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation”. Information should be located and accessed efficiently. Computers and applications should be able to process both data syntax and semantics. The next section presents the Semantic Web stack.

2.3.1 The Semantic Web Stack

Figure 4 [41] depicts the technologies that constitute the Semantic Web. At the bottom are two technologies essential to the existence of the WWW; Unicode and the Uniform Resource Identifier (URI). The next layer is XML for data syntax definition. On top of XML is the Resource Description Framework (RDF) which provides a mean to define relationships between resources on the WWW. RDF and the three layers above it deal with knowledge representation and inference. Knowledge represented in ontologies together with rules can be used in reasoning to process information. The top layer of the stack is one objective of the Semantic Web, i.e. to create an environment in which applications can verify the trustworthiness of information. A brief description of the Semantic Web stack core elements is given below:

• XML [42]

The Extensible Mark-up Language is a text format developed by WWW Consortium (W3C) to enforce a logical document structure and a strict syntax. XML documents consist of a set of legal
elements. A Document Type Definition (DTD) or an XML Schema Definition (XSD) is used to define the document structure and a list of legal elements. XML enforces the document syntax; however, it does not provide a mechanism for structuring the meaning of data. RDF, described below, can be used for such a purpose.

- **RDF [43]**

The Resource Description Framework aims to represent information on the web. It provides metadata interoperability across different resource description communities and applications. RDF can create descriptions that are both human-readable and machine processable. It represents the semantics of a description using a graph data model. The term graph is used because a collection of triples is used to represent the underlying structure of any expression in RDF. Each triple consists of a subject (class), a predicate (or property) and an object (value) where the predicate represents the relationship between the subject and the object. The nodes on the RDF graph are its subjects and objects and the arc represents the property. Arc direction is always from the subject towards the object.

![RDF Graph](image)

**Figure 3: The RDF graph**

A metadata scheme such as the Dublin Core Metadata Initiative (DCMI) standard [44] can use XML to describe its syntax and RDF to structure the meaning of the elements by describing resources with classes, properties and values. The DCMI provides standard terms to describe aspects of content such as title, creation/modification date, creator information, language, format, access rights, etc. These terms are structured as RDF triplets, for example: subject: video1, property: title, object: wild life in Africa.
Chapter 2. State of the Art and Background

Figure 4: The Semantic Web stack

- **RDF Schema (RDFS)**

RDFS is a semantic extension to RDF. It provides a framework for describing application-specific classes and properties. It is an RDF vocabulary description language that describes groups of related resources (grouped into classes) and the relationships between them.

- **Ontology [45]**

Ontologies are increasingly important in computer science, especially in the artificial intelligence domain because they relate to a powerful knowledge representation technique: DL. Ontology is used to define a domain by specifying concepts and relationships among them. DL provides a formalism to implement ontologies and OWL is the most widely used DL based ontology language.

- **DL**

DL is a family of knowledge representation languages that provide a description of a domain in terms of concepts and roles between them, based on the first order predicate logic. DL languages vary in expressiveness, complexity and decidability. Decidability and computational completeness are important features which ensure that the validity of a statement can be computed and in finite time. DL languages are powerful ontology formalisms. A number of XML syntax based DL languages have been developed; including Ontology Web Language (OWL) which is widely used and has become a W3C recommendation. OWL 2.0 is the latest version of OWL that provides maximum expressiveness without losing decidability and computational completeness.
• **OWL [46]**

OWL is a semantic mark-up language for publishing and sharing ontologies on the World Wide Web. It is an extension to RDF Schema (RDFS) [43] and provides additional vocabulary with formal semantics. OWL is based on a previous effort to build ontology languages and tools for the Semantic Web, The DARPA Agent Mark-up Language (DAML) program [47] which ended in early 2006 and was initiated by the Defence Advanced Research Projects Agency (DARPA) [48].

OWL has two versions; OWL 1.0 and OWL 2.0. OWL 1.0 has three sub languages: OWL Lite, OWL DL, and OWL Full. OWL 2.0, introduced lately, provides more expressiveness than OWL 1.0. OWL languages facilitate the machine-processing of information by providing an expressive mechanism for giving meaning to terms in vocabularies. The meaning of these terms and the relationship between them is represented by ontology. Ontologies are used to model contextual information in context-aware systems as presented in Section 2.7.3.

### 2.4 Reasoning

Reasoning in Artificial Intelligence is the process of deriving new facts from knowledge and initial facts [49]. As far as context is concerned, there are two main types of reasoning [50]: ontology (DL) reasoning and user defined rule-based reasoning. Fuzzy logic can be used in combination with the two main types of context reasoning to handle uncertainty.

#### 2.4.1 Rule-Based Reasoning

Rule-based systems are part of the logic programming paradigm. A Logic Program is specified using a rule language and is executed by a rule engine. The two main types of rule languages currently in use are Prolog and Datalog [51]. Depending on the asserted facts, the rule engine fires the appropriate rules in two main methods: forward chaining or backward chaining. Forward chaining is a bottom up approach where the reasoner starts with a set of conditions to arrive at a certain goal. Backward chaining is a top-down approach where the reasoner starts from a goal and tries to find out how to achieve it.

User defined rule based reasoning is useful to deduce high level context from low level context such as deducing the intention of the user or their activity from a set of facts. For example, if the user is in their bedroom, the lights are off and the time is 11 pm then the user is sleeping.
2.4.2 **DL Reasoning (Ontology Reasoning)**

As opposed to rule based reasoners, DL reasoners perform different reasoning tasks such as classification and realization. Classification compares every two concepts (classes) in the ontology to deduce the relationships between them. Realization computes the types (the corresponding classes or concepts) of every individual (instance) in the ontology. For example, in the content and service adaptation scenario, the instances N70, N95 and Dell 2532 are members of the concepts or classes Device, Machine and Network-Enabled-Device. Furthermore, the concept Network-Enabled-Device can be defined as a sub-concept of the concept Device. A number of DL (or ontology) reasoners have been developed such as Racer [52], Pellet [53] and Fact ++ [54].

Rule based reasoning can be combined with ontology reasoning using a rule language that is built on top of an ontology language. For example, the Semantic Web Rule Language (SWRL) [55].

2.4.3 **Fuzzy Logic**

Fuzzy Logic (FL) [56] can be used to handle uncertainty. It is useful in arriving at decisions from vague, noisy, imprecise or incomplete input by means of the mathematical concept of fuzzy sets. Fuzzy logic reasoning involves three steps: fuzzification, fuzzy reasoning and defuzzification. The fuzzification step produces fuzzy inputs from crisp inputs by applying mathematical functions called membership functions. The reasoning stage produces fuzzy outputs from the fuzzy inputs using different inference and fuzzy rules. The defuzzification process combines the fuzzy outputs and produces one crisp output that represents the decision of the system.

2.5 **Context**

Context is a core element for adaptation systems. Section 2.5.1 provides a definition of context and Section 2.5.2 presents its types. In each type, relevant parameters are listed. Section 2.5.3 surveys existing context modelling approaches.

2.5.1 **Context Definition**

Many definitions of context exist in the literature, one definition is: *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves"* [57]. The MPEG-7 Multimedia Description Scheme (MDS) [58] and MPEG-21 DIA [59] standards provide tools to describe context concepts relating to
multimedia content and multimedia adaptation, respectively. Main context concepts described are listed below.

2.5.2 MPEG-7/21 Context Concepts

• Terminal Context

A terminal in DIA is any device involved in the delivery of content to the user including mobile devices, television, audio/video players, computers, intermediate network nodes and servers. Examples of terminal capabilities specified by DIA are:

  o Codec capabilities include codec parameters such as average/maximum bit rate and buffer size for the available codecs such as image, video, audio or graphics encoders and decoders for different content formats.
  o Display capabilities describe the displays available to the user. Parameters to describe displays include resolution, screen size, colour capabilities, bits per pixel, brightness, etc.
  o Audio output capabilities specify parameters such as sampling frequency, bits per sample, signal to noise ratio, supported number of channels, etc.
  o User interaction capabilities describe the input modes supported by the terminal, for example, characteristics of available input devices such as a microphone, mouse, trackball, keyboard, etc.
  o Power capabilities include parameters such as average power consumption, remaining battery capacity and remaining battery time.
  o Storage capabilities describe parameters such as size, whether it is writable, input/output transfer rates, etc.
  o Processing power specify the performance of the Central Processing Unit in terms of the number of integer or floating-point operations per second.

• Network Context

DIA describes network context in two main categories as follows:

  o Network capability describes static network properties such as the maximum capacity and the minimum guaranteed bandwidth.
  o Network condition describes dynamic network properties such as available bandwidth, error and delay. The available bandwidth specifies maximum, minimum and average values. The error is specified in terms of packet loss rate and bit error rate. Network condition context includes a time stamp to indicate the time the measurements were taken and how long they are valid for.
Chapter 2. State of the Art and Background

• User Context

DIA specifies various user context parameters including:

- User info describes general user information using the Agent Description Scheme (DS) defined by MPEG-7 MDS. The user can be a person, a group of people or an organization. A DS describes how context parameters are used to describe concepts. For example, the Agent DS describes how context parameters such as name, address, phone number, etc are used to describe user information.

- User history is also based on MPEG-7 MDS, for example, it specifies how much certain content is used by the specifying the usage duration.

- User preferences describe information such as the types of content the user is interested in, such as sport, entertainment, news content, etc. It also describes audio preferences such as volume control, audio output device, etc. Display preferences are also specified including colour preferences, modality conversion preferences (e.g. preferring video to image conversion rather than video to audio in case adaptation is required) and priority preference (e.g. preferring certain content types, modalities or formats over others).

- Accessibility characteristics define visual or auditory impairments the user might have. Audio impairments are specified for both right and left ears in terms of hearing threshold. Visual impairments include colour vision deficiency, blindness, low vision symptoms, etc.

- Location characteristics are specified in terms of mobility and destination. Mobility is specified using directivity, update intervals and erraticity. Directivity specifies the relative angular changes in user direction. Erraticity specifies the degree of randomness in user movements, for example, to detect if the user is walking or driving. Update intervals specify parameters such as the last update location, last update time, and the new location.

• Natural Environment Context

DIA provides context parameters to describe the multimedia consumption environment such as illumination, noise levels, location and time of using the content. Both time and location are specified using the MPEG-7 Place DS and the Location DS. For example, the Place DS defines the name of the place, the coordinates, the type of the coordinates, the country, the region, etc. The Time DS and Location DS are also used by MPEG-21 DIA to describe mobility and destination.
• **Content Context**

Content context is specified by MPEG-7 MDS which provides context description tools for multimedia content. This includes context tools for content description, content management, content organization and user interaction.

- Content description tools include structural and semantic tools. Structural tools describe the content in terms of spatio-temporal segments organized in a hierarchical structure. Semantic description tools describe the content in terms of the information it contains and its relationship to other content using real world concepts such as events, places, objects, people, etc.

- Content management tools define media description tools, usage description tools and media information tools. Media description tools define parameters for video, audio, text and image content such as coding format, file size, bit rate, frame rate, resolution, etc. Usage tools describe context including how, when, where and by whom the content can be used. The creation tools provide information about the content creation process, for example, the author, director, target audience, etc.

- Content organization tools include parameters to organize related content in collections, for example to describe an album of songs. Tools are also provided to describe relationships between different collections.

- User interaction tools provide context parameters to describe user preferences and usage history.

• **Digital Rights Context**

MPEG-21 REL and RDD provide tools to describe digital rights context. RDD specifies terms that are used by REL to express digital rights, this includes terms to specify permitted actions on content such as *Adapt, Delete, Diminish, Embed, Enlarge, Modify, Print, Reduce*, etc.

Other types of context not specified by MPEG-21 and MPEG-7 include

• **Layout and Structure**

Layout and structure context describes the relationships between content items with regard to presentation. This includes relationships between content items on web pages to detect coherent web page blocks. Such context is not standardised and is automatically generated and used to break a web page into coherent blocks to improve accessibility on small displays such as in [60, 61]
2.5.3 Context Modelling

The survey in [62] presents context modelling approaches including key-value pairs, e.g. [63], rule based modelling, e.g. [64], object oriented modelling, e.g. [65], ontology modelling, e.g. [66] and mark-up modelling, e.g. [32]. A context model is characterised by its syntax and semantics (data model).

- Context Models Syntax

Most context models use XML based mark-up modelling due to the formality and interoperability XML provides with regard to syntax. Main multimedia context technologies that use XML based mark-up are MPEG-7 and MPEG-21, which provide tools to describe content and adaptation related context parameters discussed in the previous section. Other examples include the Composite Capabilities/Preference Profile (CC/PP) [67], the User Agent Profile (UAProf) [17] and the Comprehensive Structured Context Profiles (CSCP) [68].

- Semantics of Context Models

Object oriented, key value, ontology and rule-based relates to the semantics of a context model rather than its syntax. For example, a key value context model can use XML or a simple text format to represent key value pairs. With the introduction of the Semantic Web, latest context models use Semantic Web technologies, especially ontology and its underlying mathematical formalism, DL. The importance of using Semantic Web technologies such Ontology, to provide a formal definition of the underlying semantics and hence provide semantic interoperability (formal and shared understanding) has been recognized by the multimedia community [41]. Ontology specifies context concepts and the relationship among them. Various frameworks are using ontology to model context due to their expressiveness and formality; examples of such systems are presented in Section 2.7.3 and include [38, 50, 69-74]. CC/PP, UAProf and CSCP use the RDF data model explained in Section 2.3.1. MPEG-7 and MPEG-21 focus on syntax interoperability and extensibility; however, the data model is not expressed using formal semantic tools such as OWL or RDF.

- Effective Context Modelling

Effective context modelling requires effective syntax and semantics modelling. OWL is most suited for such as purpose as it is based on XML and DL, which provide powerful syntax and knowledge representation formalisms for ontology, respectively. Therefore, OWL provides formality and expressiveness with regard to both syntax and semantic.
2.6 Content and Service Adaptation Systems

Content and service adaptation has been investigated by several research projects and its significance and importance are recognised. This section provides a brief survey on the different categories of systems in this area.

The first category is oriented toward content adaptation for delivery and consumption, mostly, considering connectivity characteristics (device and network) and user preferences. The surveyed systems of this type fall into one or more of the categories specified by Table 1. A surveyed system will not be put into a category if the corresponding aspect, for example layout adaptation or adaptation management, is superficially treated. The categories of the adaptation systems are explained in Section 2.6.1.2.

Figure 5 depicts the complexity of today's content and service adaptation domain. On one hand, the user is presented with several devices of varying capabilities connected via different access technologies. On the other hand, several multimedia content and services are available. For example, news services, entertainment services, etc. An adaptation framework is necessary to bridge the gap between the user environment and content and service providers. The adaptation framework can use several adaptation tools available at the user side, the server side or the network. Next section presents main content adaptation systems from the literature.

The second category of adaptation frameworks is oriented toward adaptive ubiquitous services; they consider a wider range of context including location, time, activity, nearby objects, etc. as opposed to the first category where they mostly consider device, network, content context and the user preferences. Actually, such systems were investigated well before systems of the first category emanated (as early as 1992) [75].

2.6.1 Content Adaptation Systems (CAS)

This section presents main work published in the category of content adaptation systems (CAS).

• Motorola Museum Guide Service [33]

Motorola Labs demonstrated how a web-enabled museum visitors guide service can adapt its content to mobile users in Motorola's History museum according to the visitor's preferences, interests, location within the museum, display resolution and accepted modalities of their devices [33]. According to the user location, the system retrieves relevant content from the museum database and converts it to an XML intermediary format according to the user device, network and preferences. The Intermediary format is converted to the appropriate mark-up such as WML, XHTM, etc.
Chapter 2. State of the Art and Background

- Web page Adaptation Proxy [30]

The adaptation proxy presented in [30] adapts web documents to mobile devices to facilitate viewing and navigation by decomposition and segmentation. The proxy implemented in Java [76] performs the following functionalities: adaptation of web documents and media files, adapted content caching and user context state management. The media adaptation modules are XML-based and handle Wireless Bitmap (WBMP), Graphics Interchange Format (GIF), Joint Photographic Expert Group (JPEG) and Portable Network Graphics (PNG) images. Other XML-based modules can be plugged into the framework and used. Long web pages are segmented into delivery units which have unique URLs. The adapted content is stored in the cache and only requested units are transferred to the user device upon request. To improve user experience, a navigation page is created with index to the adapted web pages, the navigation links are labelled with keywords extracted during the adaptation process. The user is provided with navigation support links such as “next”, “previous”, “main”, etc. User sessions are created to handle context changes as part of state management and configuration.

- IBM Server Extension [34]

One of the early systems was proposed by IBM (1999) as an extension to content servers [34]. The system aims at adapting web content to client devices with varying capabilities. It consists of two major components: the InfoPyramid, a representation scheme that provides multimodal and multi-resolution representation hierarchy of the multimedia content, and a customizer that selects the best version from the InfoPyramid to suit the device and network characteristics. Although the system dynamically assesses user devices and their access networks, it chooses from a statically authored content version and only adapts for resolution and modality.
• **Context-Aware Adaptation Proxy [31, 77]**

An adaptation proxy is proposed in [31, 77] to dynamically calculate an adaptation decision based on device capabilities, network characteristics, content description and the user perception of quality. The proxy consists of a context-aware decision engine and a transcoder. The aim is to enhance the experience of users subscribed to Internet content and provide the best content version with minimizing quality degradation that may occur in the transcoding phase. The transcoder is responsible for executing adaptation decisions. The decision engine takes four inputs: user preferences profiles, device capability profiles, networking parameters and content metadata. The output is an adaptation decision represented in a transcoding strategy. The proxy uses quantization methods of content QoS according to various quality domains such as resolution, colour depth, size, download time, etc. (pre-processing phase) and a negotiation algorithm to find the content version that best meets users perception, device capability, network connection and content characteristics (real time processing phase). Content QoS defines the collective effect of the various quality domain parameters on user experience and satisfaction with a service. Different QoS values according to different user preferences are presented as score nodes in a score tree.

• **Structure-Aware Web Transcoder [78]**

This work aims at enabling delivery of long web pages to mobile devices with small screens by providing a syntax-based web transcoding system that allows universal access to web pages without manual re-authoring. The focus is on structure-aware transcoding heuristics, which preserve the original web page’s underlying layout as much as possible. The system proposes heuristics to extract the relative importance of web components using intelligent syntax analysis. However, executable content such as Java, Javascript [79], and Flash [6] are not supported. The system aims to break web pages into smaller pages without losing or degrading information. Information can be preserved if the adapted page structure is consistent with the original web page. A tree structure that has two node types (context nodes and content nodes) and two edge types (possibly nested and sibling relationships) is used to describe web page structures. The transcoding uses functions such as "grouping" (divide a web page in groups forming web components) and "summarization" (subgroups are reduced to hyperlinks, choosing representative phrases and modifying the tree based structure of the web page). This work introduces two new transcoding heuristics (techniques): generalized outlining transform and selective "elision" transform. The first heuristic detects repeated layout patterns and based on it transcodes the page in a sequence of smaller pages. The second transcoding heuristic, selective elision transform, targets web pages with complex tables to organize page layout. The technique aims to preserve table structures as much as possible by analysing table cells properties and hence deciding which
cells to remove. The shrinking factor, ratio of adapted form size to original form size, is calculated to resize the tables.

- **The XAdaptor [32]**

  The system presented in [32] adapts Hypertext Mark-up Language (HTML) pages using a rule-based decision engine. A rule-based approach is adopted for its flexibility and extensibility to new adaptation scenarios. XAdaptor adapts HTML pages to mobile devices, it considers client context (referred to as facts) which includes client device characteristics and user preferences. Client profiles are described in XML and stored in a client database. Clients’ requests are intercepted by the system which uses the content parser entity to parse the content and extracts page facts that include structural (hierarchical structure embodied in the HTML) as well as content context (referred to as facts). Page facts and client facts are fed to the inference engine which applies prolog rules to adapt the objects embodied in the HTML page. The content converter entity generates the adapted HTML page after applying fuzzy decision taking to resize HTML forms, such as buttons and labels, using a fuzzy logic based formulae that calculates the new size that would satisfy users the best.

- **Knowledge Based Adaptation Framework [80, 81]**

  This work uses a component based approach and aims at enabling content servers to use and integrate multiple specialized adaptation tools. ADT is modelled as a state space planning problem. The system accepts MPEG-7 and MPEG-21 annotated requests and outputs a plan of adaptation services. Adaptation planning is performed using component and service decomposition (OWL Services (OWL-S) [82], Semantic Web Rule Language (SWRL) [55]) techniques.

- **Distributed Content Adaptation for Pervasive Systems (DCAF) [83]**

  This system constructs an adaptation graph in order to generate the desirable content version that fits the specified user profile which contains user preference, terminal characteristics, network conditions, etc. The process consists of two steps, firstly determining the necessary transformation processes required to meet the delivery environment restrictions, and secondly finding the adaptation services for each of the transformation processes. Based on the input and output characteristics of the services, an adaptation graph is defined. The graph is optimized according to user preferences such as cost and duration of adaptation.

- **Versatile Transcoding Proxy (VTP) [84]**

  The VTP uses the concept of agent technology to enable extending the proxy functionalities by adding transcoding tools as plug-ins. The proposed system aims to provide context-aware
transcoding by accepting transcoding preference scripts from the client side to guide the transcoding process according to the user context. The concept of dynamic cache categories is adopted and the new scheme of maximum profit replacement with dynamic cache categories (DCC-MPR) is proposed. A transcoding graph is built and maintained to express the relationships between the transcoded versions of the content. Based on this graph, the system replaces cached content to improve performance.

- **MPEG-21 DIA [85]**

The MPEG group introduced DIA as part of the MPEG-21 [15] standard. DIA aims to assist multimedia content adaptation to different storage, transition and consumption environments by providing standard XML-based metadata enabling the description of terminals, networks, users and consumption environments characteristics. DIA provides tools including context description tools and a decision taking framework, these are discussed in Sections 2.7 and 2.8. However, using the tools and the decision framework is left to implementers, such as in [86-100].

- **Video Re-Composition System [101]**

The system in [101] adapts video content by content re-composition to fit small screens, un-useful regions in video frames are detected and removed to provide space for important regions to fit small displays.

2.6.1.1 The Mobile VCE AMF

A context-aware content and service adaptation framework [23] has been proposed by MVCE to provide an environment in which content and services are personalised to the user preferences and likings. The context considered includes information beyond the communication system capabilities (i.e. device, network and content). It includes user preferences, environmental context, location, activity and user characteristics. For instance, the interaction with services may be different if the user is in a shopping centre, if they have impairments or if they are sleeping.

Figure 6 provides an overview of the framework and its envisioned components. The framework receives adaptation requests accompanied with the user context from the user environment, i.e. the Personal Distributed Environment (PDE) [102] which includes all devices that the user owns or can use. The PDE consists of the Personal Assistance Agent (PAA), the Personal Content Manager (PCM) [21] and the Device Management Entity (DME). The PAA provides an interface to the outside world for the user; it hides the complexity of the user environment and enables the user to use their devices efficiently. The PCM aims at enabling the user to manage content efficiently including content access. The DME is a core part of the PDE architecture and is responsible for maintaining a set of registers that contain information about the existence, location
Chapter 2. State of the Art and Background

and capabilities of the various devices within the PDE. Furthermore, DME is the entity responsible for the coordination of these devices and it facilitates service discovery and session set-up. The DME architecture is distributed in nature, as depicted in Figure 6, in order to reduce the signalling that is required for the information exchange between its various instances in the sub-networks. Each PDE sub-network chooses intelligently a device to host the local-DME that will perform device management within its boundaries. Higher in the hierarchy, the root-DME is responsible for the interconnection of local entities.

Figure 6: PDE, AMF and adaptation

The AMF also interacts with the content and service providers to acquire content and service descriptions. Main AMF components are the AM and the Content Adaptors (CA). The AM manages contextual information including the content and service descriptions and based on them generates adaptation decisions which are handed to Content Adaptors.

The Content Adaptor entity realises the adaptation decisions on the content/service by calculating and executing an adaptation plan. The realization is termed “adaptation process” and constitutes one or more “adaptation operations”; each adaptation operation corresponds to a step in the adaptation plan. An adaptation mechanism is an implementation of an adaptation operation. Hence, there exist numerous adaptation mechanisms for one adaptation operation [103].

The AMF, i.e. the AM and the Content Adaptor are part of a hierarchical architecture as shown in Figure 6. Such architecture allows for example, a user to utilize an adaptation mechanism
(ADME) available in a laptop computer in his Personal Area Network to adapt content in order to view it in a handheld device that belongs to the same Personal Area Network. The new concept, therefore, in this architecture is the cooperation of devices belonging to the same sub-network in a PDE towards the goal of content adaptation. In case this co-operation, which will be coordinated by local AMF (AM/CA) entities, does not provide an efficient adaptation solution, an entity located in a central point in the network will be responsible for the final decisions. This entity will be the Root AMF (Root AM/CA) and it is hosted by the root DME. If the root AMF fails to adapt the content, another AMF outside the PDE is requested. The local AMF, the root AMF and the AMF have the same architecture and perform the same functionalities; however, their capabilities in terms of ADT logic and access to ADMEs differ.

2.6.1.2 Categories of Content Adaptation Systems

The systems surveyed in the previous section are classified in Table 1 under different categories. The categories are explained in this section with examples. More on content selection, content presentation adaptation and content adaptation categories is presented in [12]. In addition to these three categories, the author introduces two new categories which are: AMFs and adaptation engines. The five categories of content adaptation systems are explained below:

- **Presentation Adaptation**

  Related work in this category mainly focuses on web page adaptation to small mobile devices, example of systems that deal with this aspect include [30, 32, 33, 78].

- **Adaptation Engines**

  Most of the adaptation systems can be classified under this category, i.e. Adaptation Engines (AE), unless they describe a mechanism for a specific content type adaptation or are classified under the AMF category. An adaptation engine provides a system where different fixed tools are used for the purpose of adaptation.

- **Content Selection**

  Content selection can be applied to both push and pull services. The content delivered to the user is chosen based on user preferences, location and usage history. An example of a system that considers content selection is provided in [33].

- **Specific Techniques (Including Transcoding)**

  State of the art video and image adaptation include systems presented in [104-117]. Transcoding can be defined as converting media from one format to another. Transcoding may be used for adaptation due to unsupported formats or to fit to a constrained environment by reducing temporal
resolution (frame rate) or spatial resolution (pixels) for example. Cross modality transcoding is converting from one format of a modality to a format of another modality such as converting text to speech or video to audio. Adaptation techniques include summarization such as: text summarization [101], video mosaicing [114], temporal and spatial segmentation [117] and key frame extraction techniques [116].

Table 1: Content adaptation systems

<table>
<thead>
<tr>
<th>Systems</th>
<th>Type</th>
<th>Presentation Adaptation</th>
<th>Adaptation Engine</th>
<th>Content Selection</th>
<th>Specific Techniques</th>
<th>AMF</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Server Extension [34]</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Context-Aware Adaptation Proxy [77]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Motorola Museum Guide Service [33]</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The XAdaptor [32]</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Knowledge-Based A. Framework [80, 81]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>DCAF [83]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Structure-Aware Web Transcoder [78]</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPEG-21 DIA [85]</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Video Re-Composition System [101]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Versatile Transcoding Proxy (VTP) [84]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>The MVCE AMF</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

- AMF

As opposed to AEs which can be classified in any of the categories described above, an AMF aims to manage different tools including adaptation mechanism tools to achieve a wider range of adaptations. Thus AMFs target features such as system interoperability and extensibility to other tools. Tools such as XML (for example in [30, 33]), web services (for example in [83]),
Semantic Web services (for example in [80]), component based architectures (for example in [80]) and agent technologies (for example in [84]) are used.

Figure 7 shows management at different stages of adaptation. The contributions of this research fall into the context and decision taking management levels of adaptation.

<table>
<thead>
<tr>
<th>Content Selection</th>
<th>Context Management</th>
<th>Decision Taking Management</th>
<th>Adaptation Tools Management</th>
<th>Delivery Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content Types</td>
<td>User Context</td>
<td>General ADTE</td>
<td>ADME</td>
<td>Delivery Channel</td>
</tr>
<tr>
<td></td>
<td>Device Context</td>
<td></td>
<td>ADME</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADME</td>
<td>Specific ADTE</td>
<td>ADME</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 7: Management tasks at different levels of adaptation**

### 2.6.2 Context-Aware Ubiquitous Services Adaptation Systems (CAUS)

This section presents main frameworks of interest to this research in this category. There is an increased interest in ontology modelling due to its high expressiveness and formality [62] (Section 2.5.3). This section only includes work that uses technologies relevant to this research, mainly for context modelling.

- **Context Broker Architecture (CoBrA) [71]**

The Context Broker Architecture (CoBrA) [71] aims at creating smart spaces by enabling services, agents and devices to share their operation space context and interoperate intelligently based on the shared information. The system uses a set of common ontologies to enable the communication and representation of context.

The CoBrA Architecture is designed around a broker-centric agent architecture with the context broker as a central entity to maintain and manage a shared context model on behalf of devices and agents. The main tasks of the broker include gathering situational information from heterogeneous sources, reasoning about the gathered data to infer contextual information, enabling sharing contextual information among distributed agents, enforcing user-defined policies to protect privacy of users, enabling software agents to work together to attain shared objectives and maintaining contextual knowledge on behalf of agents and devices.
The rationale behind a centralized design is to relieve resource constrained devices, because agents rely on the server to acquire and reason about context. Moreover, enforcing policies at the broker improves security, privacy and trust.

The broker has five functional components including the knowledge base that consists of the ontologies, the reasoning engine, the context acquisition module to acquire context from internal and external sources and the team coordination module responsible for coordinating teamwork between agents.

- **Service Oriented Context-Aware Middleware (SOCAM) [66, 118]**

The Service Oriented Context-Aware Middleware (SOCAM) Framework provides support to facilitate building and prototyping of ubiquitous services that uses context to adapt their behaviour. The context model used is OWL ontology.

The SOCAM Architecture is based on a distributed middleware and consists of the following components: the context provider, the context interpreter, the context database, the context aware services and the locating service. The context providers transform acquired context to OWL representations to be used by other components. Context sources can be either internal (e.g. physical sensors deployed in a domain) or external (e.g. weather information server or a location system). The context Interpreter processes context information using logic reasoning. Its tasks include deducing high level context from low level context, preserving consistency and resolving conflicts. Along with the context reasoner, this component has a context knowledge base where other components can query or modify the (context) knowledge. The system reasons about context using a rule-based engine. The context database stores context ontologies along with context history for a sub-domain. For each domain there is one logic context database. Context-aware services are services, agents or applications that use contextual information to adapt their functionality. The locating service is used to locate context providers and to acquire the necessary context.

- **The Context Management Framework (CMF) [73]**

The CMF described in [73] aims to adapt applications to user terminals based on uncertain and imprecise context. It provides methods for acquiring and processing contextual information from an uncertain changing user environment and handing it to mobile applications to adapt their functionality according to the conditions of the environment. The framework consists of four main entities and two secondary entities. The main entities are: the context manager which acts as a server to the rest of the system, the resource server, the context recognition service and applications. The other two entities are the change detection service and the security entity. The context manager, resource servers and applications run on the mobile device. The context
Chapter 2. State of the Art and Background

The manager is responsible for storing context and answering client queries. Clients can gain context data by querying the manager or subscribing to context change notifications. The manager can also supply applications with high level context information by contacting the context recognition services. The Context Manager acquires contextual data from resource servers which post it from context data sources to the context manager's blackboard. The contextual data is then processed and delivered to clients as required. The resource servers use fuzzy logic techniques in the process of low-level context recognition. The resource servers and the recognition services transform raw low-level context into high level context defined in the context ontology. The change detection service detects any changes in context information and the security module ensures context data comes from trusted parties. Different inference mechanisms for reasoning new context can be plugged into the framework and used.

- **Context Management Framework for Distributed Applications [72]**

The work reported in [72] aims towards enabling context aware services to provide the right media at the right time and at the right place. An important enabler for this is context management. The proposed architecture aims to provide a context processing framework that can handle heterogeneous context exchange protocols and formats, and performs tasks such as context gathering, interpreting and distribution.

The proposed framework consists of the following components, the context provider, the user manager and the context-source component which consists of the context reasoner and the context wrapper. The context source component integrates contextual information from several distributed sources and provides a single point of access for context information through the context provider. The user manager manages end users' context and their access rights to context information provided by the framework. While the context wrapper encapsulates information about context concepts such as Device Location, the context reasoner is responsible for context interpretation and refinement. The framework adopts an ontology based context modelling approach using OWL-DL. Reasoning provides intelligence for the framework and reasoning tasks include context request matching, context retrieval and context derivation and estimation.

### 2.7 Context in Content Adaptation Systems

Adaptation depends on the availability of contextual information in a suitable format. Section 2.7.1 surveys the technologies relevant to context modelling in content adaptation systems. As mentioned in Section 2.7.2, context modelling in content adaptation systems (CAS) is superficially treated. Reasons for this include the limited set of context parameters considered.
Therefore, Section 2.7.3 surveys context modelling in context-aware ubiquitous systems (CAUS) where a larger set of context parameters are considered.

2.7.1 **Multimedia Adaptation Context Technologies**

This section presents the main context description standards and technologies that are used by content and services adaptation systems. A survey of multimedia context standards is presented in [119]. Implementing standards is important, especially established standards such as MPEG-7 and MPEG-21. Main multimedia context standards are:

- **The Content Description Standard: MPEG-7**

  The main multimedia content description standard is MPEG-7; it provides tools to annotate multimedia content at different stages including creation, storage and usage. The tools can describe low level features such as colour and sound features as well as temporal, spatial or spatio-temporal content structure as discussed in Section 2.5.2.

- **Domain Specific Standards**

  The International Press Telecommunication Council (IPTC) [120] introduced a family of news exchange standards [121] which includes XML based metadata to describe general news content using the News Mark-up Language (NewsML), sport content using the Sport Mark-up Language (SportsML) and event news such as conferences, summits, etc using the Events Mark-up Language (EventsML). SportsML and EventsML are based on NewsML and specify how news content including video, images, audio and text should be structured, wrapped and exchanged among players in the news domain, for example, distributing a conference information and news from conference organizers to the press using the structure and vocabulary defined in the EventsML standard.

