Policy-Based Management of Context-Aware Services in 3rd Generation Mobile Networks

Alvin Yew

Submitted for the degree of Doctor of Philosophy

Centre for Communication Systems Research
University of Surrey
October 2007
© Alvin Yew 2007
Abstract

Current 3rd Generation (3G) mobile networks have the ability to deploy and offer context-aware services. 3G service frameworks such as the Open Service Access and the Location Service provide context-aware services easy access to the user’s context. Service adaptation is fundamental to context-aware service provisioning and to the realisation of the Virtual Home Environment concept, which is now an integral part of the 3G service framework. The additional complexity in service adaptation inherent in context-aware services, however, requires a powerful and appropriate management framework to control service behaviour. Policy-based management, a proven solution in the network management field, is an appropriate framework to manage adaptable context-aware services as it reduces complexity through a rule-based approach of mapping events and conditions to management actions to achieve management goals. 3GPP has incorporated a policy management interface in the OSA specifications but has not provided any mappings or bindings to the underlying 3G network to achieve its realisation. This thesis proposes a framework that realizes the OSA policy management API while maintaining strict compliance to the plethora of 3G specifications regarding the underlying 3G network. It explores the various operating requirements for deploying the policy-based management framework including the various 3G business models, the VHE concept, and the context-aware service adaptation requirements. Solutions to managing and enforcing multi-dimensional context-aware service adaptation are also presented in this thesis. A policy information model to aid the creation of context-aware service adaptation policies was designed and developed with strict compliance to the various 3G specifications. A prototype of the framework was implemented as a proof of concept and its evaluation is provided with an empirical analysis of its performance.
# Contents

Abstract........................................................................................................................................... ii
List of Figures........................................................................................................................................ v
List of Tables....................................................................................................................................... ix
Acknowledgements............................................................................................................................. x
List of Abbreviations.......................................................................................................................... xi
List of Publications............................................................................................................................. xiii

1. INTRODUCTION ......................................................................................................................... 1
   1.1. The Deployment of 3rd Generation Mobile Communication Networks............................. 1
   1.2. Context-aware Services ........................................................................................................... 2
   1.3. Policy-Based Management .................................................................................................... 3
   1.4. Motivation ............................................................................................................................... 4
   1.5. Aims and Scope ....................................................................................................................... 4
   1.6. Contributions .......................................................................................................................... 5
   1.7. Thesis Structure ....................................................................................................................... 6

2. MANAGEMENT OF ADVANCED SERVICES IN 3G NETWORKS ............................................ 7
   2.1. Introduction .............................................................................................................................. 7
   2.2. The Virtual Home Environment Concept .............................................................................. 7
      2.2.1 Home Environment ........................................................................................................... 8
      2.2.2 Service Performance Preferences ..................................................................................... 8
      2.2.3 Networks and Terminals .................................................................................................. 8
   2.3. The Open Service Access ........................................................................................................ 9
   2.4. Related Service Architectures for Adaptable Service Provisioning ...................................... 10
      2.4.1 MOBIVAS ......................................................................................................................... 10
         2.4.1.1 General Architecture of MOBIVAS ........................................................................... 11
         2.4.1.2 Discussion on MOBIVAS ....................................................................................... 14
      2.4.2 ADMITS ............................................................................................................................ 14
         2.4.2.1 Functional and Technical Description of the ADMITS Architecture ....................... 14
         2.4.2.2 ADMITS and Adaptation ....................................................................................... 17
         2.4.2.3 Discussion ................................................................................................................. 18
      2.4.3 ISIS ................................................................................................................................... 18
         2.4.3.1 General View of the Architecture ............................................................................. 18
         2.4.3.2 The ISIS Adaptation Framework ............................................................................. 20
         2.4.3.3 Discussion ................................................................................................................. 22
      2.4.4 GIGAMOBILE .................................................................................................................. 22
2.4.4.1 Functional Description of GigaMobile Architecture ........................................................... 23
2.4.4.2 Technical Description of GigaMobile Architecture ......................................................... 24

2.5. General Discussion on Related Service Architectures ............................................................. 25

3. REQUIREMENTS FOR POLICY-BASED MANAGEMENT OF CONTEXT-AWARE SERVICES IN 3G NETWORKS ................................................................. 26

3.1. Introduction ................................................................................................................................. 26
3.2. The Impact of the VHE and Business Models on Policy-Based Management ..................... 26
  3.2.1 3GPP VHE Technical Roles ...................................................................................................... 27
  3.2.2 3G Business Models ............................................................................................................. 28
  3.2.3 Other Related Business Models ........................................................................................... 29
3.3. Factors for Placing the Policy Based Framework ................................................................. 31
  3.3.1 Framework External to the Network Operator ........................................................................ 31
  3.3.2 Framework at the Network Operator .................................................................................. 32
3.4. Using Policies to Manage Adaptation for Context-Aware Services .................................... 33
  3.4.1 Situations When Policies Can Be Enforced ........................................................................ 33
  3.4.2 Influencing the User's Context ............................................................................................. 34
3.5. Managing and Resolving Policies Relating to Different Contexts ........................................ 35
  3.5.1 Network Quality of Service ................................................................................................ 35
    3.5.1.1 Terminal Capabilities ......................................................................................................... 36
    3.5.1.2 VHE Service QoS requirements ........................................................................................... 36
  3.5.1.3 User Preferences ................................................................................................................. 37
  3.5.1.4 Network Policy .................................................................................................................. 37
  3.5.1.5 Prioritising Services ........................................................................................................... 38
3.6. Managing Multi-Dimensional Context-Aware Service Adaptation .................................... 41
3.7. Discussion and Summary .......................................................................................................... 42
  3.7.1 Requirements ......................................................................................................................... 42
  3.7.2 Chapter Contributions and Proposals .................................................................................. 43

4. POLICY-BASED MANAGEMENT FRAMEWORK FOR CONTEXT-AWARE SERVICES IN 3G NETWORKS .......................................................................................................................................... 44

4.1. Introduction ................................................................................................................................. 44
4.2. Basic Architecture of a Policy-Based Management System ................................................... 44
  4.2.1 Policy Input Tool ................................................................................................................... 45
  4.2.2 Policy Decision Point/Function (PDF) ................................................................................. 45
  4.2.3 Policy Enforcement Point/Function ..................................................................................... 45
  4.2.4 Policy Repository .................................................................................................................. 45
  4.2.5 Interaction between the Policy Decision Function (PDF) and the Policy Enforcement
      Point (PEP) ..................................................................................................................................... 46
4.3. Interfacing the Policy-based Management System with OSA Framework ......................... 46
  4.3.1 The Policy Provisioning Service Capability Feature .......................................................... 47
4.3.2 The Policy Evaluation Service Capability Feature ........................................................... 49

4.4. Static Policy Conflict Detection ......................................................................................... 49

4.5. Context Control Function and Context Gathering Function .............................................. 54

4.6. Managing and Controlling Network QoS through Policies ............................................. 54

4.6.1 The Go Interface Architecture Model .............................................................................. 56

4.6.2 Other Benefits regarding the Go Interface ....................................................................... 59

4.7. Policy-based management and 3GPP's Location Service (LCS) ....................................... 59

4.7.1 OSA Mobility API ............................................................................................................ 61

4.7.2 The LIF Mobile Location Protocol .................................................................................. 61

4.7.3 Comparison between OSA Mobility API and LIF-MLP .................................................. 65

4.7.4 Industrial Factors Influencing the Usage of the OSA Mobility API and the MLP .......... 65

4.7.5 Discussion on the Deployment of Location-Aware Services in 3G Networks .................. 66

4.8. Extracting Terminal Capabilities via the 3G network ....................................................... 68

4.8.1 CC/PP ............................................................................................................................... 68

4.8.2 UAProf ............................................................................................................................ 69

4.8.3 Processing CC/PP and UAProf ...................................................................................... 70

4.8.4 Discussion on the Usage of the OSA Terminal Capabilities API and the JSR-188 CC/PP processor ......................................................................................................................... 72

4.9. The Generic User Profile and the Service Control Function .............................................. 72

4.10. Discussion and Summary ................................................................................................. 72

5. MODELLING POLICIES FOR CONTEXT-AWARE SERVICE MANAGEMENT IN 3G NETWORKS ................................................................................................................................. 74

5.1. Introduction ......................................................................................................................... 74

5.2. Modelling Policies through the Policy Core Information Model ...................................... 74

5.3. Policy Information Model for Network Quality of Service ............................................. 75

5.4. Context-related Policy Condition Classes ......................................................................... 78

5.4.1 The Abstract ContextCondition Class ............................................................................ 79

5.4.2 The ServiceAccessCondition Class ............................................................................... 79

5.4.3 The GeographicalLocationCondition Class ................................................................... 80

5.4.4 Terminal Capabilities ....................................................................................................... 81

5.5. The SubscriberIDVariable and SubscriberIDCondition Classes ..................................... 82

5.6. Discussion and Summary ................................................................................................... 82

6. CASE STUDY AND PROTOTYPE EVALUATION ................................................................ 84

6.1. Introduction ......................................................................................................................... 84

6.2. Scenario Deployment of the Content Adaptation Server Prototype .................................... 85

6.3. The Building Blocks of Automated Content Adaptation to Terminal Capabilities ............ 86

6.3.1 Mechanisms for Obtaining Device Capabilities ............................................................. 87

6.4. Description of the Policy-Managed Content Adaptation Server Prototype ....................... 87
Figures

Figure 2.1 The Open Service Access Framework ...................................................................................... 10
Figure 2.2 MOBIVAS Service Provision Platform .................................................................................. 12
Figure 2.3 The ADMITS Architecture ..................................................................................................... 15
Figure 2.4 ADMITS End-to-end Metadata Driven Adaptation Scenario .................................................. 16
Figure 2.5 ISIS End-to-End Architecture .................................................................................................. 19
Figure 2.6 ISIS Adaptation Framework ................................................................................................... 20
Figure 2.7 GigaMobile High-Level Functional Architecture ....................................................................... 24
Figure 3.1 Service Access in the Virtual Home Environment .................................................................. 27
Figure 3.2 UMTS Forum Network Operator Centric Business Model ................................................... 29
Figure 3.3 UMTS Forum Content Aggregator Centric Business Model ..................................................... 29
Figure 3.4 UMTS Forum Content Provider Centric Business Model ......................................................... 29
Figure 3.5 Business Roles within IST VESPER ....................................................................................... 30
Figure 3.6 Deployment of the VESPER Middleware (Logical View) ....................................................... 30
Figure 3.7 Business Roles in IST Magnet .................................................................................................. 31
Figure 3.8 Policies in Action During a Service Session ............................................................................ 35
Figure 3.9 Example of Categorising Different Services by the User ....................................................... 39
Figure 3.10 Decision Process for Determining the Network QoS for a VHE Service ............................... 40
Figure 3.11 Case Study of Deciding the Amount of Bandwidth Required .................................................. 40
Figure 3.12 Policy Decision Factors regarding Service Adaptation ......................................................... 42
Figure 4.1 IETF Policy-Based Management Framework ........................................................................ 45
Figure 4.2 Functional Architecture of the Proposed Policy-Based Management Framework ................. 47
Figure 4.3 OSA Policy Management Information Model .......................................................................... 48
Figure 4.4 OSA Policy Provisioning API .................................................................................................. 49
Figure 4.5 Scenario for Potential Policy Conflict ..................................................................................... 50
Figure 4.6 Inserting Policies into the System via the OSA APIs ............................................................... 53
Figure 4.7 Go Interface Architecture Model ............................................................................................ 56
Figure 4.8 General Arrangement of the LCS ........................................................................................... 60
Figure 4.9 Overview of the MLP Framework ............................................................................................ 63
Figure 4.10 The OGC GeoMobility Server ................................................................................................. 66
Figure 4.11 JSR-188 Java APIs for Discovering Terminal Capabilities ............................................... 71
Figure 4.12 Bindings between the Policy Management Framework and the Context Gathering and Control Functions in a 3G Network ................................................................. 73
Figure 5.1 A Subset of the Core Policy Classes Defined in PCIMe ......................................................... 75
Figure 5.2 Resource Reservation with Service-based Local Policy ......................................................... 76
Figure 5.3 The Policy Condition Class Hierarchy .................................................................................... 78
Figure 5.4 Graphical Explanation of the HorizontalLocationClass Properties .................................. 81
Figure 6.1 Scenario deployment of policy-based management in a 3G/B3G network .......... 86
Figure 6.2 The Framework Prototype within the Content Adaptation Capability Server ....... 88
Figure 6.3 Example of an XML Content Skeleton ............................................................... 89
Figure 6.4 Example of a Partially Adapted XML Content Skeleton ................................... 90
Figure 6.5 State chart of the Controller Servlet ................................................................. 91
Figure 6.6 Design of the Policy Execution Logic ................................................................. 94
Figure 6.7 Sequence diagram of the Policy Execution Logic Initialisation ......................... 94
Figure 6.8 State chart of the Policy Execution Logic ......................................................... 96
Figure 6.9 Class Diagram of Important Parts of the Developed JSR-188 Processor ........... 99
Figure 6.10 XML Stylesheet for XHTML User Agents ....................................................... 100
Figure 6.11 Transformed XHTML Content from XML Skeleton .................................... 100
Figure 6.12 Frequency of Occurrences for Specific UAPref Capabilities ......................... 104
Figure 6.13 Experiment Physical Setup ............................................................................ 105
Figure 6.14 Average Response Times of Requests ......................................................... 105
Figure 6.15 Framework Throughput Performance ............................................................ 105
Tables

Table 2.1 VHE Definitions ................................................................. 7
Table 4.1 TpCommonExceptions Definition ........................................... 54
Table 4.2 Constants Associated with TpCommonExceptions ...................... 54
Table 4.3 UMTS QoS Classes .............................................................. 55
Table 4.4 Set of DTDs defined for the MLP Element Layer ...................... 62
Table 4.5 Basic MLP Services ............................................................. 63
Table 4.6 Mapping between MLP and LCS ............................................. 64
Table 4.7 OSA Tp/TerminalCapabilities datatype ...................................... 72
Table 5.1 Mapping between UAProf attribute values to PCIMe datatypes .... 82
Table 6.1 Comparison between DELI and JSR-188 Processors ................... 97
Table 6.2 Statistical Analysis of UAProf Profiles .................................... 103
Table A.1 UAProf Terminal Hardware Capabilities ................................ 119
Table A.2 UAProf Terminal Software Capabilities ................................ 121
Table A.3 UAProf Terminal Network Capabilities ................................ 125
Table A.4 UAProf Terminal Browser User Agent Capabilities .................. 125
Table A.5 UAProf Terminal WAP Characteristics ................................ 127
Table A.6 UAProf Terminal Push Characteristics ................................ 128
Acknowledgements

I would like to thank both of my supervisors Dr Antonio Liotta and Prof George Pavlou for their immense support and guidance over the course of the PhD. Prof Pavlou gave me a chance to undertake this work and I am very grateful to him. Dr. Liotta has been a pillar of support over the last few years and I owe so much to him. Such is my gratitude that I consider him more as a very good friend instead of my boss or co-supervisor. It has been an honour for me to have worked along both their sides.

I am also very grateful to my family, my inner circle, for always being there for me. Mum, Dad, Eddie, Leslie and Christine, thank you so much for just being there and supporting me through all the difficult times. It was difficult on my parents that I was away for the last 10 years and I missed them and really appreciate the sacrifices that they have made over the last decade.

I must make a mention to some very important friends who have made the PhD life interesting and fun. Firstly, the old CCSR networks group and other misfits at the University of Surrey: Sunny, Sam, Hamed, Stelios, Paris, Stéphane, Khaldoon, Carmelo, Dimitris (aka Jimbo), Panos, Ning Wang, Marinos, Siva, Small Stelios, Apostolis, and Haitham etc. Secondly, the friends I made at the University of Essex where I worked: Zhao Ke, Marco, Laura, Marcos, Raul, Chigo, Tola and Ling.
**Abbreviations**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3G</td>
<td>3rd Generation (Mobile Communication Network)</td>
</tr>
<tr>
<td>3GPP</td>
<td>3rd Generation Partnership Project</td>
</tr>
<tr>
<td>AF</td>
<td>Application Function</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>BS</td>
<td>Bearer Service</td>
</tr>
<tr>
<td>CC/PP</td>
<td>Content Capabilities/Profile Preferences</td>
</tr>
<tr>
<td>CIM</td>
<td>Common Information Model</td>
</tr>
<tr>
<td>CNF</td>
<td>Conjunctive Normal Form</td>
</tr>
<tr>
<td>COPS</td>
<td>Common Open Policy Service</td>
</tr>
<tr>
<td>COPS-PR</td>
<td>COPS Usage for Policy Provisioning</td>
</tr>
<tr>
<td>DiffServ</td>
<td>IP Differentiated Services</td>
</tr>
<tr>
<td>DMTF</td>
<td>Distributed Management Task Force</td>
</tr>
<tr>
<td>DNF</td>
<td>Disjunctive Normal Form</td>
</tr>
<tr>
<td>DOM</td>
<td>Document Object Model</td>
</tr>
<tr>
<td>DTD</td>
<td>Document Type Definition</td>
</tr>
<tr>
<td>GGSN</td>
<td>Gateway GPRS Support Node</td>
</tr>
<tr>
<td>GMLC</td>
<td>Gateway Mobile Location Centre</td>
</tr>
<tr>
<td>GPRS</td>
<td>General Packet Radio Service</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>GUP</td>
<td>Generic User Profile</td>
</tr>
<tr>
<td>HE</td>
<td>Home Environment</td>
</tr>
<tr>
<td>HE-VASP</td>
<td>Home Environment Value-Added Service Provider</td>
</tr>
<tr>
<td>HLR</td>
<td>Home Location Register</td>
</tr>
<tr>
<td>HSS</td>
<td>Home Subscriber Server</td>
</tr>
<tr>
<td>HTTP</td>
<td>HyperText Transfer Protocol</td>
</tr>
<tr>
<td>ICID</td>
<td>IMS Charging Identifier</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>IMS</td>
<td>(3GPP) IP Multimedia Subsystem</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>ISP</td>
<td>Internet Service Provider</td>
</tr>
<tr>
<td>IST</td>
<td>Information Society Technologies</td>
</tr>
<tr>
<td>ITU</td>
<td>International Telecommunication Union</td>
</tr>
<tr>
<td>JSR</td>
<td>Java Specification Request</td>
</tr>
<tr>
<td>LCS</td>
<td>(3GPP) Location Service</td>
</tr>
<tr>
<td>LIF</td>
<td>Location Interoperability Forum</td>
</tr>
<tr>
<td>LIMS-IWF</td>
<td>Location IP Multimedia Subsystem Interworking Function</td>
</tr>
<tr>
<td>MLP</td>
<td>Mobile Location Protocol</td>
</tr>
<tr>
<td>MSC</td>
<td>Mobile Switching Centre</td>
</tr>
<tr>
<td>OGC</td>
<td>Open Geospatial Consortium</td>
</tr>
<tr>
<td>OMA</td>
<td>Open Mobile Alliance</td>
</tr>
<tr>
<td>OSA</td>
<td>Open Service Access</td>
</tr>
<tr>
<td>P2P</td>
<td>Peer to Peer</td>
</tr>
<tr>
<td>PBM</td>
<td>Policy-Based Management</td>
</tr>
<tr>
<td>PCIM</td>
<td>Policy Common Information Model</td>
</tr>
<tr>
<td>PCIMe</td>
<td>Extensions to Policy Common Information Model</td>
</tr>
<tr>
<td>P-CSCF</td>
<td>Proxy Call Session Control Function (in IMS)</td>
</tr>
<tr>
<td>PDF</td>
<td>Policy Decision Function</td>
</tr>
<tr>
<td>PDP</td>
<td>Packet Data Protocol</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>PEL</td>
<td>Policy Execution Logic</td>
</tr>
<tr>
<td>PEP</td>
<td>Policy Enforcement Point</td>
</tr>
<tr>
<td>PIB</td>
<td>Policy Information Base</td>
</tr>
<tr>
<td>PLMN</td>
<td>Public Land Mobile Network</td>
</tr>
<tr>
<td>PMD</td>
<td>Pseudonym Mediation Device</td>
</tr>
<tr>
<td>PPR</td>
<td>Privacy Profile Register</td>
</tr>
<tr>
<td>PSE</td>
<td>Personal Service Environment</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>RAB</td>
<td>Radio Access Bearer</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>RSVP</td>
<td>Reservation Protocol</td>
</tr>
<tr>
<td>RTSP</td>
<td>Real Time Streaming Protocol</td>
</tr>
<tr>
<td>SAX</td>
<td>Simple API for XML</td>
</tr>
<tr>
<td>SBLP</td>
<td>Service-Based Local Policy</td>
</tr>
<tr>
<td>SCS</td>
<td>Service Capability Server</td>
</tr>
<tr>
<td>SDP</td>
<td>Session Description Protocol</td>
</tr>
<tr>
<td>SGSN</td>
<td>Serving GPRS Support Node</td>
</tr>
<tr>
<td>SIP</td>
<td>Session Initiation Protocol</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreement</td>
</tr>
<tr>
<td>SMS</td>
<td>Short Message Service</td>
</tr>
<tr>
<td>SOAP</td>
<td>Simple Object Access Protocol</td>
</tr>
<tr>
<td>SPKI</td>
<td>Simple Public Key Infrastructure</td>
</tr>
<tr>
<td>UAProf</td>
<td>User Agent Profile</td>
</tr>
<tr>
<td>UE</td>
<td>User Equipment</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>VAS</td>
<td>Value-Added Services</td>
</tr>
<tr>
<td>VASP</td>
<td>Value-Added Service Provider</td>
</tr>
<tr>
<td>VHE</td>
<td>Virtual Home Environment</td>
</tr>
<tr>
<td>VoIP</td>
<td>Voice over IP</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium</td>
</tr>
<tr>
<td>WAP</td>
<td>Wireless Application Protocol</td>
</tr>
<tr>
<td>W-HTTP</td>
<td>Wireless Profile HTTP</td>
</tr>
<tr>
<td>WLAN</td>
<td>Wireless Local Area Network</td>
</tr>
<tr>
<td>WSP</td>
<td>Wireless Session Protocol</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
<tr>
<td>XHTML</td>
<td>eXtensible HyperText Markup Language</td>
</tr>
<tr>
<td>XSLT</td>
<td>eXtensible Stylesheet Language Transformation</td>
</tr>
</tbody>
</table>
Publications