- **The Multimedia Framework Standard: MPEG-21**

  The MPEG-21 multimedia standard provides tools to extend context profiles to descriptions of multimedia content usage and consumption environment including devices, networks, user preferences, etc as described in Section 2.5.2 and Appendix A. MPEG-21 aims to enable a transparent use of multimedia content by different communities via diverse consumption environments characterised by different devices and access technologies.

- **CC/PP and UAProf**

  CC/PP proposes a framework to describe devices in terms of components, for example, a software component, a hardware component and a user preferences component. UAProf is an implementation of CC/PP, proposed by the Open Mobile Alliance (OMA) [122]. UAProf is
widely used in industry. It follows CC/PP structure and proposes a vocabulary to be used in UAProf profiles.

Section 2.7.3 and 2.7.2 present context modelling in CAUS and CAS, respectively. Chapter 4 presents AM context modelling and processing.

2.7.2 Content Adaptation Oriented Systems Context Modelling

Context modelling has not been treated in depth in state of the art CAS. Some systems do not mention the format of context profiles and how it is modelled internally, this is due to the fact that context parameters that are considered are limited. For example, the work in [77] mentions the viability of using CC/PP for user profiling, however, it does not discuss internal context representation and modelling.

Figure 8 shows context modelling in state of the art content adaptation systems (CAS). Profiles from the user environment and content provider are loaded into the systems and represented with an internal data structure. This data structure can be either defined by the programmer or as part of the API that parses the loaded profiles.

The Xadapter [32] uses XML to represent user profiles, the XML schema used does not conform to a specific standard and it is not clear whether the XML schema is also used for internal context modelling. The Versatile Transcoding Proxy (VTP) [84] supports CC/PP to provide context-awareness to the transcoding process. UAProf is the industry standard based on CC/PP and has been used in adaptation systems defined in [30, 33, 123]. The system in [81] used DIA (MPEG-21, Part 7) for environment description and MPEG-7 for content description.
2.7.3 **Context Modelling in Context-Aware Ubiquitous Systems**

Context modelling and processing are treated in more details in context aware ubiquitous systems (CAUS). In most of the surveyed CAUS, a section is dedicated to context.

Figure 9 shows context modelling and representation in context-aware ubiquitous systems. Context is acquired from context sources that are connected to sensors or applications, for example the user diary to acquire user activity context. Context from the different sources is aggregated and fused into one model. State of the art CAUS [66, 71, 72] uses ontologies to model the fused and aggregated context, an ontology reasoner is used to deduce higher level context and detect inconsistencies. Other types of reasoning such as rule-based reasoning may also be used to refine and deduce higher level context that cannot be deduced with ontology reasoning [50, 66].

The refined context is then shared by the ubiquitous services to provide intelligent and context aware behaviour to users [71].

Context modelling in three main context-aware frameworks is discussed below

- **Service Oriented Context-Aware Middleware (SOCAM) Context Modelling**

In SOCAM [66], context is represented using OWL ontologies. The SOCAM ontologies have two layers: the common upper ontology layer for general concepts and the domain-specific ontologies layer for sub-domains. The upper layer is static and is shared among different domains. The lower layer is a set of low level ontologies for defining concepts in each sub-domain such as smart-office and vehicle. This layer may be adaptive and can dynamically “re-bind” with the generalized ontology in the case of environment change. For example, if the user switches between domains, the appropriate domain-specific ontology will be bound to the common upper ontology. By separating domains, pervasive devices process context in their domain only, hence save resources in terms of ontology reasoning.
Chapter 2. State of the Art and Background

The generalized ontology (common upper ontology) consists of 14 classes and six properties to define the basic concepts of person, location, computational entity and activity. The details of these basic concepts are further defined in the domain specific ontologies. Domain specific ontologies may differ from one domain to another. They define the details of general concepts and their properties in their domain. For example, SOCAM has a domain-specified ontology for the smart home domain and the vehicle domain.

In this framework, context is classified as direct context and indirect context. Direct context can be obtained from external or internal sources. It can be further divided into sensed context (acquired from physical sensors) or defined context (acquired from virtual sensors such as web services, which usually represents the user preferences). Indirect context is inferred from direct context through context reasoning. For example, the user activity can be inferred from a set of direct context parameters. The ontologies give each context a classification category through the property element owl:classifiedAs. This property can have the values such "Sensed", "Defined" or "Deduced".

- **Context Broker Architecture (CoBrA) Context Modelling**

  In CoBrA [71], context is represented in OWL ontologies. The current version of the CoBrA ontology is defined in XML syntax and consists of 17 classes and 32 properties to define common relationships and attributes of people, places and activities in a smart space. The classes and properties of the CoBrA ontology are used to describe the concepts of "Person", "Place" and "Intention".

- **Context Managing Framework (CMF) Context Modelling**

  In the CMF [73], context information considered is temperature, touch, humidity, light, sound and acceleration. The information is represented using an ontology described in RDF. Each context is described using six properties: context type, context value, confidence (to represent the uncertainty level), sources, timestamp and attributes.

2.7.4 **Context in CAUS vs. CAS**

Table 2 summarises the different aspects of context in both CAUS and CAS, these aspects relate to context acquisition, heterogeneity, standards, context changes, context caching and reasoning.

With regard to CAUS, raw heterogeneous context is acquired from sensors and aggregated into a middleware. No standardised vocabulary could be found for context description for CAUS systems. On the other hand, a CAS acquires context profiles from user environments and service providers. Context profiles are likely to be heterogeneous as they implement different
vocabularies. Standards for CAS context modelling and description exist such as MPEG-7/21 and UAProf.

With regard to context changes and caching for CAUS, sensors notify the context aggregation system of context changes and context is cached in the middleware and updated upon context change notifications. On the other hand, CAS may perform context caching with regard to user preferences and device static features, such as device display characteristics. However, dynamic context such as network condition and battery life changes with every adaptation request. Both CAUS and CAS use ontology and rule based reasoning to refine context.

<table>
<thead>
<tr>
<th>Context Aspects</th>
<th>CAUS</th>
<th>CAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context Acquisition</td>
<td>High level context from middleware</td>
<td>Adaptation systems acquire from the user</td>
</tr>
<tr>
<td></td>
<td>Low level context from sensors</td>
<td>environment and the service providers</td>
</tr>
<tr>
<td>Heterogeneity</td>
<td>Context acquired from heterogeneous sensors</td>
<td>Context (high level) profiles likely to be</td>
</tr>
<tr>
<td></td>
<td></td>
<td>in different formats</td>
</tr>
<tr>
<td>Standards</td>
<td>Not available</td>
<td>Available, e.g. MPEG-7/21, UAProf</td>
</tr>
<tr>
<td>Context Changes</td>
<td>Context sensors notify of changes.</td>
<td>Changes with every adaptation request</td>
</tr>
<tr>
<td>Context Caching</td>
<td>Always</td>
<td>Sometimes</td>
</tr>
<tr>
<td>Reasoning</td>
<td>Ontology and rule based reasoning.</td>
<td>Ontology and rule based reasoning.</td>
</tr>
</tbody>
</table>

2.8 Decision Taking in Adaptation Systems

This section surveys decision taking in content adaptation systems. Adaptation decisions need to be taken at different stages. This includes deciding on content and functionality to be delivered, adaptation operations to be applied on the content (such as modality conversion, format transcoding, etc), the values of adapted content parameters (such as image resolution, video bit rate, etc), presentation and finally adaptation execution sequence in case several adaptation tools need to be applied to execute adaptation decisions.

Different ADT technologies are used in state of the art adaptation systems including rule based ADT, search and optimization based ADT,


Chapter 2. State of the Art and Background

2.8.1 Rule Based ADT

Examples of ADT using rule based reasoning are [32, 40, 124-129]. For example, a Prolog reasoner was employed in [51] and a Java Expert System Shell (JESS) was employed in [130]. Rule based reasoning is explained in Section 2.4.1. Adaptation rules are defined to decide on the adaptation operations that need to be applied on content, for example, image resolution reduction, modality conversion, etc.

2.8.2 Search and Optimization Based ADT

• The Context-aware Decision Engine [31, 77]

The decision process consists of two phases: the pre-processing phase that occurs before receiving requests and the real-time processing phase occurring after. The pre-processing stage generates a data structure (a linked list or a tree) of score nodes. Each node in the data structure represents a content version, its score and the steps to generate the content version. The score is calculated from user preferences and content quality parameters. For example the quality parameters for a PDF file could be colour, scaling, segment, download time and modality. Each of these parameters is modelled in a quality axis. The set of axes forms an $n$ dimensional space where $n$ is the number of quality parameters modelled as axes. The user preferences are represented in the space by specifying a quality value in each of the axes. For example, the user might set a max download time for a document by representing the time value in the download time axis. The set of values indicated in the different quality axes are aggregated to form the score for the content version. During the real-time processing phase, the decision engine uses the data structure generated at the pre-processing phase to determine adaptation decisions. When a user requests content, the system determines the device capabilities, the network parameters and the requested content metadata and searches through the tree or linked list for the best score node that matches the user preferences for the current context. A number of search algorithms were considered, a combination of scored linked list (SLL) and non-ordered relation score tree (NORST) is used for higher accuracy [31].

• DIA-ADTE [86]

Systems that use DIA-ADTE for adaptation decision taking include [36, 81, 86, 131-133]. The authors in [86] proposed a constrained optimizing problem based ADT approach. Adaptation parameters such as user characteristics, content properties and device and network capabilities represent algebraic variables for the optimization problem. The ADTE takes as input Usage Environment Descriptions (UED), Universal Constraints Descriptions (UCD) and Adaptation Quality of Service descriptions (AdaptationQoS). The output is the decisions which are the
settings of the parameters such as bit-rate, resolution and frame rate. UED, UCD and AdaptationQoS are DIA standards. UED provides description tools for multimedia usage environments including devices, networks and users. UCD provides description tools for restrictions, for example, an adaptation is needed to achieve a certain quality level, a resolution less than 50% of the device resolution, a certain maxim code size and match the device colour capabilities. AdaptationQoS assists the ADT process by describing the relationships between adaptation operations, content parameters and the resulting quality metrics.

2.8.3 Semantic Web Based ADT

The system in [80] uses Semantic Web service composition to decide on the sequence of adaptation services to invoke in order to transform the content from the original form to the adapted form. The system in [134] uses ontology to match user profiles containing device capabilities and content requirements in order to derive the necessary adaptations.

2.8.4 Fuzzy Logic Based ADT

Fuzzy Logic is described in Section 2.4.3. The system presented in [32] uses Fuzzy Logic to model user satisfaction levels with resizing HTML forms such as tables, labels and buttons. A fuzzy logic based formulae is used to calculate adapted web page form sizes that would satisfy users' presentation requirements. The system in [135] used ontology to describe user profiles and Fuzzy Logic to deal with uncertainty in user preferences.

2.8.5 Mathematical Modelling Based ADT

Mathematical functions are used in combination with other decision technologies such as search-based decision taking, rules-based decision taking, fuzzy logic, etc. For example the context-aware decision engine presented in [31, 77] uses a mathematical formula to calculate a QoS value of a content based on a number of parameters such as device capabilities and network condition. The system in [34] uses mathematical formulae to calculate a content value and compares it with content values of different stored transcoded versions of the same content. The system in [88] uses formulæ to calculate different parameters relating to vision impairments and uses them to decide on appropriate settings for the adapted visual content such as colour temperature, etc. The system in [136] used mathematical modelling in a text summarization algorithm. The algorithm detects keywords in the text and based on them summarizes text content.
2.9 Summary

This chapter has presented the background and the state of the art related to the content and service adaptation domain, and placed the author’s contributions in context. The author’s work relates to the area of context-aware service and content adaptation management. This chapter has started with defining core concepts such as adaptation, content vs. service and context. Main work in the area of content and service adaptation has been listed and adaptation systems have been classified in categories. Finally, technologies and approaches used by these systems with regard to context modelling and ADT have been discussed.

There are several approaches to context modelling and ontology is the most effective approach with regard to syntax and semantics expressiveness and formality. Ontology can be formalised using DL, a first order logic based mathematical formalism, to define classes and relationships between them. In addition to ontologies, other Semantic Web technologies such as XML, OWL and RDF are key tools increasingly used by the context and multimedia community. These technologies have also been discussed in this chapter.

Two types of adaptation systems have been identified; content adaptation systems (CAS) and context-aware ubiquitous systems (CAUS). It is noted that CAS did not treat context-awareness aspects such as context modelling and description in depth, a main reason for this is the limited set of context parameters considered. Therefore, CAUS are surveyed as they treat context-awareness aspects in more depth because they consider a larger set of context parameters.

In CAS, description standards such as CC/PP or MPEG-7/21 are sometimes mentioned but without details on how context is modelled internally. On the other hand, CAUS elaborate on internal context representation, for example, in the latest state of the art CAUS, ontologies are used to model context and ontology reasoning is used for high level context deduction. Using ontologies and ontology reasoning has several benefits as discussed in Chapter 4.

Several decision technologies have been used for content ADT. Rule based decision taking is used for its expressiveness and extensibility. Mathematical modelling is used to describe the relationships between adaptation parameters, for example user preferences with regard to size and adapted content form sizes. Mathematical based ADT and Fuzzy Logic are used in conjunction with other approaches such as rule based ADT. Some adaptation systems use search and optimization based ADT such as the optimization based ADTE proposed by MPEG-21 DIA. Different ADT technologies are combined to achieve different ADT functionalities. For example, rule based ADT provides expressive and extensible ADT with regard to specifying the adaptation operations to be applied on content for adaptation. Fuzzy Logic based ADT complement rule
based ADT by modelling uncertainty. MPEG-21 DIA complements rule based decision taking by accurately selecting adapted content parameters.

The next three Chapters (3, 4 and 5) present the author’s approach in addressing content and service adaptation management problems. Chapters 6 and 7 provide an overview of the implementation and evaluation of the author’s contributions, respectively.
Chapter 3. Context Aware AM

3 Context-Aware AM

3.1 Introduction

This chapter and the next two chapters present the contributions of the author in addressing the key content and service adaptation requirements identified in Section 3.2. The contributions of this thesis are embodied in the AM that is a central entity in an AMF. Several adaptation frameworks and architectures are presented in Chapter 2. The aim of this research is not to propose yet another framework, but rather, to introduce mechanisms including architectural and design mechanisms to support content AMFs. Key targeted features of the proposed architecture are interoperability and extensibility to tackle the complexity of the adaptation problem that arises from the heterogeneity and volatile nature of the content and service adaptation domain.

This chapter starts with identifying requirements for content and service adaptation in Section 3.2. The requirements are divided into functional and non functional requirements. Functional requirements are further divided into input, functional and output requirements. The requirements focus on the context assimilation and decision taking levels of content adaptation management with emphasis on extensibility and interoperability.

Section 3.3 presents the specification in which the contributions will be formalised. The specification is based on the Vienna Development Method (VDM) notation because it provides formality using first order predicate calculus. The adopted formal specification is explained and briefly illustrated in this chapter; and is used more extensively in the next two chapters.

Section 3.4 introduces the AM, the entity that implements the contributions presented in this thesis. Technical details about the contributions are presented in Chapter 4 and Chapter 5.

Section 3.5 presents the system design. The low level design is depicted by Figure 16 and key components of the design and interfaces, explained in Table 8, are depicted by a UML component and UML class diagrams in Figure 17 and Figure 15, respectively. Finally, Section 3.6 concludes the chapter.
3.2 Content/Service Adaptation Requirements

This section discusses main adaptation management systems functional and non functional requirements, with focus on requirements at the context management and decision taking levels.

A defining aspect of adaptation management systems is the management of several content adaptation related tools such as content transcoders, context extractors, adaptation decision engines, etc. Quality of service plays a part in the selection of adaptation tools which provide the same service, for example offering the same type of adaptation. The cost of adaptation can be traded for different quality of service parameters [137], this aspect is reflected in the AM requirements.

3.2.1 Functional Requirements

Functional requirements are described in terms of input, functionality and output requirements in the following three subsections.

3.2.1.1 Input Requirements

This section lists the requirements with regard to the input to an AMF.

- Adaptation Requests

The adaptation management system receives adaption requests which should contain the location of the requested service/content and context profiles which specify the delivery channels of the content. Details on the requirements of adaptation request are specified in Table 3.
Table 3: Adaptation request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content/Service URI</td>
<td>To locate and access the requested content/service to be adapted. This can be commercial content, freely available content on the Internet or content from the user environment; for example, adaptation may be needed to satisfy storage requirements or access with different user devices.</td>
</tr>
<tr>
<td>Context Profiles</td>
<td>To specify the locations of the context profiles for the content and the usage environment including the user, device, network and consumption environment.</td>
</tr>
<tr>
<td>Type of Request</td>
<td>The adaptation management system may consider different types of requests. An adaptation request could be to request adaptation or to inquire about its costs such as the price and/or time. This information is particularly useful if adaptation is performed at a cost and an intelligent agent is acting on behalf of the user to get the “best adaptation quote” according to criteria set by the user agent.</td>
</tr>
<tr>
<td>Constraints</td>
<td>Constraints are high priority user preferences, for example, if the user specifies the maximum time they can wait for adaptation and the cost they are willing to pay as constraints, then all other user preferences are considered as long as the time and cost constraints are satisfied.</td>
</tr>
</tbody>
</table>

- **Adaptation Mechanisms Context**

Adaptation mechanisms are implementations of adaptation operations [103]. Their context (description) is provided by the adaptation mechanism service provider and contains parameters such as those specified by Table 4. Parameters include price, performance, handled content types and invocation instructions. The adaptation mechanism could be implemented as a web service, or as an application running locally or on a remote server.

- **Requested Content/Service**

The adaptation management system may need to load the requested service or content for example for context extraction if not available in separate context profiles. For example layout context extraction tools may need to parse the service to extract layout and structure metadata. Moreover, content/service location needs to be passed to the adaptation mechanism service providers in order to load the content/service and perform the adaptation.
Table 4: Adaptation mechanisms context

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price</td>
<td>Describes the cost of using the adaptation mechanism, the cost can be specified as a function of units processed. For example, x price units per y frames or x price units per y bits. This price may vary according to some service agreement between the adaptation mechanism service provider and the AMF.</td>
</tr>
<tr>
<td>Performance</td>
<td>Describes the performance of the adaptation mechanism, it can be specified as a function of units processed. For example x time units per y frames or x time units per y bits. This performance might be set according to the price specified in the service agreement between the adaptation mechanism service provider and the AMF.</td>
</tr>
<tr>
<td>Input Parameters</td>
<td>A list of supported parameters of the supported input content. This varies depending on input content types. However, input modality, input format and input language are specified for all adaptation mechanisms regardless of their supported input content.</td>
</tr>
<tr>
<td>Output Parameters</td>
<td>A list of supported parameters of the output content. This varies depending on output content types. However, output modality, output format and output language are specified for all adaptation mechanisms regardless of the output content supported.</td>
</tr>
<tr>
<td>Invocation</td>
<td>Describes how the adaptation mechanism can be invoked. For example via a web service interface, a remote method, etc.</td>
</tr>
<tr>
<td>Description</td>
<td>Provides a description of the capabilities and properties of the adaptation mechanism.</td>
</tr>
<tr>
<td>Provider</td>
<td>Details of the adaptation mechanism service provider</td>
</tr>
</tbody>
</table>

3.2.1.2 Functionality

Functionality requirements are specified in terms of context assimilation, ADT and adaptation execution.

- Context Assimilation
  - Check the availability of contextual information for an adaptation request, otherwise context may need to be requested from the respective party, for example the user environment or the content or service provider, by sending a request context profile message.
  - Extract embedded context from content if not provided as external profiles.
  - Parse content to extract service metadata if it is not provided.
  - Maintain profiles of available adaptation operations and adaptation mechanisms.
Chapter 3. Context Aware AM

- Maintain adaptation context in a suitable data model and format.
- Format context to the required context data model and format.
- Context profiles may need to be cached, for example, to maintain registered adaptation clients. Changes to cached context need to be managed.

**ADT**

- **Content selection:** select the content or service to deliver to the user.
- **Service layout/structure adaptation:** decide which content items to remove, emphasise or de-emphasize by resizing or changing its layout. The structure needs to be adapted for services with navigation, for example XHTML services.
- **Selection of adaptation parameters:** decide on the parameters of adapted content items.

**Adaptation Execution**

- **Adaptation planning:** Break the adaptation decisions into adaptation steps which constitute an adaptation plan. Each step has to be possible, i.e. there exists a mechanism to achieve it.
- Adaptation mechanisms sequence invocation (plan execution).
- Selection of suitable mechanisms for each adaptation operation (a step in the adaptation plan) depending on time and cost constraints.
- Intermediate adapted versions caching: provide a mechanism to maintain intermediate content versions, so that successful intermediate content versions are not lost in case an adaptation mechanism service provider fails in the process of performing an intermediate step in the adaptation plan.
- Content caching: this might be practical for registered users or if the same content is requested for adaptation by a similar adaptation context. However, this might be impractical if adapted content versions correspond to a larger context parameter set (i.e. more personalization and context-awareness if offered) and hence the adapted content set is larger and adapted content versions are less likely to be reused. This is demonstrated by Figure 10.

Figure 10 depicts that in the present/future adaptation domain, the set of possible delivery contexts is large (denoted by c1, c2, c3, etc) due to the availability of different user devices with varying capabilities. Therefore, the number of adapted content versions to fit
most of the delivery context is large as well. Previously, the set of delivery contexts was limited and a small number of adapted content versions would fit most delivery contexts.

![Figure 10: Adaptation spectrum](image)

### 3.2.1.3 Output Requirements

This section outlines requirements with regard to the adaptation system output and the output of the main adaptation sub-systems.

- **Assimilated Context**

  Context parameters are assimilated in a context model. The different approaches to implement a context model are outlined in Chapter 2. The assimilated context is the interface between the adaptation context assimilation and the adaptation decision subsystems.

- **Adaptation Decisions**

  Adaptation mechanisms management, which includes execution planning of adaptation decisions, might be performed at a different location to ADT and context acquisition. In such a situation, adaptation decisions are sent to the adaptation mechanisms management entity. Table 5 outlines the parameters that constitute adaptation decisions.

- **Adaptation Response**

  An adaptation response is the notification the adaptation system sends to the adaptation requester to inform whether the adaptation is possible with the specified constrains, if any are specified. The parameters specified in an adaptation response are explained in Table 6.

- **Adaptation Results**

  Adaptation results are sent to the adaptation requester once the content is adapted. Table 7 describes the parameters specified by an adaptation result.
Table 5: Output of adaptation decisions to the adaptation mechanism management entity

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>An ID that identifies the decisions and enables mapping to the adaptation request</td>
</tr>
<tr>
<td>Content URI</td>
<td>In the case where the AMF performs selective content delivery, the decisions contain the content or service URIs to be delivered to the user.</td>
</tr>
<tr>
<td>Layout</td>
<td>This element contains the layout and structure of the adapted service. Service authoring tools that understand the layout and structure specification to generate the appropriate service presentation are needed.</td>
</tr>
<tr>
<td>Adaptation Operations</td>
<td>The list of adaptation operations to be applied on the content in order to generate the adapted version.</td>
</tr>
<tr>
<td>Content Parameters</td>
<td>Contains the parameter settings of the adapted content versions, these parameters are applied for the decided adaptation operations.</td>
</tr>
</tbody>
</table>

Table 6: Adaptation response parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Identifies the response to both the adaptation requester and adaptation provider</td>
</tr>
<tr>
<td>Result</td>
<td>Indicates whether the request can be adapted.</td>
</tr>
<tr>
<td>Constraints Met</td>
<td>If request constraints such as time and cost were specified, this parameter indicates whether they can be met.</td>
</tr>
<tr>
<td>Constraints</td>
<td>If request constraints cannot be met, the response object specifies what the cost of adaptation is (it can be specified in terms of the constraints specified by the request, for example time and cost)</td>
</tr>
</tbody>
</table>

Table 7: Adaptation result parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Identifies the adaptation result to both the adaptation requester and the adaptation provider.</td>
</tr>
<tr>
<td>Adapted Content/Service URI</td>
<td>The URI of the adapted content or service</td>
</tr>
<tr>
<td>Adapted Content Context</td>
<td>This represents the metadata or description of the adapted content or service.</td>
</tr>
</tbody>
</table>
Chapter 3. Context Aware AM

3.2.2 Non Functional Requirements

Non functional requirements relate to system-wide features rather than functionality and are very important to AMFs. This research focuses on non-functional requirements such as extensibility and interoperability. Important non-functional requirements are specified in the following subsections.

- **Modularity**
  
The modularity of the design is essential to AMF extensibility to supporting and using new adaptation related tools. Modular design entails clear interfaces are defined between sub-systems. For example, a clear interface between the context assimilation and ADT subsystems. Clear interfaces between adaptation subsystems (i.e. internal interfaces) designed with both extensibility and interoperability in mind are essential requirements.

- **Extensibility**
  
The adaptation domain is volatile and new technologies and standards emerge rapidly. Extensibility enables adaptation systems to cope with the volatile nature of the adaptation domain, and is required, for example, to support new context description standards, new content formats, new adaptation scenarios, etc.

- **Inter-operability**
  
The adaptation domain is heterogeneous in nature. The support of widely used standards is important to inter-operability. An AMF needs to be designed to use inter-operable and extensible technologies.

- **Context-Awareness**
  
Context is a core component of content adaptation systems. However, context aspects in adaptation systems has not been treated in full depth, for example, context formatting to the internal context model, and internal context model extensions need to be investigated.

- **Adaptation Management**
  
Extensive research has been conducted in the area of content and service adaptation and several adaptation techniques as well as adaptation frameworks have been developed. It can be noticed that latest research in adaptation frameworks is moving into the direction of adaptation management.

An AMF manages different tools that implement different adaptation techniques in order to achieve an effective and complete adaptation. This requirement entails the framework is able to manage existing and possibly future adaptation tools such as web service based adaptation...
mechanisms, application based adaptation mechanisms, ADT engines, context extractors, context formatters, content parsing tools, etc.

The framework needs to be designed with adaptation management in mind. Adaptation management requires that the design and architecture of the AMF is constructed with modularity, extensibility and inter-operability in mind.

3.2.3 **UML Modelling of Requirements**

The specified requirements are depicted by the UML sequence diagram in Figure 11.

There are four major subsystems: the context assimilation subsystem, the ADT subsystem, the ADME management subsystem and the adaptation management subsystem which is responsible for managing the other subsystems and guiding the adaptation process.

The diagram in Figure 11 depicts two stages, a management operations stage and a request handling stage. The adaptation management operations stage constitutes operations such as adaptation mechanisms service discovery, adaptation mechanisms applications loading, and user context registration. On the other hand, the adaptation requests handling stage deals with requests from adaptation clients. An adaptation client can be a user environment or content and service provider using the adaptation system as an adaptation proxy for its clients. The adaptation request may be for a content or service to be adapted for instant access or for a content to be adapted to satisfy storage or other terminal requirements, for later access.

Adaptation requests are received via the appropriate interface within the adaptation management subsystem which triggers context assimilation. The context assimilation process checks for the availability of context, requests usage environment context if not available, extracts embedded content context, parses services to extract layout and structure context if not provided, formats the context into the appropriate model and refines the context model. The adaptation management subsystem triggers ADT once it is notified that the context assimilation process is completed. ADT constitutes layout and structure adaptation, calculation of adaptation operations that need to be applied and calculation of adapted content and service parameters.

The decisions are passed to the ADME management subsystem which calculates an execution plan and the constraints parameters if any were specified by the request (for example a maximum cost the user is willing to pay for adaptation, or the maximum time the user is willing to wait). If no adaptation constraints such as cost and time were specified, adaptation decisions are executed and the adaptation results are sent to the user containing the necessary information such as the adapted content/service URI and the adapted content context.
Figure 11: The AM functional requirements
3.2.4 AMF Deployment

Adaptation management entails the AMF may be distributed as adaptation tools are located in different parts of the network. In addition to adaptation mechanisms web services, other adaptation services such as adaptation context assimilation services or ADT services can be managed (i.e. used) by the AMF from different locations in the network. Figure 12 shows three deployment scenarios of an adaptation management system.

![Diagram of AMF Deployment](image)

Figure 12: Deployment of content Adaptation Frameworks

Both the user environment and the content and service providers may send adaptation requests to the content adaptation system in a client server fashion (Figure 12, A). Interaction between the Content/Service (CS) providers, user environments and the AMF is possible in all directions. If the content adaptation system is deployed in the CS provider’s server (Figure 12, B), the CS provider determines whether the requested service needs to be adapted to the user context and controls the adaptation process. The CS providers may also offer adaptation services to user environments or other CS providers. For both scenarios, the content adaptation system concurrently handles multiple adaptation requests from multiple adaptation clients. Figure 12 (C)
Chapter 3. Context Aware AM

depicts the scenario where the AMF is deployed as part of the user environment and used by the user to adapt content or a service to for example display content on one of their devices or to store it in a certain format. In this scenario, the AMF can be queried if the user is within their personal environment or outside their personal environment if the AMF is connected as a server to the wide area network. In both cases, the rate of adaptation requests is much less than in the two scenarios described above. The number of external adaptation mechanisms and other adaptation tools would also be limited compared to the first two deployment scenarios as the AMF would run on a user machine rather than a powerful server as in Figure 12 (A, B).

3.3 Formal System Specification

Key parts of the system proposed in this thesis are specified formally using first order predicate calculus notation. The specification of algorithms and types is largely based on the Vienna Development Method (VDM) [24] notation. Most of the formal specification will be in the next two chapters; this section introduces the specification notation and method. VDM and the Z [138] language are the two main formal methods that aim to verify system correctness, essential for safety critical systems. In this thesis, VDM is used to specify the system formally and hence accurately, VDM is suitable as it uses the mathematical notation of first order predicate calculus.

3.3.1 Formal Specification Terminology

Below is the notation that is used to formally specify the contributions; this notation is used in next two chapters.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>::</td>
<td>is defined by (used for defining data types)</td>
</tr>
<tr>
<td>:</td>
<td>Denotes the type of a property in a data structure definition</td>
</tr>
<tr>
<td>Pre</td>
<td>Denotes the pre-condition of a function</td>
</tr>
<tr>
<td>Post</td>
<td>Denotes the post-condition of a function</td>
</tr>
<tr>
<td>Δ</td>
<td>is defined by (used for defining algorithms)</td>
</tr>
<tr>
<td>Token</td>
<td>An undefined type</td>
</tr>
<tr>
<td>SomeType-Set</td>
<td>A set that contains elements of the type: SomeType</td>
</tr>
</tbody>
</table>
3.3.2 Types

Types are defined in the following way:

\[
\text{AR} \quad \text{:: RequestedContent : token } \\
\quad \text{AdaptedContent : token } \\
\quad \text{decL : Decision-Set } \\
\quad \text{CtxtProfileL : CtxtProfile-Set }
\]

The above definition relates to the adaptation request object. \( \text{decL} \) is a set of Decision objects, the Decision object is defined in Chapter 5. \( \text{CtxtProfileL} \) is a set of \( \text{CtxtProfile} \) (context profile) objects; this object is defined in Chapter 4.

3.3.3 Functions

Functions are specified in terms of post and pre conditions. The post and pre conditions are specified formally in first order predicate calculus.

3.3.4 Algorithm

To illustrate how algorithms are formally specified, the adaptation cycle performed by the AM is defined in Figure 13. Italic font text denotes functions that are implemented for the contributions presented in this thesis and normal font text denotes functions that are outside the scope of this thesis and part of the Content Adaptor functionality. OP denotes adaptation operations that need to be applied on the content.
Chapter 3. Context Aware AM

3.4 Proposed Solution: The AM

The author proposes an adaptation management entity, the AM, which is a central part in an AMF. The aim of the AM (and the framework) is to provide an extensible, interoperable and complete adaptation. This chapter proposes the design and functional specification of the AM. Chapters 4, 5 and 6 discuss AM aspects in depth.

Key aims and objectives of the research on the AM are:

- Enhance system extensibility and interoperability using modular design with clear interfaces between AM subsystems.
- Define a clear separation between the different stages of ADT.
Chapter 3. Context Aware AM

- Define a clear interface between decision taking and context assimilation.
- Manage complexity of the adaptation problem and heterogeneity of the adaptation domain using extensible and interoperable technologies.
- Efficient management of adaptation context including adaptation operations context.
- Focus on inter-operability and extensibility of the AM and hence the AMF.

Research on the AM involves investigating requirements, mainly non functional, which are necessary to achieve an AMF able to manage the complexity of the adaptation domain, as opposed to most of the research in this area which address functional requirements of adaptation.

3.5 System Design

This section presents the design of the AM. The design is presented in two levels: high level design and low level design.

3.5.1 High Level Design

The AM performs two main tasks: context assimilation and decision taking. Modularity, extensibility and ease of maintenance must be taken into account in the design and definition of the AM. This is discussed in Chapters 4 and 5.

![Content and service AM high level design](image)

Four modules are identified to realise the separation of concerns in the tasks performed by the AM entity (Figure 14). The modules are the ACM, the Reasoning Manager (RM), the Adaptation Decision Engine and the Adaptation Management Gateway. Each of these components is described below. Chapters 4 and 5 examine in more detail the ACM and the Adaptation Decision
Engine respectively. The contributions of this research and the implementation are aimed at the components shown in Figure 14 and interfaces depicted by Figure 15. The rectangular shapes in Figure 15 denote components, the full circle symbol denotes provided interfaces and arrows denote required interfaces.

![UML component diagram of adaptation domain components and interfaces](image)

Figure 15: UML component diagram of adaptation domain components and interfaces

### 3.5.2 Low Level Design

Figure 16 depicts the low level design of the AM. Information on the implemented components of this design is available in Chapter 6. The evaluation of the design and implementation in terms of achieving research objectives is outlined in Chapter 7.
3.5.2.1 ACM

This entity is responsible for gathering the input that may be represented in different context profiles. The context profiles may originate from diverse sources in different formats. This entity transforms the input profiles into a format suitable for processing by the rest of the adaptation system. As depicted by Figure 15, this component provides the context assimilator interface to the decision engine (described by Figure 17 (c)) and requires interfaces with the user environment (to acquire delivery context), the content provider (to acquire content context), the ADME Manager (to acquire adaptation operations context) and the RM (to access ontology and rule based reasoning functionalities).

![Figure 16: AM low level design](image)

![Figure 17: Main AM interfaces](image)
The ACM contains the following components.

- **Context Extractor**
  Manages several context extraction tools (represented by E1, E2,...En in Figure 16) to extract context from multimedia content, such as embedded context, web pages layout and structure.

- **Context Formatter**
  Manages several context formatter tools (represented by T1, T2...Tn in Figure 16) to transform the context from its original form into the internal context model used by the adaptation system.

- **Adaptation (A) Operations Handler**
  Manages adaptation operations context and adaptation mechanisms (which implement adaptation operations) context.

### 3.5.2.2 RM

The RM provides reasoning functionalities to assist the ACM and the Adaptation Decision Engine. The two main functionalities are ontology reasoning and rule based reasoning. Ontology reasoning is very important to the ACM and rule based reasoning is very important to the Adaptation Decision Engine.

The interfaces provided by the RM are depicted by Figure 17 (a, b). This entity contains the following components:

- **DL-Knowledge Base (KB)**
  This component contains the internal context representation model. DL is chosen as the knowledge representation scheme for the KB in order to satisfy the requirement entailing support of Semantic Web technologies and the requirement of extensibility and interoperability for adaptation context management. A discussion on reasons for choosing DL, the DL language used and the structure of the KB is presented in Chapter 4.

- **DL Reasoner**
  This component is a DL reasoner, example of reasoners are Racer, Pellet and Fact++. Details on the chosen reasoner are available in Chapter 6.

- **DL Reasoning Handler**
  This entity provides functionality to support DL reasoning and related operations for maintaining the DL knowledge base.
• Rule-Based Reasoning Handler

This entity provides rule based reasoning functions which are essential for ADT.

3.5.2.3 Adaptation Decision Engine

The Adaptation Decision Engine implements the adaptation decision engine interface (by Figure 17 (d)) and the adaptation decision logic. The decision engine analyses adaptation context and calculates adaptation decisions. The calculated adaptation decisions may involve decomposition and segmentation of content, removing content, resizing to meet certain constraints, language translations, etc. The decisions are handed to the ADME manager, which generates a plan to execute them. The generated plan consists of implementations of the steps represented in adaptation decisions. This entity constitutes the following components.

• Rule-Based ADTE

Rule based reasoning is widely used in ADT; it satisfies the requirements of extensibility of ADT. The proposed solution uses rule based reasoning for ADT. Contributions with regard to ADT are presented in Chapter 5. The contributions related to adaptation context management aim to improve the extensibility of adaptation decision rules as discussed in Chapter 4.

• DIA-Based ADTE

The author proposes to use the MPEG-21 DIA-ADTE for accurate adaptation parameters selection.

• Specialised ADTEs

Specialised ADT engines would be used to decide on specific aspects, for example, the forms size of an adapted HTML page as described in [32]. Specialized ADTEs are outside the scope of this research.

3.5.2.4 Adaptation Gateway

The adaptation gateway component is responsible for receiving adaptation requests and sending adaptation responses. The interface to the adaptation clients implemented by the gateway is depicted by Figure 17 (f). This entity contains the following components:

• Message Translator

This entity is responsible for reading adaptation requests and invoking the necessary functions from the AM. This entity is also responsible for writing adaptation process results in the appropriate format to send to the adaptation client.
• CA Interface

This entity is responsible for communicating with the ADME manager to send adaptation decisions for execution.

• Web Services Interface

This entity is responsible for web services interactions.

• Remote Method Invocation (RMI) Interface

This interface is responsible for interacting with entities that use RMI.

Table 8 describes the interfaces specified by Figure 15 and Figure 17.

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OntologyReasoner</td>
<td>This interface is provided by the RM entity. It provides methods to aid DL reasoning.</td>
</tr>
<tr>
<td>RuleBasedReasoner</td>
<td>This interface is provided by the RM, it provides methods to aid rule-based reasoning.</td>
</tr>
<tr>
<td>ContextAssimilator</td>
<td>This interface is provided by the ACM, it provides a function to assimilate adaptation context and prepare it for decision taking.</td>
</tr>
<tr>
<td>ADecisionEngine</td>
<td>This interface is provided by the Adaptation Decision Engine entity, to enable the ADME manager to access adaptation decisions</td>
</tr>
<tr>
<td>ADMEUpdater</td>
<td>This interface is used to update the ACM with changes in the adaptation operations context such as the availability/unavailability of a new adaptation mechanism</td>
</tr>
<tr>
<td>AMGateway</td>
<td>This interface is provided by the adaptation gateway and provides methods to communicate with entities such as adaptation clients.</td>
</tr>
</tbody>
</table>

3.6 Summary

This chapter has presented the AM entity, an approach to solve the adaptation requirements specified in Section 3.2. The proposed mechanisms used by the AM have been explained in Chapters 4, 5 and 6. This chapter has presented a list of functional and non functional requirements in Section 3.2. Several research projects addressed the functional requirements of adaptation systems; this research however, addresses key non functional requirements that relate
to two key sub systems of AMFs i.e. context assimilation and ADT. The key non functional requirements are extensibility and interoperability which are vital to address the complexity that emanates from the heterogeneity and dynamicity of the content and service adaptation domain.

This chapter has defined content and service adaptation requirements and the architecture proposed to realise them. The architecture has been derived with modularity and separation of concerns in mind, and reflects the focus on adaptation management.

The next two chapters discuss in depth, mechanisms proposed by the AM with regard to the two main sub systems of AMFs i.e. context assimilation and ADT.
4 Adaptation Context Management

4.1 Introduction

This chapter presents an adaptation context management framework implemented by the ACM entity, a main component of the AM presented in Chapter 3.

Context is significant for adaptation frameworks; it constitutes the input to the adaptation system and is vital to all stages of adaptation including ADT. In essence, ADT is the processing of contextual information into adaptation decisions. The amount of information in context profiles defines the extent to which the multimedia service can be tailored to users. The efficiency of context processing and modelling greatly affects the efficiency of the adaptation system, the extensibility to new adaptation scenarios and hence the quality of adaptation.

Context, descriptions and metadata can be used interchangeably, and include any information that describes entities involved in the delivery of services to users including users and services. Thus, context encompasses the description of entities such as users, devices, access networks, usage environments, adaptation operations, content and services. Detailed information on context parameters for entities mentioned above are outlined in [85, 139].

Adaptation systems requirements such as extensibility and interoperability depend on how context is modelled and processed. To address these requirements, an adaptation context management framework is proposed, its aims, structure and advantages are discussed.

Chapter 2 discussed the different available adaptation context standards and how context is managed in existing adaptation systems. MPEG-7 and MPEG-21 are the most established standard for multimedia description and adaptation context. With the advance of the Semantic Web, there have been efforts to convert MPEG-7 and MPEG-21 to a Semantic Web compatible form i.e. ontology. This chapter presents the limitations of such conversions (automatic) and demonstrates how and why manual conversion is necessary.

The outline of this chapter is as follows. Section 4.2 discusses the requirements of context management according to the trends in context and multimedia adaptation technologies. Current adaptation systems are limited in meeting these requirements. Section 4.3 presents the solution proposed by the author, i.e. the ACM that implements the adaptation context management framework presented in this chapter, and how it satisfies requirements defined in Section 4.2.
Section 4.4 describes and formally defines the different functions of the ACM to assimilate and refine context. Section 4.5 presents how Semantic Web technologies can be used for context management. Approaches to construct ontologies from existing XML multimedia context standards are examined. Section 4.6 concludes the chapter.

4.2 Adaptation Context non Functional Requirements

Chapter 2 presented a detailed description of the main state of the art adaptation frameworks and their use of context technologies such as XML and OWL for context handling. This section highlights important issues relating to adaptation context and discusses key context modelling and processing technologies such as XML and ontology. Later sections of this chapter present the author's approach in tackling the highlighted issues.

Context, defined in Section 2.5, is a key enabler for adaptation; therefore, key multimedia service adaptation requirements are tied with context issues such as interoperability, completeness and extensibility. This section presents main context requirements and possible key solutions. As discussed in Chapter 2, existing multimedia content and services adaptation systems consider adaptation with regard to a limited set of context parameters. Thus, focus was not put on context modelling and processing. Therefore, the processing and modelling of context was either not treated at all or only treated superficially. More information on the technologies mentioned in the following subsections is available in Chapter 2.

Main context non functional requirements are:

4.2.1 Interoperability

The heterogeneity of context standards implemented by systems in the multimedia content and service adaptation domain poses interoperability problems. XML based context standards surveyed in Chapter 2 such as MPEG-7 and MPEG-21 provide syntax interoperability because XML ensures formal and platform independent syntax definition. However, XML is limited in specifying semantics. The importance of using Semantic Web technologies such as DL languages, to provide a formal definition of the underlying semantics and hence provide semantic interoperability (formal and shared understanding) is recognized by the multimedia community [41].

Moreover, context standards define different vocabularies, thus, context formatting tools are needed to understand the different context standards. Easy incorporation of new context formatting tools in the adaptation system or the easy extension of existing ones is a key requirement. The surveyed adaptation systems did not target this issue.
The ACM described in this chapter aims to ensure interoperability; this is discussed in this chapter and evaluated in Chapter 7.

4.2.2 Completeness

For an enhanced user experience, adaptation should be comprehensive and thus should take into account all available contextual information including user preferences and environment characteristics. Most adaptation systems focus only on communication environment context, i.e. device, network and content context. Adaptation context should cover information about usage environments and surrounding objects such as other available devices that could be involved in a multimodal service delivery, for example, a larger display. Content descriptions for different domains are also important. For completeness, different context standards describing different domains should be integrated. For example, NewsML and SportML can be used to complement MPEG-7 for multimedia content description. The maintainability and extensibility of the context model to more context parameters are essential requirements.

Context aware ubiquitous Systems (CAUS) consider a larger set of context parameters and have treated context processing in more detail. Although there are similarities and lessons that can be learned from CAUS and their context handling, context-awareness requirements for CAS differ from those of CAUS, as discussed in Chapter 2. Context management in the respect of this chapter does not constitute lower level tasks such as context acquisition and sensing. It rather constitutes the following:

1. Receiving context profiles from different environments.
2. Reading information from profiles of different context standards/formats.
3. Assimilating adaptation context which might be provided in different context profiles of different formats, into the AM internal context representation model.

4.2.3 Extensibility

The multimedia adaptation domain is dynamic, new technologies and user requirements emerge rapidly. This implies that adaptation context processing would need to be extended to accommodate for such changes. Such extensions should be possible and methodical. Semantic Web technologies and XML provide the mechanisms to extend context processing both semantically and syntactically.

The system should also be able to extend system context processing to new standards. Supporting an extensible system of mapping XML to the internal context modelling approach is important.
Chapter 4. Adaptation Context Management

The author adopts OWL-based context modelling as explained in this chapter. XML to OWL mapping has not been used in state of the art content adaptation systems. However, lessons have been learned from formatting and mapping approaches in other systems such as in [140], [141] and [142].

Context or web data are often represented in XML. With the introduction of the Semantic Web, the focus now is on OWL context based modelling. As far as multimedia adaptation systems are concerned, context formatting is necessary to convert input profiles into the internal system context representation model. In state of the art adaptation systems, context information is extracted directly, i.e. hardcoded in the implementation, from the supported context profiles such as CC/PP in [77] or MPEG-7/21 in [86]. However, this has interoperability and extensibility implications (Section 4.3.1).

XML to OWL mapping approaches can be divided as follows:

- Mapping to an existing model such as in [142]: mapping to an existing ontology (not automatically generated) is vital to make efficient use of the semantic capabilities of OWL and ontologies (Section 4.5.3).

- Automatically generate OWL from XML according to automatic rules: this approach is very limited as discussed in Section 4.5.3.

Extensibility is essential to ensure both interoperability and completeness. Interoperability is ensured with extensibility by enabling easy extension to new context formatting tools in order to interoperate with new context formats. Extensibility enables extending the internal context representation model to new context parameters and users requirements to provide as complete as possible adaptation.

4.3 Proposed Solution: ACM

Based on the requirements of context management as discussed in the previous section, the framework depicted by Figure 18 defines the ACM entity proposed by the author. The issues being tackled are:

- Context modelling: not treated in most surveyed systems, due to the limited set of context information used. Moreover, the use of hardcoded internal data structures poses great extensibility issues.

- No support for a mechanism to extend context handling support to new multimedia context standards.
The proposed ACM depicted by Figure 18 aims to:

- Enable mechanisms to extend the context model, and hence provide a more context aware (complete) content adaptation.

- Model context using formal methods, i.e. use a knowledge representation formalism and XML to enhance context model extensibility and interoperability both syntactically and semantically.

- Support widely used and implemented standards to enhance inter-operability.

- Provide mechanisms to ensure extensibility to new adaptation formatting tools to interoperate with new context standards.

- Define an interface between the input adaptation context and the ADT logic. The aims of such interface are as follows:
  
  - Minimize the impact, i.e. on the rest of the system, of the changing adaptation context in terms of format and especially adaptation parameters due to emerging technologies and user adaptation preferences/requirements. Thus, enabling system extensibility for both context management and ADT at minimal costs (Sections 4.3.1 and 4.3.2).
Chapter 4. Adaptation Context Management

- Reduce the complexity of the adaptation decision rules by providing context at a high level, and by handling low level decisions at the context level (Sections 4.3.1 and 4.3.2).

In order to achieve the aims stated above, the adaptation context management framework (depicted by Figure 18) that defines the ACM entity constitutes:

- The input context assimilation interface (Section 4.3.1) that deals with context formatting and that uses several context transformers to handle the different context formats such as MPEG-7/21, UAProf, etc.

- The high level context interface (Section 4.3.2) that abstracts the complexity/dynamicity of input context to the ADT logic. The interface breaks the complexity of the ADT logic by handling the lower reasoning tasks at the context management level.

The two interfaces mentioned above are described below:

### 4.3.1 Input Context Assimilation Interface

Context assimilation is the process of loading input context profiles and formatting them into the internal context modelling format. Main context assimilation functions are context extraction and context transformation (or formatting).

This interface is depicted in Figure 18. The aim of this interface is to abstract the heterogeneity of the adaptation context that is the result of implementing different context standards by user environments, content and service providers and adaptation services providers.

The interface manages several context transformation tools (depicted by T₁......Tₙ in Figure 18). The transformation tools contain statements that map concepts from input context profiles to corresponding concepts in the internal context model.

If the description of the content or service is not provided, extraction tools are used to extract descriptions from the content or services.

The internal context model is an XML based ontology and the transformation tools are XSLT based. Using an XML based ontology context model and XSLT to map (i.e. transform) input context to the model has several benefits, the key benefit is to realise extensibility in terms of semantics and syntax as follows:

- Syntax extensibility: XML based context modelling and XSLT based transformation makes it easy to implement new transformation tools to map new context formats to the internal context model. This is evaluated in Section 7.5.1.1.
Semantic extensibility: the formality of ontology and the clear hierarchy and structure makes it easy to add new context parameters. Ontology reasoning enables us to detect inconsistencies that may arise for such additions.

Adaptation systems need to be extensible as new context parameters may need to be considered, for example, as new features are supported by user devices, new context types become more available (user movement, as some devices might have accelerometers). Input context needs to be mapped to a uniform model as context profiles may be of different formats even for the same adaptation request, for example, MPEG-7 for content description and UAProf for device and network description. Hardcoded context extraction from context profiles is not extensible. If this approach is used, the coding needs to be changed for new adaptation parameters or context formats. The hardcoded extraction approach uses internal data structures to represent context before it is converted to the internal context model. With regard to the AM, the internal context model is the XML based ontology model presented in the next section.

4.3.1.1 Internal Context Model: The ADIO

This section presents the internal context model proposed by the author: the ADIO. It is termed “Adaptation Decision Interface” because it acts as an interface between context assimilation and ADT. As depicted by Figure 18, the internal context model consists of high and low level concepts. Low level concepts are used by the context assimilation interface and high level concepts are used by the high level context interface.

XML ensures syntax interoperability by enforcing formal syntax. Semantic interoperability is to have a consistent and formal understanding of context concepts such as Display, User Characteristic, Presentation Preference, etc. DL is suited for such a purpose, i.e. describing the domain concepts and formally defining the relationships between them. To ensure both syntax interoperability and semantic interoperability, a combination of XML and DL is required. Ontology Web Language (OWL) is the most widely used and deployed solution that provides such combination. The difference between DL and ontology is that ontology is a concept and DL provides a powerful formalism to implement it.

Therefore, ADIO is an OWL ontology that models the adaptation domain context and is the format to which all input context is mapped. The different tools that can be used for context modelling, including why ontologies are preferable, are outlined in [62] and discussed in Chapter 2. The terminology and concepts underlying ADIO are based on MPEG-21 DIA Universal Environment Description (UED) [139] and MPEG-7 Multimedia Description Schemes (MDS) [143]. ADIO is constructed by manual conversion of the MPEG-21 DIA UED and MPEG-7 MDS from their XML format to the ADIO OWL based format. OWL is chosen because it is based on
Chapter 4. Adaptation Context Management

XML and DL and hence provides the advantages of both. MPEG-21 DIA UED and MPEG-7 MDS were chosen for context concepts (vocabulary) because they are the most comprehensive standards that define general concepts in the adaptation domain, example of concepts are device, device display, network, user preference, device capability, etc.

As well as low level context concepts and parameters based on MPEG-7 MDS and MPEG-21 DIA UED, ADIO also consists of higher level context that can be deduced such as battery_limited. ADIO can be extended to domain specific concepts defined by domain specific standards such as NewsML and SportML. Since Semantic Web technologies such as OWL can define context concepts formally and unambiguously; extensions to such concepts can be incorporated without causing redundancies or inconsistencies. Syntax wise, XML is designed to allow extensibility. As a consequence, ADIO extension is possible and methodical. More discussion on ADIO structure and usage is available in Sections 4.4.5.1 and 4.5.2.

4.3.1.2 Mapping Input Context to ADIO

As context standards and models including ADIO are XML based, it is possible to use XSLT to map between the input profiles and ADIO. Efforts to use XSLT to map XML to OWL are discussed in Section 4.5.3. The process of plugging new XSLT formatters should be extensible and should require minimal coding; this is discussed in the implementation and evaluation chapters.

4.3.2 High Level Context Interface

This interface represents context refinement. The components of this interface are the context model represented by the ADIO and a context reasoner (ontology reasoner since an ontology based model is used). ADIO was introduced in Section 4.3.1.1 and its structure is described in Section 4.5.2.

The aim of this interface is to abstract the complexity of context to the decision taking logic. Only concepts and parameters that are handled by the adaptation decision rules are included. Furthermore, the high level context is deduced from low level context by DL (or ontology) reasoning. Thus, the number of rules in the adaptation decision rules is reduced by reducing the set of context concepts.

By defining high level adaptation concepts in the context model, such as x-limited, where x is a resource such as memory or battery power, the introduction of new adaptation parameters that affect the definition of x-limited has minimal impact on ADT. The aim is to minimize the impact
of introducing new adaptation parameters or supporting new context standards on the adaptation decision rules.

As mentioned in Section 4.2, extensibility and interoperability are key requirements to tackle the complexity of the adaptation problem. Using formal syntax and semantic technologies such as XML and DL (Ontology) is the key to addressing these requirements. ADIO provides the benefits of both XML and DL as it is based on the Ontology Web Language (OWL).

With regard to the functions and aims of the high level context interface, ADIO has the following advantages:

- **Ease of Maintenance and Development**

  The time and effort it takes to model context as ontology is much less than the time to hardcode a model in an object oriented or a rule language for example. If context profiles are changed, the code for handling the ontology does not have to be changed, unless, there is a new high level concept that needs to be defined, in which case, the coding has to be changed to extract the new high level concept and feed it to the adaptation decision system. Ontologies and DL are better suited for describing concepts in terms of classes and properties compared to other modelling approaches, and hence ontology and DL ease context model development and maintenance tasks. Moreover, there is built-in support in DL reasoning to check the consistency of changes to the context model.

- **Reduced Complexity of Adaptation Decision Rules**

  DL reasoning capabilities can be used to refine context, by defining high level adaptation concepts that function as adaptation rules. This means reducing the complexity of the adaptation decision rules as the complexity is shared between the context management entity and the decision engine. If input context profiles are converted to another context model, the adaptation decision rules have to model the rules that would be otherwise modelled using ADIO high level concepts. Two types of adaptation rules are distinguished for the AM:

  - **Higher level (Refined context) concept rules**: these are rules used to derive higher level context, for example that a device is battery limited via the `battery_limited` concept.
  
  - **Adaptation decision rules**: these are rules used to derive adaptation decisions, for example, to solve the battery limitation problem, the decision from running the rules might be to reduce image resolutions, remove videos and replace them with audio streams, etc.
• **Extensibility**

In the context of this section, extensibility is measured at both the input and output ends of the ACM rather than the extensibility of ADIO to new concepts. At the input end, extensibility is measured by the ease of developing formatting tools to map context standards to ADIO. This can be enhanced by adopting a flat structure in order to minimize and simplify the coding necessary to navigate the ontology structure (Section 4.5.2.1). At the output end, extensibility is measured by the ease of maintaining the code that extracts the necessary context to feed to the decision engine. This can also be enhanced by defining a simple ontology structure as described in Section 4.5.2.1.

• **Semantic Web compatibility**

Semantic Web technologies are playing a crucial part in the context and multimedia domain, thus, Semantic Web compatibility is important. If a Semantic Web based context standard or an import tool emerges, adoption of such a tool or standard is possible if Semantic Web APIs and technologies are supported.

• **Support**

Moreover, Semantic Web technologies have a wide community and enjoy good research and application support. Thus DL reasoning is becoming a powerful tool for reasoning. For example, there is continuing research on expressiveness extensions and performance improvements on ontology languages and ontology reasoning. The latest version of OWL, OWL 2.0 is an example [144]. Several ontology editing tools integrated with ontology reasoning tools are available, for example the Protégé editor [145] and the Pellet [53] and Fact++ [54] reasoners.

• **DL vs. Rule Based Modelling**

It should be noted that a rule based system is the only extensible approach that can be an alternative to ADIO. ADIO is used instead of a rule based system for two main reasons. Firstly, DL and Ontology are better suited for modelling a domain such as the adaptation domain while rule based systems are better suited for modelling adaptation decisions. Secondly, wide support and tools exist for ontology development and reasoning.

4.4 **Context Assimilation and Refinement Stages**

This section presents the different stages of context assimilation and refinement as depicted by Figure 19. The input to each function is indicated by normal arrows and the output of each function is indicated by the bold dashed arrows.
Chapter 4. Adaptation Context Management

Before further discussion, the following is reiterated:

- **Context assimilation**: is the process of loading context related to an adaptation request and formatting it into the internal context model, i.e. the ADIO. This process includes context extraction and context formatting.

- **Context refinement**: is the process of deriving high level adaptation context, i.e. calculating the high level concepts of the assimilated (extracted and formatted) adaptation request context.

The four main functions that are implemented with regard to context assimilation (depicted by Figure 19) and refinement are context extraction, context formatting, which formats context input to ADIO, pre-reasoning parameters calculation, which prepares ADIO for reasoning, and context refinement, which defines the values of high level concepts in ADIO using DL reasoning. These functions and related data types and predicates are formally defined in the next section.
4.4.1 Formal Specification of Context Assimilation and Refinement Functions

This section formally specifies context assimilation and refinement functions in the VDM based notation that was introduced in Chapter 3. Figure 20 depicts the high level algorithm of implemented context assimilation and refinement functions depicted by the block diagram in Figure 19.

```plaintext
1  assimilateContext (request) \Delta let request ∈ requestL in

2     if anyContextNotAvailable (request)

3     getContextProfiles (request)

4     requestUnavailableProfiles (request)

5 if contentServiceContextNotAvailable (request)

6     extractContentContext (request)

7     if contextAvailable(request)

8         formatContext (request)

9     if contextFormatted(request)

10       calcPreReasoningParameters (request)

11       refineContext(request)

12 else abort (request)

13 else abort (request)
```

Figure 20: Context assimilation algorithm

The expression "let request ∈ requestL in" in line 1 of Figure 20 denotes that the algorithm is concerned with requests from the list "requestL", which refers to the list of all adaptation requests that are supported by the system. ARequest, the type of adaptation requests, and requestL are defined in Section 3.3 of Chapter 3.

The clause.

\[ \forall request \in requestL. \text{contextNotAvailable}(request) \Rightarrow \exists prof \in CtxtProfileL(request), \text{status}(prof) = \text{unAvailable} \]
In VDM, retrieving the value of a field or a property in an object is performed for example as in the above definition: \textit{CtxtProfileL (request)}, which retrieves the value of the \textit{CtxtProfileL} field of the \textit{request} object. \textit{CtxtProfileL} is a list of \textit{CtxtProfile} objects, which are defined as follows:

\begin{verbatim}
CtxtProfile :: format : ProfFormat
    URI : token
    type : ProfType
    status : ProfStatus
    necessary : Boolean

ProfType : ProfTypeL
ProfFormat : ProfFormatL
ProfStatus : ProfStatusL
CtxtProfileL : CtxtProfile-Set
\end{verbatim}

\textit{ProfTypeL} is a list that contains the possible types of adaptation profiles. It is defined as:

\begin{verbatim}
{user, device, network, mixed, environment, content}
\end{verbatim}

\textit{ProfStatusL} is a list that contains possible adaptation profiles status. It is defined as:

\begin{verbatim}
{available, formatted, refined, unavailable}
\end{verbatim}

\textit{ProfFormatL} is a list of possible profile formats, for example, MPEG-7, MPEG-21, UAPROF, etc.

The function \textit{getContextProfiles (request)} in line 3 of Figure 20 requests the context profiles that are not available; it is defined in lines 4 to 6 of the algorithm. The function can be defined in terms of pre and post-conditions as follows.

\begin{verbatim}
getContextProfiles (request : ARequest)
    Pre contextNotAvailable (request)
    Post contextAvailable (request)
\end{verbatim}
For example, if the device context is not available, the adaptation system requests it from the user environment (or the environment where the request originated).

Line 5 of Figure 20 denotes that the adaptation system checks whether it is necessary to extract content context. Extraction may be required for embedded content context, for example in the form of Exchangeable Image File Format (EXIF) [146] metadata, or may be required for extraction of non available metadata such as web page layout.

Content or service context unavailability (line 5 of Figure 20) can be formally defined as:

\[
\forall \text{request} \in \text{requestL}, \text{ContentServiceContextNotAvailable(request)} \iff \exists \text{prof} \in \text{CtxtProfileL(request)}, \text{type(prof)} = \text{content} \land \text{status(prof)} = \text{unavailable}
\]

Status (prof) is used to check the value of the status field of the prof object and type (prof) is used to check the value of the type field of the prof object. In the case of services where more than one content item is delivered, content and service context is not available if service context or one of the content items contexts delivered by the service is not available.

ExtractContentContext (request) in line 6 of the algorithm in Figure 20 aims to extract content/service context if not available.

Line 7 of Figure 20 checks if all necessary context is available, the adaptation process would be started if other “unnecessary” context is not available. The definition of “unnecessary” and “necessary” context can differ between adaptation systems. In this thesis, necessary context is defined as the context of the device, content and network connection. Other types of context such as user preferences and usage environment descriptions would enable more context aware adaptation and customization but are not necessary to render the context to the user device. However, the set of necessary context and unnecessary context can be changed according to adaptation requirements.

The availability of context can be defined as:

\[
\forall \text{request} \in \text{requestL}, \text{contextAvailable(request)} \iff \forall \text{prof} \in \text{CtxtProfileL(request)}, \text{necessary(prof)} = \text{true} \Rightarrow \text{status(prof)} = \text{available}
\]

The function formatContext (request) in line 8 of Figure 20 aims to format the context; the function is defined as follows:
Chapter 4. Adaptation Context Management

**formatContext** (request : ARequest)

**Pre** contextNeedsFormatting(request) ∧ formattingSupported(request)

**Post** contextFormatted(request)

The pre condition of the above function can be formally defined as:

$$\forall request \in requestL. contextNeedsFormatting(request) \iff \exists prof \in CtxtProfileL(request), format(prof) \neq AMformat$$

This means that at least one context profile does not match the adaptation system internal context format (i.e. ADIO).

The post condition of the above function can be defined as:

$$\forall request \in requestL. contextFormatted(request) \iff \forall prof \in CtxtProfileL(request), necessary(prof) = true \Rightarrow status(prof) = formatted \vee status(prof) = refined$$

This means that all necessary context profiles are in the internal format that is supported by the adaptation system (ADIO). Details on context formatting are provided in Section 4.4.4.

The operation `calcPreReasoningParameters (request)` in Line 10 calculates the values of certain data type properties in the context model. More information about this is provided in Section 4.4.4.2. This operation can be defined as:

**calcPreReasoningParameters** (request : ARequest)

**Pre** contextFormatted(request)

**Post** preReasoningParamCalculated(request)

The post condition means that values have to be calculated, i.e. pre-reasoning functions have to be applied, for data type properties that satisfy the following two conditions:

1. The data type property is a function of other data type properties in the context model
2. The values of these data type properties are all assigned.
This condition can be formally defined as:

\[ \forall \text{request} \in \text{requestL}. \ \text{preReasoningParamCalculated} (\text{request}) \iff \exists \{y, x_1, x_2, \ldots, x_n\} \subset \text{dataProperties} (\text{contextModel} (\text{request})), \]

\[ \text{value}(y) = f (\text{value} (x_1), \text{value} (x_2), \ldots, \text{value} (x_n)) \land \]

\[ \{\text{value} (x_1), \text{value} (x_2), \ldots, \text{value} (x_n)\} \cap \{\text{null}\} = \emptyset \]

\[ \Rightarrow \text{value} (y) \neq \text{null} \land \text{value} (y) = f (\text{value} (x_1), \text{value} (x_2), \ldots, \text{value} (x_n)) \]

The statement \( \text{value} (x_1) \) is a function used to get the value of a property \( x_1 \), for example if the property \( \text{availableDeviceBatteryLife} \) has the value 0.4 ampere, then \( \text{value} (\text{availableDeviceBatteryLife}) = 400 \). \( \text{dataProperties} (\text{contextModel} (\text{request})) \) denotes the data properties presents in the context model i.e. instance ontology of the request. The \( "f" \) in the definition represents the pre-reasoning functions that are applied.

\( \text{refineContext} (\text{request}) \) in Line 11 of the algorithm aims at refining the context and deducing high level context from low level context parameters. This function can be defined as:

\[
\text{refineContext} \quad (\text{request} : \text{ARequest})
\]

\[
\text{Pre} \ \text{preReasoningParamCalculated(request)}
\]

\[
\text{Post} \ \text{contextRefined(request)}
\]

The post condition of the above function can be formally defined as:

\[ \forall \text{request} \in \text{requestL}. \ \text{contextRefined} (\text{request}) \iff \forall \text{prof} \in \text{CtxtProfileL(request)}, \]

\[ \text{necessary} (\text{prof}) = \text{true} \Rightarrow \text{status} (\text{prof}) = \text{refined} \]

\subsection*{4.4.2 Adaptation Mechanisms and Adaptation Operations Context Acquisition}

This stage is not related to processing adaptation requests, it mainly occurs at adaptation system start-up upon adaptation mechanisms service discovery.

- **Adaptation Mechanisms Context**

The Content Adaptor entity from the ADME management subsystem (Figure 11) constantly updates the AM with newly discovered adaptation mechanisms or with the unavailability of previously discovered mechanisms.
Adaptation Operations Context

Adaptation mechanisms are implementations of adaptation operations. All adaptation operations that are supported by the system (i.e. adaptation decision rules) are loaded at system start up. The following is the definition of the adaptation operation type (AOperation):

\[
\text{AOperation} :: \text{ID : token}
\]

- Implementing ADMEs
  - \text{opName : token}
- \text{inModality_inFormat_inLang : token}
- \text{outModality_outFormat_outLang : token}
- \text{inParameters : admeID}^m \rightarrow \text{Parameter-Set}
- \text{outParameters : admeID}^m \rightarrow \text{Parameter-Set}
- \text{admeID = token}

\text{inParameters} and \text{outParameters} are defined as maps between an ADME identifier and a set of parameters. Such mapping is necessary because, if this link cannot be identified, a problem may occur with regard to ADMEs in the following situation: The first ADME supports the input but not the output parameters of an adaptation decision. The second ADME supports the output but not the input parameters for the same adaptation decision. In such case, the decision cannot be executed because the input and output parameters need to be supported by the same ADME.

An Adaptation Mechanism (ADME) is defined as follows:

\[
\text{ADME} :: \text{ID : token}
\]

- \text{inParameters : Parameter-Set}
- \text{outParameters : Parameter-Set}
- \text{opName : token}
- \text{inModality_inFormat_inLang : token}
- \text{outModality_outFormat_outLang : token}

79
Furthermore, an adaptation parameter is defined as follows:

\[
\text{Parameter} \quad ::= \quad \text{attributes} \quad ::= \quad \text{name} \rightarrow \text{value} \\
\text{name} \quad ::= \quad \text{token} \\
\text{value} \quad ::= \quad \text{token}
\]

The algorithm depicted by Figure 21 explains in detail how adaptation mechanisms and adaptation operations context is managed, specifically, how adaptation mechanisms are added to the adaptation system upon discovery.

```
1 addADME (request) \triangleq \text{let adme} \in \text{requestL in admeList} \land \text{op} \in \text{opList}

2 \text{if correspondingOPExists (adme)}

3 \quad \text{op} = \text{correspondingOP (adme)}

4 \text{if not active (op)}

5 \quad \text{activate (op)}

6 \quad \text{addToOp (op, adme)}

7 \text{else return}
```

Figure 21: Adaptation operations management

Before explaining the algorithm the following definitions are introduced:

- **activeopList**: The list of adaptation operations that have corresponding active ADMEs.
- **opList**: The list of all adaptation operations supported by the system, supported means modelled (or used in the adaptation decision rules).
- **admeList**: The list of adaptation mechanisms which the adaptation system can discover (or load in case of applications) and invoke to execute adaptation decisions.
- **activeadmeList**: The list of adaptation mechanisms that can be invoked to realise adaptation decisions (i.e. has been discovered or loaded).
Chapter 4. Adaptation Context Management

Line 2 of the algorithm depicted in Figure 21 checks if a corresponding adaptation operation that is supported by the adaptation system exists, otherwise, the discovered adaptation mechanism is discarded (line 7 in Figure 21).