1. Introduction

1.1. The Deployment of 3rd Generation Mobile Communication Networks

The introduction of 3rd Generation (3G) mobile communication networks at the beginning of this decade was a significant technical milestone for the telecommunications industry. It sparked a 3G spectrum bidding frenzy amongst many network operators as they anticipated that the next generation of networks would herald a golden opportunity for profits based on the numerous optimistic predictions in subscriber numbers. Operators that missed out on the Global System for Mobile Communications (GSM) spectrum rights in a period when there were competing and alternative mobile cellular access technologies were keen not to "miss the boat" this time around as well.

The 3G subscriber uptake immediately after the launch of the technology, however, was slow in the UK despite the tremendous hype from the technology media and network operators -- there were only 410,600 3G subscribers after the first year of operation (March 2004) in contrast to GSM which had an estimated 40 million subscribers in the UK in 2000 [1][2]. Although the technological difference between 2G and 3G mobile cellular networks was significant, the most important being the increase in maximum bit rate supported over the radio air interface, it was apparent from the lack of subscribers that the majority of the general public viewed 3G networks as mere bells and whistles i.e. it was attractive but superfluous to their needs.

The underwhelming public response to 3G networks can be partly attributed to the failure of the network operators to create a demand for the technology. Network operators believed that the solution to this problem was to find a "killer application" which required 3G technology with video conferencing/calls being the most often used example in early marketing media. This "killer application" approach was used partly because of the phenomenal and unexpected success of the money spinning Short Message Service (SMS) of GSM networks. By using the basic logic that since SMS played a huge part in the success of GSM, network operators felt that 3G networks similarly required a "killer application" to champion its cause.

One major development that 3G network operators had to deal with was the increasing importance and popularity of the Internet and Internet Protocol (IP) networks. The National Statistics for the percentage of households with Internet access in the UK was 57% as of April 2006 [3]. Content that was important for users on the move was seen as a possible killer application for 3G networks. Some network operators began to work with content providers to develop content that they deemed pertinent to mobile users. However, there
remains the problem of content exclusivity. With the ubiquitous availability of wireless local area network (WLAN) access, potential 3G users can still access similar content in an Internet café, Internet hotspot, hotel room, or airport lounge. 3G network operators must learn to co-exist with current alternative IP network access providers, while providing added value to their product as there are currently user terminals which incorporate both WLAN and 3G network access capabilities. To maintain an edge over other access technologies, 3G operators need to develop service and content whereby key 3G network characteristic features are required, e.g. ability to deliver bandwidth intensive content and services, and uninterrupted service sessions through cellular handover.

This has led to a surge of research interest towards advanced service provisioning concepts and service creation frameworks aiming to help service providers to exploit the advantages of the technology. One important research initiative that influenced the 3rd Generation Partnership Project (3GPP), the body in charge of 3G standards, was the Parlay Group's Parlay Application Programming Interfaces (APIs) in 1998. The Parlay APIs aimed at providing services with an object-oriented service control interface that is independent of the underlying communications technologies [102][103]. As the Parlay APIs' scope was providing a unified API which could work over heterogeneous networks, the 3GPP has in turn introduced the Open Service Access (OSA) [4] which contains APIs specific to only 3G networks and technologies. The OSA enables the hosting of service applications outside 3G networks while giving them controlled access to the network resources and 3G network capabilities. 3GPP's virtual home environment (VHE) [5] concept also plays an important role and will be implemented in 3G mobile communications through OSA and other 3G specifications. The VHE concept allows users to be consistently presented with the same personalised features, user interface personalisation, and services regardless of network, terminal, and location.

1.2. Context-aware Services

One unique feature of cellular networks is that it can provide very accurate location positioning of the user without the usage of a GPS receiver compared to other access technologies such as WLAN. This factor, coupled with the ability of 3G networks to provide various guaranteed levels of network Quality of Service (QoS), has led to context-aware services mooted as a possible "killer application" for 3G networks. Context-aware services possess the capability of sensing and receiving feedback from the environment (or context) in which they operate, thus offering to the end user benefits such as greater responsiveness, flexibility, adaptability and customisability. Not only are context-aware services able to adapt their content and performance during the service session, they can also be activated
when there is a relevant change in context i.e. service activation. For example, a location-aware service can push content to the user when the user enters a particular zone in the city. The intrinsic complexities of the above mentioned example highlights that context-aware service provisioning requires an advanced service platform, framework or middleware to manage and co-ordinate service adaptation and operation.

1.3. **Policy-Based Management**

Policy-based Management is a novel approach to management that can be used to manage the collective behaviour of many individually manageable entities in a system in a coordinated manner, so as to ensure that the system's behaviour is consistent with high-level management objectives and goals. Strassner states that “Policy Management is the usage of rules to accomplish decisions” [6]. The Internet Engineering Task Force (IETF) defines policy network management in the following manner: *Network policies are a set of rules to administer, manage, and control access to network resources* [7].

Policies are created in a policy management tool by the system administrator and then stored in a policy repository so as to allow for easy access and retrieval. The PBM system should address conflict detection and resolution whenever a new policy is created (static detection/resolution) or during its execution (dynamic detection/resolution), although the latter is an extremely difficult problem [101]. The decision of the policy to apply in the system (so as to fulfill the management goals) is undertaken by a Policy Decision Function (PDF). The PDF retrieves the policy from the policy repository and then sends it to the Policy Enforcement Points (PEP) that interprets and enforces the policy. Policies may be implemented through a rule-based strategy, where conditions and events in the system are mapped onto actions to be performed by the managed entity. As such, a well-defined information model describing policy attributes such as applicable conditions, required actions, and relationships with other policies is necessary to ensure that policy data is represented in the policy repository in a manageable fashion. The generic Policy Core Information Model (PCIM) [8] is usually adopted as a starting point for models applicable to specific domains. PBM has been applied to QoS management of IP networks [9][10], routing [11], security [12][13], and programmable networks [14].

Policy-based management provides the following benefits [6]:

- Simplifying device, network, and service management.
- Requiring fewer engineers to configure the network.
- Defining the behaviour of a network or distributed system.
- Managing the increasing complexity of programming devices.
- Using business requirements and procedures to drive the configuration of the network.
1.4. Motivation

An inherited benefit of using policy-based management is that it facilitates the deployment of numerous adaptation strategies for context-aware services. This includes making the context-aware service adapt to suit the context; influencing or adapting the context's constituents if possible (e.g. change the network QoS, use more base stations to discover the user's location more accurately) to suit the service's requirements or user preferences; and a combination of the previous two approaches to derive maximum user satisfaction. As there may be many independent constituents that constitute a context, the problem of finding the optimal amount of influence to apply on each constituent in order to collectively influence the context becomes multi-dimensional. Furthermore, the sensitivity of different context constituents may significantly differ. For example, a location-based service may require the user's location to be determined extremely accurately but may be less sensitive to network QoS. This problem has not been tackled yet in the 3G context as of the writing of this thesis. Policy-based management provides natural means for addressing a number of issues and requirements related to advanced service management and context-aware services, which are significantly based on rule definition and manipulation. Policies are a suitable tool for a high-level, technology-independent service configuration and user preference specification. Through rule-based policies, users can formally specify their requirements or preferences and services can be configured accordingly. For example, a service policy could be to assign a higher priority to London customers that are connected in the London area by requesting appropriate QoS levels from the network. Similarly, as part of its profile specification, a user may request a location-dependent QoS level (e.g. a higher QoS level when he or she is connected in the London area).

1.5. Aims and Scope

3GPP has included a policy management API in its OSA framework. It has not, however, provided the specifics on the realisation of the policy management framework that will perform the tasks associated with this API. The main motivation for the body of research in this thesis is to propose the necessary features and mechanisms that should be in place in a 3G network to allow the policy-based management of context-aware services; the bigger picture being that this will in turn allow the deployment of advanced services which will give 3G networks an edge over other competing access network technologies. The main hypothesis put forward in this thesis is that it is possible to deploy a policy management framework in a 3G network to manage the adaptation of context-aware services without major modifications to current 3G specifications and business models. The
features and mechanisms of the proposed framework highlighted in this thesis are assessed though the implementation of a prototype.

The research in this thesis has achieved its aims through the proposal of a functional architecture of a policy-based management framework that is compliant with the OSA APIs and maps to the underlying 3G network reference interfaces without modifications to the existing 3GPP specifications. The research ensured that the proposed framework is also compatible with current 3G business models and the VHE concept, and therefore does not require new business roles to be created for its functional existence.

The research in this thesis was completed at the end of 2004, and therefore the related work reflected is of the period before 2004. Furthermore, the scope of the research in this thesis does not consider the service provisioning of context-aware service through heterogeneous networks as the problem area only regards the 3G mobile network environment.

1.6. Contributions

The contributions made by this thesis are as follows:

- The analysis of 3G business models and the 3GPP's Virtual Home Environment (VHE) concept, and the partitioning of functionality of a policy-based management framework that is complementary to the VHE concept and the different 3G business models. (Chapter 3.2 – 3.3)
- The presentation of generic service adaptation strategies for context-aware services. (Chapter 3.4)
- Identification of the appropriate scenarios in which policy-based management would be beneficial to context-aware services in 3G mobile networks. (Chapter 3.5)
- A proposed solution to managing and resolving multi-dimensional context-aware service adaptation through policies. (Chapter 3.6)
- The design of a 3GPP standards-compliant internal policy-based management framework for 3G network operators to deploy as a back-end to the 3G Open Service Access (OSA) Policy Management APIs. (Chapter 4)
- The proposal of a Policy Information Model (PIM) for 3G context-aware services which is compliant to IETF's PCIM standards.(Chapter 5)
- The proposal and implementation of a policy-managed content adaptation service capability server and framework which is compatible with 3GPP's OSA framework. (Chapter 6)
1.7. Thesis Structure

The thesis structure is as follows: Chapter 2 presents some background on aspects of the 3G network's service infrastructure as well as related work on frameworks and middleware for service adaptation. Chapter 3 examines the requirements for a policy-based management framework deployed for managing context-aware services in a 3G network. It also discusses the benefit and technical factors of using policies to manage context-aware services. Chapter 4 presents the proposed policy-based management framework for managing context-aware services in a 3G network. It details how the proposed framework takes into account the numerous 3G specifications thereby maintaining interoperability and compliance. Chapter 5 introduces a generic policy information model that aids the definition of policies for managing context-aware services in a 3G network. Chapter 6 presents a case study of policy-based management for a context-aware service and the design of a framework prototype that realises this case study. Finally chapter 7 concludes the thesis, highlights the key contributions made, and discusses further work which can be undertaken as a result of the research presented in this thesis.
2. Management of Advanced Services in 3G networks

2.1. Introduction
This chapter presents background information on 2 key service provisioning related specifications for 3G networks; the Virtual Home Environment (VHE) concept and the Open Service Access (OSA). It also surveys the various research projects related to service adaptation and discusses their approach to the provisioning of adaptable services.

2.2. The Virtual Home Environment Concept
There are many definitions of the term Virtual Home Environment (VHE) from various sources, each describing its interpretation of VHE enabling features. Some definitions of the VHE are presented in Table 2.1. (Note that the last two definitions are from European projects researching the VHE concept) [104].

<table>
<thead>
<tr>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>3GPP</td>
</tr>
<tr>
<td>“Virtual Home Environment (VHE) is defined as a concept for personal service environment portability across network boundaries and between terminals. The VHE ensures that users are consistently presented with the same personalised features, User Interface customisation and services in whatever network and whatever terminal (within the capabilities of the terminal and network), where ever the user may be located” [5]</td>
</tr>
<tr>
<td>GSM MoU</td>
</tr>
<tr>
<td>“Virtual Home Environment (VHE) is a system concept for service portability in the Third Generation across network borders” [15]</td>
</tr>
<tr>
<td>ITU / IMT2000</td>
</tr>
<tr>
<td>“VHE is a capability whereby a User is offered the same service experience in a visited network as in his Home system.” [16]</td>
</tr>
<tr>
<td>UMTS Forum</td>
</tr>
<tr>
<td>“VHE means that the user will have the same interface and service environment regardless of location (personalised user interface independent from the current serving network).” [17]</td>
</tr>
<tr>
<td>Eurescom P920-G1</td>
</tr>
<tr>
<td>“The Virtual Home Environment is an environment which presents the user with a common look and feel interface and service experience regardless of location, network and terminal type. The VHE is based on standardised service capabilities and personalised features that are consistently presented so that the user always &quot;feels&quot; that he is on his home network even when roaming across network boundaries” [18]</td>
</tr>
<tr>
<td>IST VESPER</td>
</tr>
<tr>
<td>“VHE main feature is that the customised environment will be following the user while he/she is roaming within different networks and using different terminals” [19]</td>
</tr>
</tbody>
</table>

The common denominator among those definitions is provisioning of customisable and personalised services, portable across heterogeneous environments. However, there are some issues with regards to other aspects of the VHE, as discussed below.
2.2.1 Home Environment

The 3rd Generation Partnership Project (3GPP) defines the 'Home Environment' (HE) as the environment that is responsible for the overall provision of services to users. In the International Telecommunication Union's (ITU) definition of the VHE, the term 'Home system' is used in comparison with the term 'visited network'. Through this comparison, it appears that the Home Environment (HE) is within the domain of the user's home network provider. The term 'home network' must be clearly defined if the VHE concept includes non 3rd Generation (3G) mobile networks in the future. For example, if a user subscribes to different mobile and Internet Protocol (IP) network providers, i.e. future 3G/WLAN integrated network, then both networks can be logically referred to as the home network. In this case, the distinction is that the former is the home mobile network, while the latter is the home IP network. This distinction is only important if the network operator provides multiple network access technologies for the user in the future, though the scope of this thesis only covers 3G networks. If the user's Personal Service Environment (PSE) is to be portable between different types of networks, it is possible that the HE is controlled by a business entity separate from the home network provider. However, it is assumed in this thesis for simplicity that the business entity which provides the user with access to a 3G Mobile Network controls the HE regardless if the business entity owns the mobile network or leases it from a network operator.

2.2.2 Service Performance Preferences

In ITU's definition of VHE, the user is offered the same service experience regardless of the network used. The expected performance of the service can be interpreted as part of the user's service experience. For server-centric services (e.g. video-conferencing), service performance is partly dependent on the Quality of Service (QoS) offered by the network. In this respect, there are three options available to ensure that the user gets the same service experience: a) let the service adapt to the network conditions; b) let the network adapt to the service requirements; and c) a mixed approach including both service and network adaptation. In the example of video conferencing, option (a) could involve adjusting the frame rate of the video to suit the network conditions. For option (b), a guaranteed QoS would be required from the network before the video conferencing begins.

2.2.3 Networks and Terminals

As all of the definitions in Table 2.1 state that the user's service experience should be similar regardless of the network used, the primary understanding of the VHE concept seems to be that of service interoperability and portability between networks. However, it is
important to note that 3GPP have added the condition of terminal-independent service functionality (limited to within the capabilities of the terminal and network) to its definition of the VHE. This presents another consideration for service providers when developing services or applications; Service providers must therefore ensure that they can adapt their applications and services to both terminal and networks. These two factors are not mutually independent as the terminal's capabilities may also determine the maximum bit rate to run the application or service. The bandwidth required from the network to support the service should therefore not exceed the maximum bandwidth supported by the terminal for reasons of costs and efficiency. In the case of a video-on-demand service, the bandwidth reserved in the network should be compatible with the maximum receivable bit rate of the terminal. For example, a 384 kb/s UMTS user should not reserve 2 Mb/s of bandwidth from the network. The terminal must therefore be able to announce its capabilities to an interested entity (e.g. network, service etc.) in order to prevent this from happening. The maximum theoretical receivable bit rate of a terminal, however, may not always be the maximum receivable bit rate in practice. For example, the user may have a contract with the network operator which limits the maximum receivable bit rate. These considerations will be further discussed in section 3.5.1.

2.3. The Open Service Access

The telecommunications landscape is continuing to develop rapidly. Evolving technologies, regulations, and changing user expectations pose challenges for incumbent telecom operators and present opportunities for new operators. Recent technological innovations allow the delivery of classic telephony services over new network technologies, such as wireless, cable, and the Internet. Today's changes in telecom regulations have blurred the lines between telecom operators, cable companies, and Internet service providers (ISPs), allowing them to provide services in areas formerly belonging solely to other domains. The arrival of new telecom and Internet network operators in the marketplace is also causing increased competition. These developments decrease the profit margins on (access) networks, the traditional source of income for telecom operators.