Checking the correspondence of an adaptation operation can be defined formally as:

\[
\forall \text{adme} \in \text{admeList}. \text{correspondingOPExists (adme)} \iff \\
\exists \text{op} \in \text{opList}, \text{adme (opName)} = \text{op (opName)} \land \\
in\text{Modality}_\text{inFormat}_\text{inLang}(\text{op}) = \text{inModality}_\text{inFormat}_\text{inLang}(\text{adme}) \land \\
out\text{Modality}_\text{outFormat}_\text{outLang}(\text{op}) = \text{outModality}_\text{outFormat}_\text{outLang}(\text{adme})
\]

\text{inModality}_\text{inFormat}_\text{inLang} and \text{outModality}_\text{outFormat}_\text{outLang} are functions used to retrieve the input and output formats, modality and language in the form of video_mpeg4_eng and audio_mp3_eng for example. This information together with the adaptation operation name determines if the adaptation operation corresponds to the adaptation mechanism, i.e. if the adaptation operation name and input and output modalities, formats and languages match.

Line 3 of the algorithm in Figure 21 assigns the ADME corresponding adaptation operation to an adaptation operation (op) object.

Line 4 of the algorithm depicted in Figure 21 checks if the adaptation operation is active. The activeness of an adaptation operation is determined by the list of implementing adaptation mechanisms and can be formally defined as follows:

\[
\forall \text{op} \in \text{opList}. \text{active(op)} \iff \text{admeL(op)} \neq \emptyset
\]

Line 5 of the algorithm depicted in Figure 21 activates an adaptation operation, the activate (op) operation is defined as follows:

activate(op) (op: AOperation)

Pre \text{op} \in \text{opList}

Post \text{op} \in \text{activeOPList} \land \text{status (op)} = \text{active}
Line 6 of the algorithm depicted in Figure 21 associates the ADME to the adaptation operation

\[
\text{addToOP}(\text{op}, \text{adme}) \quad (\text{op}: \text{AOperation}, \text{adme}: \text{ADME})
\]

\[
\begin{align*}
\text{Pre} & \quad \text{op} \in \text{opList} \land \text{adme} \in \text{ADMEList} \\
\text{Post} & \quad \text{id (adme)} \in \text{ImplementingADMEs(op)} \\
& \quad \text{adme} \in \text{activeADMEList}, \text{op} \in \text{activeOPList}
\end{align*}
\]

4.4.3 Context Extraction

Context extraction tools are maintained by the adaptation system and can be used for:

- Extraction of Embedded Metadata from Content

Metadata could be embedded in the content; in which case it needs to be extracted. An example is EXIF image metadata which can be embedded in JPEG files.

- Extraction of Features from Content

In the case where metadata is neither embedded nor external, tools can be used to parse the content and extract the physical characteristics. This has response time implications and should be avoided where possible.

4.4.4 Context Formatting

Context formatting is performed in two stages explained below. The first stage, explained in Section 4.4.4.1, formats the context profile to the internal context model (ADIO). The second stage, explained in Section 4.4.4.2, calculates pre-reasoning parameters.

4.4.4.1 Formatting Context to the Internal Model: ADIO

The adaptation management system maintains a list of formatter tools in the formatterTools list.

A context profile format is supported if a formatting tool that supports the profile format can be found; this can be formally defined as:

\[
\forall \text{request} \in \text{requestList}. \text{formattingSupported(request)} \iff \forall \text{prof} \in \text{CtxProfileL(request)}, \exists \text{ft} \in \text{formatterTools}, \text{format(ft)} = \text{format(prof)}
\]

Context formatting is crucial to interoperability. Several context standards in the content and service adaptation domain exist, such as MPEG-7, MPEG-21, CC/PP, UAProf, NewsML, etc.
The context formatting mechanism provided by the adaptation management system has to be extensible. Adoption or implementation of new context formatting tools to support new formats or extension to new features in existing formats must be possible. The use of XSLT for the first stage of context formatting is presented in Section 4.3.1.2. Details on context formatting implementation are provided in Chapter 6 and evaluation is provided in Chapter 7.

4.4.4.2 Pre-reasoning Parameters Calculation

This step is necessary, to prepare the formatted context for context refinement, because the internal context model, based on ontology (OWL), and ontology reasoning have no mechanisms to deduce types using data types reasoning based on the values of two or more data properties. For example: assume that \texttt{device.remainingBatteryPower} = x, and \texttt{service.requiredBatteryPower} = y where (device, service) are instances in the ontology, (remainingBatteryPower, requiredBatteryPower) are data type properties and (x,y) are integers. It is not possible to deduce the concept: \texttt{BatteryPowerConstrained} based on the values of x and y.

This problem can be approached in two ways:

- **Changing Ontology Structure**

For example, if the concept \texttt{BatteryPowerConstrained} is defined as follows:

\[ \exists \text{remainingBatteryPower} \leq x \]

The value of x has to be changed for each adaptation request with the value of \texttt{requiredBatteryPower}. This approach implies that the structure of the ontology is changed and re-classification is triggered. Hence this approach is not efficient.

- **Using Pre-Defined Functions and Comparing to Normalised Values**

The author proposes a pre-reasoning parameters calculation step to avoid re-classification of the ontology by avoiding changing the ontology structure. For example, the \texttt{BatteryPowerConstrained} concept can be defined as follows:

\[ \exists \text{batteryLimitation} \leq 0 \]

The data type property \texttt{batteryLimitation} denotes the value \texttt{remainingBatteryPower} – \texttt{requiredBatteryPower}. \texttt{batteryLimitation} is always compared to 0 regardless of the values of \texttt{remainingBatteryPower} and \texttt{requiredBatteryPower}. Similarly the following can be used:

\[ \exists \text{batteryLimitation} \leq 1 \]

In this case the data type property \texttt{batteryLimitation} denotes the value:

\[ \text{remainingBatteryPower/requiredBatteryPower} \]
This step of context formatting assigns the values of data type properties such as batteryLimitation, which is named a pre-reasoning parameter as it important to the context reasoning (refinement) step.

It is very important that the implementation of this step does satisfy important requirement targeted by this research, namely extensibility and maintainability. The implementation of this step is presented in Chapter 6 and evaluated in Chapter 7.

4.4.5 Context Refinement

As presented in Section 4.3.1.1, the internal context model is an ontology based structure implemented in the OWL language because OWL is the most powerful combination between XML and DL. The reasons for this choice are explained in Section 4.3.2. Hence, context refinement is a DL (Ontology) reasoning process. DL (Ontology) reasoning is presented in Chapter 2. The next section discusses how the ADIO is used for context refinement.

4.4.5.1 Using the ADIO

As mentioned in Section 4.3.2, DL (and ontologies) provides efficient modelling of domain concepts such as content and service adaptation domain concepts. The enhanced expressiveness of OWL 2.0 broadens the concepts that can be modelled, for example using data types reasoning and negative property assertions (Sections 5.4.3.1 and 5.4.3.2). By defining high level concepts using fixed values and including a pre-reasoning parameters calculation step (Section 4.4.4.2), re-classification is avoided and performance is enhanced. The ontology reasoner compares the pre-reasoning parameters to the fixed values as described in Section 4.4.4.2.

In order to enhance extensibility and maintainability, adaptation rules that can be defined as high level concepts are defined in ADIO rather than being defined as part of adaptation decision rules (Section 4.3.2). High level concepts are more affected by context parameters changes, for example to accommodate for new adaptation scenarios as a result of extending to new device capabilities or new user requirements. An adaptation decision rule can be of the form:

\[
\text{Battery-limited (deviceX)} \rightarrow \text{someDecision (deviceX)}
\]

This rule would not be affected if the parameters that affect the concept battery-limited change. What is affected is the high level concept battery-limited. If for example battery-limited is defined as follow:

\[
device.\text{requiredBattPower} < device.\text{availableBattPower}
\]
Then, the definition of battery Limited can be required to change if for example a user preference, with regard to the remaining battery power, needs to be taken into account. For example

\[
\text{device.requiredBattPower} \text{< } \text{device.availableBattPower - user.remainingBattPowerPreference}
\]

OWL 2.0 enables defining high level concepts such as battery-limited, using data types reasoning, and hence reduces the complexity of the adaptation decision rules.

### 4.5 Ontology and DL for Adaptation Context

This section examines the use of DL and ontology for content and service adaptation context management. Two main scenarios of using ontologies and DL are identified:

- **Adaptation Domain Ontology (ADO)**: i.e. as a means to describe the entities that exist in the adaptation domain. Such ontology is used for adaptation request annotation. More discussion on ADO is provided in Section 4.5.1.

- **Adaptation Decision Interface Ontology (ADIO)**: i.e. in order to model context internally in the adaptation system. Such ontology is different from ADO and is used as an interface between context assimilation and ADT as presented in Section 4.3. More on ADIO is presented in Section 4.5.2.

Table 9 presents the main differences between ADIO and ADO. It might be possible that parts of an ADO ontology are used as the internal context representation model. However, a different ontology that better serves certain requirements could be implemented, i.e. ADIO in this thesis.

<table>
<thead>
<tr>
<th>Feature</th>
<th>ADIO</th>
<th>ADO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standardisation</td>
<td>Not important</td>
<td>Crucial</td>
</tr>
<tr>
<td>Wide scope</td>
<td>Only context used by the system</td>
<td>Crucial</td>
</tr>
<tr>
<td>Reasoning Efficiency</td>
<td>Crucial</td>
<td>Desirable</td>
</tr>
<tr>
<td>Human readability</td>
<td>Not important</td>
<td>Desirable</td>
</tr>
<tr>
<td>Simple Structure</td>
<td>Crucial</td>
<td>Not important</td>
</tr>
<tr>
<td>Compactness</td>
<td>Crucial</td>
<td>Not important</td>
</tr>
</tbody>
</table>
The first column in Table 9 represents features relevant to ADIO and ADO. The first feature is standardization; the benefits of having a standardized ADO are obvious. Such ontology is a major task and would play the role of standards such as MPEG-21. Standardization is not as important for ADIO; in fact, an ADIO might be optimized and hence not shared with competitors, especially as ontology languages become more expressive and are used to perform more complex reasoning tasks.

The scope of ADO would be as large as possible covering as many adaptation concepts as possible. ADO construction requires a collaborative effort from the content adaptation community. On the other hand, ADIO models only contexts that are relevant to the system in order to reduce context model size and hence achieve better performance. This means that ADIO is optimised for reasoning and access, which requires a simple ontology structure. The requirement of clear hierarchy and classification is more important for ADO, and hence ADO structure would be more complex in terms of the number of sub-classing levels and relationships between ontology classes (Section 4.5.3.5).

Both ADO and ADIO ontologies should be constructed manually and based on current context description standards, which are mostly XML based. The design decisions on the structure of ADO and ADIO vary as will be illustrated with examples in Section 4.5.3.

Regardless of an ontology being an ADIO or ADO, there are three scenarios of constructing and using ontologies for context description in the multimedia domain.

- **Manually Construct an Ontology**
  Systems such as in [73] construct an ontology that models concepts according to application requirements.

- **Manually Construct an Ontology Based on Existing Standards**
  Systems such as in [147, 148] use vocabulary from existing standards such as MPEG-21 to manually construct an ontology. Manual conversion examines and models each XML element according to its semantics.

- **Automatically Construct an Ontology Based on Existing Standards**
  Systems such as in [141] automatically converts XML elements into ontology using fixed rules, for example converting every XML complex type to an ontology class.

In Section 4.5.3, both manual and automatic conversion approaches are examined in the light of both ADIO and ADO. MPEG-21 part 7 DIA and parts of MPEG-7 MDS were converted manually; examples to show the inefficiency of automatic conversion are given in Sections 4.5.3.3 to 4.5.3.6.
4.5.1 ADO

This section discusses the ADO. A shared ontology between the different entities involved in content/service adaptation is a vital step towards the Semantic Web and towards efficient multimedia application and processing including adaptation [41]. Such ontology would enable multimedia content and multimedia adaptation requests to be annotated semantically and hence lead to more relevant adaptations. Being built on top of RDF and XML, OWL enjoys the syntax interoperability of XML and builds on RDF to take context processing to a higher level enabling computers to process the meaning of terms through relationships between them. This is achieved by means of reasoners such as RACER and Pellet, a list of other DL reasoners is available in [149]. What makes ontology and OWL suitable for the task is that it is based on XML and DL which are powerful syntax and knowledge representation formalisms, respectively.

The first step in the development of ADO is to identify the main sub domains of adaptation. This requires knowledge of the physical characteristics of the different entities involved in the adaptation process. These entities can be defined in ADO as part of one of the following sub domains: content/service, user, terminal, network and natural environment. MPEG-7 and MPEG-21 serve the purpose of identifying a comprehensive adaptation-related vocabulary set by offering several tools (description schemes) to annotate multimedia content at different levels and consumption/creation environments respectively. DIA (Part 7) [139] and MPEG-7 (part 5) MDS are most relevant to adaptation and should form the basis of ADO. MPEG-21 DIA provides tools to describe the usage environment including: usage characteristics (user info, usage history, preferences and physical characteristics such as disabilities), Device characteristics (display, memory and battery), Network characteristics (error characteristics and bandwidth) and Natural environment characteristics (noise and illumination). MPEG-7 provides multimedia description tools. Constructing ADO from available XML standards is discussed in Section 4.5.3.

Constructing the adaptation ontology from MPEG-21 and MPEG-7 has direct benefits. MPEG research in multimedia environment description is established and their standards are well accepted within the multimedia community. Many frameworks and environments already use the MPEG-7 standard and the interest in MPEG-21 is reflected by several research projects based on its parts, for example, part 7 (DIA) is used by adaptation related research projects [36, 81, 86, 131-133]. Reasons for using OWL over other context modelling approaches including XML are discussed in Section 4.3.2.

Interest in ADO has resulted in the creation of a taskforce by W3C, the multimedia annotations task force, within the best practices and deployment working group [150] to study efforts to
convert MPEG-7 to Semantic Web standards, mainly OWL and RDF, and to provide a common ground for researchers in this area.

4.5.2 ADIO

The ADIO, introduced in Section 4.3.2, is proposed by the author to achieve extensible and efficient context modelling and processing. This section discusses ADIO structure. ADIO aims to interface between adaptation context assimilation and ADT. The interface is based on the concepts defined in MPEG-21 DIA and MPEG-7 and is implemented in OWL 2.0. The application of ADIO is internal context modelling and differs from ADO; hence design decisions differ between ADO and ADIO.

4.5.2.1 ADIO Concepts

The ADIO contains low and high level concepts. In order to enhance extensibility, a simple structure is adopted. A simple ontology structure would enhance extensibility with regard to:

- **Developing formatting tools**: a simple structure eases the task of creating/ extending a context formatting tool by requiring less transformation rules compared to a complex structure which requires more concepts in the ontology and hence more transformation rules.

- **Access to ontology**: a simple structure simplifies the coding required to access the ontology and extract the required information to feed into ADT. If changes to the context model are required, changes to coding designed for a simpler structure requires less maintenance.

A simple structure means a flat ontology structure. Examples where a flat ontology structure can be applied rather than a complex ontology structure at expense of semantics are given in Section 4.5.3.5.

- **ADIO Low Level Concepts**

For reasoning efficiency, it is important that only context that is required by adaptation decision rules (i.e. is part of the input to the decision system) is modelled. For example, if there is no adaptation rule that uses the location context, location context should not be modelled as part of ADIO. Limiting the scope of ADIO to necessary context will reduce the ontology size and hence improve reasoning efficiency.

Examples of low level concepts are depicted by the UML class diagrams in Figure 22.
Aiming for a simple structure is noticed from the low level parameters depicted by Figure 22. For example, the CodecCapability type contains fields that represent average and maximum bitrates. This is used as opposed to pointing to a bit rate parameter that would be an instance of a BitRate type which would contain the average and maximum bit rate properties. This is further discussed in Section 4.5.3.5.

- **ADIO High Level Concepts**

High level concepts aid the extensibility of the adaptation systems by abstracting the decision taking from changes to modelled context by defining as much high level concepts as possible. The evaluation of ADIO in terms of extensibility and ease of maintainability is discussed with the evaluation of system extensibility to new adaptation scenarios and system maintainability in terms of changing user requirements in Chapter 7. Examples of high level concepts are list below:

- **Simple High Level Concepts**

The definition of simple concepts does not involve other concepts, an example is given:

\[
\text{SomeModalityLimited} \equiv \exists \text{requiredModality}(\text{someModality}) \land \lnot \exists \text{supportedModality}(\text{someModality})
\]

Where \text{someModality} can be video, image, text, audio or graphics and \text{supportedModality} and \text{requiredModality} are data type properties. The ability to assert negative property assertions (\lnot \exists \text{supportedModality}(\text{someModality})) is crucial for this concept definition and is among the features introduced in the latest version of OWL (OWL 2.0). The concept \text{SomeFormatLimited} can be defined in the same way. Another example is:

\[
\text{SeverelyBandwidthLimited} \equiv \exists \text{bandwidthLimitation} < -2000
\]
bandwidthLimitation is a data type property the value of which is calculated by subtracting the
required bandwidth from the available bandwidth. This is calculated in the pre-reasoning
parameter calculation stage as described in Section 4.4.4.2. In the same way, other limitations can
be defined, for example the BatteryLimitation concept.

Complex High Level Concepts

Complex concepts may involve other concepts in their definition, for example, the
NoVideoDeliveryRequired concept can be defined as follows:

\[
\text{NoVideoDeliveryRequired} = \text{SeverelyBandwidthLimited} \cup \text{VideoLimited} \cup \exists \text{userFirstPreferredModality}(\text{Images})
\]

OWL 2.0 enables defining high level concepts, which would otherwise have to be defined as part
of the adaptation decision rules. Defining high level concepts as part of ADIO reduces the
complexity of the adaptation decision rules as discussed in Section 4.3.2.

4.5.3 Construction of Ontology from XML Schema Definition

This section examines efforts to bring the Semantic Web closer to reality, specifically, using
Semantic Web technologies such as ontology languages. One of these efforts is to automatically
convert XML data into OWL data by means of automatically converting XSD. In this section, the
XML schema definition is the MPEG-21 DIA and MPEG-7 MSD schemas for multimedia
adaptation and description respectively.

It should be noted that constructing ontology such as ADO should not aim to completely replace
XML, in this case MPEG-7/21. An example where ADO cannot replace MPEG-21 is for
modelling AdaptationQoS which specifies mathematical constructs such as stack functions, look
up tables, switch functions and variables.

4.5.3.1 Automatic Conversion

Figure 23 (b) depicts the process of automatic conversion from XML to OWL. Automatic
conversion converts XML elements into OWL constructs using fixed rules, for example
converting every XML complex type to an OWL class. Automatic conversion of an XML tree
structure results in an ontology that describes the relationships between the tree elements instead
of describing the relationships between the semantics embodied by the tree elements. For
example, there exist several efforts, stated in [151], to automatically construct ontology
representations of MPEG-7/21 by means of XSL transformation according to the rules specified
in [141]. Although automatic conversion expresses the XML-based standards in an OWL or RDF format, it does not add much to the expressiveness provided by the XML version. In [140], an ontology based on MPEG-7/21 has been constructed using automatic conversion rules, and was used to facilitate the development of knowledge-based multimedia applications such as multimedia information retrieval, filtering, browsing, interaction, extraction, segmentation, and content description.

4.5.3.2 Manual Conversion

On the other hand, manual conversion (Figure 23 (a)), such as in [148], examines and models each XML element according to its semantics. For an expressive OWL representation of the XML-based standards, manual conversion is necessary. Rather than merely writing an XML document tree in an OWL format, as performed by automatic conversion, manual conversion could remove XML elements or add OWL concepts depending on the semantics of the XML element being converted from XML to OWL. This is because some XML elements in XML documents do not have semantic significance and are created, for example, to make the syntax easily readable or well structured.

Manual conversion is necessary only once, depicted by the “Static” stage in Figure 23 (a), when constructing the ontology. Input context represents instance ontologies, which consist of instances such as “Nokia 95”, “Tom” and “contentX”, and can be converted automatically, depicted by the “Dynamic” stage in Figure 23 (a), using XSLT by the context formatter (denoted by T Rules in Figure 23). The definition of the concepts: device, user and content are pre-defined in the manually constructed ontology.
There are no set rules on how to manually convert an XML schema description into OWL ontology. XML schema does not describe the semantics of the relationship between its elements and attributes. It only describes the structure of an XML document (i.e. nesting of elements) and supported types of elements and attributes. The converter (a human) has to examine the elements and the attributes of the XML schema and study their semantics and translate them into OWL constructs.

As opposed to automatic conversion, manual conversion results in a clearer hierarchy that makes more sense to the human reader and organizes relationships between ontology concepts according to the semantics of elements as described in the XML specification (MPEG-7/21 in this example). Automatic conversion on the other hand, merely translates the XML syntax to OWL Syntax. Manual conversion could be described as manual construction of an ontology using a vocabulary set defined by an XML standard (MPEG-7/21 in this example) by carefully assigning the relationships between the defined classes according to the vocabulary semantics specified in the XML standard.

In the next subsections, examples are given to compare between manual and automatic conversion with regard to different aspects such as ontology application, accuracy of modelling and efficiency of reasoning.

4.5.3.3 Modelling Input Validation Elements

The XML specification in Figure 24 denotes that either multimedia content creation information or its reference is specified. Such restriction does not need to be implemented in OWL as it is not necessary for reasoning. Properties to represent both CreationInformation and CreationInformationRef can be implemented, however creating complex reasoning statements to ensure that any description that contains both is flagged inconsistent is unnecessary. To model such restrictions, complex statements involving the union, intersection and complement-of OWL constructors are required. For efficiency, input validation could be left to the input interface.

```
<choice minOccurs="0" maxOccurs="1">
  <element name="CreationInformation" type="mpeg7:CreationInformationType"/>
  <element name="CreationInformationRef" type="mpeg7:ReferenceType"/>
</choice>
<choice minOccurs="0" maxOccurs="1">
  <element name="UsageInformation" type="mpeg7:UsageInformationType"/>
  <element name="UsageInformationRef" type="mpeg7:ReferenceType"/>
</choice>
```

Figure 24: MPEG-7 choice tag (1)
The following choice example is different from the above example because of the unbounded max-occurrence restriction of the choice tag.

```xml
<complexType name="MixedContentType">
  <complexContent>
    <extension base="mpeg7:AudioVisualContentType">
      <choice minOccurs="1" maxOccurs="unbounded">
        <element name="Image" type="mpeg7:stillRegionType"/>
        <element name="Video" type="mpeg7:VideoSegmentType"/>
        <element name="Audio" type="mpeg7:AudioSegmentType"/>
        <element name="AudioVisual" type="mpeg7:AudioVisualSegmentType"/>
        <element name="ContentRef" type="mpeg7:ReferenceType"/>
      </choice>
    </extension>
  </complexContent>
</complexType>
```

**Figure 25: MPEG-7 choice tag (2)**

The specification in Figure 25 denotes that there could be \( n \) (image, video, audio, audiovisual or contentRef) elements, where \( n \) is \( \min = 1 \) and \( \max = \text{unbounded} \), in any order. The reason for using a choice tag instead of the sequence tag in this case is to denote that ordering of elements is not important. This description is not required at all to be implemented in OWL when converting. In such cases (i.e. \( \text{maxOccurs} = \text{unbounded} \)), the choice can be modelled with a single property, for example in this case: image_video_audio_audioVisual_contentRef.

Other syntax restrictions in XML are for example modelling required attributes; this can be easily implemented in OWL using cardinality exactly 1.

### 4.5.3.4 Modelling Unnecessary XML Elements

Figure 26 shows MPEG-21 elements that should be ignored during the manual conversion process. Such XML elements are ignored because their aim is to define the XML document structure rather than define semantics between the XML elements.

Figure 27 and Figure 28 show the ontology structure of terminal codec capability specification from MPEG-21 DIA resulting from automatic and manual conversions, respectively.

In order to evaluate the difference between the manually and automatically constructed ontologies, the number of concepts involved in the definition of the following statements can be compared:

- User with video codec capability enabled device
- User with video codec capability enabled device but only with image decoding capability device.
The following can be observed from the class diagrams depicted by Figure 27 and Figure 28:

- Classes such as UsageEnvironmentPropertyBaseType, DIADescriptionType, DIABaseType, CodecCapabilityBaseType, TerminalCapabilityBaseType are present purely for syntax purposes and have no reasoning significance, they certainly make the XML document easily extensible and structured, however, they are not useful in terms of semantics (semantic relationships between the terms being modelled).

- The second hierarchy is more descriptive and yet simpler in terms of the relationships between entities such as user and device.

Modelling a user who has a video codec capability enabled device with the automatically generated ontology requires 7 ontology instances; on the other hand, the same situation can be modelled using the manually generated ontology with only 3 instances. Instance reasoning is more complex and time consuming than concept reasoning [152]; the fewer instances present in an instance ontology the more efficient the reasoning process is.
Figure 27: Modelling device capabilities with automatic conversion

Figure 28: Modelling device capabilities with manual conversion
4.5.3.5 Considering a Flatter Structure

To minimize the number of instances and hence improve reasoning, the following could be considered. For example, instead of defining a class `Frame` (having properties `height` and `width`) instances of which are assigned to a `Video` instance via the `has-frame` property, properties `frame-height` and `frame-width` could be assigned directly to a `Video` instance.

The following example is given to illustrate the design decisions based on the application and requirements of the ontology, for example, the reasoning efficiency requirement for ADIO and the structure and clear hierarchy requirement for ADO.

The user audio preference would be modelled as follows with automatic conversion:

\[
\text{(A) User } \equiv \exists \text{ userCharacteristic userCharacteristicBaseType}^* \\
\quad \text{AudioPresentationPreference } \equiv \text{ userCharacteristicBaseType} \\
\quad \text{AudioPresentationPreference } \equiv \exists \text{ volumeControl zero2one}
\]

**Definition 1: Modelling user audio preference automatically (option (A))**

On the other hand, manual conversion could model the preference in two different ways as follows:

\[
\text{(B) User } \equiv \exists \text{ hasAudioPresentationPreference AudioPresentationPreference} \\
\quad \text{AudioPresentationPreference } \equiv \exists \text{ volumeControlPreference zero2one}
\]

\[
\text{(C) User } \equiv \exists \text{ volumeControlPreference zero2one}
\]

**Definition 2: Modelling user audio preference manually (options (B and C))**

Options (B) and (C) could be adopted for different reasons. Option (B) is suited for scenarios where many users have the same preference; a number of instances can be created and some instances may be used for more than one user. However, before adding such a preference, the system needs to check if the preference already exists. Moreover, if user profiles are not cached, this option is not relevant. Hence, Option (C) is chosen if user profiles are not retained and because the ontology might be used for reasoning (in the case of ADIO) to serve real time requests. Moreover, the number of combinations of audio preferences increases with the number
of audio preferences considered and hence the number of instances representing audio preference gets larger.

Another example is the audible frequency range depicted by Figure 29. Definition 3 and Definition 4 show the result of converting audible frequency range concept manually and automatically, respectively.

```xml
<element name="AudibleFrequencyRange" minOccurs="0">  
  <complexType>  
    <sequence>  
      <element name="StartFrequency" type="float"/>  
      <element name="EndFrequency" type="float"/>  
    </sequence>  
  </complexType>  
</element>
```

**Figure 29: MPEG-21 audible frequency range**

**Definition 3: MPEG-21 audible frequency range modelling with manual conversion**

User $\equiv \exists$ audibleStartFrequencyPref. float $\land \exists$ audibleEndFrequencyPref. float

AudibleFrequencyRange $\equiv \exists$ startFrequency. float $\land \exists$ endFrequency. float

**Definition 4: Audible frequency range preference modelling with automatic conversion**

However, it should be noted that this approach can only be used if the cardinality is 1, if the cardinality is greater than 1, this approach leads to errors. The following example is given to illustrate this:

ClassY $\equiv \exists$ SomePropertyX. ClassX

ClassX $\equiv \exists$ SomePropertyC. ClassC $\land \exists$ SomePropertyV. ClassV

If the class ClassX is eliminated and two direct properties are used instead as follows:

ClassY $\equiv \exists$ SomePropertyXC. ClassC $\land \exists$ SomePropertyXV. ClassV
Then the instances of ClassC and ClassV can no longer be linked together if the cardinality of SomePropertyX is greater than 1. In such a situation, using the class ClassX and the property SomePropertyX is necessary.

4.5.3.6 Incorrect Modelling with Automatic Conversion

An example from MPEG-21 (Figure 30) is given to illustrate where automatic conversion results in incorrect modelling:

```xml
<complexType name="codecCapabilityBaseType" abstract="true">
   <complexContent>
      <extension base="dia:DIABaseType">
         <sequence minoccurs="0" maxOccurs="unbounded">
            <element name="Format" type="mpeg7:ControlledTermUseType"/>
            <element name="CodecParameter" type="dia:CodecParameterBaseType" minoccurs="0" maxOccurs="unbounded"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 30: MPEG-21 codec capability

Modelling the XML specification in Figure 30 with automatic conversion will not only give an inefficient OWL structure, but will result in incorrect modelling. This schema suggests the following format is possible:

```xml
<Format> Format-1 </Format>
<CodecParameter>Para-1 </CodecParameter>
<CodecParameter>Para-2 </CodecParameter>
<CodecParameter>Para-3 </CodecParameter>
<Format> Format-2 </Format>
<CodecParameter>Para-4 </CodecParameter>
<CodecParameter>Para-5 </CodecParameter>
<CodecParameter>Para-6 </CodecParameter>
```

The XML processor is able to associate Para-1, Para-2 and Para-3 with Format-1 as they directly follow in the XML document. However after loading an OWL file, this sequence is lost (if the same structure is modelled using OWL). The semantics on this element entails that a
structure (possibly named Format) which is linked to a format and several parameters must be created. Manual conversion has to carefully check for situations like this. What causes this situation is the max occurrence unbounded with the sequence element.

4.5.3.7 Validating the Ontology

In order to validate the ontology resulting from automatic conversion from XML schema to OWL, the following can be performed:

1. Populate the ontology with data and check that the original XML data is represented adequately with the OWL model.
2. Revise the ontology against the semantics of the elements specified in the XML document. As indicated by Section 4.5.3.3, some restrictions are for syntax validation and do not influence the semantics of the elements.
3. Use an ontology reasoner to check if there are any inconsistencies among the classes in the ontology.

The author manually converted MPEG-21 DIA UED and high level MPEG-7 MDS elements. The resultant ontology has been validated using these steps.

4.6 Conclusions

This chapter has presented a context management framework implemented by the ACM, part of the AM presented in Chapter 3. The aim of the proposed framework is to manage the complexity of adaptation domain context and enhance system interoperability and extensibility.

The author’s approach to addressing key context management requirements for content and service adaptation systems has been presented. The identified requirements are interoperability, extensibility and completeness. These requirements are very important to solve the heterogeneity, dynamicity and hence the complexity problem of the of the adaptation domain. These problems which affect the context management subsystem impact the whole adaptation system. Extensibility is identified as the most important requirement and is key to both interoperability and completeness.

The author’s approach has been to define a context management framework which consists of two parts, the context assimilation interface and the high level context interface. The context assimilation interface abstracts the heterogeneity of the context profiles from the rest of the system by formatting and assimilating context into the internal context representation model, the ADIO. The high level context interface defines high level context and absorbs changes in context
parameters or adaptation scenarios from the rest of the system and hence enhances extensibility and maintainability.

The internal context representation model is the ADIO. It is implemented in OWL 2.0 which provides more expressiveness including data types reasoning to define more high-level context concepts compared to using previous ontology languages such as OWL 1.0.

ADIO has been presented as part of the solution for solving extensibility for content and service adaptation. Using latest OWL expressive extensions, ADIO is able to define more high-level context concepts. However, the manner in which these new OWL expressive extensions are used, such as in defining high-level context concepts, can hinder extensibility and could lead to maintenance problems. Hence, the author has proposed a pre-reasoning parameters calculation step. This step ensures that ADIO high level concepts definition is extensible and requires minimal maintenance as demonstrated in Chapter 7.

ADIO should be distinguished from an ontology which aims to annotate adaptation requests; such ontology could be referred to as the ADO. A comparison has been made between ADIO and ADO. For example, a main requirement for ADIO is reasoning performance. It is important that the structure of ADIO is as efficient as possible, for example via limiting the use of instances as instances reasoning is more complex than concepts reasoning. This is achieved by aiming for a flatter ontology structure as limiting the number of concepts and object properties limits the number of instances that need to be used.

ADIO and ADO would be constructed using vocabulary of existing multimedia standards in order to enhance interoperability. Most of these standards are XML based; the construction of an ontology based on an XML based standard has been discussed. Different design decisions would need to be taken for ADIO and ADO. It has also been demonstrated, with examples, that automatic conversion from XML to OWL lead to several problems and that manual conversion is necessary.
5 Adaptation Decision Taking

5.1 Introduction

Content and service adaptation is a complex problem. Adaptation decisions are derived from context i.e. descriptions of entities such as users, usage environments, devices, content and networks. New content, device and network technologies emerge rapidly; thus, ADT needs to be extensible to new adaptation scenarios.

Section 5.2 discusses the most important content and service adaptation decision requirements and Section 5.3 presents the author's approach to satisfying them to achieve effective and extensible ADT.

As described in Chapter 2, there are several standards to describe context. The Semantic Web envisages a world, where information is described using a uniform system that provides both formality and universality in terms of syntax and semantics. OWL, ontology, DL and XML are among the technologies being explored for this vision. Moreover, the multimedia and context communities are moving towards Semantic Web multimedia content annotation and semantic based context modelling. Hence, the compatibility of a content and service adaptation system with key Semantic Web technologies is crucial. A core technology for the Semantic Web is DL, Section 5.4 presents a novel DL based rule based reasoning that is implemented with Semantic Web technologies. Sections 5.4.2 to 5.4.4 describe how rules are defined in this approach and Section 5.4.5 presents how the defined rules can be run with a DL reasoner. The motivations behind this approach are presented in Section 5.4.6. DL reasoning is complex, Section 5.4.6 presents measures that can be taken to improve the performance of the proposed reasoning approach. Finally Section 5.5 concludes the chapter.

5.2 ADT Requirements

Content and service ADT is a complex problem due to the heterogeneity of the adaptation context, the dynamicity of the adaptation domain and the variability of user preferences. The author proposes the context management framework, presented in Chapter 4, to assimilate context into a homogenous model in order to solve the context heterogeneity problem. With regard to ADT, the following requirements can be identified:
Chapter 5. Adaptation Decision Taking

• Extensibility
Extensibility is necessary in order to be able to accommodate for new adaptation scenarios, especially that delivery environment capabilities may change continuously with the rapid technological developments. Moreover, new content technologies are introduced and new user preferences emerge.

• Effectiveness and accuracy
The surveyed adaptation decision approaches in state of the art adaptation systems target different aspects of ADT. For example, the MPEG-21 DIA-ADTE is effective in selecting adapted content parameters. Accuracy is ensured using context provided by the content provider (in the form of AdaptationQoS profiles) detailing which content parameters to select in the case of different adaptation operations and different usage environment constraints. Accuracy and effectiveness is required for all aspects of ADT including deciding on layout and structure, adaptation operations and deciding adaptations for user characteristics such as colour vision deficiency. To ensure accuracy and effectiveness, different ADT approaches that target different ADT aspects have to be integrated.

• Completeness
A combination of more than one decision technology is important not only to ensure accuracy of decision taking, but also to cover all aspects of ADT as accurately as possible.

5.3 Proposed Approach: Two Stages of ADT
The author proposes an ADT framework where several ADT technologies can be plugged in and used. The diagram in Figure 31 depicts two main stages of ADT in such a framework.

As discussed in Chapter 4, the interface between the decision taking and adaptation context management is the Adaptation Decision Ontology Interface (ADIO). This interface provides context in a high level refined format. Providing context in a refined state entails that low level adaptation decisions such as deducing that a device is battery limited with regard to a service are performed by the context management framework and thus reducing the size and complexity of ADT rules.
There are two main layers of decision taking in the proposed framework. The first layer of decision taking is responsible for taking decisions such as:

- Which content items to remove, emphasise or de-emphasise.
- The resources allocated for each content item.
- The necessary adaptation operations to be applied to each content item.

The second layer of decision taking aims at accurately selecting adapted content parameters values. Figure 31 shows the two stages of ADT.

The AM Adaptation Decision Engine (AM-ADE) does not implement a decision technology but rather aims to manage several decision taking entities, including the rule-based decision engine and other specialised decision taking engines such as a presentation adaptation algorithm or a HTML form resizing decision taking engine.
Specialised ADTEs can be used for a variety of reasons, including calculating restrictions on content items, for example deciding on colour characteristics of video or image content based on user vision disabilities [88]. Novel decision taking techniques that might be specific to a single adaptation parameter such as web page form sizes [32] can be plugged in and used. This is to provide an accurate and complete ADT.

At the core of the second layer decision taking is MPEG-21 DIA-ADTE. Figure 32 presents how the two layered approach is defined with the MPEG-21 DIA-ADTE framework.

DIA-ADTE is used for a number of reasons:

- The selection of adaptation parameters is based on AdaptationQoS metadata provided by the service or content provider. This provides a more accurate decision taking with regard to adapted content parameter settings.
- The DIA-ADTE is generic and is not specific to certain content types.

The MPEG-21 standard is an important standard for multimedia and is based on the widely recognised standard; MPEG-7. Once MPEG-21 DIA is widely adopted, interoperability with other
MPEG-21 DIA based systems is ensured. Moreover, more content providers would provide AdaptationQoS which ensures accurate ADT.

The proposed ADT framework addresses the requirements specified in Section 5.2 as follows:

- **Effectiveness and Accuracy**

A layered approach that uses different decision taking technologies ensures effectiveness and accuracy as adaptation decision is provided for different adaptation aspects. For example, a layout decision engine decides on the structuring and layout of content items [30, 84], a fuzzy logic decision engine is used to decide on modelling user preferences with regard to form sizes [32], the MPEG-21 DIA is used to calculate parameters of adapted content versions [86], a specialised ADT is used to adapt colours for vision deficient users [88], etc. the ADT Engines are managed by the AM-ADE which uses a rule based system to perform first layer ADT and to manage the other ADTEs.

- **Extensibility**

Adopting extensible ADT technologies is necessary to ensure system extensibility. AM-ADE uses a rule based system in order to ensure the extensibility of the adaptation decision rules to new adaptation scenarios. A modular design is also essential, interfaces between the AM-ADE and the other ADTEs need to be clearly defined.

- **Completeness**

Extending the ADT functionality with specialised ADTE ensures completeness of adaptation. Therefore, extensibility is important for completeness as well as accuracy and effectiveness of adaptation.

5.3.1 **ADT Management Algorithm**

This section explains how the AM-ADE manages the ADT process. Figure 34 depicts the algorithm which can be specified in VDM as depicted by Figure 33.
Chapter 5. Adaptation Decision Taking

Figure 33: ADT management algorithm

Next sections explain the algorithm specified in Figure 33, relevant stages depicted by Figure 34 are mentioned.

Line 1 of Figure 33 starts the definition of the function \( \text{takeDecisions(request)} \), this function is called after context assimilation and refinement is completed to calculate the necessary Decisions on the content or service in order to adapt it to the user context. The main properties of a Decision object are:

\[
\begin{array}{c|c}
\text{Decision} & \text{OP} \\
\hline
\text{inParameters} & \text{Parameter-Set} \\
\text{outParameters} & \text{Parameter-Set} \\
\text{constraints} & \text{Constraint-Set} \\
\text{alternativeOP} & \text{AOperationList} \\
\end{array}
\]

The OP field represents the adaptation operation that needs to be executed in order to realise the decision. An adaptation operation can be reducing temporal resolution, removing content, transcoding to a different format or converting to a different modality, etc.

The \( \text{inParameters} \) field describes the properties of the content on which the decision needs to be applied; this includes the format, bit-rate, size, resolution, etc.
The outParameters field represents the parameters of the decision or the adapted content, i.e. the parameters under which the adaptation operation (specified by OPid) is applied to produce the adapted content. For example, if the OPid represents an image resolution reduction operation, the output parameters represent the new horizontal and vertical resolutions of the adapted content.

The constraints field is calculated from the context. The output parameters are calculated based on the input parameters and the constraints. The difference between a constraint and parameter is that the constraint represents a range, for example resolution < x while output parameters represent exact values.
A constraint can be defined as follows:

| Constraint :: constrainedParameter Parameter |
|-------------|------------------|
|             typeOfConst               |
|             TypeOfConstraint ∈ TypeOfConstraintsList |

TypeOfConstraintsList is defined as follows:

\{ ≥, ≤, >, <, =, ≠ \}

The alternativeOP field of the Decision type is used if more than one adaptation operation needs to be composed in order to achieve the adaptation. If this field is not null, then the OP field will be null.

The `takeDecisions(request)` function can be defined in terms of post and pre conditions as follows:

\[
\text{takeDecisions (request : ARequest)}
\]

\[
\text{Pr contextRefined (request)}
\]

\[
\text{Post decisionsSupported(request)}
\]

The pre condition specifies that the context must be assimilated and refined before this function is invoked; this condition is described in Chapter 4.

The post condition (\(\text{decisionsSupported (request)}\)) means that all decisions have a corresponding adaptation operation that is active and supports the decision parameters. This condition can be defined formally as:

\[
\forall \text{requ ∈ requList. decisionsSupported (requ) } \iff \forall \text{dec ∈ decl(requ) status (OP(dec)) = active } \land \text{parametersSupported (dec)}
\]

The predicate \(\text{parametersSupported (dec)}\) is formally defined as:

\[
\forall \text{dec ∈ declList parametersSupported(dec) } \iff \text{inParameters(dec) ⊆ inParameters (OP(dec)) } \land \text{outParameters(dec) ⊆ outParameters (OP(dec))}
\]
Line 2 of Figure 33 (calcServiceRequirements (request)) (step 1 in Figure 34) calculates service requirements by analysing content and service metadata. Service requirements are the required bandwidth, the required memory, required processing power, required battery life, etc. There is dependency between some of these requirements, for example, the required battery power depends on required processing power and bandwidth (data transmitted to the device).

If a service delivers more than one content item at a point in time, than the service requirements are the combination of content requirements delivered at a that point of time.

Line 3 of Figure 33 (CalcDeliveryEnvConstraints (request)) (step 2 in Figure 34) calculates the constraints on the service imposed by delivery environment capabilities and user preferences, such as the maximum bandwidth supported, the maximum battery life available, maximum resolution, etc. The constraints are calculated by comparing service requirements, deduced from service descriptions, and the delivery environment capabilities deduced from the user device and connectivity descriptions.

The calculated constraints and the calculated service requirements in line 2 of the algorithm (step 1 in Figure 34) are used to decide upon adaptation parameters in line 7 of the algorithm 2 (step 7 in Figure 34).

Line 4 of Figure 33 (decideAdaptationOperations (request)) (step 3 in Figure 34) defines the decisions required on the service without deciding on the exact adaptation parameters; for example to decide that resolution needs to be reduced below a value $X$ without deciding on the exact value of the resolution. The layout and the structure of the service are also decided in this step. The process of deciding layout and structure requires metadata about the layout and structure of service components including the importance of each component to the service, or an extraction tool to extract such metadata. This is outside the scope of this thesis, related work includes [60].

Decisions taken at this step are represented by the decision object defined earlier in VDM. Such decisions can refer to:

- Removing a content item, which corresponds to a remove adaptation operation
- Degrading/enhancing a content item; this can correspond to several adaptation operations depending to the degradation or enhancement to be applied, for example in terms of temporal resolution, special resolution, etc.
- Changing of modality, this corresponds to several possible adaptation operations, for example video-to-audio or audio-to-text, etc.
Chapter 5. Adaptation Decision Taking

The decisions mentioned above are taken to meet the service requirements (line 2 of Figure 33, step 1 in Figure 34). This takes into account delivery environment constraints (line 3 of Figure 33, step 2 in Figure 34) which may include the situation of the user. For example, the user driving means that the delivery environment is constrained in terms of receiving video content.

Line 4 of Figure 33 can be formally defined in terms of post and pre conditions as follows:

\[
\text{decideAdaptationOperations} \quad \text{(request : ARequest)}
\]

\[\text{Pre} \quad \text{requirements,} \text{constraints} \text{calculated (request)}\]

\[\text{Post} \quad \forall \; \text{dec} \in \text{decL(request)} \quad \text{constraintsSupported} (\text{dec})\]

The post condition indicates that the constraints on the adapted content should be supported by the adaptation operations, specifically, the operation output parameters. Moreover, the chosen adaptation operations need to be active. This can be defined formally as:

\[\forall \; \text{dec} \in \text{decL(request)} \quad \text{constraintsSupported} (\text{dec}) \iff \text{status} (\text{OP(dec)}) = \text{active} \land \text{constraints} (\text{dec}) \subseteq \text{outParameters} (\text{OP(dec)})\]

Lines 5 and 6 of Figure 33 are the checks carried out by the \text{decideAdaptationOperations} function, to ensure that all adaptation operations are available and support the constraints defined in line 3 of the algorithm (depicted by step 2 in Figure 34).

The condition \text{someOPNotSupported (request)} (step 6 in Figure 34) can be defined as:

\[\forall \; \text{requ} \in \text{requList. someOPNotSupported (requ)} \iff \exists \; \text{dec} \in \text{decL(requ)} \quad \text{status} (\text{OP(dec)}) = \text{inactive} \lor \neg \text{constraintsSupported} (\text{dec})\]

Line 6 of the algorithm (\text{calcAlternativeToUnavailableOperations (request)}) calculates alternative adaptation operations for unavailable adaptation operations. Two situations are possible:

a. Calculate an alternative operation (step 8 in Figure 34) to perform the same adaptation by composing two or more adaptation operations.

b. Change the modality (steps 12 and 13 in Figure 34), for example, if a video transcoding operation is not available to adapt to a different format, then a change to image or audio modality is decided depending on user preferences on modality conversion priorities.

In option (a), a number of adaptation operations are joined (\(\otimes\)) to achieve the same adaptation of the unavailable adaptation operation, this is defined as follows:
Chapter 5. Adaptation Decision Taking

\[ \forall \text{op} \in \text{OPList}. \exists \{\text{op1}, \text{op2} ... \text{opi}\} \subseteq \text{activeOPList}, \ \text{alternativeActiveOP} (\text{op}, \text{altOP}) \iff \]
\[ \exists \ \text{altOP} \in \text{OPList}. \ \text{altOP} = (\text{op1} \Join \text{op2} \Join ... \Join \text{opi}) \land \ \text{ID} (\text{op}) = \text{ID} (\text{altOP}) \land \]
\[ \text{opName} (\text{op}) = \text{opName} (\text{altOP}) \]
\[ (i \in \mathbb{N} \land i > 1) \]

The ID field is used because it is composed from input and output modality, format and language.

The index (i) is at least 1 because each adaptation operation is unique and composition would involve at least 2 operations. A maximum to the index (i) should be considered for ensuring acceptable performance in terms of both finding the sequence and executing the adaptation using the composition of the operations in the sequence.

The input and output parameters of the adaptation operation and its alternative needs to satisfy the following condition:

\[ \forall \text{requ} \in \text{requList}. \forall \text{altOP} \in \text{OPList}. \forall \text{dec} \in \text{decL} (\text{requ}) \]
\[ \text{alternativeOPSuitable} (\text{altOP}, \text{dec}) \iff \text{alternativeActiveOP} (\text{dec}(\text{op}), \text{altOP}) \land \]
\[ \text{inParameters} (\text{dec}(\text{op})) \subseteq \text{inParameters} (\text{altOP}) \land \]
\[ \text{outParameters} (\text{dec}(\text{op})) \subseteq \text{outParameters} (\text{altOP}) \]

Once all defined adaptation operations are active and supports the decisions (steps 5 or 9 in Figure 34), line 7 in the algorithm (step 7 in Figure 34) \((\text{calculateAdaptationParameters(request)})\) uses the MPEG-21 DIA decision taking approach to calculate adaptation parameters. This approach is used because it is based on metadata about the relationships between the constraints on the service and suitable adaptation parameters as described in Section 5.3.

The function pre and post conditions are defined as follows:

\[
\text{calculateAdaptationParameters} \ (\text{request} : \text{ARequest})
\]
\[
\begin{align*}
\text{Pr} & \ \forall \text{dec} \in \text{decL(request)} \ \text{constraintsSupported(dec)} \\
\text{Post} & \ \forall \text{dec} \in \text{decL(request)} \ \text{parametersSupported(dec)}
\end{align*}
\]

The pre condition, \(\text{constraintsSupported(dec)}\), is defined in this section when explaining line 4 of the algorithm and the post condition, \(\text{parametersSupported(dec)}\), is defined in this section when explaining line 1 of the algorithm.
Once decisions (adaptation operations) have been successfully calculated including the corresponding adaptation parameters (line 7, step 7 in Figure 34) for all content items (steps 10 and 14 in Figure 34), the Adaptation Decision Engine generates adaptation instructions (step 15 in Figure 34) and sends them to the entity responsible for execution. If decisions still need to be defined on some content items (step 11 in Figure 34), the algorithm goes back to defining adaptation decisions for this content as explained in this section (step 3 in Figure 34).

Adaptation instructions are composed of decision objects, represented by the Decision class defined in this section.

5.4 DL Based Rule-Based ADT

5.4.1 Introduction

With the proliferation of the Semantic Web, another form of reasoning, DL (ontology) reasoning, has emerged. DL reasoning employs a DL language and a DL reasoner. DL is a family of knowledge representation languages based on the first order predicate logic. A DL language specifies a domain in terms of concepts (or classes) and roles (or relationships) among the concepts. Hence DL languages can be used as formalisms to implement ontologies. A DL specification is an ontology specification, and thus DL and ontology can be used interchangeably in this thesis. A number of XML DL based ontology languages have been developed. The most widely used ontology language is the Ontology Web Language (OWL). OWL 2.0 is the latest and most expressive version of OWL.

DL reasoners operate differently from rule engines. The “traditional” use of OWL is to use DL (ontologies) to classify types and define their instances.

This section proposes a novel approach that uses DL outside its traditional use. The approach provides a method to specify rules using a DL (ontology) language, and run the rules using a DL reasoner. The defined rules are referred to as “the rules ontology”. The rules ontology fulfils functionalities provided by a Logic Program (e.g. Datalog rules) and not by the traditional use of DL. The advantages of this approach are highlighted in Section 5.4.6.

This approach can be used in scenarios where reasoning revolves around a central entity. In this thesis the scenario is content adaptation and the central entity is the user. However, this approach can be extended to other scenarios where there is no single entity, for example reasoning on relationships between family members. This requires implementing support to manage the input variables to decision taking, in order to maintain the rules that apply to each variable. This is not necessary for the adaptation scenario, because all the rules relate to the central entity i.e. the user.
5.4.2 Defining Rules in the Rules Ontology

A rule is of the form \((\text{RuleBody} \rightarrow \text{RuleHead})\). The proposed approach specifies rules using the following DL constructs: class sub-typing \((\subseteq)\), class equivalence \((\equiv)\), intersection \((\cap)\) and union \((\cup)\). A RuleBody is a set of premises and a RuleHead is a set of conclusions. Rules are modelled as depicted by Figure 35. Firstly, three classes are created, the Rule, the RuleBody and the RuleHead classes. These classes are subclasses of the top concept \(\bot\), to which all concepts are either a direct or an indirect subclass.

\[
\begin{align*}
\text{Rule} & \subseteq \bot \\
\text{RuleBody} & \subseteq \bot \\
\text{RuleHead} & \subseteq \bot
\end{align*}
\]

To add a rule, a class that represents it \((\text{Rule}_n)\) is created as a direct subclass of the Rule class. The \(n\) in \(\text{Rule}_n\) is an identifier that is unique to each rule.

\[
\text{Rule}_n \subseteq \text{Rule}
\]

Each rule (for example Rule_1, Rule_2, etc) has a rule body and a rule head represented by a Body_\(n\) and a Head_\(n\) classes respectively. Body_\(n\) and Head_\(n\) are subclasses of the classes RuleBody and RuleHead respectively. Body_\(n\) is asserted equivalent to its respective Rule_\(n\), so that if the premises in the Body_\(n\) hold, the rule to invoke is known (by extracting the equivalent class of Body_\(n\)). Head_\(n\) is asserted as a super type of its respective Rule_\(n\) and thus it is a super type of its respective Body_\(n\) (because Body_\(n\) is equivalent to Rule_\(n\)).

\[
\begin{align*}
\text{Body}_n & \subseteq \text{RuleBody} \\
\text{Rule}_n & = \text{Body}_n \\
\text{Head}_n & \subseteq \text{RuleHead} \\
\text{Rule}_n & \subseteq \text{Head}_n
\end{align*}
\]

The relationships between Rule_\(n\), Body_\(n\) and Head_\(n\) are defined in such a way because of the following: Sub-typing is defined as: \(A \subseteq B \iff \forall x \ x \in A \Rightarrow x \in B\). For \(x\) to be part of \(A\) or \(B\) it has to match all \(A\) and \(B\) element characteristics. The sub-typing (or super-typing) implication resembles that of rules i.e. if a set of premises (or conditions) described in the rule body hold
(similar to $x \in A$), then it is implied ($\Rightarrow$) that the set of conclusions defined in the rule head also hold (similar to $x \in B$).

Each body and head has a unique identifier $n$ to distinguish them from other rule bodies and heads respectively. For example, Head_1, Head_2, Body_1 and Body_2.

A rule Body is a set of premises and a rule Head is a set of conclusions

$$\begin{align*}
\text{Body}_n &= \{\text{premise}_1 \land \lor \text{premise}_2\} \ast \land \lor \{\text{premise}_3 \land \lor \text{premise}_n\} \\
\text{Head}_n &= \{\text{conclusion}_1 \land \text{conclusion}_2 \land \ldots \land \text{conclusion}_n\}
\end{align*}$$

Each conclusion or premise is either a named class or an unnamed class

$$\begin{align*}
\text{premise}_n &= \text{namedClass} \mid \text{anonymousClass} \\
\text{conclusion}_n &= \text{namedClass} \mid \text{anonymousClass}
\end{align*}$$

A named class is a class definition that has been assigned a name, for example

$$\text{battery\_limited} = \exists \text{battery\_limitation} < x$$
∃ is the existential quantifier. In this case, the class definition ∃ battery_limitation < x is given the name battery_limited and is defined as: there exists (∃) a battery_limitation data property relationship that has a range of values smaller than some value x. If the definition is not given the name battery_limited, then it remains as an anonymous or unnamed class. It is impractical and unnecessary to name all classes as the set of possible premises and conclusions (representing conditions in the user context and adaptation decisions respectively) is very large. Premises or conclusions are named when the definition is lengthy, complex and likely to be re-used. For example:

```
remove_video_condition = (∃user_requires(Summary) ∩ ∃preferred_modality_1(image)) ∪
                        (∃device_limitation(video_incapable)) ∪
                        (∃battery_limitation < 0.2 ∩ ∃preferred_modality_1(video))
```

The named class remove_video_condition represents three situations with the same outcome: a decision to remove video content. Naming this class saves the effort of having to type its definition each time it is included in a rule body. The same usage of named and un-named classes applies to conclusions.

5.4.3 Representing Context as Premises in Rule Bodies

User preferences are specified with class definitions of the form

```
∃ user_preters (some_preference)
```

Which means: there exists a user_preters object property relationship that has the value some_preference, where some_preference is an instance (or individual) in the ontology. User preferences can be defined as instances in the ontology. A premise that describes a user preference is formed by asserting the respective preference with the object property user_preters. If user_preters is modelled as a data type property, the user preference some_preference is modelled as a string data type.

A similar approach can be used to specify device, network and environment context. For example:

```
∃ has_device_property (some_property)
```
Chapter 5. Adaptation Decision Taking

Where: has _device _property is an object relationship and some_property is an instance in the ontology.

In some cases, the priority or ordering of some preference needs to be specified. For example, the priority of preferred modalities which can be specified as follows:

\[ \exists \text{Prefers_modality}_1(\text{video}) \cap \exists \text{Prefers_modality}_2(\text{audio}) \cap \exists \text{Prefers_modality}_3(\text{text}) \]

Prefers_modality_n is an object property and n is the order of preference or priority \((0 \leq n \leq 4)\). Video, audio, text and image are instances in the ontology. The above class definition means that the user’s first preferred modality is video, then audio and then text. Naming properties depends on the scenario being modelled. It is also noted that a data property can be used instead of an object property to define premises. For example, Prefers_modality_n could be modelled as a Datatype property and video, audio and text as string values.

5.4.3.1 Data Types Reasoning

Data types reasoning is required for user preferences or other context elements that involve data types comparison. For example, the depth range and maximum delay frame preferences of the 2D to 3D Stereoscopic Conversion preference are specified as follows:

\[ \exists \text{2D}_3D\text{-stereoscopic_Depth_range} \geq |x| = x \ (0 \leq x \leq 1) \]
\[ \exists \text{2D}_3D\text{-stereoscopic_Max_delay_frame} \geq |x| = x \ (x \geq 0) \]

Where 2D_3D_stereoscopic_Depth_range and 2D_3D_stereoscopic_Max_delay_frame are data type properties. Both class definitions shown above are anonymous (unnamed) classes.

In some cases, user preferences or other context elements have dynamic values and two or more parameters may need to be compared. For example, the available battery resource \((a)\), the preferred remaining resource by the user \((u)\) and the required resource \((r)\) by the service. Such cases are modelled by creating a data property (e.g. battery_limitation) and adding the following class definition to rule bodies that require it:

\[ \exists \text{Battery}\_\text{limitation} \geq |x| = x \]

Where \(x = a - r / u\). For example, \(x (a-r / u) \leq 1\) entails adapting by reducing \((a - r - u / r) \times 100\) % of the service battery resource requirement. Such predefined functions (e.g. a-r / u) are used to
avoid changing the ontology for each request and hence avoid reclassification. This eliminates a considerable overhead.

5.4.3.2 Dealing with DL Mono-tonicity

DL languages are monotonic, which means that addition of assertions would never invalidate any previously inferred knowledge. Monotonic logic requires the Open World Assumption (OWA) [153]. OWA, as opposed to Negation as Failure (NF), means that the absence of some information does not negate it. This goes against the requirements of the adaptation scenario as well as several other scenarios. For example, if mp4 is not present in the supported formats list of a device profile, it is required to deduce that mp4 is not supported. To get around this impediment, the construct of object property negation, recently implemented in OWL 1.1, is used. For example, if a device does not have mp4 in its supported formats list, the following is asserted:

\[-(\exists \text{supports format}(mp4))\]

Where \(-\) denotes the negation symbol, \text{supports format} is an object property and mp4 is an instance in the ontology. The class \text{mp4-limited} can then be described as:

\[\text{mp4\_limited} \equiv \neg \{\exists \text{supports format}(mp4)\} \cap \{\exists \text{required format}(mp4)\}\]

5.4.4 Representing Adaptation Decisions as Conclusions

In the content and service adaptation scenario, conclusions (that constitute rule heads) represent adaptation decisions that need to be applied on the content or service. The decisions are specified as follows:

\[\exists \text{decision}_n \text{(someDecision)}\]

Where \text{decision}_n is an object property or a data type property and \text{someDecision} is an instance in the ontology or a string. The identifier \(n\) indicates the order in which the decision should be applied. In some cases several decisions can be applied to satisfy a certain decision aim. For example, to satisfy the decision aim of reducing the memory requirements of a service, the decisions which could be applied are; reducing temporal or spatial resolution, removing certain content, etc. Decision Aims are modelled as follows:
∃decision_aim_n.someDecisionAim

Where \( \text{decision_aim}_n \) is an object property, \( n \) is the order in which the decision aim is to be satisfied and \( \text{someDecisionAim} \) is a class in the rules ontology that has instances representing decisions that satisfy the decision aim (\( \text{someDecisionAim} \)). Such instances might be attributed to more than one \( \text{someDecisionAim} \) class as one decision might satisfy more than one decision aim.

5.4.5 Running the Rules with a DL Reasoner

The algorithm to run the rules ontology is depicted by Figure 36. The implementation of this approach is depicted by Figure 37.

Reasoning with the rules ontology is performed in the following steps:

5.4.5.1 Static Reasoning: Rules Ontology Classification

1. At start-up, the reasoner loads the rules defined in the ontology and classifies them before receiving any requests. Now, the rules are organized into a hierarchy that models explicit as well as implicit sub-typing relationships between them.

5.4.5.2 Dynamic Reasoning: Running the Rules upon Receiving Requests

2. Extract context from profiles.

3. Calculate pre-reasoning parameters using pre-defined functions. These functions are used to compare parameters without having to change the rules ontology, because the result of pre-reasoning functions are compared to uniform values set in the rules ontology. For example, evaluating \( x-y < 0 \) instead of \( x < y \).

4. Build a class representing a rule body (\( \text{tempRuleBody} \))

5. Add the rule body class (\( \text{tempRuleBody} \)) to the reasoner

6. Invoke the reasoner to get the types and super-types of \( \text{tempRuleBody} \) (which would be one or more of the \( \text{Body}_n \) classes).

7. Choose the \( \text{Body}_n \) classes that are direct super types of \( \text{tempRuleBody} \) (the direct super types represent the most specific rules which apply)
Chapter 5. Adaptation Decision Taking

8. Extract the equivalent classes to the chosen Body\_n classes (which would be Rule\_n classes)\(^1\)

9. Extract the super types of the Rule\_n classes extracted from step 8 (which would be Head\_n classes)

10. Extract the equivalent classes to the Head\_n classes extracted in step 9 (which represent adaptation decisions in this scenario).

---

\(^1\) This step (8) could be dropped because body\_n is equivalent to rule\_n. If dropped however, the super types which need to be extracted in step 9 are for the body\_n class rather than the rule\_n class.
Chapter 5. Adaptation Decision Taking

5.4.6 Implementing for Performance

Although several research efforts are focusing on DL optimization techniques for reasoning algorithms and promising results have been shown [154], some measures can be taken to improve performance. These measures are:

- **Reduce size of ontology**
  
The terminology ontology (ADO, Section 4.5.1) is separated from the rules ontology. I.e. the rules ontology does not import or use concepts defined in the terminology ontology. The terminology ontology is the ontology that defines the concepts in the domain. Concepts in this scenario are for example: user, device, preference, content, etc. Concepts that are used should be defined in the rules ontology separately from the terminology ontology. If the terminology ontology is used (i.e. imported), all concepts defined are involved in reasoning and hence the reasoner would be classifying a larger and more complex ontology.

- **Pre-Classification**
  
The reasoning ontology is classified at system start-up.

![Diagram](image.png)

*Figure 37: DL based rule reasoner implementation*
• Avoid Re-Classification

Introduce measures in order to avoid changing the ontology and hence avoid reclassification. For example using predefined functions to compare parameters

• Avoid Instance Reasoning (ABox Reasoning)

All reasoning requests are reduced to class requests because instances reasoning is less effective [152].

5.4.7 Motivation

This novel approach is proposed for the following reasons:

• The context community, including multimedia metadata and context modelling in general, is moving towards the direction of Semantic Web context modelling. Thus, in scenarios such as the adaptation scenario, Semantic Web technologies support is crucial. This approach is Semantic Web compatible.

• The integration of a tool based on this approach in any system that supports the Semantic Web is possible because the necessary APIs and reasoner would be available.

• Off the shelf tools such as OWL development environments and reasoners are available. Several such tools are used by large communities and enjoy good support.

• There is an increased interest in DL reasoning. DL has become an active research field and DL reasoning optimisation and expressiveness extensions are among the priorities of the DL research community.

• It is possible to make use of DL built-in semantics (disjointness for example) and reasoning services (classification and consistency check). The following are a few examples:
  
  o Equivalence: ram = rm

  o Disjointness: video_unable ≠ deliver_video. Disjointness can be used for example to ensure that rules can not contain the first class as a premise and the second as a conclusion at the same time

  o Consistency check: To make sure rules are defined consistently, using disjointness for example

• Structured maintenance, development and extension of rules because of the clear structure of ontologies.
• This approach is syntax independent; it depends on DL theory rather than a specific DL language such as OWL.

5.5 Conclusions

This chapter has dealt with content and service ADT in the heterogeneous and volatile content and service adaptation domain. The nature of the domain entails changing requirements and changing adaptation scenarios. Therefore, flexible and extensible ADT is required.

The first part of this chapter has proposed an ADT framework to ensure the key required features of ADT i.e. extensibility, accuracy, effectiveness and completeness. The chapter did not propose a new decision taking approach, but rather proposed an entity, the AM-ADE, that manages several adaptation decision engines to achieve the mentioned key requirements.

Decision taking is managed in two stages. The first stage defines the adaptations that need to be performed and the restrictions that need to be satisfied with the adaptations. For this purpose a rule based approach has been adopted. Specialised adaptation decision technologies should be used for specific aspects such as adapting visual content for vision colour deficiencies. Rule based decision taking is necessary to ensure extensibility to new adaptation scenarios. The second stage of decision taking is based on MPEG-21 DIA-ADTE. This approach has been chosen because it offers accurate and effective ADT as it is based on metadata provided by content providers that relates adaptation parameters to adaptation restrictions. The novelty of the proposed decision framework includes combining rule based ADT with MPEG-21 DIA-ADTE.

The second part of this chapter has proposed a novel approach for rule based reasoning using DL. Due to the proliferation of the Semantic Web, DL (ontology) reasoning has gained momentum. Based on the research being conducted in this field and the progress so far, the author has expected that ontology reasoning will continue to improve in terms of performance and expressiveness extensions. However, due to the nature of DL languages, some adaptation decision rules cannot be modelled, for example triangle relationships such as the uncle-of relationship. Such relationships can only be modelled using rules as part of a logic program. Thus, Section 5.4 has proposed a DL based approach to sufficiently specify rules that would otherwise need to be specified in a Logic Program such as Prolog or Datalog. This approach can be used in scenarios where reasoning revolves around a central entity. The scenario used in this thesis is content and service adaptation and the central entity is the user. The approach can be applied to other scenarios such as reasoning on customer data or problem diagnosis such as equipment fault detection or patient diagnosis.
Chapter 5. Adaptation Decision Taking

DL reasoning is complex and can be expensive because an update to the ontology requires entire reclassification to maintain consistency. However, later research is investigating DL reasoning optimization techniques. A possible optimization is incremental reasoning, which allows updates to ontology without triggering entire reclassification. Incremental reasoning and other optimization techniques are being investigated and promising results have been shown [154]. However, incremental reasoning is not yet fully supported in DL reasoning. Section 5.4.6 has explained how certain measures are used to avoid changing the ontology and hence avoiding reclassification.

Chapter 6 demonstrates how the architecture of the AM-ADE and the proposed DL-based rule base engine are implemented. Chapter 7 presents an evaluation of the contributions that has been presented in this chapter.
6 Implementation and Proof of Concept Demonstration

6.1 Introduction

This chapter presents the implementation of a prototype as a proof of concept for the contributions presented in Chapters 3, 4 and 5. The prototype implementation constitutes the AM entity, a central component in the content and service adaptation management test bed. This chapter is important to demonstrate the feasibility of the solutions proposed by the author.

Not all aspects related to implementation are mentioned, rather, only those significant to the contributions presented in this thesis are mentioned. Throughout this chapter, parts of the content and service adaptation management test bed components that were not implemented by the author are clearly highlighted.

The chapter starts with presenting the experimental results in Section 6.2. Section 6.2.1 presents the test bed, of which the AM is a central and a significant entity. This section describes the components of the test bed and focuses on the part implemented by the author: the AM, the tools used and the deployment approach are discussed. Section 6.2.2 presents the proof of concept demonstration. The start-up, receiving requests, processing requests and delivering responses stages are discussed and screen shots are used for illustration. Section 6.3 discusses system implementation, starting with a description of implemented AM components in Section 6.3.1. Sections 6.3.2 through to Section 6.3.5 discuss specific implementation issues related to topics addressed in this thesis, mainly issues related to adaptation context assimilation and ADT. Section 6.4 concludes the chapter.

6.2 Experimental Results

This section is divided in three parts. The first part (Section 6.2.1) presents the test bed, of which the implemented prototype constitutes a major part. The second part (Section 6.2.2) presents the scenarios used in the demonstration. The third part (Section 6.2.3) presents the result from the demonstration.
6.2.1 The Content and Service Adaptation Test Bed

This section presents the test bed for content and service adaptation management, part of which is the AM prototype implemented by the author as a proof of concept demonstration of the contributions presented in this thesis.

Figure 38: Content and service adaptation management test bed

Figure 38 depicts the test bed; the description of the components is presented below. The test bed is run on 4 laptop computers, a wireless router and mobile terminals. The following components were implemented by the author:

- **AMF-1 and AMF-2 Servers**

The AMF-1 and AMF-2 servers run an AMF. The AMF consists of an AM and a Content Adaptor (CA) (the Content Adaptor is responsible for calculating the time and cost of adaptation, finding the invocation sequence of adaptation mechanisms and invoking the adaptation mechanism, this entity is outside the scope of this research and is implemented by other researchers working on the content adaptation test bed). The AMF runs on Apache Tomcat server and communicates via web services (deployment is presented in Section 6.2.1.1). The reason for using two AMF servers is to offer two choices for adaptation, for example, at different prices or processing times. At this stage one of the AMFs provides a positive result and the other a negative result with regard to being able to adapt the content.
Chapter 6. Implementation and Proof of Concept Demonstration

The following sections briefly describe the test bed components that were not implemented by the author.

- **Content and ADME Server**

  This machine hosts the adaptation mechanisms and content that is requested by the user. The ADMEs are implemented using the FFMPEG\(^2\) library, which provides several tools for video and audio content transcoding to several formats. The following ADMEs are provided: a MOV video to 3GPP video transcoder, a MOV video to MP3 audio transcoder and a key frame extractor from a MOV video into JPEG images. Both the ADMEs and the content are hosted in Apache Tomcat servers.

- **DA+ PAA**

  This machine represents the Dispatcher Agent (DA) and the Personal Assistant Agent (PAA). The dispatcher agent manages the interface between the AMF and the user environment (through the PAA). It communicates with the AMF through web services.

- **WLAN**

  The computers including the mobile terminal are interconnected via a WLAN, using the wireless router.

- **Mobile Terminal**

  The mobile terminal runs the mobile Personal Assistant Agent (PAA), which communicates with the Dispatcher Agent and PAA agent at the DA+PAA machine. The mobile terminals being used are the NOKIA N80 (connected through WLAN) and the NOKIA N70 (connected via a Bluetooth link).

6.2.1.1 **Deployment**

The Eclipse [155] platform has been used as an integrated development tool for the implementation of the AM. The Eclipse platform has been chosen because it offers several tools to facilitate code development and maintenance such as refactoring tools and auto code generation. The AM prototype can be demonstrated either by running from the integrated apache server in Eclipse or by deploying on a standalone apache server. The second option is preferred as running within Eclipse requires more resources. However, the first option, i.e. running the AM within Eclipse, is ideal for testing.

\(^2\) http://www.ffmpeg.org/
To deploy the AM as a web service, it is first exported as a WAR (Web Application Archive) and placed in the web applications folder of the Apache Tomcat server. The Web Service Description Language (WSDL) [156] file used to describe the AM service is described in Section 6.3.4.

Upon apache server start-up, the WAR file is extracted and is ready to receive requests. However, an instance of the AM web service is not created until a first request is received. Therefore, upon receiving the first request, initialization steps are executed, which might result in a longer response time for the first request. For this reason, if the AM is to be deployed as a real life service, it is preferable to send a dummy request (or a start-up request) to trigger initialization such as loading context formatters, loading reasoning ontology, discovering ADMEs, etc. more details about initialisations at start up are mentioned in Section 6.2.3.1.

6.2.1.2 Tools

The following tools were used for the deployment of the AM:

- Apache Tomcat Web Server

The AM run as a web service, the service is contained in the Apache Tomcat web server. The web service was deployed as a WAR file, which is extracted by the Tomcat server upon Tomcat start-up.

- Apache Axis

Axis is used as the Simple Object Access Protocol (SOAP) [157] engine, to process SOAP messages exchanged between the dispatcher agent and the AM.

6.2.2 Demonstration Scenarios

Three adaptation scenarios are used for the same content. The content is a MOV video file. The three scenarios are described in Table 10.

Extending the demonstration to the three scenarios is used to evaluate the extensibility of the system as explained in Chapter 7.

The three scenarios are represented by three sets of context profiles of the user preferences, usage environment and content descriptions.
Table 10: Demonstration scenarios

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video-Video</td>
<td>This scenario is applicable if the user device does not support the original video content format, or a downgrade in resolution is required to save resources such as battery or bandwidth. The example used here is MOV to 3GPP adaptation.</td>
</tr>
<tr>
<td>Video-Audio</td>
<td>This adaptation is applicable if for example the user's preferred modality is audio, or if the user device is not video capable. The example used is MOV to MP3 adaptation.</td>
</tr>
<tr>
<td>Video-Images</td>
<td>This scenario applied to a number of situations including if the user is interested in screen shots from a video content or if the device is not video capable and the user prefers image modality to audio modality. The example used is MOV to 3GPP adaptation.</td>
</tr>
</tbody>
</table>

6.2.3 AM Proof of Concept Demonstration

This section presents the results from the proof of concept demonstration by running the scenarios described in the previous section. The next sections describe the start-up of the AMF, the reception of adaptation requests, the processing of adaptation requests and the delivery of the adaptation response.

6.2.3.1 System Start-up

At system start-up, the adaptation management system loads the necessary tools and initialises system parameters.

Figure 39 depicts the pop up window to show the main stages of the initialization phase at AM system start-up.

All functionalities apart from ADME discovery are implemented by the author and are related to the solutions presented in this thesis. The following subsections explain each of the initialization steps.

- **Web Service Gateway created**

This step indicates that the entity that manages web services requests has been created. Information on this entity is presented in Section 3.5.2.4.
• **Context Transformers for MPEG-21 and MPEG-7 have been successfully loaded**

The XSLT based MPEG-7 (MDS) and MPEG-21 (DIA) transformers are loaded, these transformers are used to format MPEG-7 (MDS) and MPEG-21 (DIA) profiles into the internal AM context model. Information about transformation tools’ implementation is presented in Section 6.3.2.2.

• **Adaptation rules successfully loaded**

The adaptation rules used by the Adaptation Decision Engine are loaded; more details are presented in Section 6.3.3.1.

• **Adaptation operation descriptions loaded**

The description of adaptation operations that are supported by the system are loaded as presented in 6.3.2.1.

The steps “**ADME discovery engine created**” and “**Discovering ADMEs**” are part of the Content Adaptor functionality (implemented by other researchers working on the test bed). Three ADMEs are implemented for the demonstration and are discovered at system start up.

• **New ADMEs discovered and added successfully**

The AM receives the ADME profiles from the Content Adaptor and updates the list of active and inactive adaptation operations.

![Figure 39: System start-up](image-url)
6.2.3.2 Receiving Requests

The AM can receive three main types of requests:

- **Adaptation inquiry requests**: An adaptation inquiry request is sent in order to inquire about the time and cost of adaptation.
- **Adaptation execution following an adaptation inquiry request**: The user environment may send an adaptation execution request following an adaptation inquiry request if the time and cost specified are suitable for the user environment.
- **Adaptation execution requests**: The user environment may send an adaptation execution request without sending an adaptation inquiry request. This type of request is a combination of the two requests stated above but without sending adaptation time and cost to the user and waiting for an execution request.

The pop up window in Figure 40 was used to show the type of adaptation request received by the AM.

![Figure 40: Types of requests](image)

6.2.3.3 Processing Requests

In order to demonstrate the processes involved in answering an adaptation request, pop up windows are used to indicate the different adaptation stages of an adaptation process. The pop up windows are depicted by Figure 41.

For each inquiry or adapt request, a pop up window is displayed showing the request ID and the different stages of adaptation. The window can be closed at any time. The text displayed on the window is updated according to the progress of adaptation.

The main adaptation stages as shown by Figure 41 are:

- **Adaptation inquiry request received**
- **Adaptation process started (inquiry request)**
• **Starting context acquisition**

This stage includes checking context profile availability, downloading the context profiles, determining type and format, using the appropriate context formatter, applying pre-reasoning functions and refining the context. Some implementation aspects on context acquisition and assimilation are discussed in Section 6.3.2.

• **Starting ADT**

This includes feeding the refined context into the adaptation rules, running the rule engine, extracting adaptation decisions, calculating adaptation parameters, etc. Implementation aspects of decision taking are discussed in Section.

![Figure 41: Demonstrating adaptation stages](image)

- **Calculating adaptation time and cost**: The Content Adaptor entity calculates the time and cost of adaptation and finds a sequence of adaptation mechanisms to achieve the adaptation. This is outside the scope of this research.

- **New adaptation execution request received**

- **Adaptation process started (execution request)**: The AM retrieves the ADME invocation sequence and triggers the Content Adaptor to perform the adaptations
- **Invoking adaptation mechanisms**: This is outside the scope of this research and is part of the Content Adaptor functionalities.

- **Content successfully adapted and sent to user**: The AM sends the adapted content or service URI back to the user environment.

### 6.2.3.4 Content Delivery

The adapted content URI is delivered to the user environment by the dispatcher agent (outside the scope of this research), in the demonstration, the Java mobile emulator, N80 and N70 have been tested as presented in Figure 42 and Figure 43. Section 6.3.4 presents information on the web services interface between the user environment and adaptation management system.
6.3 System Implementation

The implementation is described in two parts. Section 6.3.1 describes the implemented system components. The rest of the sections describe implementation aspects related to context assimilation and ADT.

6.3.1 Implemented Components

Figure 44 shows a detailed architecture of the AM. The implementation of the components is outlined below. Dashed italic shapes denote components that have either not been implemented or that are provided. Sections 6.3.2 through to Section 6.3.5 present implementation of functionalities provided by the following implemented components:

- **ACM**
  
  This entity is responsible for managing adaptation context. It has three main components: the context extractor, the context formatter and the adaptation operations context handler. The implementation of this entity is discussed in Sections 6.3.2.1 and 6.3.2.2.

- **Context Extractor**
  
  The extractor manages several extraction tools to either extract embedded metadata from content or to parse the content and extract metadata on physical characteristics of the content. This entity is briefly discussed in Section 6.3.2.2.

- **Context Formatter**
  
  This entity manages several formatting tools to handle different context formats. The AM implements a context formatter for parts of MPEG-7 (MDS) and MPEG-21(DIA) profiles. Implementation issues of this component are discussed in Section 6.3.2.2.

- **RM**
  
  This component implements reasoning functionalities that aid context management and decision taking such as ontology reasoning and rule based reasoning. Implementation aspects of this entity are discussed in Sections 6.3.2.6 and 6.3.3.1.

- **Adaptation Decision Engine**
  
  The decision engine Controls the flow of decisions, extracts the refined context and invokes the necessary ADTEs. This entity’s implementation is discussed in Section 6.3.3.
- Adaptation Gateway

The gateway creates the web services interface to receive requests from web service clients and the message translator entity, which reads the web service requests and writes adaptation results to a suitable web service response. The gateway contains and manages adaptation clients. The Content Adaptor Interface manages the interaction between the AM and the Content Adaptor, both the AM and the Content Adaptor run in the same Java Virtual Machine (JVM). The AM creates the Content Adaptor adaptation mechanisms discovery engine and triggers discovery. The discovered ADMEs are sent to the AM to be associated with the corresponding adaptation operations. Implementation issues of this entity are discussed in Section 6.3.4.

![Figure 44: AM implemented components](image)

6.3.2 Adaptation Context Assimilation

Context assimilation is a key part of the adaptation process. This section discusses implementation aspects related to this stage including: adaptation operation context management, context formatting and context refinement.

6.3.2.1 Adaptation Operation Context Management

Adaptation operations context describes the adaptation mechanisms that can be applied to realise the adaptation decisions. An adaptation operation description corresponds to one or more adaptation mechanisms that are discovered as adaptation services by the Content Adaptor.
Chapter 6. Implementation and Proof of Concept Demonstration

Upon system start-up, the AM loads supported adaptation operations descriptions from an XML file. The structure of the XML operation description is depicted by Figure 45. The description specifies only the static attributes including format, modality and language for both the input and output of the operation. These attributes, together with the adaptation operations name (OPName) uniquely, identify the adaptation operation; therefore, an adaptation operation can be identified by an OPID defined as follows:

\[
\text{(A)} \quad \text{OPName:inModality_inFormat_inLang-outModality_outFormat_outLang}
\]

If the input and output language parameters are not relevant, for example for a video format transcoding operation, then a default value such as noLang takes the value of inLang and outLang.

Adaptation operations handling implementation is depicted by Figure 46. Adaptation mechanism updates such as the availability or unavailability of a new adaptation mechanism are received from the Content Adaptor. To find the corresponding adaptation operation for an ADME (\text{findOPforADME}), an adaptation operation ID of the form depicted by (A) is formed out of the ADME parameters, and is searched in the supported and active adaptation operations lists which are implemented as hash tables for efficient searching. \text{updateOPwithADME} adds (or removes, depending on the type of update) the ADME Id to the list of implementing ADMEs of the OP object and updates the input and output parameters of the OP object. Then, the ADME is added to (or removed from) the list of active adaptation mechanisms.
6.3.2.2 Context Extraction

Context extraction is necessary if no content context is provided. Several tools can be used; one of the latest technologies to extract embedded metadata is Adobe's Extensible Metadata Platform (XMP)\(^3\).

Context extractors used by the AM implement the interface specified by Figure 47. The main method extractContext takes as a parameter the content and returns a context profile object.

\(^3\) http://www.adobe.com/products/xmp/
Context formatting is performed in two stages, XSLT formatting and pre-reasoning parameters calculations. Aspects related to these steps are explained in Sections 6.3.2.3 and 6.3.2.4.

6.3.2.3 Context Formatting with XSLT

Context input may be in the form of several context standards, such as MPEG-7/21 and CC/PP. These standards are XML based; therefore, the AM uses XSLT for context formatting. The context formatter manages several XSLT transformation tools to format different context standards to the required format, which is the ADIO. ADIO description and implementation are provided in Sections 4.3.1.1 and 6.3.2.5, respectively. The XSLT tools contain statements that map constructs from input context profiles to corresponding concepts in ADIO. Two XSLT transformation tools for parts of MPEG-21 DIA and MPEG-7 MDS have been implemented to format MPEG-21 DIA and MPEG-7 MSD to the internal context representation model (ADIO).

In order for the adaptation manager to use the XSLT formatter, the formatter is added to the list of supported formats described in an XML file that is loaded at system start-up (Figure 48). This approach aims at enhancing the system extensibility to new formats as will be evaluated in the next chapter.

![Figure 48: Adding formatters to the system](image)

6.3.2.4 Context Formatting with Pre-reasoning Parameters Functions

As described in Section 4.4.4.2, the formatted context needs to go through a pre-reasoning parameter calculation step. This is because there is no mechanism within ontologies or ontology reasoners to calculate the value of data properties that are functions of other data properties specified in the context. For example, the following may need to be calculated:

\[
\text{(B)} \quad \text{Device.batteryLimitation} = \text{Service.batteryRequirement} - \text{Device.batteryAvailable}
\]

Device and Service are OWL classes (or instances) and batteryLimitation, batteryRequirement and batteryAvailable are data type properties.
This step is called pre-reasoning parameters' calculation, because the calculated parameters (Datatype properties in the ADIO) are important for the context refinement (based on ontology reasoning) step, as explained in Section 4.4.4.2. They define the types of several instances in the ontology, for example the types of the limitations on a device such as \textit{batteryLimited}, \textit{memoryLimited}, etc.

It is possible to perform such calculations within the context if a rule-based approach was adopted for context modelling (and using a rule engine that offers this capability such as JESS). Since ontology, the chosen context model represented in ADIO as described in Section 6.3.2.5, has no mechanism to support such calculations; the pre-reasoning parameters calculations step is introduced.

Extensibility is a key feature that has to be taken into account; the implementation presented here is evaluated for extensibility in Section 7.5.2.2 of the next chapter.

In order to enable ease of extensibility and maintenance, pre-reasoning functions are defined in an XML format as stacks, they are loaded upon system start-up by the context formatter and applied to the formatted context. The implementation is generic and describes any type of function. To add pre-reasoning functions, such as the one described in (B), the XML file that describes the function needs to be extended.

The functions are described as post fix expressions and evaluated using a post fix expression evaluator. For example, the function depicted by Figure 49 will look like the following after being extracted:

\[
\text{Devicel.batteryAvailable service1.batteryRequirement user1.batteryPreference - - Device1.batteryLimitation}
\]

The last argument: \textit{device1.batteryLimitation} is the result of the function. Thus the postfix expression evaluator will apply the following calculation:

\[
device1.batteryLimitation = \text{Device1.batteryAvailable} - \text{service1.batteryRequirement} - \text{user1.batteryPreference}
\]

Pre-reasoning function descriptions consist of three parts:

1. Arguments: specified by the \texttt{<Arg>} tag. The arguments are specified in the following way: (individual | Class).data property. This syntax is used to find the property to which the result of the calculation is assigned or a property from which an argument is read.
2. Operators: these can be -, *, /, + and are specified with the <OP> tag
3. Constants: specified by the <Const> tag, instead of retrieving a value from the ontology, a constant can be used if the value applies to all context profiles, for example, an average remaining battery power preference.

![Functions diagram](image)

**Figure 49: Pre-reasoning functions**

Another issue with OWL reasoning is the open world assumption as discussed in Section 5.4.3.2, which is deducing negative knowledge from its absence. For example, the following cannot be achieved in OWL: if the fact that a device someDevice supports a format someFormat is not present in the context, then someDevice does not support someFormat. This can be solved with the latest version of OWL using negative property assertions, i.e. explicitly asserting that someDevice does not support someFormat.

Figure 50 illustrates how negative property assertions can be added to context based on information that is not available. The <NegativeProperty> tag indicates which property needs to be added as negative axioms. For example, device1.supportedFormat.

The <Provided> tag indicates the data type property to retrieve the positive knowledge from, i.e. the context that is available. For example in this case it is device1.supportedFormat (and the context could be the set providedSet= {mp4, mp3, jpg}).

The <Required> tag indicates the data type property, values of which are compared against the values retrieved from the property specified in the <Provided> tag. For example in this case it is the service1.requiredFormats (and the context could be the set requiredSet = {mp3, bnp, png}).

(requiredSet – providedSet) represents the set of values on which negative property assertions are added. In this example, the set is {bnp, png} and the negative property is supportedFormat. Therefore, the following is asserted:
Chapter 6. Implementation and Proof of Concept Demonstration

- Not device1.supportsFormat (bmp)
- Not device1.supportsFormat (png)

It is possible now for the ontology reasoner to deduce that device1 is for example bmpLimited and pngLimited because it is explicitly stated that the formats bmp and png are not supported. In ontology reasoning, the open world assumption (OWA) entails that all knowledge has to be explicitly stated i.e. it cannot be simply inferred from absent statements.

![Figure 50: Pre-reasoning negative properties calculations](image)

Sections 6.3.2.5 and 6.3.2.6 present context refinement implementation aspects which involve ontology reasoning on the context model implemented as the ADIO.

6.3.2.5 Context Refinement using the ADIO

Interoperability and extensibility of the adaptation system are key aims of this research, thus interoperable and extensible mechanisms must be adopted. XML ensures syntax interoperability by enforcing formal syntax. Semantic interoperability is to have a common and formal understanding of context concepts such as Display, User Characteristic and Presentation Preference. DL is suited for such a purpose, i.e. describing the domain concepts and formally defining the relationships between them. To ensure both syntax interoperability and semantic interoperability, a combination of XML and DL is needed. Ontology Web Language (OWL) is the most widely used and deployed solution that provides advantages of both XML and OWL. The difference between DL and ontology is that ontology is a concept and DL provides a powerful formalism to implement it.
ADIO is an OWL ontology that models the adaptation domain context. ADIO is the format to which all input context is converted. The base terminology and concepts underlying the ADIO are defined in MPEG-21 DIA Universal Environment Description (UED) and MPEG-7 Multimedia Description Schemes (MDS). The ADIO is constructed manually based on concepts from MPEG-21 DIA UED and MPEG-7 MDS in addition to the higher level concepts described in Section 4.5.2.

OWL is based on XML and DL and hence provides the advantages of both. MPEG-21 DIA UED and MPEG-7 MDS were chosen because they are the most comprehensive standards that define general concepts in the adaptation domain such as device, device display, network, user preference, device capability, etc.

There are several versions of OWL; the version used for ADIO implementation is OWL 2.0, which is the latest version of OWL that provides data types reasoning and negative property assertions, features which are essential for ADIO implementation. The implementation of ADIO was aided by the Protégé [145] interface, specifically version 4, which supports OWL 2.0 and provides integrated Pellet and Fact++ ontology reasoners. ADIO structure and concepts are discussed in Section 4.5.2. The API used is the OWL API [158], it is the only API that supports the full features of OWL 2.0 [159], it is supported by Protégé, Pellet and Fact++.

6.3.2.6 Context Refinement using Ontology Reasoning Support Functions

The RM provides functions to aid ontology reasoning. These functions are necessary for both context refinement and ontology based rule based reasoning. The main support functions are represented in the UML class diagram in Figure 51 and described in Table 11.

<table>
<thead>
<tr>
<th>OntologyReasoningHandler</th>
</tr>
</thead>
<tbody>
<tr>
<td>+loadOWLAPIOntology(in source, in uri, in hashMap)</td>
</tr>
<tr>
<td>+loadOntologyToReasoner(in OWLOntology, in OWLReasoner)</td>
</tr>
<tr>
<td>+createOWLReasoner() : &lt;unspecified&gt;</td>
</tr>
<tr>
<td>+classifyOntology()</td>
</tr>
</tbody>
</table>

Figure 51: Support functions for ontology reasoning
6.3.3 ADT

As explained in Section 5.3, the adaptation management system adopts a two layer ADT approach, the first layer uses a rule based approach together with specialised ADTEs and the second layer uses the MPEG-21 DIA based ADT to select the parameters of the adapted content. This thesis proposes a new approach of rule based reasoning based on the Semantic Web technology: DL. The theory behind this approach is explained in Section 5.4. This section outlines implementation aspects of the first layer ADT. This thesis does not deal with the specialised adaptation decision engines or the MPEG-21 DIA adaptation engine. Their availability is documented in the state of the art such as in [86] for the MPEG-21 DIA-ADTE and in [32] for a HTML form resizing adaptation decision engine based on fuzzy logic.

6.3.3.1 Ontology Based Rule Based Reasoning

The main functions to support ontology based rule based reasoning, which is used for first layer ADT, are presented in the UML class diagram depicted by Figure 52 and described in Table 12. These functions also use the functions for ontology reasoning explained in Section 6.3.2.6.

Table 11: Ontology reasoning support functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>loadOWLAPIOntology</td>
<td>Loads an ontology from an input stream, a reader or a file. The hash map</td>
</tr>
<tr>
<td></td>
<td>input parameter contains the physical URIs of the imported ontologies.</td>
</tr>
<tr>
<td>loadOntologyToReasoner</td>
<td>Loads an ontology to the reasoner knowledge base.</td>
</tr>
<tr>
<td>createOWLReasoner</td>
<td>Used to create an OWL reasoner based on either Pellet or Fact++.</td>
</tr>
<tr>
<td>classifyOntology</td>
<td>Classifies an ontology on the reasoner knowledge base.</td>
</tr>
</tbody>
</table>

Table 12: Support functions for ontology based rule based reasoning

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>+loadRulesOntology()</td>
</tr>
<tr>
<td>+composeTempBodyClass()</td>
</tr>
<tr>
<td>+getRuleHead()</td>
</tr>
<tr>
<td>+extractDecisions()</td>
</tr>
</tbody>
</table>

Figure 52: Support functions for ontology based rule based reasoning

142
Table 12: Ontology based rule based reasoning support functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>loadRulesOntology</td>
<td>Loads the rules described in an ontology.</td>
</tr>
<tr>
<td>composeTempBodyClass</td>
<td>Composes a temp rule body from the refined context.</td>
</tr>
<tr>
<td>getRuleHead</td>
<td>Finds the head (represents the decisions) that corresponds to the temp rule body.</td>
</tr>
<tr>
<td>extractDecisions</td>
<td>Extract the decisions from the rule head.</td>
</tr>
</tbody>
</table>

Figure 53 shows a screen shot of DL (Ontology) based adaptation rules. The heads represent the conclusions of the rules, including adaptation decisions. The bodies represent the conditions, and the rules link the bodies with the heads. Rule classes are made equivalent to the corresponding body classes, Figure 53 (b, c), and head classes are made super classes of the corresponding rule classes Figure 53 (a).

Figure 53: Adaptation rules written with the Protégé ontology editor
6.3.4 **Interfacing**

There is an internal Java based interface between the AM and the Content Adaptor entities. The Content Adaptor is responsible for calculating the cost of adaptation (in terms of time and money for this demonstration), finding the sequence of adaptation mechanisms and invoking them. The Content Adaptor entity is outside the scope of this research.

The interface between the adaptation management system and adaptation clients is based on web services; the AM web service description used in the demonstration is described by Table 13 and Table 14.

### Table 13: The WSDL web service interface: Adaptation requests

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Action</strong></td>
<td>Defines the type of request: adapt content or inquire about adaptation cost.</td>
</tr>
<tr>
<td><strong>Constraints</strong></td>
<td>Acceptable time and cost for the user.</td>
</tr>
<tr>
<td><strong>Request ID</strong></td>
<td>The ID assigned by the adaptation system to the request.</td>
</tr>
<tr>
<td><strong>Client ID</strong></td>
<td>The ID assigned by the adaptation system to the client.</td>
</tr>
<tr>
<td><strong>Context URI</strong></td>
<td>Context profiles of usage environment and content.</td>
</tr>
<tr>
<td><strong>Service URI</strong></td>
<td>The location of the content or service to be adapted.</td>
</tr>
</tbody>
</table>

### Table 14: The WSDL web service interface: Adaptation response

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Request ID</strong></td>
<td>The ID assigned by the adaptation system to the request.</td>
</tr>
<tr>
<td><strong>Client ID</strong></td>
<td>The ID assigned by the adaptation system to the client.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>Defines if the service can be adapted, or if the constraints can be met.</td>
</tr>
<tr>
<td><strong>Context URI</strong></td>
<td>The location of the adapted content context.</td>
</tr>
<tr>
<td><strong>Constraints</strong></td>
<td>The time and cost of the adaptation.</td>
</tr>
<tr>
<td><strong>Adapted Service URI</strong></td>
<td>The location of the adapted service or content.</td>
</tr>
</tbody>
</table>
6.3.5 Handling of Adaptation Requests

Figure 54 depicts how adaptation requests are handled internally. Each request is associated with an adaptation client, an adaptation request and an adaptation process object. The adaptation client object holds all requests from a client and an adaptation process is created to invoke AM functionalities to process the request. At start-up, the AM creates hash tables to manage requests, processes and clients.

6.4 Summary

This chapter has outlined the implementation aspects of the AM entity. It has demonstrated that the solutions proposed by the author are feasible in terms of implementation. The chapter has presented the technologies and tools that have been used. For example, the Java, OWL, XML and WSDL languages, the Eclipse and Protégé Development environments, the Pellet and Fact++ reasoners, the OWLAPI API and the XML transformation and Web services technologies.

The chapter has presented the AM implementation, a major component in the content and service adaptation test bed.

The AM implementation is a prototype to serve as a proof of concept. Some parts of the implementation, for example, the MPEG-21 DIA formatter does not support the entire MPEG-21 DIA standard, it covers parts of it, and the rest can be implemented in the same manner.

This chapter has laid the way to the evaluation chapter in order to assess aspects of the solutions that have been proposed by the author based on the implementation that has been outlined in this chapter and the research aims that have been presented in Chapter 1.
Chapter 7. Evaluation

7 Evaluation

7.1 Introduction

Evaluation is a very important part of the research thesis to assess the author's contributions in terms of achieving the aims and objectives of the research. Several methods of evaluation may be used according to the research aims and objectives and the available benchmarks and evaluation tools. Section 7.2 discusses these issues with regard to the evaluation methods that may be used for content adaptation management systems. Section 7.2.6 outlines the evaluation methods adopted for the AM including performance modelling and analysis, performance testing and implementation and demonstration use cases. Section 7.3 presents Performance Analysis Process Algebra (PEPA) evaluation results [160]. Section 7.4 presents performance testing of the implemented AM prototype. Section 7.5 presents the main method of evaluation for this research, i.e. implementation and demonstration use cases. Demonstration use cases are used to verify the degree to which the research contributions fulfil the non functional requirements of extensibility, interoperability and maintainability. Section 7.6 concludes the chapter.

7.2 Evaluation Methods

This section outlines main methods of evaluation that may be applied to content adaptation management systems (Table 15), and identifies the methods that are adopted in this thesis. Three main categories of evaluation methods can be identified:

7.2.1 Performance Modelling and Analysis

There are several performance modelling and analysis methods such as process algebra, queuing networks, etc. A survey is available in [161].

These methods can be applied at different stages of system development including architecture, design and implementation. They are especially suitable for systems with complex interactions such as distributed and real time systems. They can be used to predict system performance in terms of parameters such as response time, utilization and throughput. Analysed performance parameters depend on the properties of the system being evaluated. As indicated by Table 15,
performance modelling and analysis method is not used in the surveyed adaptation management systems; this may be due to several reasons including:

- **Research Aims**

With regard to content and service adaptation frameworks, contributions are mainly related to architectures and frameworks that aim at using new technologies and tools, such as XML and Semantic Web technologies, for the purpose of adaptation as opposed to for example, introducing optimised algorithms to enhance adaptation system performance. Therefore, in most state of the art adaptation frameworks, main part of evaluation is based on non performance based parameters such as usability.

- **Tools and APIs**

The APIs and tools used in proof of concept implementations may not be mature; this greatly affects system performance and hence renders system performance modelling and analysis less relevant.

- **Benchmarks**

Research in this domain has not matured enough to focus on measuring performance parameters and setting benchmarks to evaluate against them.

### 7.2.2 Implementation and Demonstration Scenarios

This type of evaluation is very common in this domain. It is often accompanied with experimental testing evaluation, explained in the next section. The implementation of proposed solutions demonstrates their feasibility. Demonstration scenarios or use cases are used, input and output are identified and key features of the implemented system are demonstrated. Demonstration scenarios can be used for two purposes:

- **System Feasibility/Practicality Demonstration**

The output of the system is examined to check the correctness of implementation with respect to the input. This is the first step to prove that the proposed solution is applicable in real life. Performance experimental testing is used to evaluate the viability of the proposed solutions.

- **Non Performance based Features Evaluation**

This category is used to evaluate the proposed solution in terms of non performance based features. In the context of this research; this includes extensibility and interoperability of the adaptation system. Two types of scenarios can be used; black box and white box scenarios.
7. Black Box Scenarios

Black box scenarios are agnostic (hence the term "Black Box") to the system internal properties and implementation. Parameters external to the adaptation system are varied through input. For example, the system in [86] evaluates the universality of decision taking using several adaptation scenarios involving different types of content and adaptation decision parameters. The system successfully takes decisions regardless of the semantics of the content type.

7. White Box Scenarios

As opposed to black box scenarios, this category does not depend on the input to the system; it rather depends on modifying internal system structure, for example, modifying the system to support certain functionality. The research reported in [32] and [84] uses white box scenarios to examine system extensibility by showing the steps required to extend the functionality of the system to certain tools.

7.2.3 Performance Testing

Implementation can only prove the feasibility of the proposed solutions, performance tests are necessary to evaluate their viability. It is important to highlight the tools and APIs that are used, the test bed capabilities and the inputs to the system. This information helps analyse performance testing results. This category considers performance parameters, some of which are general to most systems such as response time. Conclusions about the scalability of the proposed solutions may be drawn by varying input parameters and observing system behaviour, however, to a limited extent compared to performance modelling and analysis evaluation. Experimental testing can be either performed for the overall system or for specific subsystems; the latter is used if different parts of the system have different performance parameters or different properties that affect the performance parameter.

7.2.4 User Validation

There are several usability evaluation techniques [162] [163] including usability testing, heuristic evaluation, cognitive walkthroughs, focus groups and questionnaires.

In usability testing, users are observed by usability experts while using the system to assess the practicality and system ease. Heuristic evaluation is also performed by experts to assess usability against guidelines derived from human-computer interaction, graphics design, cognitive psychology, etc. Cognitive walkthrough can be used at different stages of system design where a group of evaluators consider certain goals with regard to using the system, and determine the
steps and difficulty involved in achieving the goals. Focus groups are used to assess an initial design, by presenting it to users, gathering their feedback and discussing their requirements. Questionnaires are a very popular method of evaluation because a good amount of information can be gathered in a short time and at a low cost. For content adaptation, a number of criteria are presented to users to rate the adapted content against some scale to indicate their satisfaction or agreement with a statement about the adapted content such as: "the adapted content is rendered clearly to the user device". Questionnaires are particularly practical with websites as a wide range of users can access the system and evaluate it.

Usability testing, focus groups and questionnaires involve user validation. The system in [32] used a user study questionnaire to evaluate adapted websites with their website content adaptation system. The system in [136] used usability testing to evaluate the proposed text summarization system. 50 students were invited to use the system and gave their feedback. In [164], a group of students were invited to use the proposed medical education learning system that adapts content according to user preferences. User feedback with regard to how content adaptation affected their learning experience was collected.

### 7.2.5 Evaluation Methods in State of the Art Adaptation Frameworks

Table 15 presents evaluation methods used in main state of the art content and service adaptation frameworks. All presented frameworks use demonstration scenarios for implementation feasibility evaluation. An implementation is required for all types of evaluation except for performance modelling and analysis. Therefore, demonstration scenarios are in most cases implicitly used.

Demonstration scenarios for implementation feasibility evaluation [31, 32, 34, 84, 124, 127, 165] differ from white box/black box demonstration scenarios in that the latter two evaluation methods aim to evaluate certain features targeted by the research, while the former aims to prove research contributions' feasibility. For example, the work reported in [86] uses black box demonstration scenarios to evaluate the universality of the ADT system. The evaluation in [86] is black box because only input to the system is varied without modifying the system internally. White box demonstration scenarios are used in [32] to demonstrate system extensibility to new adaptation scenarios by extending to new content parsing tools, extending the rule base and extending adaptation procedures.

Performance testing is also common [32, 126, 129, 164, 166, 167]. The main performance parameter that is tested is response time. For example, The system in [164] evaluated the use of semantic web services by comparing system response time when using semantic web service versus normal web services. The system in [167] evaluated system response time of the proposed
adapation middleware with regard to the number of adaptation tasks. The next section presents the evaluation methods adopted for the AM proposed and developed in this thesis.

7.2.6 AM Evaluation

This section outlines the approaches adopted to evaluate the contributions discussed in this thesis.

- **Performance Modelling and Analysis**

Section 7.3 presents the performance modelling and analysis of the adaptation management system using Performance Evaluation Process Algebra (PEPA). The evaluation is based on a prototype implementation to evaluate the AM system design.

- **Demonstration Scenarios/Implementation Feasibility**

Chapter 6 presented the implementation of the AM, three adaptation scenarios are used in order to demonstrate the implementation feasibility of the authors’ contributions.

- **Experimental Testing**

Although this research focuses on improving non performance based features such as interoperability and extensibility, it is nonetheless important to test the viability of real life deployment, at least for long term.

- **White Box Implementation Scenarios**

This research aims at tackling the complexity of the content and service adaptation problem by proposing solutions for extensibility and interoperability. White box demonstration scenarios are used (Section 7.5) to evaluate the proposed solutions in this regard. The term “white box” is used because internal system modification is part of the evaluation.
Table 15: Evaluation in content and service adaptation frameworks

<table>
<thead>
<tr>
<th>System</th>
<th>Research Aims</th>
<th>Performance modelling &amp; analysis</th>
<th>Demonstration scenarios (white box)</th>
<th>Demonstration scenarios (black box)</th>
<th>Demonstration scenarios (feasibility)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[86] MPEG-21 DIA based ADTE</td>
<td>Universal decision taking agnostic of adaptation parameter semantics</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>[129] Rule-Based Adaptation System</td>
<td>An adaptation system based on a rule based approach</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>[34] IBM Adaptation Server</td>
<td>Appropriate content version to communication environments</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>[31] Context Aware Adaptation System</td>
<td>Adapt content according to user context including preferences</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>[32] The XAdaptor</td>
<td>Flexible adaptation system</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>[84] The Versatile Transcoding Proxy</td>
<td>Extensible adaptation system to adaptation tools</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>The AM (this research)</td>
<td>Manage adaptation complexity by ensuring extensibility and interoperability</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

7.3 Performance Modelling and Evaluation of the Design

A formal analysis detects deadlocks or flaws in system design. The AM design has been evaluated [160] using an early AM prototype with Performance Evaluation Process Algebra (PEPA) [168]. PEPA is a high-level model specification language for low-level stochastic models. It describes a system as interactions between components as they engage in activities.

PEPA can be used to build a performance model based on system design and evaluate the design in terms of performance parameters such as response time, throughput and utilization. The performance model was built based on the functional architecture and adaptation working cycle modelled in UML sequence/activity diagram as described in Section 7.3.1. The entities, activities
and messages exchanged were modelled in PEPA notation to build the PEPA performance model as described in Section 7.3.3.

7.3.1 The Adaptation Working Cycle

Figure 55 shows the adaptation working cycle of an early AM prototype. The actions on arrows represent shared activities between the components that define the start and end of the arrow. The actions represented in boxes are actions executed by the respective component. The Context Provider (CP), the Context Reasoner (CR) and the ADTE of the evaluation AM prototype provide the functionality of the ACM, RM and the DIA based ADTE of the final AM prototype discussed in this thesis, respectively. In the evaluation AM prototype, the first layer of ADT is performed by the CR and the second layer of decision taking is performed by the ADTE.

The adaptation cycle works as follows. Upon receiving an adaptation request, an Adaptation Process (AP) is created (create_ap). The AP invokes the CP to gather (gather_c) the necessary contextual information and the CP checks if the necessary context is available (availability). If context was not provided with the request it is requested from the user environment or content and service provider (requ_c). The Adaptation Gateway (AG) forms the message and sends it to the intended entity (requ_msg). After all necessary context is gathered, the CP checks if additional information needs to be extracted from the content or service. If so, extraction tools are used to extract descriptions from the content or service (extract_c). Once all necessary context is available it is loaded (load_c) using the provided URIs. The context is formatted (format_c) by the Context Formatter (CF) entity and is sent back to the AP (rtm_f_c and set_f_c). The AP invokes the CR to take the first layer decisions (take_1l_dec) using the formatted context, a Reasoning Task (RT) is created for this purpose (create_r_task). The RT loads the context in the knowledge base of the DL reasoner (load_c_kb).

The DL reasoner computes the types (get_types) of instances in the knowledge base, for example, the types of a certain user device are: battery-limited, etc. These types are used to calculate constraints (calc_constraints) on service components (content). For example, a video file must have a frame rate and a size smaller than a certain value while maintaining a certain quality. The service layout (set_layout) is also defined. The AP then invokes ADE to take the second layer decisions (set service parameters, set_2l_dec) such as the resolution and frame rate of content element of the service. Once the ADT process is completed, the results are returned (set_2l_dec) to the AP which asks the AG to compose a message and send it to the content adaptor to execute the decisions (dec_msg).
7.3.2 Introduction to PEPA

Performance Evaluation Process Algebra (PEPA) [168], developed by Hillston in the 1990s, is a high-level model specification language for low-level stochastic models. It describes a system as interactions between components as they engage in activities. In this subsection, a brief introduction to PEPA is given; more information is available in [168]. Each activity in PEPA can be defined as a pair \((a, r)\) where \(a\) is the action type and \(r\) is the activity rate. The rate specifies the parameter of an exponential distribution. If a component \(P\) completes an activity \((a, r)\) and then behaves as \(Q\), it is denoted \(P \equiv (a, r)Q\) and the transition may be denoted as \(P \xrightarrow{(a,r)} Q\). The name and interpretation of combinators used in the PEPA language, which express the individual activities and interactions of the components, are as follows.
7.3.2.1 PEPA Syntax

Prefix: \((a, r).P\): Prefix is the basic mechanism which describes the behaviour of the system. Such a component will subsequently behave as \(P\) after it carries out the activity \((a, r)\), which has an action type \(a\) and a duration which satisfies exponential distribution with parameter \(r\).

Choice: \(P+Q\): The component \(P+Q\) represents a competition between two components. The system may behave either as \(P\) or as \(Q\). The activities of both \(P\) and \(Q\) are enabled. The choice is resolved by a race policy; the component whose activity is completed first proceeds, the other is discarded.

Cooperation: \(P \parallel L \parallel Q\): The cooperation combinator describes the synchronisation of \(P\) and \(Q\) over the activities in the cooperation set \(L\). For any activity whose action type is contained in \(L\), \(P\) and \(Q\) must cooperate to achieve the activity. However, they will proceed independently and concurrently with any activity whose action type is not in \(L\).

Parallel: \(P||Q\): The component \(P||Q\) represents two concurrent but completely independent components, meaning the cooperation set is empty. This is simply a shorthand notation for \(P \parallel^\infty \parallel Q\).

Hiding: \(P/L\): Hiding makes the activities whose action types are in \(L\) invisible for an external observer. The component \(P/L\) behaves as \(P\) except that any activities of types within the set \(L\) are hidden.

Constant: \(A \equiv P\): Constants are components whose meaning is given by a defining equation such as \(A \equiv P\), which gives the constant \(A\) a similar behaviour as the component \(P\).

7.3.2.2 Semantics and Performance Measures

PEPA has a structured operational semantics. This is a set of transition rules which define the evolution rules for each combinator of the language. The formal semantics map the PEPA language to a discrete state space, continuous time Markov chain (CTMC) [168]. The infinitesimal generator matrix of the CTMC can be obtained automatically by the PEPA tools. Solving the matrix equation characterising the global balance equations enables deriving the steady state probability distribution for the CTMC, upon which performance measures such as throughput and utilisation can be derived. Moreover, the matrix may be used as the basis for transient analysis, allowing measures such as response time distributions to be calculated.

7.3.3 System Analysis

This section defines a PEPA model based on the functional architecture and adaptation working cycle presented in Section 7.3.1. The model is comprised of nine components, corresponding to
the main components of the functional architecture, i.e., the AG, AP, CP, Context Extractor (CE), CF, CR, RT, the DL Reasoner (DLR) and the ADTE components. Each of the components has a repetitive behaviour, reflecting its role within the working cycle. Below, PEPA definitions for the AG and CP components are shown.

AG: The behaviour of the AG starts with the arrival of an adaptation request, represented as activity \textit{receive1}. The subsequent action is to create an AP, \textit{create\_ap}, which is shared with the AP. The following event is one of two choices depending on the response of the AP. The AP may either invoke the AG to request contextual information, \textit{requ\_msg}, if no context was provided or go directly to wait for the result of the AP. When the requested context arrives, the AG sets the provided information to the AP, represented by \textit{receive2} and \textit{set\_info} respectively. After receiving the result and processing the \textit{de\_msg} message, the AG completes a working cycle and goes back to the initial state of waiting for adaptation requests. AG PEPA definition is depicted by Figure 56.

\begin{align*}
AG\_receive1 & \equiv (\text{receive1, } r\_\text{receive1}) \cdot AG\_\text{create}\_ap \\
AG\_\text{create}\_ap & \equiv (\text{create}\_\text{ap, } r\_\text{create}\_\text{ap}) \cdot AG\_\text{choice} \\
AG\_\text{choice} & \equiv (\text{requ}\_\text{msg, } r\_\text{requ}\_\text{msg}) \cdot AG\_\text{receive}\_2 \\
& \quad + (\text{dec}\_\text{msg, } r\_\text{dec}\_\text{msg}) \cdot AG\_\text{receive}\_1 \\
AG\_\text{receive}\_2 & \equiv (\text{receive2, } r\_\text{receive2}) \cdot AG\_\text{set}\_\text{info} \\
AG\_\text{set}\_\text{info} & \equiv (\text{set}\_\text{info, } r\_\text{set}\_\text{info}) \cdot AG\_\text{dec}\_\text{msg} \\
AG\_\text{dec}\_\text{msg} & \equiv (\text{dec}\_\text{msg, } r\_\text{dec}\_\text{msg}) \cdot AG\_\text{receive}\_1
\end{align*}

Figure 56: AG PEPA Definition

CP: After an AP has been created, it invokes the CP to gather the necessary context, \textit{gather\_c}. The CP checks whether the contextual information is available (availability). If context was not provided it is requested and set, represented by the activities \textit{requ\_c} and \textit{set\_rc}, respectively. After all necessary contextual information has been gathered, the CP checks whether additional information needs to be extracted from the content or service. If context extraction is needed, then \textit{extract\_c} and \textit{set\_ec} are invoked. Once all necessary context is available, it is loaded (\textit{load\_c}), formatted by the CF entity (\textit{format\_c}) and set back to the AP (\textit{rtn\_f\_c} and \textit{set\_f\_c}). A working cycle of the CP is thus completed. CP PEPA definition is depicted by Figure 57.
The definition of PEPA models for other components is similar, and thus omitted. The system equation, which specifies how the complete model is constructed from the defined components, is depicted by Figure 58. The equation specifies how the components interact, by forcing cooperation on some activity types. The number of independent instances of some components, such as APs, varies in some experiments.

**Figure 58: Adaptation system PEPA equation**

7.3.4 Parameter Settings for Performance Modelling and Model Checking

Table 16 presents parameter settings for the functions shown in Figure 55. The settings are used for the evaluation of the AM design. The durations, represented in milliseconds, were obtained by running experiments on an early prototype of the AM on a Linux machine (1.99 GHZ processor and 512 MB memory). The rate represents how many activities can be completed in a time unit, one second in this case. The most time consuming task is the first layer decision taking \(\text{take}_1\text{l }\text{dec}\) task because it involves DL reasoning.
Table 16: Parameters setting for PEPA evaluation

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Time</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>receive</td>
<td>Receives an adaptation request</td>
<td>1</td>
<td>1000</td>
</tr>
<tr>
<td>create_ap</td>
<td>Creates an AP to process the receive request</td>
<td>5</td>
<td>200</td>
</tr>
<tr>
<td>gather_c</td>
<td>Gathers the context</td>
<td>125</td>
<td>8</td>
</tr>
<tr>
<td>availability</td>
<td>Checks the availability of the context</td>
<td>5</td>
<td>200</td>
</tr>
<tr>
<td>requ_c</td>
<td>Requests context from user side or content or service provider side</td>
<td>1</td>
<td>1000</td>
</tr>
<tr>
<td>requ_msg</td>
<td>AG forms the request to get the context</td>
<td>2</td>
<td>500</td>
</tr>
<tr>
<td>receive</td>
<td>A message is received (with the requested context)</td>
<td>1</td>
<td>1000</td>
</tr>
<tr>
<td>set_info</td>
<td>information provided (requested context) is set to AP</td>
<td>1</td>
<td>1000</td>
</tr>
<tr>
<td>set_rc</td>
<td>The requested context is set to the process</td>
<td>1</td>
<td>1000</td>
</tr>
<tr>
<td>extract_c</td>
<td>Extracts context if necessary from the service and content</td>
<td>1</td>
<td>1000</td>
</tr>
<tr>
<td>set_ec</td>
<td>Sets the extracted context to the process</td>
<td>1</td>
<td>1000</td>
</tr>
<tr>
<td>load_c</td>
<td>Once context profiles are known, they are loaded from their URIs</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>format_c</td>
<td>Context is formatted to the required format</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>rtm_fc</td>
<td>Formatted context is returned to the CP</td>
<td>1</td>
<td>1000</td>
</tr>
<tr>
<td>set_fc</td>
<td>Formatted context is set to the AP</td>
<td>1</td>
<td>1000</td>
</tr>
<tr>
<td>take_1l_dec</td>
<td>Defining service layout and constrains</td>
<td>338</td>
<td>2.96</td>
</tr>
<tr>
<td>create_r_task</td>
<td>RT is created</td>
<td>5</td>
<td>200</td>
</tr>
<tr>
<td>load_c_kb</td>
<td>Context is loaded to the DL reasoner knowledge base</td>
<td>20</td>
<td>50</td>
</tr>
<tr>
<td>get_types</td>
<td>The types of instances (devices, content …) are defined</td>
<td>200</td>
<td>5</td>
</tr>
<tr>
<td>set_types</td>
<td>The types are returned to the reasoning task</td>
<td>1</td>
<td>1000</td>
</tr>
<tr>
<td>calc_constraints</td>
<td>Constrains are calculated</td>
<td>50</td>
<td>20</td>
</tr>
<tr>
<td>set_layout</td>
<td>Service layout and structure is defined</td>
<td>50</td>
<td>20</td>
</tr>
<tr>
<td>gen_1l_decs</td>
<td>Service layout and constraints are integrated in decisions</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>rtm_1l_dec</td>
<td>Decisions are returned to the CR</td>
<td>1</td>
<td>1000</td>
</tr>
<tr>
<td>set_1l_dec</td>
<td>Decisions are set to the AP</td>
<td>1</td>
<td>1000</td>
</tr>
<tr>
<td>take_2l_dec</td>
<td>AP requests second layer decisions</td>
<td>156</td>
<td>6.41</td>
</tr>
<tr>
<td>calc_parameters</td>
<td>Service parameters are calculated</td>
<td>150</td>
<td>6.67</td>
</tr>
<tr>
<td>set_2l_dec</td>
<td>Second layer decisions are set to the AP</td>
<td>1</td>
<td>1000</td>
</tr>
<tr>
<td>dec_msg</td>
<td>A decision message is formed and sent to the content adaptor for execution</td>
<td>5</td>
<td>200</td>
</tr>
</tbody>
</table>

4 In the case where all context is available and extraction is not needed
5 exclude communication overhead and time for the user environment to respond
6 depends on extraction tools and content properties such as size, etc.
7.3.5 Performance Analysis

Sections 7.3.5.1 and 7.3.5.2 show how the system performance changes with the variation of some parameters. It is considered that each type of component has only one instance in the system. Section 7.3.5.3 illustrates how the number of adaptation processes and the working mechanism of the AG affect the system performance. Experiments are conducted using the PEPA Workbench and associated tools [169].

7.3.5.1 Response Time Analysis

The response time is the interval between the arrival of a request, represented by the activity receive1, and the outcome of the system, indicated by the activity dec_msg. Response time measures the duration of a client’s wait for the output of the adaptation management process.

The cumulative distribution function of the system’s response time is given under the parameter setting in Table 16. As can be seen from Figure 59, the response time is sensitive to the take_1ll_dec rate, especially when the rate is below 2.96. The value 2.96 corresponds to an average time of 338 ms, as shown in Table 16. This result indicates that the client’s waiting time can be significantly decreased by reducing take_1ll_dec’s duration i.e. improving its performance.

On the other hand, not all rates have a significant impact on system response time. As illustrated in Figure 59, the rate of calc_parameters with average process latency of between 122 (corresponds to a rate of 8.2) and 193 (corresponds to a rate of 5.2) ms has little impact on the response time.

The response time is affected by other factors besides the performance of components and their interactions, for example, the number of adaptation requests, which is synonymous to the number of adaptation processes; this is discussed in Section 7.3.5.3. The impact of activities’ rates on other performance measures, such as throughput and utilisation is discussed in the next section.
Chapter 7. Evaluation

7.3.5.2 Throughput and Utilisation Analysis

The throughput of an activity is defined as the average number of activities completed by the system during one time unit (1 second for this evaluation) [168]. Since \textit{dec msg} represents the final outcome of the system, the throughput of this activity reflects system performance. Figure 60 shows that the throughput of \textit{dec msg} strongly depends on the rate of \textit{take\_1I\_dec}.

However, \textit{calc\_parameters}, has little influence on the throughput, may be due to its longer duration. Figure 60 illustrates that the rate of DL Reasoner’s \textit{get\_types} also impacts the throughput of \textit{dec\_msg}. Thus, increasing these rates, i.e. decreasing the duration, can improve the throughput. By increasing the rates of the activities mentioned above, not only the throughput can be improved, but also the other components’ proportion of idle time can be reduced. To illustrate this, the definition of utilisation is introduced. Utilisation is the proportion of time that a component spends in different states. Figure 61 illustrates that when the reasoner and the ADE enhance their performance, e.g. increase the rates of \textit{get\_types} and \textit{calc\_parameters}, the AP’s proportion of idle time is reduced, which means that the AP is more efficiently utilised.
Chapter 7. Evaluation

Figure 60: Impact of different functions on system throughput

Figure 61: Proportion of $AP_{set_{2l}\_dec}$ vs. $calc\_parameters$ and $AP_{set_{1l}\_dec}$ vs. $get\_types$ rates
7.3.5.3 Bottleneck Analysis

The AM is designed to work concurrently, i.e. dealing with several adaptation processes and reasoning tasks simultaneously. The AG receives and sends messages from and to entities in the AMF. In order to achieve an efficient AM operation, the gateway should support concurrency. If the gateway operates sequentially, there would be a jam of messages at the gateway resulting in reduced performance. The following experimental results illustrate this point.

Suppose there is only one gateway in the system operating sequentially, Figure 62 shows that the response time sharply increases with the number of APs. The response time here is the duration of processing an adaptation request, i.e. the interval between the create_ap and the dec_msg activities inclusive. Furthermore, the proportion of waiting time for creating the AP object, i.e. the utilisation of create_ap, increases sharply from near 0 to above 0.5 as shown in Figure 63, when the number of APs increases. In addition, Figure 64 indicates that if the AG operates sequentially, the throughput of dec_msg remains unchanged regardless of the number of APs. However, if the system runs multiple gateway instances (the number of the gateways should not be less than the number of APs) or a single gateway instance that supports concurrency, a significant improvement in throughput can be achieved while the idle proportion of each AP is kept at an extremely low level (Near 0 as shown in Figure 64).
7.4 Performance Testing

Chapter 6 has presented the proof of concept demonstration; hence, the feasibility of the contributions of this thesis is demonstrated. However, this section presents experimental testing to evaluate the contributions in terms of viability. The experiments were conducted using a Windows XP machine with a 2.0 GHz CPU and a 512 MB RAM.

As discussed in Section 7.2.6, experimental testing is not the main method of evaluation in this thesis. This is because the research aims and objectives are not related to performance optimization with regard to adaptation systems. The research aims and objectives are rather related to non-functional system features such as extensibility and interoperability. Therefore, the main method of evaluation used in this research is white box demonstration use cases as presented in Section 7.5. Nonetheless, it is important to ensure that the proposed contributions are deployable, at least in the long term. This section analyses the performance of the implemented AM to ensure that they would be deployable once related technologies have matured and more computational resources are available.
Figure 65: AM performance analysis

Figure 65 depicts performance testing results. Only AM functionalities are included, i.e. test bed functionalities such as planning adaptation decisions and invoking adaptation mechanisms to execute the decisions are not included.

The chart in Figure 65 shows the duration of different stages of adaptation processing in milliseconds, specifically, reading requests, context formatting, context refinement and ADT.

The performance chart shows five requests including an initial request which represents the first request received and processed by the AM. Except for the first request, all requests are processed under 1 second. The initial request is processed in approximately 6 seconds, 4.6 seconds of which represent the duration of ADT. ADT is based on the DL based rule based reasoning approach presented in Chapter 5. Upon receiving the first request, the rules ontology is classified in the reasoner knowledge base, hence the longer duration of ADT for the first request. Context refinement is also based on ontology reasoning and takes more time for the first request as ADIO is classified for the first time. There is a subtle increase in response time for request 4; this is due to an increase in DL reasoning response time (DL based rule based reasoning). Such increase is related to the instability of the DL reasoner implementation rather than the complexity in the request.

The “reading of request” and the “context formatting” stages take similar durations for all requests. The “reading the request” stage reads details such as context profiles and the requested...
content URI. Context formatting includes XSLT formatting (Section 6.3.2.3) and applying pre-reasoning parameters functions (Section 6.3.2.4).

It is recognised that the performance of the AM may significantly decrease if the system is under load; however, this is mainly due to the used APIs and the complexity of the used technologies, mainly DL reasoning. This section aims to evaluate the AM in terms of viability of the solutions, however, not necessarily for the short term.

### 7.5 Non Performance Based Parameters Evaluation

This section presents the main method of evaluation of the author’s contributions. Whereas the proof of concept implementation and performance testing provides an evaluation in terms of implementation feasibility and viability, non performance based parameters’ evaluation examines the contributions of this thesis in terms of achieving the research objectives outlined in Chapter 1.

The main aim of the research is to address the following two main problems:

1. **The complexity in the adaptation domain context**: This problem arises from the fact that several context adaptation parameters need to be considered. Moreover, new adaptation parameters are constantly introduced as new technologies emerge.
   a. This problem requires the adaptation process to be extensible to new adaptation scenarios, and to use extensible and maintainable context modelling and decision taking technologies.

2. **The heterogeneity of the adaptation domain**: This problem is due to the existence of several standards such as context standards and several content and device technologies. Furthermore, new standards and technologies may emerge.
   a. This problem requires the adaptation management system to provide mechanisms to interface and interoperate with the main standards and technologies in the adaptation domain.
   b. This problem also requires the adaptation management system to provide mechanisms to extend the adaptation process to support new context standards and adaptation tools.

The following subsections address the evaluation of the contributions proposed in this thesis with regard to the requirements 1(a), 2(a) and 2(b), which are addressed in depth in Chapters 4 and 5.

Technologies such as XML, Ontology and rule-based decision taking are known to be extensible. However, the way in which these technologies are employed may affect maintainability, for example the degree of hard coding maintenance that is required to extend the context model to certain adaptation parameters. The following sections are not merely evaluating the adaptation
7. System Extensibility Evaluation

Extensibility with respect to the content and service adaptation domain can be evaluated from different angles including:

- Extensibility to new context formats/extensions to existing supported formats
- Extensibility to new adaptation operations.
- Extensibility to new adaptation scenarios.

Each of the above extensibility aspects will be treated in the following subsections.

7.5.1 System Extensibility to New Context Standards

Context profiles for the requested content, the delivery environment and the user preferences descriptions are very important to adaptation systems as they contain information based on which adaptation decisions are taken. Therefore, the extensibility of the adaptation system to new context formats or parameters is crucial.

Context format standardization efforts such as MPEG-7 exist. However, the emergence of new context profile formats outside the standards is inevitable due to several reasons. New context formats may be introduced due to new adaptation parameters and content types (not covered in the existing standards) emerging as a result of technological advances. The emergence of new context parameters, such as new terminal capabilities and new content descriptions, or the emergence of domain specific descriptions, such as descriptions for entertainment content, social networking content, etc, entails either the development of new context formats or extending existing ones. In both situations, the adaptation system needs to be extended to support the new context parameters.

This section evaluates the adaptation management system in terms of its extensibility to new context formats or adaptation parameters. Evaluation in this section is based on a scenario about extending the AM context processing to a new context format. Because, extending to new context formats includes extending to new adaptation parameters of already supported context formats. Hence, evaluating extensions to new context formats includes evaluating extensions to new adaptation parameters.
As outlined in Section 6.3.2.3, the adaptation management system prototype implements an MPEG-7 and MPEG-21 DIA context formatting tools to support parts of the MPEG-7 MDS context description profiles and MPEG-21 DIA usage environment context profiles.

The User Agent Profile Standard (UAProf) is a device context description standard based on Composite Capabilities/Preferences Profiles (CC/PP). Several mobile manufacturers provide UAProf profiles for most of their mobile devices. Figure 66 depicts some UAProf concepts and the corresponding ADIO concepts that they need to be mapped to, i.e. formatted to. The mapping is indicated by arrows.

In order to evaluate system extensibility to new (or extensions of existing) context formats, this section presents the steps that are needed to support a new format, which is UAProf, which might introduce new context parameters. CC/PP based profiles, such as UAProf, are composed of a number of components, and are extensible to new adaptation parameters by extending existing components or creating new ones.

![Diagram of UAProf and ADIO concepts](image)

Figure 66: UAProf and ADIO

Context profiles based on CC/PP (i.e. RDF), OWL or another model are all based on XML. To support UAProf context profiles, a context formatter based on XML transformation (XSLT) that

---

7 [http://w3development.de/rdf/uaprof_repository/](http://w3development.de/rdf/uaprof_repository/)  
http://nds1.nds.nokia.com/uaprof/NE71-1r100.xml
Chapter 7. Evaluation

forms the input profile to the AM internal context representation model (ADIO) needs to be implemented. Implementing the XSLT based formatter is straight forward.

In order to add the new context formatter (i.e. the newly implemented XSLT transformer) to the AM, an entry is added to the context formatters' description file, depicted by Figure 67. The added entry enables the AM to load the new context formatter automatically at system start up. No maintenance of the coding is required.