Users are demanding new services at reasonable prices, often tailored to their changing business models and personal situations. Mobile telecommunications operators face certain problems when they want to introduce new services rapidly in response to the growing mobile communications market. Traditionally, such services often are tied closely to a particular network operator. Thus, a new service must be created by the service provider separately on each network, and inter-working problems may have to be solved for each additional service. Such problem solving is expensive and compromises the rapid
development of new services that customers require. The introduction of the Open Service Access (OSA) was meant to tackle such problems by providing service providers with a standardised interface with which their service can interact with the 3G mobile network.

Open Service Access (OSA) was developed by 3GPP to allow service providers to develop server-centric services requiring 3G network capabilities in an unrestrictive manner via a framework [4]. The creation of the OSA stemmed from the Parlay Group’s initiative of providing services with access to network resources and capabilities. Figure 2.1 illustrates a high level view of the open service access. The key component of the OSA is the framework server and its Application Programming Interface (API). These allow applications to discover the network functionalities that are exposed by the network provider. The network provider is not obliged to expose the entire functionality of the network through the framework API, and therefore service discovery is a key feature in both Parlay and OSA.

2.4. Related Service Architectures for Adaptable Service Provisioning

This section presents the prominent European projects in the area of service architectures and service provisioning up to 2003. Their research focused on the management of service adaptation to the user's context and includes the impact of user customisation on service adaptation. The related work highlighted in this section is relevant to this thesis for two main factors: User customisation is a key feature of the VHE concept, which is considered in the design of the proposed policy-based management framework. The related work also involves the management of the adaptation of context-aware services, which is the main scope of this thesis.

2.4.1 MOBIVAS

MOBIVAS was an Information Society Technologies (IST) project funded by the European Union as part of the Fifth Framework Programme (FP5) for research, technological development and demonstration (RTD). The objective of the project was the development of an innovative and modular network platform for seamless and dynamic provision of
advanced Value Added Services (VAS) over 3rd Generation Mobile Telecommunication systems.

2.4.1.1 General Architecture of MOBIVAS

Figure 2.2 illustrates the MOBIVAS service provision platform in the overall end-to-end network architecture. The platform can be used to support advanced adaptability features in mobile service delivery [21].

The primary components of the proposed platform are the following:
- Reconfiguration Control and Service Provision Manager (RCSPM),
- Value-Added Service Provider (VASP),
- End-User Terminal (EUT) and
- Charging, Accounting and Billing system (CAB).

The RCSPM is responsible for the automatic service registration and deployment by third party Value-Added Service Providers (VASPs). Its primary task is the reconfiguration of the underlying network components for efficient service provision and the management interactions between mobile network providers and VASPs. Moreover, the RCSPM caters for service adaptability within and in-between highly variable environments/domains by providing a mobile portal to the end user for personalised and context-aware service discovery and access.

The RCSPM component consists of the following modules:
- The User Interaction Management Module (UIMM) is responsible for providing the user with a personalised, context-aware mobile portal for service discovery and downloading. The UIMM maintains context information per user session. This information is retrieved from various repositories such as user profile databases, value-added service (VAS) databases and the underlying network. Context information is delivered to the Reconfiguration Manager (RCM). The latter, after taking various reconfiguration actions, returns the results to the UIMM for presenting a concrete view of customised service provision to the end user. Moreover, it handles user authentication, based on Simple Public Key Infrastructure (SPKI) certificates issued by a trusted third-party.
The Reconfiguration Manager (RCM) is responsible for network, platform and service reconfiguration actions. The RCM reconfigures the Differentiated Services (DiffServ) Edge Routers (DSERs) during VAS management procedures such as registration, deregistration and updates triggered by the VASP. The RCM comprises a generic adaptation module for intelligent profile matching. An important characteristic of the adaptation module is its independence of the profile types that need to be matched. In the context of the MOBIVAS architecture, the adaptation module is also used for the customisation of the service discovery procedure and for service adaptation. Notably, the adaptation module is an autonomous component that accepts input profile data and returns adaptation decisions. Instances of the adaptation module can be activated elsewhere in the system and used by any other component of the platform or the service itself.

The Packaging and Downloading Module (PDM) packages the necessary software components and supporting resources for service execution. A single archive is produced, which is dynamically tailored to the context of the particular VAS selection request.

The VAS Registrar Module (VASREGM) is exploited by the VASPs for automatic service deployment. The VASREGM includes a Web interface for service management operations, including addition, removal and update of Extensible Markup Language (XML)-encoded descriptors of the service attributes. Based on these attributes, the
VASREGM co-ordinates the service deployment process, including any required management actions (e.g., interactions with the OSA/Parlay framework) and reconfiguration actions (e.g., of the DSERs).

- The Value-Added Services Database Module (VASDBMM) comprises a front-end of a database for keeping service profile information.
- The User Database Management Module (UDBMM) handles user profile information required for customised service provision. This data concerns user interface preferences, service-independent user preferences (language, pricing level preferences, and QoS specific metrics) as well as the list of the user’s favourite services. The stored profiles can be viewed and modified by the user at any time. Different profiles may be associated with a single user. Each of these profiles shall reflect a particular, distinct status (e.g., “leisure/business”, “home/travelling”). More than one profile can be active for a single user at any time.

The CAB system collects and processes all chargeable information, apportions the revenue between the involved business players (e.g., VASPs, service platform providers, network operators) and imposes flexible billing schemes for debiting end-users. The system should allow the application of arbitrarily complex billing models and tariff schemes to be dynamically reconfigurable by the RCSPM. The CAB system is an autonomous component that is not tightly bound with the network infrastructure of a particular network operator. Thus, it can process charging data and provide bills for users served by different connectivity providers.

The VASP component of the platform handles secure terminal access to a repository of application clients. This component accommodates Web interfaces to RCSPM functionality that can be exploited by the VASPs in order to perform service management operations such as service deployment.

The End-User Terminal (EUT) component resides in the Mobile Terminal. The EUT caters for service discovery, secure downloading and lifecycle management (e.g., start/stop/abort/resume) of application clients, as well as initial user registration with the platform and subsequent user login/authentication. The EUT software can be downloaded and dynamically installed, provided the user holds either a valid subscription with a network operator that has a contract with the service platform provider or a certificate issued by a trusted third party. The terminal may also host part of the user session context information as well as an instance of the aforementioned generic adaptation module, which can be useful for locally making adaptation decisions that are reached and actuated during service execution.
2.4.1.2 Discussion on MOBIVAS

The MOBIVAS platform was designed primarily from the value-added service provider’s viewpoint. It allows the value-added services to obtain the user’s context through the OSA framework and therefore it is designed to be deployed on top of OSA. This incurs additional management overheads to obtain details about the user’s context. The MOBIVAS platform is related to this thesis as it views the OSA as key to service provisioning in 3G networks. It, however, does not provide any avenues for services to influence the user’s context (e.g. network Quality of Service) in the 3G network. The proposed policy-based management framework in this thesis differs from the MOBIVAS platform by addressing this issue.

2.4.2 ADMITS

ADMITS (Adaptation in Distributed Multimedia IT Systems) [22] project was initiated at the University of Klagenfurt, Austria. It realizes a distributed multimedia architecture, which adapts its behaviour to changing resource availabilities. The goal in the ADMITS project is to access multimedia resources, from different types of terminals and/or over diverse networks and make this access interoperable and transparent.

2.4.2.1 Functional and Technical Description of the ADMITS Architecture

The ADMITS project realizes a distributed multimedia architecture which adapts its behaviour to changing resource availabilities. The individual components shown in Figure 2.3 play different roles in the adaptation process.

These components are connected physically by the network and semantically by MPEG-7 [23] Multimedia Descriptions and MPEG-21 [24] DIA (Digital Item adaptation) Descriptions that flow over the network. MPEG-7 defines the Variation Description Scheme, which enables standardised communication of A/V data in different representations.

This distributed multimedia system comprises several components, such as media servers, meta-databases, proxies and routers. This subsection describes the whole architecture shown in Figure 2.3.

- **Meta-Database**: This component supplies the meta-data for the adaptation engines. Also, it is for the user an entry point to the distributed multimedia architecture. It is implemented as an MPEG-7 Multimedia Data Cartridge relying on cartridge technology of the Oracle databases management system (DBMS) [25]. The metadata supports content-based queries that are based on low and/or high-level indexing information. It supports as well adaptation based on meta-information, such as the definition of transcoding procedures with the help of MPEG-7 Variation Descriptors.
• **Adaptive Virtual Video Server:** This component provides means for offensive component adaptation (allocation of additional components instead of use of the best out of the available ones). It stores the raw A/V data and has a distributed architecture containing a number of nodes. The server is called virtual, because it is able to change the set of the actually allocated physical nodes on demand. In particular, it is composed of a media storage server and a meta-database. Content indexing of relevant metadata is done in two ways: The media is first analysed for semantic and structural content (Step 1a) and for its adaptation capabilities (Step 1b). Therefore, two categories of metadata are extracted and this is done in two steps:

![Figure 2.3 The ADMITS Architecture](image)

- **Step 1a:** Video segmentation is carried out and semantically meaningful information is extracted (e.g. where events, persons and objects are the important entities of a segment). Furthermore, low-level descriptors are extracted, for instance, the MPEG-7 DominantColor and the ScalableColorType. MPEG-7 Semantic and structural Descriptors enable content based queries.

- **Step 1b:** MPEG-7 Variation Descriptors are produced expressing the relationship between the available variations of the media and their characteristics regarding media information (e.g. file size and frame rate) and quality.
The ADMITS server provides delivery functionality for media and its descriptive metadata. Therefore, the media server contacts upon a request the meta-database for descriptions (Step 1a). These descriptions are attached to the media stream and delivered to the client (Step 1b).

- **Adaptive proxy cache**: This component supports media adaptation. The proxy cache implements quality-aware versions of the LRU and greedy dual-size algorithms [26]. Initially, the cache stores full videos with their respective metadata descriptions in the form of MPEG-7 Variations Descriptors, and it reduces their quality (and thus their size) in integral steps, before fully deleting them. The cache can offer both short start-up delay and acceptable quality. The essential innovation is that the quality reduction process is driven by metadata in a standardised format (MPEG-7), which is readily provided in ADMITS by the meta-database (Step 3). For each adaptable video, the cache expects just the simple triple parameter: transcoding operation, resulting size and resulting quality.

- **Adaptive routers**: In contrast to the proxy cache, the operations that a router can perform on a media stream are fairly limited, since the forwarding speed must not be compromised significantly. A router can also cope with dynamically varying network load and multicast media streams along heterogeneous links or towards clients with different capabilities. Adaptation operations performed by routers would include dropping of video frames, of certain enhancement layers, or of entire MPEG-4 Elementary Streams.
• Adaptive query and presentation interface: The adaptive query and presentation interface supports the specification of search criteria for media resources, displays results from the database (metadata), proposes means for selecting the media from the database results, and opens players for the selected media. It is the final adaptation level in the end-to-end scenario as depicted in Figure 2.4, Step 4. It is adaptive in the sense that all interface components adjust dynamically to the usage environment. The usage environment encompasses the client's terminal capabilities (e.g., hardware, software) and the usage preferences (e.g., user prefers only audio files).

2.4.2.2 ADMITS and Adaptation

The ADMITS project deals with the adaptation of multimedia data. This adaptation process is performed by the ADMITS' components. Adaptation in multimedia systems is usually restricted to stream-level adaptation which evaluates to defensive, reactive media adaptation. The ADMITS project proposes a combination of stream-level adaptation with an offensive, proactive, system-level approach to adaptation. Offensive adaptation can proactively handle situations where the resources of the server nodes and/or network links become saturated. The combination of the two approaches enables the overall, end-to-end quality of service. Many Descriptors are proposed in MPEG-7 that can support multimedia resource adaptation in network nodes on the delivery path to the client. In particular, the VariationSet Description Scheme (DS) provides hints as to when to apply an adaptation and which algorithm to use. The VariationSet DS may be understood as holding an unbounded set of variations of a multimedia resource described by a Variation DS. It allows one to bind several versions of the same resource without specifying how these versions are obtained. For instance, if the source resource is a video, then the variations may consist of an image, audio, or several videos with lower temporal, colour or spatial resolutions obtained respectively by temporal, colour or spatial reduction. These MPEG-7 Variation DS can be used in connection with quality-based video caching at a proxy server. Although initially whole videos are stored at the proxy, their qualities can be changed according to some criteria. This measure is, for example, beneficial to enrich the cache replacement strategy of the proxy. Two cases are distinguished: (1) quality reduction, i.e., the proxy only reduces quality; (2) quality adaptation, i.e., the proxy can reduce or enhance the quality. A proxy cache can benefit from a VariationSet description to enhance its cache replacement strategy. When there is a need to replace a video in the cache, the proxy applies a quality reduction to the video, thus reducing its data size, instead of deleting it.
2.4.2.3 Discussion
The ADMITS project builds an experimental distributed multimedia system for investigations into adaptation. The MPEG-7 technology is widely used and provides tools, such as the variation description scheme, which enable the components to communicate adaptation metadata in a standardised form. The major distinguishing feature of the ADMITS approach is that (MPEG-conforming) meta-data are used to steer the adaptation processes in components on the delivery path to the client (e.g., proxies, routers). The ADMITS project is related to this thesis as it tries to tackle the issue of adapting the provisioning of services according to the user’s context. The approach in this thesis differs by utilising a policy-based management approach to making decisions regarding adaptation. Furthermore, the framework described in this thesis utilises specific 3G network components and technologies whereas ADMITS was designed to work over a generic IP based network.

2.4.3 ISIS
ISIS [27] stands for "Intelligent Scalability for Interoperable Services". This IST project started in September 2002 and ended in February 2004. The goal of ISIS was to design, implement and validate a multimedia framework that allows for audio-visual content to be created once and adapted to a wide range of service scenarios by customisation to different transport characteristics and end device capabilities as well as personalisation to end-user preferences.

2.4.3.1 General View of the Architecture
This project makes a good state-of-art in three different technological areas of work:
• Development of innovative content representation formats with an emphasis on scalable content management techniques.
• Design of an integrated customisation and personalisation framework able to deal with complex multimedia content formats and metadata.
• Demonstration of the end-to-end platform accessing the content over various networks, including fixed and wireless networks as well as fixed and mobile terminals.
To achieve its objectives, the ISIS project [27] has analysed, specified and developed a complete end-to-end production chain which is presented in Figure 2.5.
This architecture has been designed by specifying and developing a set of player tools to play-back and interact with personalised and adaptive content (still images, audio, video, vector graphics, etc.). According to Figure 2.5, the client's interaction with the server side basically consists in two different requests:

- **The request for a personalised catalogue:** First of all, (1) the user requests a personal catalogue. This request reaches the user modelling engine which creates a catalogue request Uniform Resource Locator (URL) with additional parameters to identify terminal, network and user preferences. The Media Client creates and sends (2) an HTTP request for this personal catalogue to the Customisation and Personalisation Server (C&P Server) with (3) attached parameters for User Preferences, Terminal Capabilities and Network Conditions. On the server side, the P&C Server temporarily caches this user information and generates, according to the user's preferences indicated in the user profile and merging the suggestions coming from server-side user modelling engine, an HTML personal catalogue containing a session ID. The scope of this process is the generation of a "personal home page" where only the available content (from the Content Description Data Base) considered interesting for this specific user.

- **The request for a customised media stream:** Now that the user has his personal catalogue, s/he can navigate between the descriptions of contents on his terminal. S/he selects a specific content description in order to play it (6). The User Modelling Engine (6') on the client side uses this description and allows the learning of the user's preferences and the
update and the improvement of the processing of the User’s profile. The selection, followed by the SessionID, is then forwarded to the Media Client. This request (Selected Content (Session ID)) is sent to the Media Server (7). The Media Server interacts with The C&P Server, which, thanks to the Session ID, knows the appropriate information about the User for the customisation process (8). Finally, the Media Server replies to the Media Client with the customised content (9) and the User can access the content at optimum quality with regard to this available and his/her preferences (10).

2.4.3.2 The ISIS Adaptation Framework

The ISIS adaptation framework [28], as shown in Figure 2.6, is centred on the Media Customiser (MC).

![Figure 2.6 ISIS Adaptation Framework [28]](image)

The MC is the unit that takes as input the user’s selection of content from the Streaming Server (the unit that receives the adapted resources and streams them to the terminal), the description of the content from the Content Description Database and the user’s constraints description from the Temporary Cache Database, decides on the optimal form of the resources for the given constraints and then applies the corresponding adaptation as described in the content’s DID (Digital Item Declaration). The output of the Media Customiser is a stream of data of the adapted resource which are input to the Streaming Manager. The MC consists of two units as depicted in Figure 2.6:

- **The optimiser** analyses the user’s constraints and tries to best match them against a number of adaptation options of the scalable content, as described in the Content Description. The selected adaptation is then passed to the Resource Adaptor.

- **The Resource Adaptor** is the module that applies the decided adaptation and sequentially pipes the data of the adapted stream to the Streaming Server, which then packetises and
2. Management of Advanced Services in 3G networks

streams the data to the end-user. In order to support the wide range of adaptation use cases in the scope of ISIS, two instantiations of the Media Customiser were designed in ISIS adaptation framework. These instantiations are:

- **gBSDMediaCustomizer**: the default Media Customiser, based on the MPEG-21 DIA gBSD [29] technology. It is used in all adaptation use cases involving scaling operations on scalable content.

- **3DMeshMediaCustomizer**: this plug-in is used exclusively for the adaptation of 3DMesh content. Due to the need for backchannel information and specific optimisation intelligence required for the adaptation of 3DMesh, it was not possible to use a generic processor like gBSDLmediaCustomizer for this purpose.

In a more detailed manner, this is how an adaptation session is realised within the ISIS architecture:

1. The Streaming Server receives a Real-Time Streaming Protocol (RTSP) Uniform Resource Identifier (URI) from the client. The URI includes the presentation (service) name (corresponds to one of the MC instantiations) as well as the SessionID. The correct instance of the MC is instantiated and the information included in the URI is passed to the algorithm.

2. The Optimiser uses the SessionID to retrieve the DID containing the user's constraints descriptors from the Temporary Cache database. The SessionID is the DID of the constraints' DID. This DID has been previously aggregated and saved in the cache by the Personalisation Server.

3. The Optimiser uses the presentation name to retrieve the DID of the content from the Content Description Database. The presentation name is the DID (Digital Item Identification) of the DID as well as the ID used to save it in the Database.

4. The Optimiser analyses the adaptation options in the content description and the user's constraints to determine the optimal adaptation. The decision is passed to the Resource Adaptor (MC1) in the form of adaptation type (in the case of gBSD-based adaptation, the Extensible Stylesheet Language Transformation to be used) and adaptation parameters (4). The interaction between the two modules (MC1) is internal to the Media Customiser and does not necessarily need to be in the form of a formal interface.

5. The Resource Adaptor retrieves the corresponding bitstream and applies the selected adaptation.

6. The adaptation is applied sequentially and produced data is piped through the plug-in API to the Streaming Server.

**The transmoder** is another plug-in module which is very similar and operates in parallel to the MC in the overall architecture. Transmoding is defined as a change of modality based on
the substitution of the Digital Items resource with an alternate version that is of another modality.

The ISIS transmoder plug-in comprises the following transmoding tools:
- Text to image
- Graphics2D to video
- Graphics2D to image
- Video to graphics2D (slide show)

2.4.3.3 Discussion
The goals sets by the ISIS project were to adapt several different media types towards unknown network, target device, and user constraints and personalisation. Many aspects have been covered regarding to adaptation such as
- A modular client-server architecture, scalable to lightweight platforms.
- The use of two adaptation techniques: the Transmoding, the Bitstream Description Transforming.
- The Personalisation Server.
- Affectiveware platform coupled with a Context Manager providing pertinent contextual information and enhanced user experience.

Further work is left to be done such as dynamic adaptation, session mobility, more decoders and more integration efforts (only one module including all the adaptation techniques).

The ISIS project is related to this thesis as it considers the user's context when provisioning content and services to the user. The ISIS project places emphasis on adapting the content to the user's context. The work in this thesis differs by also considering the possibility of influencing the context (e.g. network QoS) to suit the user's preferences and service operating requirements.

2.4.4 GIGAMOBILE
GigaMobile (Telematics for advanced mobile applications) began early in 2000 and finished by the end of 2002 [30]. It is a GigaPort application project that aims to extend knowledge and expertise on customised mobile services that can be realised on next generation mobile networks. The GigaMobile project focuses on customisation of mobile end-user services on 3G mobile networks like the Universal Mobile Telecommunications System (UMTS). Customisation here stands for the adaptation of end-user services to personal preferences, location, network and terminal capabilities, and the entire context in which the service is used. The goal of GigaMobile project is to obtain an integral framework and demonstrators that specify and exhibit the functionality of generic services for customising mobile end-user
services. GigaMobile can deal with the customisation of a video service by adapting dynamically the video quality to the available bandwidth.

2.4.4.1 Functional Description of GigaMobile Architecture

Figure 2.7 shows the functional decomposition of the architecture into three pillars.

- The function User Interaction covers all interaction functionality necessary to interact with end users. Functionality encompasses on the one hand presentation logic, e.g. input and output conversion such as encoding and decoding, and on the other hand all user input functions. The User interaction function consists of two sub functions: User input handling and Presentation. Note that this function need not equate with all functionality of the client device in the underlying system architecture.

- The function End-user service covers the functionality of (end-user) services themselves. There is a distinction between the different service functions and the service logic that ties these functions together. Furthermore, service logic adaptation functionality is part of this function. Finally, service accounting functionality is incorporated. Note that the end-user service functionality may be distributed over both client device and back-end servers (e.g. in case of a fat client).

- Content handling covers all functionality regarding content management. Content handling is directly connected to end-users only with respect to content presentation. In principle, end-users can only execute services. As a result, content and content handling is always accessed through the use of a service. Content handling is partitioned into three sub functions: Content access, Content adaptation and Content accounting. Context discovery encompasses all functionality related to the discovery of information regarding the context of the end-user.

- The function User discovery covers all functionality regarding discovery and management of users with respect to user preferences, demands and wishes. The function can be subdivided into two main parts: user profile management, and user search & matching functionality.

- End-user service discovery allows users to find services, by naming or specifying the (category of) service that they are looking for (e.g., “find me a local news service”). The function covers all functionality regarding discovery and management of services with respect to user profiles. The function can be subdivided into two parts: service directory management, and service search & matching functionality.

- Billing & payment encompasses functionality related to billing and payment

- The function Session control & management sets up, modifies and releases communications sessions. Authentication is part of the set-up process.
2. Management of Advanced Services in 3G networks

Well, the figure you've provided shows the high-level functional architecture of GigaMobile. Let's break down the main points:

- **First Pillar: User Interaction**
  - User Input Handling
  - Presentation
  - Service Logic
  - Service Function

- **Second Pillar: Content Discovery**
  - Positioning Information Discovery
  - Device and Network Capabilities Discovery
  - User Discovery
  - User Matching and Search
  - Service Directory Management

- **Third Pillar: Session Control and Management**
  - User Authentication
  - Lifecycle Control and Management
  - Global Roaming
  - Network Selection
  - Handoff Execution

**Figure 2.7 GigaMobile High-Level Functional Architecture [30]**

- **End-to-end QoS control & management** ensures that the QoS level that the end-user perceives is such that the service he or she uses is (and remains) useful. The control aspect of this function governs the QoS of a single communications session. The management aspect governs the QoS of a session at a time scale that transcends the lifetime of an individual session.

- The **Global roaming** function allows a mobile device to connect to the infrastructure when the end user wants to set up a session. It also ensures that the device remains connected when it roams during a session.

- The function **Notification usage & control** notifies listeners of service events, e.g., the signal from a ‘buddy’ application that says that the user’s friend is nearby. It also controls the registration of listeners to events.

2.4.4.2 Technical Description of GigaMobile Architecture

Personalisation is the key-issue tackled in the project. In order to offer personalised services in a consistent manner and to accelerate the development of such services, the GigaMobile project presented a Personal Service Environment (PSE). The PSE assists a user in finding, adapting and using services that satisfy the user’s needs given the personal profile, mobility and context [31].

The PSE implementation follows the Web services paradigm whose architecture is loosely coupled. Specific PSE components will be generated on a development and deployment environment for negotiating brokerage agents with learning, anticipating, and adaptive capabilities [31]. These brokerage agents manage matching and adaptation of service
2. Management of Advanced Services in 3G networks

content and logic, ensuring optimal QoS delivery. The PSE is a holistic approach to the personalisation of mobile data services in a scalable and efficient way by using both centralised and distributed functionality and intelligence [31].

2.4.4.3 GigaMobile and Adaptation

In the GigaMobile project end-user services are adapted to service context (e.g. personal preferences, location, network and terminal capabilities) in order to provide value-added mobile end-user services. GigaMobile enables the customisation not only across vertical chains, but also horizontal ones.

The project studied three categories of generic services:

- **Personalisation**: Adapt services to the user position and orientation (location-awareness) and user preferences and profile.
- **QoS adaptation**: Adapt services to the quality of network and the terminal.
- **Brokerage**: Broker between personal requirements and service capabilities. This involves, besides a brokerage mechanism, an integrated representation of the user as well as a representation of services by means of service profiles.

The GigaMobile project is related to this thesis as it explores context-aware service provisioning while considering the concept of a user Personal Service Environment (PSE) which is related to 3GPP’s VHE concept because the VHE allows a user’s PSE to be available in a visited network. The work in this thesis expands on these ideals by considering the usage of 3G interfaces as specified by 3GPP. Most importantly, the framework in this thesis incorporates the OSA, a key enabler of the VHE concept, to fulfil the some of the components in the GigaMobile functional architecture.

2.5. General Discussion on Related Service Architectures

All of related projects pertaining to context-aware service adaptation presented in the previous subsections do require a functional component which ultimately decides on the method of service adaptation. However, none of them used a policy-based management approach for this task. As there are many aspects of the user’s context that will contribute towards the adaptation decision, the adaptation process requires a decision algorithm that is consistent with the service’s adaptation goals, efficient and conclusive. For example, how should the service then adapt if it is not able to adapt to the user’s preference? The proposed framework for managing context-aware services in this thesis utilises policies, which are represented as a hierarchical structure of rules, to manage the service to achieve its adaptation goals.
3. Requirements for Policy-Based Management of Context-Aware Services in 3G networks

3.1. Introduction

As the first phase (Release 99) of 3G mobile networks have been deployed across the world over the last few years, most of the 3G standardisation efforts behind the scenes have been on the technical aspects of the radio access network, the core network and the user terminal. There was a discernible lag in research and standardisation on the service infrastructure for the new advanced 3G services (e.g. context-aware services such as location-aware services). This is slowly being addressed with the introduction of new service-related specifications such as the Open Service Access and the Virtual Home Environment (VHE). Service management aspects are also addressed with the introduction of policy based management components within the 3G core network and at the OSA. This chapter examines the requirements of a policy based management framework that is compatible with current 3G specifications. While 3GPP has introduced policy-based management functionality into its OSA APIs, it has not explained how policies can be beneficial or be used to manage the adaptation of context-aware services. This chapter's main focus is to address this issue by expanding and proposing when and how policies should be applied to manage context-aware services in a 3G network while satisfying the requirements that the framework must satisfy when operating with a 3G network.

Section 3.2 examines the Virtual Home Environment concept and how it impacts the deployment of the policy-based framework developed in the proposed architecture. Section 3.3 explores the different alternatives for the physical deployment of a policy-based framework to manage context-aware services. Section 3.4 examines the use of policies to manage context-aware services. Section 3.5 examines how to create and enforce policies that deal with different aspects of the user's context.

3.2. The Impact of the VHE and Business Models on Policy-Based Management

The VHE concept has a technical model where several role players are involved in the provisioning of services to the user. It is important to determine where the policy-based management framework is located within these role players as the division of management tasks between the different roles has to be clearly defined and partitioned. There are two key requirements for the placing of the framework within any business model; it will certainly need to be compliant with 3GPP's technical views on the VHE. Finally, it should be
deployable within a viable business model whereby the revenue-sharing relationship between service providers and network operators is realistic and not far-fetched. This section discusses the various alternatives in placing a policy based framework within the 3GPP business model.

3.2.1 3GPP VHE Technical Roles

There are 4 main technical roles defined by 3GPP for the VHE [5]; the user, the Home Environment (HE), the Home Environment Value-Added Service Provider (HE-VASP), and the Value-Added Service Provider (VASP). The home environment, which is provided by a network operator, is responsible for managing the user profile, and for controlling access and the overall provisioning of the user’s personalised services. The HE-VASP is a value added service provider which has privileged access to VHE-enabling service toolkits such as the OSA, and is managed by the home environment which therefore guarantees user accessibility when the user roams to a visited network. The VASP offers a service to the user but has restricted (or no) access to VHE-enabling toolkits.

![Figure 3.1 Service Access in the Virtual Home Environment](image-url)
3. Requirements for Policy-Based Management of Context-Aware Services in 3G networks

3.2.2 3G Business Models

The UMTS forum has predicted 3 main possible business models for billing and charging which will be dominant in the future [32]. In the service delivery scenario, the UMTS Forum envisages 3 main technical roles, and a business entity can play more than one technical role. For example, a network operator can also be a content aggregator. The 3 technical roles are:

a) **Network Operator** (NO). The key function of the network operator is to provide access and transport services. A network operator is typically a 3G license holder.

b) **Content Aggregator** (CA). A content aggregator could perform the function recognised today as a mobile portal. The key function of the content aggregator will be to package and offer services from one or several content providers.

c) **Content Provider** (CP). The role of the content provider is to provide services ("content" or applications) that add value to access and transport services. Value-added services can be produced by the content provider itself or purchased from others.

The 3 main business models are:

a) **Network Operator centric business model** (Figure 3.2): In this model, the network operator plays both content aggregator and network operator roles. This is currently the predominant business model operating today. The user only pays one bill to the network operator for all his subscribed services. The management of services and the user's home environment is the entire responsibility of the network operator.

b) **Content Aggregator centric business model** (Figure 3.3): This model was developed for content aggregators to enter the 3G market. It can also cater to future virtual mobile 3G network operators who do not own the network or the content services. In this business model, the user may have 2 contracts; one for the content aggregator, and a separate contract for a network operator. It is also possible for the user to have a single contract with the content aggregator, which settles network charges on the user's behalf. This business model is probably the most complicated to implement and manage as the transactions required for billing and service provisioning are more than the other 2 business models.

c) **Content Provider centric business model** (Figure 3.4): In this model, the content provider plays both content provider and content aggregator roles. This model caters mainly for "one-off" value added services such as the downloading of ring tones and music. The user is expected to have a separate contract with the network operator.
3. Requirements for Policy-Based Management of Context-Aware Services in 3G networks

3.2.3 Other Related Business Models

There are a few European projects that propose different technical roles for providing the VHE. They are namely the IST Magnet and the IST VESPER European projects. The
VESPER project’s main aim was to facilitate the deployment of the VHE and the management of services which conform to the VHE concept. In the IST VESPER project, the VHE is managed by a VHE provider on behalf of the user. There is a clear separation between the network provider/operator and the VHE provider as shown in Figure 3.5. Figure 3.6 shows that the VHE provider interacts with the network provider through the OSA/Parlay interfaces, and the VESPER middleware is placed within the VHE provider which is logically between the service provider and the network operator [19][33].

![Figure 3.5 Business Roles within IST VESPER](image)

![Figure 3.6 Deployment of the VESPER Middleware (Logical View)](image)

The IST Magnet project’s main focus is the deployment of Personal Area Networks in the next generation Internet and its business roles model adheres more closely to the UMTS Forum’s model. The following roles are defined in IST Magnet [34]:

- **Equipment manufacturer**: the company that manufactures and sells the Personal Network (PN) devices, network elements etc.
3. Requirements for Policy-Based Management of Context-Aware Services in 3G networks

- **Network operator**: a company that manages and maintains a large network for rent.
- **Internet Service Provider**: the company that provides the Internet access and other value added (e.g. Voice over IP) services.
- **Personal Network (PN) Service Provider**: the business company or organisation that delivers the PN services to the users over the networks.
- **Application Provider**: the company that provides the applications for the PN service provider or the end users.
- **Payment Service Provider**: the company that handles payment transactions in a secure and trustworthy manner.
- **Content Provider**: the company that provides the information-based services (like online shopping, web surfing, chat rooms, accessing data) to users or PN service providers.
- **Content Aggregator**: the company or organisation that gathers and packages the web content from different sources for reuse or resale.

The Personal Network (PN) Service Provider has a role which is similar to IST VESPER’s VHE provider and is also in charge of all service adaptation decisions. Figure 3.7 shows the business roles envisaged by IST Magnet.

3.3. Factors for Placing the Policy Based Framework

There are two possible locations for deploying the policy-based management framework that manages VHE services; at the network operator, or external to the network operator (e.g. at a value-added service provider).

3.3.1 Framework External to the Network Operator

It is possible to place a policy-based framework outside of the network operator, in value-added service providers etc. However, this requires more complexity in the implementation of the framework, particular in the task of context gathering from the network operator (e.g. location, available network QoS etc.) when the framework tries to perform context-aware
service adaptation. More transactions and network traffic between the framework and the network operator are also required for the same purpose. There are other issues as well. 3GPP states that network policies (e.g. network QoS) always take precedence over application-based policies. Policy conflicts would then be difficult to diagnose, detect and resolve as the policies from 2 different policy framework systems housed in the different business roles are operating at the same time. It does, however, seem suitable for the context aggregator centric business model proposed by the UMTS forum.

3.3.2 Framework at the Network Operator
In this approach the home environment/network operator is primarily responsible for the management of the user’s services. If the policy-based management framework is housed within the network operator, it would be easily accessible by services through the OSA policy based interfaces. The OSA interface provides sufficient capabilities to enable applications to request:

a) To manage the application’s policy-related information
This allows applications to create, modify and delete policies, policy events and to activate, deactivate and modify policy rules. Policy rules may be expressed with simple data types (such as integers or string) or more complex data types (such as Boolean values, time, lists, meta-variables etc). Expression of policy rules shall take into account these complex data types as well as allow for a feature rich set of operands and allow for ability to define user specific functions.

b) To manage policy event notification
This allows applications to register for specific policy events. Once registered for such events, the application shall receive notification of the events until it explicitly requests the termination of the notification request.

c) To collect policy statistics
This allows an application to collect policy related statistics from the network. Examples include success or failure of operations on policies and time stamps of policy events.

d) To request policy evaluation
This allows an application to request that a set of policies is evaluated by the network.

Placing the framework at the network operator has several key benefits. Firstly, all policy-based management tasks are performed by a single business entity which lowers the complexity as opposed to interoperating policy framework from numerous business entities. Secondly, the policy management interfaces are standardised in the industry as opposed to developing an ad-hoc solution using an ad-hoc policy definition language. Finally, it is
suitable for the network operator-centric business model proposed by the UMTS Forum, which is the dominant 3G business model in use today. As such, it was decided that the policy-based framework will be designed and developed for deployment at the network operator.

3.4. Using Policies to Manage Adaptation for Context-Aware Services

The availability of a policy-based management framework facilitates the possible deployment of adaptation strategies via policies by context-aware service providers using the advanced context-sensing technology that 3G networks provide. It is important to clarify the manner in which policies are used to manage adaptation for context-aware services. This section discusses the key generic use cases, from a context-aware service’s perspective, for a policy-based framework located at the network operator and made accessible through the OSA interfaces [106].

3.4.1 Situations When Policies Can Be Enforced

The 3GPP OSA policy management interface allows for policy event notification. This allows for management-related interaction and feedback from the 3G network to the service. Policies can, therefore, be either wholly enforced within the network or through the coordinated efforts of the network and the service application. Policy event notification is particularly useful for service activation and service configuration of context-aware services.

Context awareness can also be exploited for service activation, e.g., time- and location-driven service activation. Services can then be activated and ‘pushed’ to the user as soon as specific time or location conditions are met. An example is a location-aware traffic-warning service that alerts the user of congested roads as soon as he or she enters a metropolitan area. Such a service can then use a policy which requests for event notification only for the metropolitan areas in which it serves.

Context-driven service configuration consists of configuring/delivering the service (service components, parameters, and content) on the basis of the operating context. For instance, different drivers or codecs may be needed in a multimedia streaming application, depending on user preferences (e.g., image resolution), terminal capability (e.g., display resolution and colour depth), and available end-to-end QoS level.

Another important feature is context-driven run-time adaptation. In this case the service will exploit context information to trigger appropriate adaptation actions with the ultimate goal of meeting the user expectations regardless of location, terminal, and network fluctuations. One or more adaptation actions may be possible for a given service and context. The service content may be adapted in order to accommodate network-changing
conditions — e.g. a video streaming service may dynamically modify the compression ratio of the codec in response to a varied end-to-end QoS level. Alternatively, there may be an option for the service to dynamically request additional network resources in order to satisfy a user request for increased video resolution.

There are three general high-level service adaptation strategies available for developers of context-aware services: let the service adapt to the context; allow the service to influence the context to suit its operation and service requirements; or a hybrid approach encompassing the previous two strategies. The hybrid approach is particularly useful if the service adapts differently to various aspects of the user’s context. For example, the service could adapt to the user’s terminal capabilities, but at the same time request for more network QoS for better performance. There are however some factors to consider when dealing with these generic service adaptation strategies. 3GPP specifications state that the access network policy (e.g. the user’s QoS contract with the network operator) takes precedence over all application-defined policies. It is therefore important for service providers to test their policies using the OSA Policy Evaluation interfaces to ensure that their policies work in the appropriate manner.

3.4.2 Influencing the User’s Context

Service providers have to realise that there are limits in influencing the context as an adaptation strategy. For example, terminal capabilities cannot be changed during service adaptation without the user physically changing the terminal. Another appropriate example is that requesting more accuracy for location sensing from the network is also dependent on the location sensing technology that the network supports. Furthermore, location sensing methods such as network-assisted Global Positioning System (GPS) also require certain capabilities from the user’s terminal. Hence, a service adaptation policy requesting the network to always use network-assisted GPS to determine the user’s location might not always work.