```xml
<Formatter>
  <Description>A formatter for UAProf profiles</Description>
  <FormatterURI>c:\formatters\uaprofFormatter.xslt</FormatterURI>
  <Identification>
    <Namespace>
      http://www.openmobilealliance.org/tech/profiles/UAPROF/ccpschema-20021212
    </Namespace>
    <Tag>rdf:Description.prf:component</Tag>
  </Identification>
</Formatter>
```

**Figure 67: Adding the UAProf formatter to the formatters’ description**

A mechanism is needed for the AM to associate context profile formats with the appropriate formatter. In this scenario, the type of the profile can be recognized by the following: xmlns:prf=’http://www.openmobilealliance.org/tech/profiles/UAPROF/ccpschema-20021212#” and the root tag name <rdf: RDF>.

The pop up window in Figure 68 shows that the AM has successfully loaded the new context formatter for UAProf profiles.

**Figure 68: UAProf context formatter successfully loaded**

Figure 69 shows the formatted context using Protégé 4 [145]. This indicates that

1. The formatter works correctly. If the formatter did not work correctly, the Protégé editor will generate an exception and will not display the context as depicted by Figure 69.
2. The new context formatter was loaded successfully as indicated by Figure 68. Otherwise, getting the output of Figure 69 is not possible.
Chapter 7. Evaluation

3. The AM successfully recognised the new format and successfully determined the appropriate formatter to use. Formatting a UAProf context profile would have generated an unsupported profile format exception before adding the UAProf context formatter. Such exception was not generated and Figure 69 indicates that the AM was able to format the profile to ADIO using the appropriate formatter. If the wrong formatter was used, the generated formatted context will be empty and will not show any contextual information.

Supporting the new context profile (UAProf) was possible, and no coding maintenance was required to the AM implementation except from XML coding. The same process can be applied for any XML based context profile. Extending context formatting to new adaptation parameters as a result of extensions to a supported context format requires extending the corresponding XSLT formatter.

New context formats or the extension of existing ones may introduce new context or adaptation parameters that are not modelled by the internal context model, for the AM, this represents the ADIO. Hence extensions to ADIO may be required; this aspect is evaluated in Section 7.5.1.3.

Figure 69: The result of formatting displayed with Protégé
7.5.1.2 System Extensibility to New Adaptation Operations

This section evaluates system extensibility in terms of supporting a new adaptation operation/technique. However, aspects related to the Content Adaptor such as web services discovery and invocation are outside the scope of this thesis. Aspects that are addressed here relate to extending context management and decision taking.

As new content and device technologies and user preferences emerge, new content adaptation operations/techniques are introduced. Examples of state of the art multimedia adaptation techniques are presented in [170]. It is important to be able to extend the system to support such adaptation techniques. Such extensions require the following:

- **Extension to the adaptation operations context**: The Content Adaptor discovers adaptation operations and informs the AM with their availability. The AM maintains adaptation operations context to recognise the discovered adaptation mechanisms (which implement the adaptation operation/technique) that can be used.
- **Extension to ADT rules**: This is necessary to incorporate decisions that correspond to applying the new content adaptation technique. A new adaptation technique, implemented by a discovered adaptation mechanism, is supported if it corresponds to a decision in the adaptation decision rule base.

The extensibility to new adaptation techniques is assessed in the following two sections in terms of the two aspects mentioned above.

- **Extending Adaptation Operations Context**

This is achieved by adding an entry to the possibleOP.xml file. The entry includes the operation ID, description, name and input and output parameters.

The implemented AM prototype supports two adaptation operations, a MOV to 3GPP video transcoding operation and a MOV to MP3 video to audio conversion operation.

The aim is to extend the prototype to a new adaptation operation, which is a MOV to JPEG video to images key frame extraction. This operation could be useful in a number of situations including:

- The user device does not support video
- The user is interested in image content for key frames in the video
- To save resources such as battery resources or bandwidth.
In order to enable the AM to recognise this new adaptation operation upon discovering an adaptation mechanism that implements this operation, the following entry is added to the possibleOP.xml:

```xml
<OP>
  <OPDescription>Extracts key frames from video content</OPDescription>
  <OPName>KeyFrameExtraction</OPName>
  <InputParameters>
    <Format>mov</Format>
    <Modality>video</Modality>
    <Language>eng</Language>
  </InputParameters>
  <OutputParameters>
    <Format>jpeg</Format>
    <Modality>image</Modality>
    <Language>eng</Language>
  </OutputParameters>
</OP>
```

**Figure 70: Added adaptation operation to the possibleOP.xml file.**

The added entry allows the AM to recognise the discovered adaptation mechanisms that implement the newly added adaptation operation. The description of the added adaptation operation is depicted by Figure 70. The parameters specified in the description enable the AM to associate the discovered adaptation mechanisms to the operations they implement as described in Section 6.3.2.1. The name of the operation and the parameter names should be standardised, MPEG-7 and MPEG-21 provides classification schemes such as in [139] to provide a shared terminology in this respect.

- **Extending the Adaptation Decision Rule Base**

To make use of the new adaptation technique, the adaptation decision rule base needs to be extended to support it. The adaptation management system employs a rule based approach in the first layer of decision taking as described in Section 5.3. Rule based systems are recognised for their extensibility. In order to support the new operation, a new rule such as the following is added:

```
videoContent(contentX) ∧ videoLimited(deviceX) ∧ 
secondPreferredModality(image) → extractKeyFrames(contentX)
```

The screen shot in Figure 71 shows the AM output on the mobile device emulator. The discovered adaptation mechanism service that corresponds to the new adaptation operation is successfully applied; the adaptation rule that uses the new adaptation technique was invoked to match the user context (user device does not support video).
Extending the AM to the new adaptation operations required extension to the operations descriptions XML file and extension to the rule base in OWL. Minimal hard coding maintenance was required.

7.5.1.3 **System Extensibility to New Adaptation Scenarios**

The extensibility to new adaptation scenarios can be required for several reasons, such as:

- To satisfy new user requirements.
- To use new adaptation operations.
- To use newly introduced context (adaptation) parameters, for example the user location. New adaptation parameters can be introduced to the adaptation system when supporting a new context format or extending an existing one. For instance, extending adaptation to domain specific concepts defined by domain specific standards such as NewsML and SportML could be an example.

In all the scenarios mentioned above, extensibility can affect the following:

- The internal context model (ADIO), in terms of both low level adaptation parameters and high level adaptation concepts. The difference between the low level adaptation parameters and high level adaptation concepts is explained in Section 4.5.2.1.
- The adaptation decision rule base.

This section will evaluate extensibility to new adaptation scenarios using a demonstration scenario. The different steps that are required are explained.
• **New Adaptation Scenarios to Satisfy New User Requirements**

The scenario is as follows: The user is subscribed to a video content server; the subscription includes access to videos via the user’s home cinema system and mobile device (an adapted version is provided). For example, during a sporting event, the user watches the reports or the matches in their home cinema system to relax after a stressful working day. The user is interested in a wide range of sports, and would like to use their train journey at the end of the day to select which reports or matches they would like to view at home. In order to do so, the user requires a summary of the videos when accessed during their journey. The adaptation system does not currently support location (or activity) context, and this type of user preference.

New adaptation context that needs to be added is

- User-location: train.
- User activity: commuting.
- Threshold of video length upon which summarisation is decided.

Inferring high level context such as user activity or location is outside the scope of this research. If an intelligent entity that deduces such knowledge is not present, a preference such as summary-required can be specified by the user to indicate a situation where the user is not interested in long versions or would like to select among several long versions to view at home (or with a bigger display).

The context model used by the AM (ADIO) is implemented in OWL. A new high level adaptation concept is created which is defined as follows in DL notation:

```
summary_required ≡ (∃userLocation(train)) ∩
(∃videoSummaryLengthThreshold < 0)
```

Adding new low level context related to new adaptation scenarios will not affect adaptation rules that relate to the high level concept. For example, a new scenario could be that users would like to get the summary of the service during their lunch break; hence, the following new low level parameters need to be added:

- user-location: office
- user activity: lunch break
The context model is extended with the new context parameters without affecting ADT rules, only the high level concept summary-required is modified as follows:

\[
\text{summary - required} \equiv (\exists \text{userLocation(train)} \cup \\
\{ \exists \text{userActivity(lunchBreak)} \}) \cap \\
\{ \exists \text{videoSummaryLengthThreshold} < 0 \}
\]

The definition of the summary-required concept involves a pre-reasoning parameter: the videoSummaryLengthThreshold data property, which is calculated based on the original video content length and the user required video length. The required pre-reasoning function to calculate the value of videoSummaryLengthThreshold is added to the system by adding an entry to the pre-reasoning functions XML file as depicted by Figure 72. Pre-reasoning parameter function implementation is discussed in Section 6.3.2.4.

\[
\begin{align*}
\text{Function} & \\
\text{<Arg>} & \text{content1.videoLength</Arg>} \\
\text{<Arg>} & \text{user1.maxVideoLength</Arg>} \\
\text{<Op>} & - \\
\text{<Arg>} & \text{summary-required.videoSummaryLengthThreshold</Arg>}
\end{align*}
\]

Figure 72: Post fix expression described in XML for videoSummaryLengthThreshold

The adaptation decision rules need to be extended by adding new rules that corresponds to the new adaptation scenario. For example:

\[
\text{summary - required(contentX)} \rightarrow \text{summariseVideo(contentX)}
\]

If the new adaptation scenario is added because of the availability of a new adaptation operation, the system also needs to be extended to support the new adaptation operation; this aspect is evaluated in Section 7.5.1.2.

As Semantic Web technologies such as OWL can define context elements formally and unambiguously; extensions to such elements can be incorporated without causing redundancies or inconsistencies. Syntax wise, XML is designed to allow extensibility. As a consequence,
extending a context model based on XML and Semantic Web technologies is possible both syntactically and semantically.

The proposed approach to calculate pre-reasoning parameters is extensible without requiring any modifications to the AM implementation because an XML document that describes how the pre-reasoning parameters are calculated is loaded at start up and applied to context profiles during the context refinement stage (Sections 4.4.4.2). The adaptation system reads the XML file and performs the calculations described as postfix expressions. Details about pre-reasoning functions and describing them as postfix expressions are presented in Section 6.3.2.4.

Section 7.5.1 has demonstrated that AM extensibility to new context formats, new adaptation operations and new adaptation scenarios requires minimal coding maintenance. Only XML and OWL data is modified, for example using user interfaces such as Protégé and XML Spy [171]. Therefore, the contributions presented in this thesis satisfy the requirement of extensibility.

7.5.2 System Maintainability Evaluation

System maintenance could be required at different levels. As far as this thesis is concerned, system maintenance could be required for the following:

- Modifications to the internal context model (ADIO), either by extending it or changing the definitions of adaptation concepts.
- Modification of context formatting, in case modifications are applied to the context model (ADIO).
- Modifications to ADT rules, either for extending to new adaptation scenarios or modifying existing rules.

These aspects are evaluated in the following sections.

7.5.2.1 Internal Context Model (ADIO) Modification

The internal context model, ADIO, consists of low level and high level concepts as described in Section 4.5.2.1. Maintaining the ADIO model may be required for extending or updating its structure, for example to improve the performance of reasoning on the context model (ADIO).

The context model can be maintained using the Protégé IDE, which provides a good user interface to modifying ontologies. Reasoners are integrated with Protégé in order to check ontology consistency. Ontologies are known for their ease of maintainability and extensibility, therefore they have been chosen to implement the ADIO.
It is important to ensure that changes to the context model can be easily reflected in parts of the system that depend on it, specifically, context formatting tools and pre-reasoning function calculations. This aspect is evaluated in the following two sections.

7.5.2.2 Context Formatting Modification

If the structure of the internal context model, i.e. ADIO is modified, the context formatting tools have to be modified to reflect the changes. Maintenance is required at two levels:

- XML to ADIO formatting
- Pre-reasoning functions’ calculation

XML to ADIO Formatting Modifications

XML IDEs such as Stylus Studio [172] or XML Spy can be used to modify the context formatter to reflect the changes in ADIO. For example, if ADIO is to be changed to use data type properties instead of object properties to improve reasoning performance, this has to be reflected in the corresponding formatter.

As context formatting from XML to ADIO is implemented entirely in XSLT, no modifications to the AM coding that handles context formatters is required.

ADIO Pre-reasoning Parameters Calculation Modifications

During the context refinement step, ADIO pre-reasoning parameters (Section 4.4.4.2) define the types of high level adaptation concepts, such as the high level battery-limited-device concept.

The user remaining battery power preference can be specified in the user preferences, however, if it is not specified, a default value is assigned.

If modification is required to the default remaining battery power preference value, changes are made to a XML file that describes ADIO pre-reasoning functions as described in Figure 73 (for example from 15% to 20%).

![Figure 73: Pre-reasoning functions modifications](image-url)
If changes are applied to the ontology structure including simple changes such as renaming concepts, the changes will have to be reflected in the XSLT formatter as well as the pre-reasoning functions description file. For example, if the adaptation system is upgraded from adapting a single content to adapting multiple content items, i.e. the introduction of the concept service that represents the delivery of all content types, then pre-reasoning functions such as the function that calculates the memory-limited-device property value have to be updated as depicted by Figure 73.

7.5.2.3 Adaptation Decision Rules Modification

Rule based systems are known for their ease of maintainability compared to other decision technologies. The AM uses a rule based reasoning approach. The approach (Section 5.4) uses DL concepts to represent rule dependency, OWL as the syntax and protégé as the rules editor. The person modifying the rules needs to be familiar with the theory behind the used approach (i.e. using class sub-classing and equivalence between rule bodies, heads and rule classes); however, modifying or adding a rule is straightforward.

For ease of maintainability of the rule base, especially when extending the rule base, classes representing conditions that are likely to occur in several rule bodies and decisions likely to appear in several rule heads can be created. Therefore, modification to the conditions do not require modification to all the rules that use it, rather modification is required only to the class that represent the set of conditions or conclusions.

For example, the following adaptation and possible reasons for it could apply:

Adaptation: change modality from video to images, text and audio depending on the context. For example the service might be adapted to text and images only, audio only, audio and images only or text only.

Main Reason: this adaptation could be triggered by several reasons such as:

- Limited battery life to support the video content including where the service would exhaust the remaining charge to a level below the user’s remaining battery power preference.
- Limited buffer memory or processing power, for example due to running other services and applications that the user does not want to terminate.
- Limited device capability that does not support video content.
The summary-without-video-content class is defined as follows:

\[
\text{summary - without - video - content} \equiv \\
(\text{severelybatteryLimited} \cup \text{memoryLimited} \cup \text{videoLimited}) \cap \\
\text{serviceIncludesVideo}
\]

If the definition of this class needs to be changed, it only needs to be changed once and not at all rules; this eases rule base development. The changes could include adding more situations that correspond to this condition, or changing an existing situation (for example severelyBatteryLimited) in this condition (i.e. summary-without-video-content).

For example, the condition could be extended to cover the following situation: The user is roaming in a different network where data charges are higher than user's data charge preference.

\[
\text{summary - without - video - content} \equiv \\
(\text{severelybatteryLimited} \cup \text{memoryLimited} \cup \text{videoLimited} \cup \text{expensiveData}) \cap \\
\text{serviceIncludesVideo}
\]

A change may also be required if one of the situations that corresponds to this condition (summary-without-video-content) needs to be modified, such modifications will only need to be applied to the respective class. For example, changing the severelyBatteryLimited class by changing the threshold of remaining battery power according to which the reduction in allocated battery resource to the service is decided.

A rule that uses this class could be for example:

\[
\text{summary - without - video - content}(\text{serviceX}) \rightarrow \text{extractKeyImages}(\text{serviceX}) \land \\
\text{removevideo}(\text{serviceX})
\]

7.5.3 System Interoperability

System interoperability is very important in a heterogeneous domain such as the content and service adaptation domain. It is not possible to claim that the system is interoperable with all the tools and technologies that are available. However, interoperability can be ensured by extensibility and by implementing dominant interoperable standards and technologies. The
Chapter 7. Evaluation

The prototype implementation of the AM includes support for key technologies in the content and service adaptation domain such as:

- MPEG-7-MDS and MPEG-21-DIA: MPEG-7-MDS is the most recognised standard for multimedia content description, it enjoys wide support from the multimedia community. MPEG-21-DIA is built on top of MPEG-7 and provides tools to describe the usage environment, for example, user preference, device and network description, natural environment characteristics, etc. The importance of MPEG-21 DIA is reflected on the research work based on it such as [86] and [88].
- UAP: UAP is widely used in industry to describe device and connectivity characteristics.
- XML: context profiles including MPEG-7 and MPEG-21 are XML-based. The AM supports XML based profiles.
- Semantic Web technologies: the multimedia community recognises the importance of Semantic Web technologies [41] for future multimedia standards; the AM supports Semantic Web technologies such as the Ontology Web Language (OWL).

The support of MPEG-7 MDS, MPEG-21 DIA and UAP is not complete, what has been implemented aims to provide a proof of concept demonstration. The extension to the whole standards can be achieved in the same manner.

The most important enabler to interoperability is extensibility to new standards and tools. It is not possible to claim that a system is interoperable with all technologies, however, if a system is easily extensible to new standards and technologies, interoperability with the new standards and technologies is guaranteed. Section 7.5.1 evaluates system extensibility with regards to different aspects.

7.6 Conclusion

This chapter has presented an evaluation of the contributions proposed by the author in this thesis. Section 7.2 has presented different methods of evaluation that can be used in the domain of content and service adaptation management. These evaluation methods have been categorised and identified in main content and service adaptation systems in Table 15.

It is clear that evaluation depends on the research aims and objectives. The main aims and objectives of this research are to provide mechanisms to ensure extensibility with minimal maintenance of adaptation management systems to solve the adaptation domain complexity and heterogeneity problems. Therefore, the main method of evaluating the contribution of this thesis is white box demonstration scenarios. The term white box is used in “white box testing” [173], and
Chapter 7. Evaluation

refers to involving the knowledge of internal system structure in order to modify the system to evaluate for certain system characteristics. For example, in this thesis, the system has been modified as part of white box scenarios to evaluate system extensibility and maintainability.

The evaluation of the AM in terms of non performance based parameters, namely, extensibility, and ease of maintenance, has shown that the contributions of this thesis succeed to a great extent in achieving the aims and objectives of the research stated in Chapter 1. The success has not been merely measured in terms of the possibility of extensibility and maintainability, but rather, by the easiness at which extensibility and maintainability are ensured. For example, by identifying the number of steps required to extend the functionality of the AM to a new context extraction tool, a new adaptation operation or a new adaptation scenario.

It has been noted that the technologies used such as XML, ontology, OWL and rule-based reasoning are extensible and easily maintainable. However, the evaluation has assessed the extensibility and maintainability of the way in which these technologies have been employed.

With regard to interoperability, it is not possible to claim that the prototype is fully interoperable with all tools and technologies in the adaptation domain. However, key tools and technologies are supported such as XML, OWL, Ontology and MPEG-7/21. Interoperability with other existing or new technologies is ensured if the AM is extensible.

Other methods of evaluation such as experimental testing are also important and have been used in this chapter in order to prove the viability of the contributions in the sense that they can be applied in the real world. The experimental results have shown that the AM achieved reasonable performance, taking into account the complexity of the technologies used (such as DL reasoning) and the immaturity of the technologies and APIs. It has been concluded that the contributions can be deployed in the long term once the technologies and APIs used mature and more resources are available.
8 Conclusions

8.1 Summary

This section presents a summary of research issues and the mechanisms proposed to address them as follows:

- From studying the domain, it is evident that more content and service adaptation tools and technologies are continuously emerging including, adaptation operations and implementing adaptation mechanisms, metadata (context) description tools, adaptation decision techniques, device and networking technologies, etc.

- As a result of such a dynamic and heterogeneous domain, adaptation requirements are volatile. New user requirements emerge for new scenarios which are made possible by new device and networking features. Moreover, new content technologies are introduced to satisfy user requirements and to exploit the available enhanced device and networking features.

- In addition to classifying adaptation systems into the categories of content selection, content presentation adaptation and content adaptation systems, this thesis has classified state of the art systems into adaptation systems or adaptation management systems.

- This thesis has been concerned with two main areas of adaptation management, namely context assimilation and ADT. Other aspects of management include adaptation operations' invocation.

- The content adaptation domain complexity emanates from the heterogeneity of the adaptation requirements domain (user requirements, content requirements and device and access technologies) and the heterogeneity of the adaptation tools domain (adaptation decision techniques, context formatting tools, adaptation operations, etc.).

- It should be noted that adaptation management aspects are implicitly treated in some state of the art adaptation systems. However, it is necessary that the adaptation management aspect has to be explicitly treated in adaptation systems requirements, design and implementation.

- In this thesis, it has been emphasized that adaptation management has to be designed and implemented with two key principles in mind, specifically: extensibility and interoperability.
Conclusions

- Extensibility is more important because it ensures interoperability with new adaptation tools.

- Key enablers for extensibility implemented in the AM and discussed in this thesis included:
  - Modular design and separation of concerns: this is reflected in the design of the AM. For example, a two layer decision taking approach is proposed to clearly separate between decision taking, which represents actions that need to be taken on the content, and decisions relating to adapted content parameters setting such as setting the bit rate and resolution of the adapted content.
  - Clear interfaces between AM modules:
    - For example, the ADIO interfaces between the context assimilation subsystem and the ADT subsystem. ADIO provides an interface of high level context that reduces the impact of context changes on ADT and hence enables extensibility with minimal maintenance.
    - The context formatter uses XSLT as an interface between input context profiles and the internal context model. Any changes to the context are reflected in the XSLT transformer without requiring hard coding maintenance.
  - Implementing extensible, interoperable technologies such as:
    - XML technologies for syntax extensibility and interoperability.
    - DL for semantic extensibility, interoperability and ease of maintenance.
    - Rule based ADT.
    - MPEG-21 ADTE was used to decide on parameters of adapted content.
  - Avoiding hard coding and using generic implementations where possible:
    - For example, using start up files to load and add context formatters to the system. This approach is very extensible and requires minimal maintenance.
    - Loading adaptation operations with an XML based start-up file.
    - Using postfix expressions described in an XML based start-up file to execute pre-reasoning parameter functions. This approach is extensible and requires minimal maintenance. Editing or adding new pre-reasoning functions due to changing context or changing user requirements requires minimal maintenance as demonstrated in Chapter 7.

As mentioned in Chapter 1, this thesis has investigated the use of Semantic Web technologies, mainly DL for content adaptation. In this regard, two main contributions have been discussed in this thesis:
• DL based rule based reasoning: As mentioned above, rule based ADT is among the important extensible technologies that are used for the implementation of the AM. An important contribution of this thesis is a novel approach to perform DL based rule based reasoning. This approach has been proposed because of the increasing research and application related to Semantic Web technologies and in particular DL. DL reasoning is already an important and widely known reasoning technology in addition to rule based reasoning. Optimizations to DL reasoning and extensions to DL languages are being investigated; the future of DL is promising.

• The use of DL (implicitly ontology) to model context was discussed in Chapter 4. It was concluded that two different applications can be identified:
  o Adaptation domain ontology (ADO): this application is for describing context for content and service adaptation request annotation.
  o ADIO: this application emphasises more on reasoning efficiency which has implications on the structure of the ontology.
  o Both applications require manual construction of the ontology, and entail different design decisions. For example, reasoning efficiency is crucial for ADIO and hence a simple structure is adopted.

8.1.1 Prototype Implementation

A prototype of the AM was implemented and represents a central part of the content and service adaptation management test bed. The prototype and the test bed were used to demonstrate the feasibility of the contributions of the author and to evaluate the research against the aims and objectives set in Chapter 1.

It is important to note that evaluation has to be planned in accordance with the aims and the objectives of the research. The main aim of this research was to tackle the complexity of the adaptation domain by providing extensible and interoperable adaptation management. Thus, evaluation demonstration scenarios were developed to evaluate the extensibility and interoperability of the system to using new context formatting tools, extending adaptation decision rules, extending the context model to new context parameters or supporting a new adaptation operation. The result of the evaluation in Chapter 7 shows that the aims of this research have been satisfied to a great extent.
8.2 Future Work and Research Topics

Research in the area of content and service adaptation is rather vast. Considerable research has been conducted in this domain, part of which has been discussed in this thesis. This section outlines topics and areas that can follow from the contributions outlined in this thesis. Future directions with regard to the two main themes addressed in the thesis are:

- **Adaptation Management**
  - Enhancing system extensibility: In order to enhance extensibility, clear and generic interfaces need to be defined and ideally standardised for:
    - ADT: interfaces (or decision languages) for different types of adaptation decisions need to be defined. This is useful to interact between the managing adaptation decision entity and the specialised adaptation decision engines.
    - Interface between ADT and adaptation decisions execution: similarly to the point addressed above, such an interface, i.e. a specification language for the final adaptation decisions, would enhance interoperability with external content adaptation execution management entities.
  - Ease of development and maintenance: currently, editing the pre-reasoning functions is performed via XML. A user interface can be implemented to abstract the syntax of the postfix expressions that are used to specify the functions and hence reduce errors.
  - A mechanism to automate mapping assimilated context to adaptation rules would enhance maintainability. Such mechanism would enable retrieving the values of newly added (or existing) high level concepts to input them to the corresponding newly added (or existing) adaptation rules. If no such mechanism exists, hard-coding is required to enable such mapping.

- **DL Based Rule Based Reasoning**
  With regard to this contribution, possible future work includes:
  - Designing and implementing a variables management layer to generalize the use of the approach to more scenarios and not only scenarios where reasoning revolves around a central entity, which is the user in this thesis.
  - Implementing a user interface to edit the rules: currently, in order to write rules, the developer needs to be familiar with OWL, Protégé and the details of the proposed approach. Such complexity should be abstracted from the developer.
  - Implementing a mechanism to transform XML based rules or a simple rule format into rules of this approach and hence hide the complexity of the approach. It is noted that the complexity of the approach relates to the syntax, i.e. the unusual way of using OWL to represent rules as specified in Chapter 5.
8.3 Research Application

This section explores the use of the conclusions of this thesis in different domains. Learnt lessons can be applied to different domains as follows:

- **Non functional requirements**: whether in computer science or other fields, it is important to observe non functional requirements. Extensibility is the key requirement that needs to be observed in all stages of development. Extensibility ensures interoperability with future tools and technologies. However, extensibility is only useful if it is realised with relatively minimal maintenance. In fact, the principles outlined in this paragraph can be applied in a wide range of domains. For example, when designing and implementing (which can also mean manufacturing) an artefact, the extensibility principle is crucial to the adaptability and the relevance of the artefact or application to different deployment scenarios.

- **The management aspect**: when a variety of services (or tools) are available, being able to manage a number of services to offer a composite service can in itself provide a valuable service. The management process needs to be flexible in terms of using new services or replacing others, hence the principle of extensibility discussed above.

8.4 Epilogue

Content and service adaptation management systems manage the wide range of existing tools in order to satisfy the varying users' requirements with regard to adapting the wide range of multimedia content available to heterogeneous and complex delivery environments. Such management systems have to be designed with extensibility and interoperability in mind. Extensibility ensures interoperability by supporting existing and emerging tools.

Semantic Web technologies and rule based reasoning are among the tools used by an AMF to ensure extensibility and interoperability. Though having a different application compared to rule based reasoning, DL, a core Semantic Web technology, has become an established form of reasoning. Extending the use of DL to applications beyond ontology reasoning enables such applications to benefit from the wide research and application support DL enjoys.
Appendix A: MPEG Description Standards

MPEG-7 [174] and MPEG-21 [175] are International Standardization Organization (ISO) standards developed by the MPEG group. Previous MPEG standards dealt with multimedia encoding such as MPEG-1, MPEG-2, MPEG-4 and MPEG-3.

MPEG-7 is the first multimedia description standard from the MPEG group and was named the Multimedia Content Description Interface. It aims to describe multimedia content at different levels including audio and video features, content creation, content management, etc. MPEG-21 builds on MPEG-7 description tools and aims to provide description tools for multimedia delivery environments, DRM, etc.

• MPEG-7: The Multimedia Content Description Interface

The motivation behind introducing MPEG-7 is the sheer amount of multimedia content that is available. Without interoperable and standardised metadata, multimedia content management including storage, search and retrieval become very difficult and may render multimedia content useless. MPEG-7 aims to provide interoperable, application independent tools to describe multimedia content. MPEG-7 was developed by experts from various fields including broadcasting, electronic manufacturing, academia, telecommunication services, etc.

The audio visual description tools MPEG-7 provides are used to create descriptions (metadata files). MPEG-7 description tools may be used to describe information related to content creation, usage, storage, low level content features, semantics of information contained in the content, content structure and interaction with the user. The MPEG-7 standard has the following parts:

Part 1: MPEG-7 Systems [176]

This part of the standard describes the binary format to encode MPEG-7 descriptions and the architecture of terminals that process them. A terminal in MPEG-7 context is an entity that uses MPEG-7 descriptions and could be a standalone multimedia application or part of an application system. The terminal architecture specifies interfaces. However, their implementation is not defined and is left to application developers. Interfaces that are specified include the Binary Format Interface which describes the formats of binary representation of MPEG-7 descriptions and the Binary/Textual Decoder interface, to transform the binary data to MPEG-7 Descriptions.
Appendix A: MPEG Description Standards

Part 2: Description Definition Language (DDL) [177]

DDL is an XML based language and is a core component of MPEG-7. DLL specifies the syntax of MPEG-7 description tools by specifying the syntax of Descriptors and Description Schemes. DLL also specifies how extensions or modifications to Description Schemes may be added. Descriptors define the syntax and semantics of features such as colour, texture, motion, time, etc. Description Schemes specify the relationships between its elements which may be Descriptors or Description Schemes. DDL uses XML schema to define data type’s to describe audio visual features descriptions. Such data types include vectors, matrices, time data types, content locating data types, etc.

Part 3: MPEG-7 Visual [178]

Visual description tools include Descriptor and Description Schemes for the following visual content features: colour, texture, shape, motion, localisation, face recognition and others (i.e. features other than the previous six main features). Descriptor and Description Schemes use basic structures such as grid layout, time series, 2D-3D multiple views, 2D coordinates and temporal interpolation. A grid layout is used to split an image into a set of rectangular regions that are equal in size. This enables each region to be described separately, for example, in terms of its colour, shapes it contains and motion characteristics of its components. Motion Descriptors describe properties such as the pace of action in a video content. For example, a scene relating to goal scoring in a football match, or a car chase would be described as high activity scenes. On the other hand, a scene for a character sitting at a coffee shop would be described as low activity. Other Descriptors that can be used for visual content can be found in [178].

Part 4: MPEG-7 Audio [179]

MPEG-7 Audio consists of tools to describe low level audio features relating to the audio signal (spectral, parametric and temporal features) and high level audio features such as musical instruments timber, audio signature, melody description, sound recognition and indexing, spoken content description, etc. Audio signature description tools allow recognising audio signals and hence providing a unique audio content identifier. Melody description tools provide a mean to describe monophonic melodic information to enable melodic similarity matching. Sound recognition and indexing tools provide a mechanism to categorise and index sounds, for example, to segment and index sound tracks. Spoken content description tools provide a mechanism to describe the words spoken in an audio stream. More information about low level and high level Audio description tools are provided in [179].
Appendix A: MPEG Description Standards

Part 5: MPEG-7 Multimedia Description Schemes (MDS) [58]

While MPEG-7 Visual and Audio provide tools to describe visual and audio features, respectively, MPEG-7 MDS provides tools to describe generic multimedia content including video, images, audio and text. This includes describing information related content, (structure, model, semantics), how it is used, created and managed. MDS description tools are discussed in Section 2.5.2 and more information can be found at [58].

Part 6: MPEG-7 Reference Software: The Experimental Model (XM) [180]

MPEG-7 XM provides a simulation platform for Descriptors, Description Schemes, Coding Schemes (to encode and decode Descriptors and Description Schemes) and the DDL. XM application can be server or client applications. Client applications use the Descriptor and Description Schemes being tested and server application creates the Descriptor and Description Schemes being tested.

Part 7: MPEG-7 Conformance Testing [181]

MPEG-7 conformance testing specifies guidelines and procedures to test for the conformance of an implementation to the MPEG-7 standard. For example, to check for the conformance of generated description files to the syntax specified by MPEG-7.

Part 8: MPEG-7 Extraction and Use of Descriptions [182]

MPEG-7 Part 8 includes information on how MPEG-7 description may be extracted and used by applications. This includes information on Part 6 related to descriptions extraction implementation and alternative approaches. More information can be found at [182].

Part 9: MPEG-7 Profiles and levels [183]

MPEG-7 Part 9 defines profiles and levels, namely, the Simple Metadata Profile, which is used for describing media clips, the User description Profile, which is used to describe user preferences and related user context, and the Core Description Profile, which is used to describe general multimedia content.

Part 10: MPEG-7 Schema Definition [184]

This part of the standard contains all XML schema description tools specified in Parts 2, 3, 4 and 5.

Part 11: MPEG-7 profile Schemas [185]

This part contains the actual XML schema for the profiles defined in MPEG-7 Part 9.
Part 12: MPEG-7 Query Format [186]

MPEG-7 Part 12 describes the input and output interfaces between a database of MPEG-7 descriptions and a client application that needs to query the descriptions.

- MPEG-21

MPEG-21, termed the Multimedia Framework Standard, aims to provide mechanisms to enable different players/Users in the multimedia delivery chain to use multimedia content transparently across a wide range of network and devices. The concept of a User in the MPEG-21 framework denotes entities that interact with the MPEG-21 framework and use Digital Items. This includes individual users, communities, organizations, content producers, companies and governments. Digital Item is a central concept in MPEG-21 and denotes a digital object that embodies content and that has a standard representation, identification and description (metadata). A Digital Item constitutes the unit of distribution and transaction in the MPEG-21 framework. For example, a Digital Item could represent an album of images which contains images, text description of images and sound files that are used when the album is being navigated. Digital Item representation, identification and description are specified by different MPEG-21 parts as will be discussed. MPEG-21 parts are as follows:


Part 1 of the MPEG-21 standards introduces the aims and objectives of MPEG-21 Multimedia Framework and the requirements that need to be fulfilled in order to achieve them. In MPEG-21 Part-1, core concepts i.e. the Concept of User and Digital Item, explained in the previous section, are introduced.

Part 2: MPEG-21 Digital Item Declaration (DID) [188]

This part of the standard clearly specifies the components of the unit of transaction in the MPEG-21 Framework: Digital Items. The components of a Digital Item and how they are described needs to be generic in order to be able to cover the different content formats and types that are available. DID describes Digital Item components and how they relate to the containing Digital Item and to other components within the containing Digital Item using a model, a representation and a schema. The schema and representation specify the syntax of using the DID model in DID documents. The DID model describes a set of terms that can be used to define Digital Items, for example, Container, Item, Fragment, Component, Anchor, Descriptor, Condition, Selection, Choice, Annotation, Assertions, Resource, Predicate, etc. The semantics of these terms are defined in [188]. For example, the Component term is used to bind a Resource to its relevant Descriptors which contain information about the resource. Anchor is used to bind Descriptors to
a Fragment corresponding to a specific part of a resource. A Condition consisting of Predicates is used to describe elements that are optional. A Condition also links the optional resources to Selections, i.e. specific decisions, which determine their inclusion into the resource.

**Part 3: MPEG-21 Digital Item Identification (DII) [189]**

MPEG-21 DII specifies a mechanism for unique identification, including uniquely identifying Digital Items and parts within them and uniquely identifying Intellectual Property, Descriptions, types related to Digital Items, Containers, Components and / or Fragments within them. DII identifies, based on URIs, are included in DID documents.

**Part 4: MPEG-21 Intellectual Property Management and Protection (IPMP) [190]**

MPEG-21 IPMP builds on MPEG-4 IPMP and provides a standardized mechanism to retrieve IPMP tools and exchange messages between IPMP tools and terminals or other IPMP tools. IPMP aims to control the usage of Digital Items throughout their lifecycle. IPMP has two main components, the IPMP Digital Item Declaration Language (DIDL) and IPMP information schemas. IPMP DIDL allows providing a protected representation through encryption and digital signature derived from a DID for a Digital Item that requires protection. IPMP information schemas specify the syntax and structure of IPMP DIDL.

**Part 5: MPEG-21 Rights Expression Language (REL) [191]**

The REL and the Rights Data Dictionary (RDD), explained below, are very important tools for DRM. The REL is used to declare permissions on a Digital Item using terms from the RDD. REL and RDD aim to provide an interoperable mechanism to enable Digital Items to be published exchanged and distributed without violating digital rights. Examples of relevant resources to REL and RDD include movies, music, electronic books, software, etc. REL also aims to enable Users to specify digital rights on resources they own in order to, for example, protect personal data.

REL uses a simple model that consists of four entities and defines the relationships between them. The entities are Right, Principle, Resource and Condition. The entities relate to each other using the Issued to, Associated with and Subject to relationships. The REL model is depicted by Figure 74. Rights on specific resources are issues to a certain party (Principal) subject to some conditions.
Appendix A: MPEG Description Standards

Part 6: MPEG-21 Rights Data Dictionary [192]

MPEG-21 Part 6 defines a dictionary of terms to be used by the REL to specify permitted actions on Digital Items. The terms include *Adapt, Delete, Diminish, Embed, Enlarge, Modify, Print, Reduce, etc.*

Part 7: MPEG-21 Digital Item Adaptation (DIA) [59]

DIA provides tools to facilitate Digital Item adaptation to different delivery context characterised by different device and networking technologies. Important DIA tools for this thesis are the Usage Environment Description (UED) tools and the ADT framework.

UED tools build on MPEG-7 Descriptors and Description Schemes and extend enable extending descriptions to terminals, networks, usage environment, user preferences and characteristics, etc. Examples of concepts that can be described using UED tools are presented in Section 2.5.2.

In addition to UED, DIA provides ADT tools including AdaptationOoS and Universal Constraints Description UCD. This is discussed in Section 2.8.2.

Part 8: MPEG-21 Reference Software [193]

MPEG-21 Part 8 describes software that implements the other parts of MPEG-21. This includes information on the implemented modules and how they can be used. The software may be used to validate MPEG-21 descriptions or to demonstrate how certain specifications may be implemented.

Part 9: MPEG-21 File Format [194]

Part 9 of the MPEG-21 standard aims to define a file format that can define both binary data (such as MPEG-4) and textual data such as XML in order to represent the complex information contained in Digital Items.
Part 10: MPEG-21 Digital Item Processing (DIP) [195]

DIP provides tools to add declarations to allow Users to specify functionality as part of the Digital Item DID. As mentioned previously, Users (with capital U) in the context of MPEG-21 are entities that interact with the MPEG-21 framework and include content producers. A main component of DIP is the Digital Item Method (DIM). DIM provide tools to express operations to be applied on Digital Items at the level of DID. Such operations have to be implemented by the terminal that receives and processes the Digital Item.

Part 11: Evaluation Tools for Persistent Association Technologies [196]

This part of the MPEG-21 standard identifies requirement with regard to persistent identification and association of Digital Items and technologies that can be used such as file headers, watermarks and fingerprints. These technologies are explained in [196].

Part 12: MPEG-21 Resource Delivery Test Bed [197]

This part of the MPEG-21 Standard describes the test bed architecture for streaming applications. The test bed includes a media player, a server and a network simulator.

Part 13: MPEG-21 Scalable Video Coding (SVC) [198]

SVC aims to define a new scalable video coding format with high compression performance.

Part 14: MPEG-21 Conformance Testing [199]

This part of the standard provides tools to check for the performance of descriptions to the different parts specified by the MPEG-21 standard.

Part 15: MPEG-21 Events Reporting [200]

Event Reporting provides tools to specify, detect and act upon events. Such events can be related to how Digital Items are used. For example, an event could be the adaptation of a Digital Item or some resources it contains to a terminal. Events may not be related to Digital Items, for example, an event could be a specific transaction that occurred between two Users. MPEG-21 Event Reporting specifies events in terms of; the conditions that are attached to the occurrence of the event, the syntax and semantics of event information, the intended recipients and parameters related to the event report such as time, delivery mechanism, priority, etc.

Part 16: MPEG-21 Binary Format [201]

MPEG-21 Part 16 aims for efficient storage and interfaces of descriptions by specifying a binary format for MPEG-21 XML based descriptions.
Part 17: Fragment Identification of MPEG Resources [202]

MPEG-21 Part 17 aims to provide a mechanism to index and identify parts within resources. More information is available in [202].

Part 18: Digital Item Streaming (DIS) [203]

MPEG-21 DIS aims to provide mechanisms to stream Digital Items and defines the Bit-stream Binding Language (BBL) for this purpose.

Part 19: Media Value Chain Ontology Vision (MVCO) [204, 205]

The MVCO is the latest part of MPEG-21 under development. The aim is to construct an ontology that describes the media value chain. It currently consists of 60 classes and 20 properties implemented in OWL and will be accompanied with a Java API. The ontology describes Intellectual Property concepts (Work, Adaptation, Instance, Copy, Product, etc), user roles (Creator, Adaptor, Producer, etc), and actions (CreateWork, MakeManifestation, MakeCopy, Produce, etc) as described in Figure 75.

Figure 75: MVCO Structure
References


References


References


References


References


References


References


References


References


References