There are 2 main reasons for influencing the user’s context. Firstly, the service can influence the context so as to meet the minimum service operating requirements. This could include the network QoS for a video-on-demand service. Policies for influencing the context for this purpose would be appropriate during context-driven service configuration as if this policy fails at that stage, the service would be made not available to the user. The other main reason for the service to influence the user’s context is to improve the service performance to the standard as required by the user and stated in the user’s profile. Policies motivated by this reason could be enforced during context-driven service configuration as well as context-driven runtime adaptation. An example for the latter scenario could be to increase the
network QoS during service operation or to increase the location sensing accuracy (when the user moves from a non-urban to an urban area).

![Figure 3.8 Policies in Action During a Service Session](image)

### 3.5. Managing and Resolving Policies Relating to Different Contexts

The availability of a policy-based management framework facilitates the possible deployment of adaptation strategies via policies by context-aware service providers using the advanced context-related technology that 3G networks provide. This includes the Location Service (LCS), the Generic User Profile (GUP) framework, the different classes of network QoS the network supports, and the technology for obtaining terminal capabilities which the 3GPP network has. Adaptation strategies defined through policies can allow services to adapt to the context in many ways. For example, a location-aware service that is adaptable to network conditions may place greater emphasis on the user’s location rather than the network QoS. The service could then request for the network to use the most accurate location sensing technology possible as its first priority in its adaptation process. Services could also decide to influence the context, e.g. request for more network QoS, or to adapt to the current context during service operation. 3GPP, however, has stated that network policies always take precedence over application defined policies [4]. One of the most important aspects of network policies is the network operator’s QoS policy, which includes the different 3GPP QoS classes offered and defined by the network (e.g. streaming, conversational etc.), and the level of QoS offered to the user as stipulated in the user’s contract. Since network QoS is one of the fundamental applications of policy-based management, and has higher precedence over other application defined policies, issues of policy resolution and enforcement relating to QoS should be explored first. This section proposes a decision process in obtaining the required network QoS for the context-aware service to operate.

#### 3.5.1 Network Quality of Service

It is important to derive the network QoS requirements for service operation in the VHE before performing any QoS management through policies. There are many factors that can be used in determining the network QoS requirements. These can be all or a combination of
the following: the terminal capabilities, the VHE service’s QoS requirements, the user’s personal preferences, and the access network policies.

3.5.1.1 Terminal Capabilities
The terminal capabilities may be important in determining the VHE service’s QoS requirements. For example, a content-based VHE service may adapt its service to the screen size of the user’s terminal (for example, use smaller pictures). The VHE service and the 3G network obtain information about a terminal’s capabilities through the Content Capabilities/Profile Preferences (CC/PP) framework (possibly by using the CC/PP exchange protocol) [35][36]. After examining the capabilities of the user’s terminal, the VHE service can then decide on the appropriate service model that best suits the terminal before delivering the service. Another important consideration regarding the issue of terminal capabilities regarding QoS is the amount of bandwidth the terminal supports. This capability represents a “hard” upper threshold and therefore the amount of bandwidth reserved in the network should never be more than the terminal’s capabilities.

3.5.1.2 VHE Service QoS requirements
An adaptable service changes its service model to achieve certain predefined goals in its service delivery. These goals could vary from user satisfaction through its perceived performance, to service accessibility catering for a wide range of terminal types. A good example of an adaptable service is a real player video service that adapts the video media bit rate to suit the bandwidth in the user’s access network. A common effect experienced by most services is that the performance of a service is proportional to the level of QoS delivered by the network. Deriving QoS requirements from the network for adaptable services involves more complexity than that of non-adaptable services. This is because the manner in which the service chooses its service model will have to be considered. For example, the choice of service model could be related to terminal characteristics. Consider a layered video service with two service models represented by two separate video files with video resolutions of 160’120 and 320’240 pixels respectively. The decision on the choice of the video file to deliver to the user is based on the user’s terminal maximum screen resolution. Within each of the two service models, the video can adapt to the available bandwidth in the network giving various levels of user satisfaction (e.g. adjusting the frame rate of the video). If the manner in which an adaptable service chooses its service model is known in advance, dynamic QoS reservation in the network can be performed more efficiently. The ability of the service to request for a specific or preferred level of QoS is
pertinent for the category of push services, i.e. services which are initiated by the service provider instead of the user.

3.5.1.3 User Preferences
Choosing the preferred QoS is one of the essential methods that a user can use to personalize a service. This subsection presents some additional user personalization options that deal with QoS allocation in the network. It is possible to deduce the optimum QoS on behalf of a user for a particular service if the user’s terminal capabilities and the VHE service QoS requirements were the only two factors considered. However, the user may not always desire to have the best QoS available in the network for his VHE service. Reasons for this include the service subscription and the cost of the network usage. The user may have a contract with the service provider that differentiates the service offered at the application and/or the network level within the service domain. For a video-on-demand service, this could be achieved by assigning different video servers with different traffic policies to different classes of subscriber [37]. A customer may then have the cheapest subscription to the video services where the video is of the smallest available resolution regardless of his terminal screen resolution. This factor is dependent on factors within the service’s business model and results in the user having a predefined service profile for the particular service.

3.5.1.4 Network Policy
In some cases, the access network policy on resource and QoS allocation may have an upper bound level of QoS available to the user that is lower than that of the terminal capabilities. 3G terminals are categorised into different classes according to their bandwidth capabilities. The user, however, may have a pre-established Service Level Agreement (SLA) with the network provider on the maximum amount of QoS that s/he is allowed to request and use.

By considering all the factors presented in this section regarding the choice of QoS for a VHE service, it is possible to find an ideal level of QoS to request and reserve from the network. Figure 3.10 shows the factors considered in the four-step decision process to achieve the desired QoS parameters that will enable the VHE service to deliver the same personalised ‘look and feel’ regardless of network or terminal as required in the VHE concept. As each step of the decision process is completed, the range of the QoS narrows. In some cases, one or more factors in Figure 3.10 need not apply in the decision process. For example, if the user did not personalise the service, then the user service profile factor becomes irrelevant. For such a scenario, there will be a range of QoS levels that are applicable for the service at the end of the decision process. Choosing a suitable QoS level may then depend on factors discussed in the next section.
3.5.1.5 Prioritising Services

Although the VHE concept specifies that a service should be delivered in accordance with the user’s preferences, there is only a finite amount of resources available to deliver the service. For the issue of network QoS, the user terminal’s maximum bandwidth throughput is a resource which has constraints during service delivery. 3G terminals are categorised into different classes with each class supporting varying maximum amount of bandwidth throughput. The maximum bandwidth throughput capability is important when the user is running multiple bandwidth-intensive services on the user terminal as the user’s service preferences may not be fulfilled because of a lack of bandwidth.

High level policies specified within the user profile can be a useful solution to this problem. A generic simple example is “IF video Conferencing service is running, THEN video frame rate should be at least 8 frames per second and audio should be in sync”. The policy action in this policy is interpreted into various different low-level actions required to effect and enforce the policy. In this example, a certain bandwidth must be reserved from the network for the video frames to be at least 8 frames per second. There, however, must be a mechanism in place if this bandwidth cannot be delivered to the user because the user is concurrently operating other bandwidth-intensive services. A graceful solution to this problem is for the user to personally state the importance of the various services by grouping them into 3 different categories:

- **Gold service** = the user satisfaction should be met at all times (i.e. influence the context if necessary to meet user’s expectation) and the network should ensure that the service always be accessible unless the user’s context is unable to support it.
- **Silver service** = the user is satisfied with the best available quality according to the user context, but service should still be accessible (i.e. influence the context if necessary to meet minimum service operation requirements)
- **Bronze service** = the service is delivered in a best effort manner. If service’s operating requirements cannot be met, then user is happy not to have the service.

Services within each category can also be prioritised by the user. For example, the user may decide through a policy that a peer-to-peer (P2P) file sharing service and a movie-on-demand video streaming service from different content providers are to be in the silver service category. The user can also fine tune this decision by giving the movie-on-demand a higher priority for user resources than the P2P file sharing service if both services are concurrently running.

There might be a scenario whereby the operation of 2 gold services cannot be maintained at the same time because of a lack of resources. This would then violate the guidelines of the gold service category which states that the user satisfaction must be met at all times. If it
is not possible to influence the context, e.g. request for more network QoS for 2 bandwidth intensive gold category services, then the lower priority gold category service will be downgraded to the silver service category as the user's context is unable to support both gold services running concurrently.

The five factors for determining Network QoS requirements and their relative importance is depicted in the diagram below [105]. As one traverses down the layers of the inverted triangle, the policy decision on the appropriate network QoS becomes more finely tuned as the range of applicable QoS parameters decreases. At the very top layer, the terminal's QoS capabilities and the access network policy always take precedence over every other factor, followed by the minimum network QoS required to support the service (so as to fulfil the Bronze category of VHE services). This is followed by the service adaptation QoS decision required to deliver the VHE experience to the user (i.e. to cater to the Gold and Silver category of services). This is further followed by any further user preference for each individual VHE service which could affect the parameters of the network QoS required (e.g. no audio for a foreign language movie in a movie-on-demand service. User is comfortable with reading subtitles and less bandwidth is therefore required). The final decisive factor is to consult the VHE service priorities so as to resolve any conflicts when concurrent services are demanding for an amount of network QoS which cannot be delivered to the user.
3. Requirements for Policy-Based Management of Context-Aware Services in 3G networks

Figure 3.10 Decision Process for Determining the Network QoS for a VHE Service

Figure 3.11 Case Study of Deciding the Amount of Bandwidth Required

Figure 3.11 shows four possible suggested ways in which to make QoS decisions when two VHE services (labelled 1 and 2) with similar service adaptation policies and QoS requirements are active. In case 1, the user activates the second VHE service (labelled 2) while still using service 1 with a terminal which has a maximum bit rate of 56 Kbits/sec. The two services would normally require a bandwidth of 42 Kbits/sec to provide an optimal VHE service experience to the user. The oversubscription of bandwidth occurs because without coordinated QoS management, the two services each believe that they are still requesting a bandwidth within the terminal capabilities. Case 2 illustrates the effect of coordinated QoS management. In this case, the overall decision process is based on balancing the user satisfaction derived on the quality of both services, resulting in an equal share of the terminal’s maximum supported bandwidth as proposed in [38]. This method is well suited for balancing two Bronze category VHE services. In case 3, the allocation of bandwidth between the two services is based on fulfilling the one with the highest priority. When the user activates the second service, a decision is made based on the terminal
3.6. Managing Multi-Dimensional Context-Aware Service Adaptation

Although there can be many factors which constitute the user’s current context, this thesis only considers the factors mentioned in various 3GPP specifications (e.g. the virtual home environment). These factors are: the network QoS, the user profile, the user’s location, and the user’s terminal capabilities. The previous subsection highlighted that network policies always take precedence over application-specific policies. With network QoS included in this category of network-specific policies (other network-specific policies include billing and charging), this aspect of the user’s context is always considered first when managing multi-dimensional context-aware service adaptation. It is possible to attempt to prioritise the other context factors (e.g. user terminal capabilities), but a “one solution fits all” approach to prioritising the remaining context factors is rigid and would deny service developers creativity in catering different adaptation strategies for their customisable context-aware services. For example, a location-aware service may place a higher priority to policies dealing with a user’s location rather than the user’s terminal capabilities.

The provisioning of the context-aware service is therefore highly customisable by the service provider through the use of policy-based management. The only constraint faced by the service provider is that the network QoS policy will always take precedence over other types of policies regarding service adaptation. The following factors regarding service adaptation in the proposed framework can be prioritised if required:

- User resources e.g. available bandwidth
- Service behaviour/performance user preferences.
- Context sensing e.g. a location-based service may place greater importance to location accuracy for customer satisfaction, than bandwidth/network QoS monitoring.
Prioritising policies can be beneficial if policies regarding one of these factors affect the evaluation of policies regarding another factor. For example, the user’s profile may state a preference for a more accurate location sensing technique than the service’s default choice. Alternatively, the service provider can place equal priority to all three factors because they affect service adaptation independently. These factors combine to provide the final policy decision on the adaptation of the service as shown in Figure 3.12.

![Figure 3.12 Policy Decision Factors regarding Service Adaptation](image)

3.7. Discussion and Summary

This chapter has discussed certain key requirements for a policy-based framework for context-aware services which can be deployed within a 3G network. Firstly, there is the issue of the physical deployment of the framework. The author of this thesis has recommended that the framework be deployed within the network operators’ domain. This is to facilitate compliance to the Virtual Home Environment concept (particularly the Open Service Access framework), as well as to adhere to a recommended feasible business model (i.e. the network operator centric model suggested by the UMTS forum). The chapter has examined and proposed the key moments during a service’s operation whereby policies will play an important role. Finally, the chapter proposed how policies can be used effectively to manage context-aware services. The requirements and contributions in this chapter are summarised in bullet point below.

3.7.1 Requirements

The requirements for implementing a policy-based management framework for context-aware services in a 3G network are as follows:

1. Framework must be compliant with 3GPP’s technical views on the VHE. This requirement allows the value-added service providers to manage the adaptation of their service regardless of the network used, location, and terminal capabilities of the user.
2. Framework must be deployable within a viable and realistic business model. This requirement ensures that the proposed framework's impact is significant as it is deployable in a business environment which is advantageous to the business entities involved.

3. The framework must provide methods/interfaces for service providers to create and deploy their policies. This requirement allows policy-based management to be executed on their services.

4. Network QoS policies will always have a higher priority over other policies. This requirement is explicitly stated by 3GPP.

3.7.2 Chapter Contributions and Proposals

This chapter has presented the following key proposals for policy-based management of context-aware services in 3G networks.

- Proposed that policy-based management framework should be located within the network operator's domain to satisfy requirements 1, 2, and 3.
- Proposed a method of obtaining the required network QoS level for the user when the policy decision has to consider requirement 4.
- Proposed the beneficial usage of policies to control service adaptation during a context-aware service's lifecycle in terms of when and how. This contribution provides service providers with guidelines of how to incorporate the use of policies when designing the various states of service adaptation and operation.
- Proposed the key factors to consider that will affect a policy decision regarding the adaptation of a context-aware service to many different aspects of the user's context. This contribution provides service providers with guidelines to design the manageability of the provisioning of their context-aware service when there are many aspects of the user's context that will affect the operation of the service.

4.1. Introduction

This chapter builds on the previous chapter by presenting a key contribution in this thesis: the design and the proposal of the functional architecture for a policy-based framework in a 3G network to manage the adaptation of context-aware services. 3GPP has provided an initial requirement by introducing and standardising OSA APIs for policy-based management. 3GPP, however, has not specified instructions for the mapping of the policy-based management interfaces to the functional components in a 3G network. Contrastingly, some other interfaces such as the OSA call control API do provide detailed mappings between the API and the 3G network. This chapter presents the proposed functional architecture of policy-based framework and its mapping from the OSA policy management APIs to the 3G network components to facilitate the management of context-aware services through policies.

4.2. Basic Architecture of a Policy-Based Management System

Recent research in policy-based management (PBM) has mainly favoured an architecture which uses a basic manager-agent model. The Internet Engineering Task Force’s (IETF’s) RFC on a framework for policy-based admission control adopts such a model and introduced terms, such as Policy Enforcement Point, that are now widely understood and accepted in the PBM research community [39]. The IETF has also introduced a standard called the Common Open Policy Service (COPS) protocol which is used to transfer policy information between policy management components/entities in their policy framework [40]. The 3GPP standards regarding policy management maintain compliance and use the COPS protocol, thus maintaining the importance of interoperability with the IETF’s work. Our architecture is similar to the PBM framework described by the IETF [39] (shown in Figure 4.1) and this section describes the basic functional components of our PBM architecture which is derived from IETF’s policy-based management framework [105].

4.2.1 Policy Input Tool
The policy input tool is used by the policy author to specify policies and to introduce them into the system. A policy input tool ensures that the policies are not conflicting, contain no anomalies, and are specified correctly.

4.2.2 Policy Decision Point/Function (PDF)
The policy decision point/function is the functional component in the system which decides the policy to deploy and enforce when policy events occur. Examples of policy events could be events reported by the policy enforcement points, when new policies are inserted into the system, or when existing policies in the system are amended.

4.2.3 Policy Enforcement Point/Function
The policy enforcement point/function is the functional component in the system which is connected to the managed entities it is in-charge of and enforces the appropriate policies to control their behaviour. It interacts with the policy decision point/function to request for assistance in deciding the policy to enforce and to receive new policy data from the policy decision point/function.

4.2.4 Policy Repository
The policy repository is used to store all the policies which apply to the system. It is typically a directory or a database and is used by the policy decision point to retrieve policy data. The repository is used to hold policy conditions and policy actions. The Network Operator can populate the repository with the conditions and actions that it can support. These may indeed be based on 'off-line' negotiations with the application developer. The application developer uses the conditions and actions in the Policy Repository to create rules for his/her own application. Policies are stored as configuration data for the policy enforcement points in the system.
4.2.5 Interaction between the Policy Decision Function (PDF) and the Policy Enforcement Point (PEP)

The Policy Enforcement Point (PEP) interacts with the PDF by using COPS in two ways: When a PEP first commences operations, it uses COPS to request from the PDF all the relevant policies which it needs to apply so that when an event occurs, it can then map that onto an appropriate policy. If a PEP is unable to find an appropriate policy which matches the policy event, it can make a request via COPS to the PDF for an appropriate policy to apply.

4.3. Interfacing the Policy-based Management System with OSA Framework

This section presents the two key policy-based management interfaces, the policy provisioning API and the policy evaluation API, introduced by 3GPP in the OSA. These two interfaces are used by the proposed framework in this thesis.

The IETF’s detailed specifications on policy-based management were used as a reference by the 3GPP (3G’s standardisation body) when they introduced their Policy-based Management functionality in the Open Service Access (OSA) interfaces, which is an important part of the VHE and 3G service framework [5][41]. These PBM related interfaces provide an avenue for future 3rd party service providers to perform PBM via an OSA gateway, thereby allowing advanced and flexible service management and control. It also provides content and service providers with a feature previously lacking for the VHE; a well-established framework in which to flexibly deploy, manage, and control adaptable services. An OSA client must use the OSA framework interface to discover and bind to the Policy Management interfaces.

In release 6 of the OSA interfaces, the PBM functionality is represented by two different service capability features: Policy Provisioning and Policy Evaluation [42]. The content service provider uses a policy management tool to interact with the Policy Provisioning Service Capability Server (PP-SCS) via the OSA gateway to define, create, modify, and delete policies for content adaptation in the policy repository housed in the network operator’s domain. The Policy Evaluation Service Capability Server (PE-SCS) is used by the content provider to evaluate policies and generate policy events into the system so as to control the adaptation strategy and the provisioning of the content. The PE-SCS is particularly important when new policies are added, deleted, or modified in the policy repository as it is used to evaluate if there is a need to assert a change in policy for the current managed system. Figure 4.2 shows a high level functional overview of the proposed architecture. The shaded portion of the figure represents the additional functionality required.
for the application of policy-based management to context-aware services, and is a key contribution of this thesis.

4.3.1 The Policy Provisioning Service Capability Feature

The main functionality of the OSA Policy Provisioning Service Capability is to allow OSA clients to introduce policies into the policy-based management system in a 3G network in a transactional manner. More specifically, it allows clients to create, retrieve and delete policy groups, rules, actions and conditions. The policy groups, rules, actions and conditions must adhere to an OSA Policy Management Information Model which is very similar to the IETF Policy Common Information Model. A class diagram of the OSA Policy Management Information Model is shown in Figure 4.3.

Figure 4.2 Functional Architecture of the Proposed Policy-Based Management Framework

A significant difference between the IETF Policy Information Model and the OSA Policy Management Information model is that the latter introduces subclasses to both abstract PolicyCondition and PolicyAction classes. In the OSA information model, policy conditions can be defined as an expression, or as an event. An example of a policy expression condition can be $x > y$ which requires evaluation to ascertain if the condition has occurred. A policy event condition, however, matches policy event attributes to those specified in an IpPolicyEventCondition class. A policy event can be generated by the 3G network or by an
OSA client. One example of a policy event generated by the 3G network could be the termination of a service session by the user. The event attributes for such an example could be the service ID and the user. Figure 4.4 shows the interface classes supported by the Policy Provisioning Service Capability Feature as specified by 3GPP.
4.3.2 The Policy Evaluation Service Capability Feature
The Policy Evaluation Service Capability Feature is used by an OSA client to subscribe to policy event notifications and to generate policy events as well. It also allows OSA clients to request for the evaluation of policy rules on demand. This service capability feature accommodates the policy provisioning model described in IETF’s COPS Usage for Policy Provisioning (COPS-PR) specification whereby an external application (besides the Policy Enforcement Point) can influence the policy decision process at the Policy Decision Point/Policy Decision Function through generating policy events into the policy framework [43].

Figure 4.4 OSA Policy Provisioning API [42]

4.4. Static Policy Conflict Detection
As was stated in section 4.2.4, the policy repository stores both reusable policy conditions and policy actions which can be safely inserted by an OSA client without affecting other policy data. It is important, however, to check that there are no policy conflicts when new policy rules are created and inserted into the policy repository. The IETF states that a policy conflict occurs when the actions of two rules (that are both satisfied simultaneously)

contradict each other [7]. The entity implementing the policy would not be able to determine which action to perform. The implementers of policy systems must provide conflict detection and avoidance or resolution mechanisms to prevent this situation.

The research in this thesis deals specifically with using policies to drive the adaptation of context-aware services so as to fulfil the VHE concept. Optimal service operation is achieved by making the service accessible and behaves according to the user’s expectations at any given time. Policy conflicts can therefore occur if one policy states that an adaptation policy action is required and another policy states that the service requires no adaptation. Another example of a policy conflict is that given a certain specific user context, 2 service adaptation policies decisions specify different policy actions on the same service application/component.

Lupu et al. introduced several models for categorising modality conflicts in policy specification [44]. Lupu splits policies into two general categories: Authorisation policies, and Obligation policies. Authorisation policies deal mainly with security and access control. Obligation policies are actions that a subject is obliged to perform on policy targets. A “negative Obligation” policy specifies the actions that a subject should/must not perform on policy targets. Lupu states that policy conflicts may arise when two or more policies with modalities of opposite sign refer to the same subjects, actions and targets. This occurs when there is a triple overlap between the sets of subjects, targets and actions as shown in Figure 4.5.

This section does not delve into the different mechanisms and algorithms for detecting policy conflicts as the subject is too broad a scope to be covered appropriately in this thesis. This thesis assumes that there is an effective policy conflict detection mechanism in place and suggests its functional location and its associated mappings in the framework. The remainder of this section examines the required interactions and modifications in the OSA API to incorporate the policy conflict detection functionality.
There are 2 methods for inserting policies into a policy management system in a 3G network [42]. The first method is to create new policy conditions and actions objects and insert these directly into a policy rule. The second method is to use the policy repository to find an existing policy condition or action, and reuse it by assigning it to the policy rule.

Figure 4.6 shows an example of both approaches and the sequence diagram is explained below.

1. Opens the transaction bracket.
2. Creates a rule object in the group by passing the name as parameter. The method returns the reference to the new rule object.
3. Closes the transaction bracket.
4. Opens the transaction bracket.
5. After having created the rule object, one can "fill" it with actions and conditions. Here a condition is created on the rule object, thus becoming a part of the rule. Conditions defined in such a way cannot be reused in other rules. For this the repository approach should be used. Parameters passed are the condition name and the condition type. Returns a reference to this condition object. Note that: the type of condition object that is to be created must be one of those specified in TpPolicyConditionType. The method createCondition() is used to create a new instance of a condition type in the repository or rule. This method passes the name of the condition, the type of the condition and an appropriate set of attribute-value pairs. Note that it is necessary to include, within the conditionAttributes argument of createCondition(), all those attribute-value pairs that are not inherited from IpPolicyCondition - if the inherited attribute-value pairs are included in this argument then their assigned values will override the values assigned prior to this assignment. Thus, for example, if the new condition type to be created is TpPolicyExpressionCondition, then the attribute named "Expression" and its value must be included in conditionAttributes. Note that this call may throw an exception if the value of "Expression" cannot be parsed. The steps to create an action object instance are similar to those taken to create a condition object instance. We use the method createAction() to create a new action instance. Note that an action object must be one of those specified in TpPolicyActionType. It is necessary to include all the attribute-value pairs that are not inherited from IpPolicyAction, in the actionAttributes argument of createAction().
6. Closes the transaction bracket.
7. Now the repository approach is used, i.e. reusable condition or action objects. In this example we reuse an action. For that purpose we ask at the IpPolicyManager interface
for a reference to a named repository. The repository name is passed. Returns the reference to the repository.

8. If we know already the name of the action object one retrieves the action directly by passing the name as parameter. Otherwise one has to retrieve the name first by using an action iterator. Returns a reference to the action object.

9. Opens the transaction bracket.

10. Now, the action(s) must be assigned to the rule. Furthermore and different to the conditions, one has to assign an ordering number to the action. Passed parameter is the action list, which is a list of action reference/sequence pairs.

11. After having created or retrieved all needed conditions they must be assigned to the rule. This is done by passing the list of conditions to that method. This is explicitly done by passing TpPolicyConditionList again consisting of TpPolicyConditionListElements which contains the reference the IpPolicyRule object created with message 2. If the rule is active, this will then cause the expression defined in the condition to be evaluated (as often as necessary). Note that the binding between the variables referenced in the expression and the instances of the variable available is done each time the expression is evaluated. That is, when evaluating a variable reference, each enclosing domain is searched in order (from closest to farthest) for a matching variable. If one is found, it is used. If no matching variable is set, the expression condition fails (evaluates to FALSE). Activation of actions is done similarly.

12. Closes the transaction bracket.

Figure 4.6 Inserting Policies into the System via the OSA APIs [42]

The checks for conflicts occur after each action has been inserted into the rule. The OSA’s transactional capabilities can be used to reject unsuitable policies by throwing exceptions to the OSA client. The checking of policy conflicts and the generation of appropriate exceptions are the main responsibilities of a static policy conflict detection function in a policy-based management system. Table 4.1 and Table 4.2 show the OSA specifications regarding exceptions generated by the OSA framework. From Table 4.2, the most appropriate exception code to generate when a static policy conflict occurs is 000Eh. This code is inserted into the “ExceptionType” field of the TpCommonExceptions object, while the “Extrainformation” field will contain the description of the conflict.

<table>
<thead>
<tr>
<th>Structure Element Name</th>
<th>Structure Element Type</th>
<th>Structure Element Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExceptionType</td>
<td>TpInt32</td>
<td>Carries a constant from the list in the table below</td>
</tr>
<tr>
<td>ExtraInformation</td>
<td>TpString</td>
<td>Carries extra information to help identify the source of the exception, e.g. a parameter name</td>
</tr>
</tbody>
</table>

Table 4.2 Constants Associated with TpCommonExceptions

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_RESOURCES_UNAVAILABLE</td>
<td>000Dh</td>
<td>The required resources in the network are not available</td>
</tr>
<tr>
<td>P_TASK_REFUSED</td>
<td>000Eh</td>
<td>The requested method has been refused</td>
</tr>
<tr>
<td>P_TASK_CANCELLED</td>
<td>000Fh</td>
<td>The requested method has been cancelled</td>
</tr>
<tr>
<td>P_NO_CALLBACK_ADDRESS_SET</td>
<td>0011h</td>
<td>The requested method is refused because no callback address has been set (this may be the result of a timing issue between setting the callback address and invoking the method)</td>
</tr>
<tr>
<td>P_METHOD_NOT_SUPPORTED</td>
<td>0016h</td>
<td>The method is not allowed or supported within the context of the current service agreement.</td>
</tr>
<tr>
<td>P_INVALID_STATE</td>
<td>02E8h</td>
<td>Unexpected sequence of methods, i.e., the sequence does not match the specified state diagrams.</td>
</tr>
</tbody>
</table>

4.5. Context Control Function and Context Gathering Function

The context control and the context gathering functions (shown in Figure 4.2) deal with the network QoS, user location, and user terminal capabilities aspects of the user and service operating context. The context control function is in charge of changing and controlling the user’s and service’s operating context as part of the context-aware service’s adaptation process. In our architecture, it specifically deals with the configuration of the network QoS. The context gathering function deals with gathering information about the user and service operating context. This includes gathering and requesting updates on the status of user’s location, the network QoS, and the user’s terminal capabilities. Sections 4.6, 4.7 and 4.8 will examine the mapping of the Context Control and the Context Gathering functions to the standardised components and internal interfaces of a 3G network.

4.6. Managing and Controlling Network QoS through Policies

3G networks incorporate a packet-switched core network to transport data, and it is possible for network operators to introduce different levels of Network QoS. This section presents 3GPP’s specifications involving policy-based management for network QoS. This content in this section does not represent any contribution in this thesis but is presented for completeness as network QoS is a key part of the service’s operating context. Subsection
4.6.2 does, however, discuss the implications of the 3GPP specifications to the contribution of the proposed service prioritisation method highlighted in subsection 3.5.1.5.

6 different QoS classes have been introduced by the 3GPP for 3G networks and they are created from 4 different UMTS traffic classes [45]. Table 4.3 shows the various QoS classes. The UMTS classes are: conversational, streaming, interactive, and background. The conversational UMTS traffic class provides a guaranteed bit rate and has a limit on maximum transfer delay. It caters for bandwidth intensive, delay sensitive, bi-directional traffic. The Streaming UMTS traffic class also provides a guaranteed bit rate but its limit on the maximum transfer delay is less stringent than the conversational UMTS class. It is used for bandwidth intensive, delay sensitive, downlink traffic. The Interactive UMTS traffic class was designed mainly for Internet surfing. It is split into 3 different best-effort QoS classes with different traffic handling priorities thus resulting in different network delays expected with each QoS class. There is no bandwidth guarantee associated with this class. The background UMTS traffic class is a best effort QoS class whereby the bandwidth guarantees and network delay are not a concern for the user’s application. This class was designed to be used with messaging applications similar to the Short Message Service (SMS) in GSM.

<table>
<thead>
<tr>
<th>QoS class</th>
<th>UMTS Traffic Class</th>
<th>Traffic Handling Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Conversational</td>
<td>N/A</td>
</tr>
<tr>
<td>B</td>
<td>Streaming</td>
<td>N/A</td>
</tr>
<tr>
<td>C</td>
<td>Interactive</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>F</td>
<td>Background</td>
<td>N/A</td>
</tr>
</tbody>
</table>

NOTE: QoS class represents the highest class that can be used for the bearer.

Armed with the ability of providing various levels of network QoS for a service session, a 3G network must decide the appropriate amount of network QoS subject to maximum authorised QoS allowed for the user. 3GPP has standardised the policy management interfaces and the network architectural components required for network QoS policy authorisation during a service session establishment and service session modification (for adaptation) [46]. The two interfaces which will be used primarily for policy-based management in a 3G network are:
- Go interface – This interface is used between the policy decision function and the policy enforcement point located in the GGSN. 3GPP has specified that COPS-PR will be the protocol used in this interface [46].
- Gq interface – This interface is used between the (service) application function and the policy decision function. This allows the policy provisioning model, whereby an external policy event can be fed into the policy decision function, to be employed in the 3G network. 3GPP has specified that the Diameter protocol is to be used for the Gq interface [47][48].

Figure 4.7 shows the Go interface architecture model as specified by 3GPP. Any authorised service application can interact with the 3G network's policy components using the Gq interface, and the Proxy Call Session Control Function (P-CSCF) in IMS has been identified as a likely "Application Function" candidate which will use the policy-based management functionality provided by the 3G network. Section 4.6.1 explains the Go interface architecture model in greater detail.

NOTE: For clarity in the diagram, network elements that are not involved in service-based local policy are not presented here (e.g. radio network elements, SGSN, etc).

Figure 4.7 Go Interface Architecture Model [46]

4.6.1 The Go Interface Architecture Model
The Gateway GPRS Support Node (GGSN) is responsible for the policy-based authorisation, i.e. to ensure that the requested QoS is in-line with the "Authorised QoS". The GGSN needs the "Authorised QoS" information of the Packet Data Protocol (PDP) context for the uplink as well as for the downlink direction. Therefore, the "Authorised QoS" information for the combination of all IP flows of each direction associated with the media component as determined by the PDF is used.
In case of an aggregation of multiple media components within one Packet Data Protocol (PDP) context, the "Authorised QoS" for the bearer is provided by the PDF as the combination of the "Authorised QoS" information of the individual media components.

The GGSN shall perform the proper mapping between the IP QoS information and the UMTS QoS information. This mapping is performed by the Translation/mapping function which maps the "Authorised QoS" information for the PDP context into authorised UMTS QoS information. Furthermore, the GGSN shall derive the highest allowed UMTS Traffic class for the PDP context from the QoS class in the "Authorised QoS" according to Table 4.3.

The QoS class values given by the PDF are equal for both the uplink and the downlink directions. The Data rate within the "Authorised QoS" information for the bearer is the combination of the data rate values of the "Authorised QoS" of the individual IP flows of the media components. In the case of real-time UMTS bearers (conversational and streaming traffic classes), the GGSN shall consider the Data rate value of the "Authorised QoS" information as the maximum value of the 'Guaranteed bit rate' UMTS QoS parameter, whereas the 'Maximum bit rate' UMTS QoS parameter is limited by the subscriber and service specific setting in the Home Location Register (HLR)/Home Subscriber Server (HSS) and by the capacity/capabilities/service configuration of the network. In the case of non-real-time bearers (interactive and background traffic classes) the GGSN shall consider, the Data rate value of the "Authorised QoS" information as the maximum value of the 'Maximum bit rate' UMTS QoS parameter.

The UMTS Bearer Service (BS) Manager receives the authorised UMTS QoS information for the PDP context from the Translation/mapping function. If the requested QoS exceeds the authorised QoS, the UMTS BS Manager shall downgrade the requested UMTS QoS information to the authorised UMTS QoS information.

The GGSN may store the authorised QoS for the binding information of an active PDP context in order to be able to make local decisions when the User Equipment (UE) requests for a PDP context modification.

The Gate Function represents a user plane function enabling or disabling the forwarding of IP packets. A gate is described by a set of packet classifiers that identify IP flows associated to the gate. The packet classifier, which is received from the PDF in an authorisation decision, includes the standard 5-tuple (source IP address, destination IP address, source port, destination port, protocol) explicitly describing a unidirectional IP flow.

The GGSN installs the packet filter corresponding to the packet classifier. The packet classifier includes the status that the gate shall be set to. The commands to open or close the gate lead to the enabling or disabling of the passage for IP packets. If the gate is closed all
packets of the related IP flows are dropped. If the gate is opened the packets of the related IP flows are allowed to be forwarded. The opening of the gate may be part of the authorisation decision event. The closing of the gate may be part of the revoke authorisation decision event.

The following policy decision point functionalities for Service-Based Local Policy (SBLP) are identified:

- **Authorisation function:**
The PDF shall be able to provide an authorisation decision upon receiving a bearer authorisation request from the GGSN. The PDF shall authorise the request according to the stored session and media related information received from the Application Function (AF).
The PDF shall use the binding information to determine the AF session(s) and the set of IP flows. Multiple sets of binding information and multiple AF sessions may be involved, if flows from separate AF sessions are multiplexed in the same PDP context. Based on the IP flows, the PDF shall determine the authorised QoS, packet filters, and gate status to be applied. The authorised QoS specifies the maximum allowed QoS class, and the data rate for the set of IP flows identified in the binding information.
The PDF shall be able to provide updates to the authorisation decision, if receiving modified service information from the AF which changes the QoS and packet classifiers for PDP contexts which are already established.

- **Revoke function:**
The PDF may revoke the authorisation of resources at any time. Revoke Authorisation for General Packet Radio Service (GPRS) and IP resources is communicated by the PDF to the GGSN.

- **Approval of QoS Commit / Removal of QoS Commit:**
The PDF may allow or deny the usage of the PDP context for the selected IP flow(s) by controlling the correlated gate(s). The "Approval of QoS Commit" command may either be part of the authorisation decision, or the PDF may provide a separate decision with the "Approval of QoS Commit" command to open the gate. The "Removal of QoS Commit" command is a separate decision to close the gate(s) e.g. when a media IP flow(s) is put on hold.

- Actions due to Indication of bearer release:
When the GGSN informs the PDF of bearer deactivation, the PDF shall remove the corresponding authorisation request state. Additionally, the PDF shall inform the AF about this deletion event.

- Actions due to Indication of bearer modification:
When the PDF receives an indication of bearer modification of the maximum bit rate to or from 0 kbits/s, the PDF shall inform the AF about this modification event.

- **Generation of authorisation token:**
  The PDF generates an authorisation token for the AF session as specified in 3GPP's specification for the Gq interface [47]

- **Mapping service information to "Authorised QoS" parameters:**
  To perform proper authorisation, the PDF shall map the necessary service information containing session and media related information to "Authorised QoS" parameters.

- **Charging identifiers exchange:**
  The PDF shall send the AF charging information, if provided by the AF, as part of the initial authorisation decision(s) for all the bearer authorisation request(s) that correspond to the respective AF session.

### 4.6.2 Other Benefits regarding the Go Interface

Section 3.5.1.5 discussed the need for prioritising services when the user's network resources were limited. The key requirement for enabling the prioritising of services is the ability to detect when service sessions are started and terminated. The Go interface fulfils this requirement by seeking authorisation when a new PDP context is established, and can provide notification when an existing bearer is modified or terminated. Since the IP flow information is required when establishing a PDP context, one needs only to match the IP flow information to a service's IP address (stored in a repository such as the GUP) to identify the service being accessed. This makes it possible to track the number of active services that the user currently uses at any time, therefore making service prioritisation possible.

### 4.7. Policy-based management and 3GPP's Location Service (LCS)

The Location Service (LCS) framework was developed by the 3GPP to enable 3G networks to support location-aware services [49]. The LCS can be accessed through the Open Service Access (OSA) Mobility Service Capability Feature interfaces or by using the Le interface via an external LCS client. The most important component within the LCS from a LCS client's perspective is the Gateway Mobile Location Centre (GMLC). The GMLC is the first node an external Location Application accesses to use the LCS in a 3G network. The GMLC uses many reference interfaces to communicate with other components of the LCS and the 3G network to manage location positioning requests from LCS clients. Figure 4.8 shows the general architecture of the LCS and the GMLC's role within the LCS. The Le and the Lg interfaces are the more significant interfaces in the LCS architecture as they are used to

convey the location data of the user to the LCS client. The Lg interface is used by the GMLC to convey a location request to the Mobile Switching Centre (MSC) or Serving GPRS Support Node (SGSN) currently serving a particular target User Equipment (UE) whose location was requested. The interface is used by the MSC or SGSN to return location results to the GMLC.

Figure 4.8 General Arrangement of the LCS [49]

The GMLC may request routing information from the HLR or HSS via the Lh interface. After performing registration authorisation, it sends positioning requests to either the SGSN or MSC Server and receives final location estimates from the corresponding entity via the Lg interface. Information needed for authorisation, location service requests and location information may be communicated between GMLCs, located in the same or different PLMNs, via the Lr interface. The target UE's privacy profile settings shall always be checked in the UE's home Public Land Mobile Network (PLMN) prior to delivering a location estimate.

The GMLC receives requests for the user’s location either via the Le interface from a LCS client or via the OSA mobility Service Capability Server (SCS) from an OSA client. The Le interface is compatible with the Location Interoperability Forum’s (LIF) Mobile Location Protocol (MLP) [52]. It is useful to discover the capabilities of the GMLC from a location-client application's perspective by examining both the Le interface, and the interface
between the GMLC and the OSA Mobility SCS. Sections 4.7.1 and 4.7.2 discuss the two interfaces in detail respectively, and section 4.7.3 compares the two interfaces for common functionalities and significant differences. Sections 4.7.3 to 4.7.5 represent the contribution in this thesis in finding an appropriate mapping for the location aspect of the context-gathering functionality of the proposed policy-based management framework.

4.7.1 OSA Mobility API

The OSA Mobility API is one of several APIs in the OSA framework available to OSA clients and is supported by an OSA Mobility Service Capability Server (SCS) [51]. The direct interface between a GMLC and an OSA Mobility SCS is proprietary and is not standardised by 3GPP (as shown in Figure 4.8). The OSA Mobility API was designed to support location-aware applications through 5 main API methods:

a) Request Location Report.
This method is used by an application to request a location report on the user to the LCS on demand.

b) Request Extended Location Report
This is similar to a normal location request but includes specific requirements on the accuracy of the location positioning in the location report, and a limit on the maximum response time for the LCS to serve the location request. The maximum response time serves as a “Quality of Service” parameter.

c) Periodic Location Reports (Start and Stop)
The location application can start or terminate the receipt of periodic location reports on the user from the LCS. The application includes a parameter representing the amount of milliseconds between each location report when requesting a periodic location report from the LCS.

d) Triggered Location Reports (Start and Stop)
The location application can specify a geographical region in which it wishes to be informed by the LCS if the user is entering or leaving the specified geographical region.

e) User Status Reports
The location application uses this feature to discover if the user is reachable (i.e. if the user’s terminal is switched off or not working).

4.7.2 The LIF Mobile Location Protocol

The Mobile Location Protocol (MLP) was designed and specified by the Location Interoperability Forum (LIF) to be completely compliant the the Le interface between the GMLC and a LCS client application in the 3GPP LCS framework [52]. The MLP is
essentially a XML-based framework at the application level and its specification deals largely with the definition of the XML elements which is used in the information transfer across the Le interface. show a general overview of the MLP framework. On the lowest level, the transport protocol defines how XML content is transported. Possible MLP transport protocols include HTTP, Wireless Session Protocol (WSP), SOAP and others.

The Element Layer defines all common elements used by the services in the service layer. Table 4.4 shows the current defined set of Document Type Definitions (DTDs) making up the element layer of MLP:

Table 4.4 Set of DTDs defined for the MLP Element Layer

<table>
<thead>
<tr>
<th>DTD Name</th>
<th>MLP Element Layer Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP_ID.DTD</td>
<td>Identify Element Definitions</td>
</tr>
<tr>
<td>MLP_FUNC.DTD</td>
<td>Function Element Definitions</td>
</tr>
<tr>
<td>MLP_LOC.DTD</td>
<td>Location Element Definitions</td>
</tr>
<tr>
<td>MLP_RES.DTD</td>
<td>Result Element Definitions</td>
</tr>
<tr>
<td>MLP_SHAPE.DTD</td>
<td>Shape Element Definitions</td>
</tr>
<tr>
<td>MLP_QOP.DTD</td>
<td>Quality of Position Element Definitions</td>
</tr>
<tr>
<td>MLP_GSM_NET.DTD</td>
<td>GSM Network Parameters Element Definitions</td>
</tr>
<tr>
<td>MLP_CTXT.DTD</td>
<td>Context Element Definitions</td>
</tr>
</tbody>
</table>

Some of the descriptions of the DTDs in Table 4.4 are self-explanatory though a few of them require further clarification. The Identify Element Definitions specifies the XML tags that identify the end user with which to apply the location positioning requests and results. The Function Element Definition specifies the XML tags that deals with the characteristics of a location request e.g. priority, start time, stop time, location (e.g. current, last etc.). The Result Element Definitions specifies the XML tags that contain the result and the result codes of the positioning request e.g. Method Not Supported. The Context Element Definitions specifies the XML tags that contain the identity and authorisation of the MLP client requesting location request on the end user.

The Service Layer defines the actual services offered by the MLP framework. Basic MLP Services are based on location services defined by 3GPP, while Advanced MLP services have not been specified by the LIF yet.
There are a number of different possible types of location services. Each implementation of location server can select which services it wants/needs to support. The basic MLP services are described in Table 4.5. A mapping between MLP services and the location procedures defined in 3GPP’s specification regarding the functional description of the Location Services framework is also provided in Table 4.6.

Table 4.5 Basic MLP Services

<table>
<thead>
<tr>
<th>Service</th>
<th>Description</th>
</tr>
</thead>
</table>
| Standard Location Immediate Service | This is a standard query service with support for a large set of parameters. This service is used when a (single) location response is required immediately (within a set time) or the request may be served by several asynchronous location responses (until a predefined timeout limit is reached). This service consists of the following messages:  
  - Standard Location Immediate Request  
  - Standard Location Immediate Answer  
  - Standard Location Immediate Report |
| Emergency Location Immediate Service | This is a service used especially for querying of the location of a mobile subscriber that has initiated an emergency call. The response to this service is required immediately (within a set time) or the request may be served by several asynchronous location responses.  
  This service consists of the following messages:  
  - Emergency Location Immediate Request  
  - Emergency Location Immediate Answer  
  - Emergency Location Immediate Report |
| Standard Location Reporting Service | This is a service that is used when a mobile subscriber wants an LCS Client to receive the MS location. The position is sent to the LCS Client from the location server. Which LCS application and its address are specified by the MS or defined in the location server.  
  This service consists of the following message:  
  - Standard Location Report  
  - Standard Location Report Answer |
Emergency Location Reporting Service

This is a service that is used when the wireless network automatically initiates the positioning at an emergency call. The position and related data is then sent to the emergency application from the location server. Which LCS application and its address are defined in the location server. This service consists of the following message:
- Emergency Location Report

Triggered Location Reporting Service

This is a service used when the mobile subscriber’s location should be reported at a specific time interval or on the occurrence of a specific event.
This service consists of the following messages:
- Triggered Location Reporting Request
- Triggered Location Reporting Answer
- Triggered Location Report
- Triggered Location Reporting Stop Request
- Triggered Location Reporting Stop Answer

Table 4.6 Mapping between MLP and LCS

<table>
<thead>
<tr>
<th>Location procedures defined in 3GPP [49]</th>
<th>Services defined in MLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circuit Switched Mobile Terminating Location Request CS-MT-LR</td>
<td>LCS Service Request</td>
</tr>
<tr>
<td>CS-MT-LR without HLR Query - applicable to North America Emergency Calls only</td>
<td>LCS Service Response</td>
</tr>
<tr>
<td>Packet Switched Mobile Terminating Location Request PS-MT-LR</td>
<td>LCS Service Request</td>
</tr>
<tr>
<td>Network Induced Location Request NI-LR</td>
<td>LCS Service Response</td>
</tr>
<tr>
<td>Packet Switched Network Induced Location Request PS-NI-LR</td>
<td>Location Information</td>
</tr>
<tr>
<td>Mobile Terminating Deferred Location Request</td>
<td>LCS Service Request</td>
</tr>
<tr>
<td>Mobile Terminating Deferred Location Request</td>
<td>LCS Service Response(Provide Subscriber Location ack)</td>
</tr>
<tr>
<td>Mobile Terminating Deferred Location Request</td>
<td>LCS Service Response(Subscriber Location Report)</td>
</tr>
<tr>
<td>Combined Periodical/Deferred Mobile Terminating Location Request</td>
<td>LCS Service Request</td>
</tr>
<tr>
<td>Combined Periodical/Deferred Mobile Terminating Location Request</td>
<td>LCS Service Response(Provide Subscriber Location ack)</td>
</tr>
<tr>
<td>Combined Periodical/Deferred Mobile Terminating Location Request</td>
<td>LCS Service Response(Subscriber Location Report)</td>
</tr>
</tbody>
</table>
4. Policy-based Management Framework for Context-aware Services in 3G Networks

<table>
<thead>
<tr>
<th>Location procedures defined in 3GPP [49]</th>
<th>Services defined in MLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancellation of a Deferred Location Request</td>
<td>LCS Cancel Service Request</td>
</tr>
<tr>
<td></td>
<td>Triggered Location Reporting Stop Request</td>
</tr>
<tr>
<td></td>
<td>LCS Cancel Service Response</td>
</tr>
<tr>
<td></td>
<td>Triggered Location Reporting Stop Answer</td>
</tr>
<tr>
<td>Mobile Originating Location Request, Circuit Switched CS-MO-LR</td>
<td>Location Information</td>
</tr>
<tr>
<td></td>
<td>Standard Location Report</td>
</tr>
<tr>
<td></td>
<td>Location Information Ack</td>
</tr>
<tr>
<td></td>
<td>Standard Location Report Answer</td>
</tr>
<tr>
<td>Mobile Originating Location Request, Packet Switched PS-MO-LR</td>
<td>Location Information</td>
</tr>
<tr>
<td></td>
<td>Standard Location Report</td>
</tr>
<tr>
<td></td>
<td>Location Information Ack</td>
</tr>
<tr>
<td></td>
<td>Standard Location Report Answer</td>
</tr>
</tbody>
</table>

4.7.3 Comparison between OSA Mobility API and LIF-MLP

A quick glance at the features of the OSA mobility API and the MLP described in the previous 2 sections reveal that they are very similar in functionality with one exception; the OSA Mobility API offers the ability to provide periodic location updates, whereby the location application can start or terminate the receipt of periodic location reports on the user from the LCS. By deduction, one can conclude that this is an additional feature provided by an OSA mobility SCS, and that this functionality is not inherent in a GMLC. A LCS client using the Le interface (e.g. through the MLP) can replicate this functionality through a combination of using a timer (to time the period between requests) and the Standard Location Immediate Request basic MLP service. Otherwise, there is no significant functional difference in using either the OSA mobility API or the MLP.

4.7.4 Industrial Factors Influencing the Usage of the OSA Mobility API and the MLP

Having examined both the OSA Mobility APIs and the LCS Le interface, it appears that there is no great difference in functionality between both approaches to perform location positioning related tasks. It is important, though, to examine other current industrial initiatives and specifications and their possible impact on the choice of interface for the policy-based management framework.

The Open Geospatial Consortium (OGC) is an international industry consortium of 342 companies, government agencies and universities. Its main aim is to aid the deployment of location-based services by promoting interoperability through the development of open solutions and interface specifications [53]. The OGC has released an implementation specification of the OpenGIS® Location Services which targets its main aim of promoting interoperability. The OpenGIS® Location Services implementation, which is termed the GeoMobility Server, is shown in Figure 4.10.
The GeoMobility Server incorporates the 5 core parts of the OpenGIS® Location Services specification: The directory service, the gateway service, the location utility service (geocode/reverse geocode), the presentation service, and the route service. The directory service provides access to an online directory to find the nearest or a specific place, product or service. The gateway service uses the MLP to obtain position data of 3G mobile users from the 3G networks LCS. The location utility service performs as a geocoder by determining a geographical position, given a place name, street address or postal code. It also performs as a reverse geocoder by determining a complete normalised place name, street address or postal code given a geographical position. The presentation service renders geographical information for display on a mobile terminal. Finally, the route service determines a navigation route for a mobile user between two different geographical locations.

4.7.5 Discussion on the Deployment of Location-Aware Services in 3G Networks

From the previous section, it would appear that the OCG’s OpenGIS® Location Services implementation, i.e. the GeoMobility Server, offers better support and features for service developers than the alternative OSA mobility APIs. Since the OSA mobility API has similar functionalities as the MLP, which the GeoMobility Server uses, it is conceivable that the OCG’s framework would be more attractive to developers of location-aware services. It is important, though, to remember that the GeoMobility Server and the OSA mobility APIs were designed and developed for different purposes; the OSA mobility APIs provides...
location positioning functionalities in the 3G network from the LCS, whereas the GeoMobility Server aims to provide a complete solution to deploying location-aware services but requires assistance from the LCS to obtain the mobile user’s location. The following points present the various factors if one was to use the OSA mobility API approach over the policy-based management via the MLP/Le interface approach:

- There are less management overheads for a location-aware service to use the MLP instead because the MLP connects a LCS client directly to the GMLC. If the OSA mobility API is used by a location-aware service, then there are the additional overheads from the requirement of the service to discover and bind to the OSA mobility service capability server.

- Using the OSA mobility APIs negates the need for a policy managed location-aware service to use the functionalities provided by the OSA policy management API. It is still possible to build a policy management framework on top of the OSA though the business models and factors discussed in sections 3.2 and 3.3 have highlighted the disadvantages of this approach. Not using the OSA policy management APIs also makes it difficult to deploy location-aware services that are adaptable to other aspects of a user’s context. For example, the OSA policy management API currently provides the only means for the service to manage the network QoS according to its adaptation requirements as there is no “network QoS” API available from the OSA.

- The OSA does not provide numerous functionalities to deploy location-aware services compared to dedicated frameworks such as the OCG’s GeoMobility server. However, this may also be seen as a benefit as a more lightweight customised framework with less management overheads can be developed and deployed on top of the OSA if a service provider only needs a small subset of functional features that the GeoMobility server provides.

- Policy-based management offers a more graceful automated usage of location triggering and location positioning requests from the LCS. Consider the following policy rule (written in pseudo code)

```
If user is in location A, then perform service adaptation action A
```

The initiation of a location aware service by the user immediately requires the evaluation of this rule during service configuration, thus generating an automated location positioning request from the LCS. This is done by the location requesting a policy evaluation from the policy evaluation service capability server via the OSA policy management API.

If the above policy rule example is for a push service, then the service must be triggered if the user enters a location. This requires a Triggered Location Reporting Request from the policy management framework to the LCS. This is automatically performed by the policy management framework when the policy rule was initially enabled (as stated in [42]).

The three factors above present a balanced argument on the benefits of using one approach over the other. The conclusion is that the using the OSA mobility API for managing the adaptation of a location-aware service requires less implementation effort and complexity and is advantageous for location-aware services which do not consider other aspects of the user's context. A policy-based approach via the MLP interface, however, provides the means for location-aware services that can adapt to multiple aspects of the context (e.g. terminal capabilities and network QoS) to specify the complex adaptation strategies in a flexible manner.

4.8. Extracting Terminal Capabilities via the 3G network

There are 2 general methods for a 3G network to determine a user's terminal capabilities. Firstly, the user may utilise the Generic User Profile (GUP) to store the user's terminal capability. This method is efficient when the user primarily operates a single terminal and does not switch terminals often as the user profile was designed to store information that does not require frequent updates. If the 3G network requires information on the user's terminal capabilities, it would then access the user's profile to retrieve it. The second method requires an information exchange mechanism between the terminal and the network for the terminal to declare its capabilities on demand. Both methods require that the schema of the terminal capabilities information be standardised (e.g. through a schema or vocabulary). The CC/PP framework provides terminals a standard vocabulary to describe their capabilities and the framework has been incorporated into the Open Service Access (OSA) interfaces [35][54]. User terminals are able to push information about its capabilities to a server using either the Wireless Profile HTTP (W-HTTP) protocol, Real-Time Streaming Protocol (RTSP) or CC/PP Exchange protocol, thus allowing 3G networks to obtain the user's terminal capabilities on demand if required [56][57][36]. W-HTTP and CC/PP Exchange Protocol are both based on HTTP but contain additional headers to insert terminal capabilities information.

4.8.1 CC/PP

The CC/PP vocabulary was created by the World Wide Web Consortium (W3C) to tackle the generic problem of publishing capability and preference information so as to allow
customizable and adaptable services to be delivered to consumers. It is based on the Resource Description Framework (RDF) and can therefore be seen as applying W3C's semantic web initiative whose goal is to "smoothly interconnect personal information management, enterprise application integration, and the global sharing of commercial, scientific and cultural data" [58][59].

In order to fulfil this goal, RDF itself is usually serialised into XML through an RDF model and syntax specification [60]. RDF is a framework with which to describe web resources through a directed graph model theory, whereby all information is represented as either a node or a property arc (i.e. an edge). A node represents either a resource (not necessarily web-based) or a literal, and a property arc represents a property of a resource.

CC/PP uses the RDF serialisation specification to publish information about device capabilities in XML. For example, it is possible to declare that a CC/PP component (RDF resource) has a screen size (RDF Property) of dimension '128x96' (RDF Literal). More specifically, the CC/PP vocabulary defines the structure of a CC/PP document and the data types of CC/PP attributes that can be used in the document (e.g. Integer, Sets of String Literals etc.). Every CC/PP-based profile has an internal hierarchy as follows: a CC/PP profile is used to aggregate one or more CC/PP components; a CC/PP component is then used to aggregate one or more CC/PP attributes.

The CC/PP vocabulary, however, does not specifically declare the attributes that device must support (e.g. Colour Capable etc.). This is usually done through extending the CC/PP vocabulary to create another more specific and appropriate vocabulary to describe the capabilities and preferences in a particular domain. Many CC/PP derived vocabularies do exist but the Open Mobile Alliance's User Agent Profile (UAProf) vocabulary is most comprehensive for describing terminal capabilities and was used for the proposed framework prototype [61]. The UAProf vocabulary was designed specifically for Wireless Application Protocol (WAP) devices to communicate their capabilities to WAP gateways so as to allow 'pushed' and 'pulled' content to be adapted and delivered to the consumer. Furthermore, a large proportion of the UAProf vocabulary is also applicable to non-WAP devices.

**4.8.2 UAProf**

There have been 6 different versions of the UAProf vocabulary since the year 2000, and 7 different UAProf components types have been defined so far. They are as follows: HardwarePlatform, SoftwarePlatform, BrowserUA (to describe the capabilities of the terminal browser), NetworkCharacteristics, WapCharacteristics, PushCharacteristics, and MiscCharacteristics (to describe the capabilities of the
terminal with regards to using Multimedia Messaging Services). UAProf builds on the CC/PP vocabulary by defining the attributes that can exist within each UAProf component (e.g. `BrowserName` belongs to the `BrowserUA` component), and associates each attribute value with a datatype (e.g. `BrowserName` can only be defined using a XML Literal). There are 91 different attributes described in the UAProf vocabulary that can be used to construct a UAProf terminal capability profile. A valid UAProf device capability profile, however, need not declare all 91 attributes – it may even contain 1 attribute only.

The UAProf specification also provides guidelines on combining many individual device capability profiles to form a single resolved terminal profile. This is useful when a consumer accesses the service through a personal area network containing many devices (e.g. a laptop with Bluetooth connection to a 3G phone accessing a 3G network). A terminal transmits the various device profile information as an ordered list within a special header of a HTTP-based request. When combining and resolving different device profiles, the UAProf specification stipulates that every UAProf attribute has a resolution rule that is declared as `locked`, `append` or `override`, as explained below:

a. **locked** - the resolved attribute value is that of the first occurrence of the attribute in the ordered list;
b. **append** - the resolved attribute value is the union of all values of the specific attribute in the list;
c. **override** - the resolved attribute value is that of the last occurrence of the attribute in the ordered list.

### 4.8.3 Processing CC/PP and UAProf

If a user terminal decides to use CC/PP or UAProf to declare its capabilities, there still exists the issue of deploying a (software) processor that is capable of interpreting CC/PP and UAProf. Sun Microsystems has taken the lead in this area by specifying a Java standard API (JSR-188) for processing CC/PP and UAProf [62]. Other Java-based APIs for processing CC/PP do exist, such as the DELI (Delivery Context Library), but they do not have as much industrial backing as Sun Microsystems’s offering [63]. The proposed framework, therefore, utilises a JSR-188 compliant CC/PP processor to obtain the user’s terminal capabilities on demand [108]. The inclusion of the JSR-188-compliant CC/PP processor represent the contribution in this thesis in finding an appropriate mapping for the location aspect of the context-gathering functionality of the proposed policy-based management framework, bearing in mind that the processor’s inclusion is not part of any 3GPP specifications. This short subsection will explain the inner workings of Sun Microsystems’s JSR-188 processor. Figure 4.11 shows a UML diagram featuring the Java API interfaces which are used to
represent the terminal's capabilities. JSR-188 specifies that all queries regarding terminal capabilities are performed through methods defined in these standardised Java interfaces. The purposes of the various methods in the classes are self-explanatory as evident by their naming convention.

In the JSR-188 specifications, profiles are created via the newProfile method in a ProfileFactory class which implements the abstract factory design pattern [64]. The newProfile method in the ProfileFactory class allows various different object types as arguments to create a profile. The various Java object types which can be used to create a profiles object include javax.servlet.http.HttpServletRequest, java.io.InputStream and java.net.URL. The ability of the ProfileFactory class to process a javax.servlet.http.HttpServletRequest object provides a JSR-188 compliant processor a direct method of create a profile objects from examining HTTP headers.
4.8.4 Discussion on the Usage of the OSA Terminal Capabilities API and the JSR-188 CC/PP processor.

The OSA currently has an API, the terminal capabilities API, for OSA clients to obtain the user’s terminal capabilities. Table 4.7 shows the TpTerminalCapabilities datatype returned by the OSA framework when an OSA client requests for a user’s terminal capabilities [55]. The documentation for the sequence element name in the TpTerminalCapabilities datatype clearly shows that the OSA only returns either the CC/PP headers or an unparsed RDF CC/PP or UAProf file to the OSA client. The OSA does not process the CC/PP headers or the RDF file, and therefore a CC/PP processor such as the JSR-188 is still required to extract the individual terminal capabilities (e.g. the terminal’s screen size). The policy management framework, therefore, requires the incorporation of a CC/PP or UAProf processor so that policy conditions dealing with individual terminal capabilities attributes can be detected allowing the appropriate policy actions to be performed.

Table 4.7 OSA TpTerminalCapabilities datatype

<table>
<thead>
<tr>
<th>Sequence Element Name</th>
<th>Sequence Element Type</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TerminalCapabilities</td>
<td>TpString</td>
<td>Specifies the latest available capabilities of the user's terminal. This information, if available, is returned as CC/PP headers as specified in W3C and adopted in the WAP UAProf specification. It contains URLs, terminal attributes and values, in RDF format, or a combination of both.</td>
</tr>
<tr>
<td>StatusCode</td>
<td>TpBoolean</td>
<td>Indicates whether or not the TerminalCapabilities are available.</td>
</tr>
</tbody>
</table>

4.9. The Generic User Profile and the Service Control Function

The Generic User Profile is a 3GPP specification that deals with the management, synchronisation, user access and storage of the user’s service and network subscription information, personal preferences and service customisation information [65]. The GUP allows users to customise and store the user’s preferences for prioritising different services as explained in section 3.5.1.5. The GUP specification allows for the distribution of the user profile across different entities, e.g. user terminal and network, and therefore there are multiple instances of it shown in Figure 4.2.

The Service Control function shown in Figure 4.2 considers the scenario whereby network operators may use policies to manage in-house value-added services. The PEP can therefore perform policy actions on the service application without going through the OSA.

4.10. Discussion and Summary

The OSA policy management APIs specification in its current form only provides a shell with no proposed mappings to the 3G network to provide the specified functionalities. This chapter described the proposed numerous functional entities in a policy-based management framework for a 3G network that can realise the OSA policy management APIs while
considering strict compliance with existing 3G network specifications. Figure 4.12 shows the expanded view of the proposed policy management framework (shown in Figure 4.2) with details of the specific mappings proposed in this chapter between the policy management framework and the components in a 3G network that are responsible for gathering and controlling the user’s context.

![Diagram of Policy Management Framework](image)

Figure 4.12 Bindings between the Policy Management Framework and the Context Gathering and Control Functions in a 3G Network
5. Modelling Policies for Context-Aware Service Management in 3G Networks

5.1. Introduction

It is envisioned that the traffic of mobile telecommunications will be delivered with an all-IP network in the near future [66]. Popular current circuit-switched services will migrate to their equivalent IP-based version in the future as part of the 3GPP IP Multimedia Subsystem (IMS) [67][68]. One could, therefore, design a policy information model for service management with the consideration that all future 3G services will have the common characteristic of being IP-based.

It is tempting to over-specify an information model for managing context-aware services for 3G networks. For example, one could define some specific policy actions classes for all context-aware services. The implications of doing this would be to specify that every service management application have to support a certain API or logic to support this policy action. This would be contrary to many current open standards and specifications for flexible service delivery, e.g. W3C's web services, Web Services Description Language, W3C's Semantic Web initiative, and the Web Services Policy [69][70][59][71]. The definition of common specific policy action classes would also hinder the adoption of dynamic service composition frameworks which allow services to declare their interfaces to be discovered dynamically [72][73].

3GPP has not specified any specific policy information model classes for usage with its OSA policy management interfaces with the exception of network Quality of Service (QoS). This chapter presents and explains the design of the policy information model for other aspects of the user's context so as to allow their incorporation into the management of context-aware services through OSA policy management interfaces. Sections 5.2 and 5.3 present key background information on the IETF's policy core information model and policy information models for network QoS. Sections 5.4 and 5.5 present a key contribution in this thesis: the design of a policy information model classes for representing the user, the service, the user's terminal capabilities, and the user's location.

5.2. Modelling Policies through the Policy Core Information Model

A policy can be expressed through a rule-based notation where the occurrence of a set of conditions will effect a set of actions. A policy expressed in this way is often referred to as a policy rule. The set of conditions in a policy rule can be expressed as either an ORed set of ANDed sets of condition statements or an ANDed set of ORed sets of statements. Individual

Condition statements can also be negated. These combinations are termed, respectively, Disjunctive Normal Form (DNF) and Conjunctive Normal Form (CNF) for the conditions [8].

The Policy Common Information Model (PCIM), developed jointly by the IETF and the Distributed Management Task Force (DMTF), defines a set of classes and properties with which one can model policies. PCIM was created by extending DMTF's Common Information Model (CIM) [74]. Figure 5.1 shows a subset of the PCIM core class hierarchy including the extensions introduced in January 2003, often referred to as Extensions to Policy Common Information Model (PCIMe) [75]. In Figure 5.1, the PCIMe CompoundPolicyCondition and CompoundPolicyAction classes serve as containers to hold multiple SimplePolicyCondition and SimplePolicyAction classes respectively, and allow policies to be declared in DNF and CNF combinations in an extensible manner. Most of the other classes in Figure 5.1 are named less ambiguously and will not be explained in this section ([8] and [75] provide a more detailed explanation and definition of the PCIM and PCIMe classes).

![Figure 5.1 A Subset of the Core Policy Classes Defined in PCIMe](image)

5.3. Policy Information Model for Network Quality of Service

This thesis does not attempt reinvent the wheel with regards to creating a policy information model for network quality of service as there are a few detailed specifications from the IETF
and 3GPP which deals with this aspect. The IETF has released a Quality of Service Policy Information Model (QPIIM) which covers both Reservation Protocol (RSVP) signalling and the Differentiated Services (DiffServ) QoS architecture [76]. The DiffServ architecture is used in a significant part of the 3G core network to provide network Quality of Service. The IETF has also defined a Policy Information Base (PIB) for Diff Services networks, while 3GPP has defined a reference interface (named the Go interface) between the GGSN and the Policy Decision Function (PDF) [46]. The Go interface specifies how the GGSN and the PDF interacts using the Common Open Policy Service (COPS) protocol to communicate policy management information. A PIB has also been defined by the 3GPP specifically for the Go interface. The GGSN can request for network QoS authorisation from the PDF through the Go interface. Policies can also be provisioned by the PDF to the GGSN through the Go interface. For example, the PDF can request the modification of a user's current PDP context with a new set of QoS parameters. External Application Functions can also provision policies via the PDF by using the Gq interface. Figure 5.2 shows a use-case sequence of resource reservation and QoS authorisation through service-based local policies in a 3G network for the activation of a secondary PDP context as defined by 3GPP [45].

1) The UE sends an Activate (Secondary) PDP Context message to the SGSN with the UMTS QoS parameters. The UE includes the Binding Information in the Activate PDP Context message.

2) The SGSN sends the corresponding Create PDP Context message to the GGSN.
3) The GGSN sends a COPS REQ message with the Binding Information to the PDF in order to obtain relevant policy information.

4) A PDF generated authorisation token enables the PDF to identify the authorisation status information. If the previous PDF interaction with that AF had requested this, or if the previous interaction with the AF did not include service information, the PDF sends an authorisation request to that Application Function.

5) The AF sends the service information to the PDF.

6) The PDF shall authorise the required QoS resources for the AF session if the session description is consistent with the operator policy rules defined in the PDF, and install the IP bearer level policy in its internal database. This is based on information from the Application Function.

7) The PDF sends a COPS DEC message back to the GGSN.

8) The GGSN sends a COPS RPT message back to the PDF, which may also trigger a report message to be sent from the PDF to the AF.

9) The GGSN maps IP flow based policy information into PDP context based policy information and uses the PDP context based policy information to accept the PDP activation request, and sends a Create PDP Context Response message back to SGSN.

10) Radio Access Bearer (RAB) setup is done by the RAB Assignment procedure.

11) The SGSN sends an Activate (Secondary) PDP Context Accept message to UE.
5.4. Context-related Policy Condition Classes

This thesis considers the 4 main aspects of the user’s context: the user’s terminal capability, the user’s location (for location-aware services), the user’s network Quality of Service (QoS), and the user’s current active services. The previous section describes briefly the 3GPP specification regarding the modelling of policies for the network QoS. The following subsections explore the definition of Policy Condition classes and variables for other aspects of the user’s context for use in the proposed policy-based framework. These policy condition classes are associated with specific policy variable classes which represent aspects of the user’s context. Figure 5.3 shows the main Policy Information Model classes related to policy conditions for context-aware services in 3G networks.

![Figure 5.3 The Policy Condition Class Hierarchy](image)
5.4.1 The Abstract ContextCondition Class

The abstract PolicyCondition class from the PCIM was extended to include an abstract ContextCondition class. The ContextCondition class serves as a superclass for all the policy condition related classes in the system. Its only property, ContextType, is used to describe the context aspect which the condition is associated with (e.g. location).

There are 3 main policy condition classes which extend from this abstract ContextCondition class: TerminalCapabilitiesCondition, ServiceUsageCondition, and GeographicalLocationCondition.

5.4.2 The ServiceAccessCondition Class

The ServiceAccessCondition class is used to model the number of concurrent service sessions that a user has. Section 3.5.1.5 discussed the applicability of policies to prioritise services when the user has more than one active service session as there is limited amount of resources available at any time to run a service (e.g. network Quality of Service). It is thus important to detect changes in a user’s service session, e.g. initiation, termination, in order effect the management actions required to fulfil the user’s preference and policies for service prioritisation. As 3G networks migrates towards an all-IP infrastructure [66], detecting active service sessions would require the detection of user access to an IP-based service access point for that particular service (e.g. IP-based IMS node). Active service sessions can be easily detected as a PDP context has to be activated by the Gateway GPRS Support Node (GGSN) in a 3G network to support the service session. This means that there are no additional requirements or modifications needed in a 3G core network to support the detection of concurrent active service sessions. The ServiceAccessCondition class aggregates a ServiceSessionIsActiveVariable class and a PolicyBooleanValue class. The ServiceSessionIsActiveVariable is defined to be used to specify policy conditions when a particular service session belonging to a service is active.

ServiceAccessCondition class:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServiceType: (String CIM datatype)</td>
<td>A String describing the type of the service (e.g. Voice over IP call)</td>
</tr>
<tr>
<td>ServiceID: (uint16 CIM datatype)</td>
<td>An unsigned integer describing the ID number of the service</td>
</tr>
<tr>
<td>ServiceIPAddress (PolicyIPv6AddrValue datatype)</td>
<td>The (Destination) IP Address of the Service</td>
</tr>
<tr>
<td>ServicePortNumber: (uint16 CIM datatype)</td>
<td>The (Destination) Port Number of the Service</td>
</tr>
<tr>
<td>ServiceProtocolID: (uint16 CIM datatype)</td>
<td>The Protocol ID used by the service</td>
</tr>
</tbody>
</table>
5.4.3 The GeographicalLocationCondition Class

The GeographicalLocationCondition class designed with a view that it must be compatible with the current OSA specifications for the mobility service. The OSA mobility service interface allows services to specify the conditions to be notified concerning a change in user location [51]. By ensuring compatibility with the OSA specifications, service providers can then have a choice between using policy-based management (i.e. via a PDF) and the OSA mobility service capability feature (SCF) interfaces to perform location positioning tasks.

The GeographicalLocationCondition aggregates the HorizontalLocationVariable, the UsedLocationVariable, and AltitudeVariable class. A new policy value class, HorizontalLocationValue, was created to represent a HorizontalLocationVariable used in a GeographicalLocationCondition. The UsedLocationVariable is used to specify the location positioning technique used in ascertaining the user’s location (e.g. Assisted-Global Positioning System). This variable is used by the service to instruct the LCS via policies the preferred location accuracy for the service to perform optimally [107]. The AltitudeVariable is used to specify the user’s altitude. Figure 5.4 explain in greater detail the properties of the HorizontalLocationValue class.

HorizontalLocationValue class:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Longitude (real32 CIM datatype)</td>
<td>Longitude of the position used in the trigger.</td>
</tr>
<tr>
<td>Latitude (real32 CIM datatype)</td>
<td>Latitude of the position used in the trigger.</td>
</tr>
<tr>
<td>AreaSemiMajor (real32 CIM datatype)</td>
<td>Semi major of ellipse area.</td>
</tr>
<tr>
<td>AreaSemiMinor (real32 CIM datatype)</td>
<td>Semi minor of ellipse area.</td>
</tr>
<tr>
<td>AngleOfSemiMajor (uint16 CIM datatype)</td>
<td>Angle of the semi major of the ellipse area.</td>
</tr>
<tr>
<td>Criterion (uint16 CIM datatype)</td>
<td>Trigger criteria with regard to the ellipse area. Value 0 indicates that the user is entering the area. Value 1 indicates that the User is leaving the area.</td>
</tr>
</tbody>
</table>

Note that the coordinate system for latitude and longitude follows the Geodetic System 1984 (WGS 84) [77].
5.4.4 Terminal Capabilities

As was mentioned in section 4.8, the user's terminal capabilities are declared by the terminal using the CC/PP framework and structure. There is, however, a specialised CC/PP compliant structure and vocabulary called UAProf which was created specifically for mobile terminals by the Open Mobile Alliance (OMA) [61]. The UAProf has 77 different terminal capability attributes available for a terminal manufacturer to describe the terminal's capabilities. A policy administrator defines these individual terminal capability attributes as "PolicyVariable"s and uses the aggregation PolicyVariableInSimplePolicyCondition to link it to a policy condition [75]. Each UAProf terminal capability attribute is therefore mapped to its equivalent PolicyVariable class (77 in total) [108]. For example, PolicyBitsPerPixelVariable is derived from the UAProf terminal capability attribute BitsPerPixel. The 77 different terminal capability attributes defined in UAProf are shown in numerous tables representing different terminal capability categories in the Appendix A.

There is still the requirement to select an appropriate PolicyValue (class) for each of the mapped UAProf-based PolicyVariable class before a policy administrator can create policies dealing with terminal capabilities. A UAProf terminal capability attribute value datatype can be one of 7 data types: Literal, Rational, Set, Sequenced, Boolean, Number and Dimension. The PCIMe specifications have defined a standardised way of matching PolicyVariables to PolicyValues, which was adopted for the proposed framework. PCIMe states that a PolicyIntegerValue can be defined as a single integer, a set of integers, or a range of...

integers. A PolicyStringValue can be defined similarly except that it cannot be expressed as a range. PolicyBooleanValue is always defined singularly. Table 5.1 shows the mapping created in order to match the UAProf attribute values to the PCIMe datatypes values.

Table 5.1 Mapping between UAProf attribute values to PCIMe datatypes

<table>
<thead>
<tr>
<th>UAProf datatype class</th>
<th>PCIMe datatype class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Literal, Rational, Set, Sequenced</td>
<td>PolicyStringValue</td>
</tr>
<tr>
<td>Boolean</td>
<td>PolicyBooleanValue</td>
</tr>
<tr>
<td>Number, Dimension</td>
<td>PolicyIntegerValue</td>
</tr>
</tbody>
</table>

The mapping in Table 5.1 is self-explanatory with one exception: the Sequenced datatype. The Sequenced datatype is used in the UAProf to indicate a preference of the consumer’s terminal as is an ordered list. Coincidentally, UAProf device attributes that are defined using Sequenced datatypes are those regarding (human) language usage. Therefore, there exists a problem of resolving user's language preferences against the defined policy of the content service. In such a case, the component in charge of policy enforcement takes the first literal value (i.e. the most preferred language) encountered in the Sequence and attempt to find a matching policy rule for it. If the content service is not adaptable to that language, it repeats the process for the rest of the languages in sequential order.

5.5. The SubscriberIDVariable and SubscriberIDCondition Classes

While the previous section defined the various conditions of a user’s context which can be defined in policy rules, there may be a requirement for the policy decision function to determine the users that the defined policy conditions is associated with. This is pertinent for push services which can be triggered and activated by changes in a user’s context, e.g. the user’s location. For example, to represent the following policy rule (written in pseudo code), there must be a policy variable class to specify user A in the policy rule.

If user A enters location A, then perform adaptation action A

As such, the SubscriberIDVariable class was created to represent the user for use in a policy condition, i.e. the SubscriberIDCondition class. The SubscriberIDVariable class was created as a direct subclass of the abstract PolicyVariable class in PCIMe, and its associated class for its policy value is the PCIMe’s PolicyIntegerValue class as the subscriber identifier is a numerical integer.

5.6. Discussion and Summary

This chapter has discussed the proposal of a generic Policy Information Model for key aspects of a user’s context in 3G networks. The compliance to current 3G standards
regarding network Quality of Service, Location Service (LCS), and terminal capabilities as well as IETF's policy core information model was carefully considered during the design of the model. The information model also included the definition of the policy condition class to detect the operating status of services that the user has subscribed to. This policy condition class allows the policy framework and the network operator to prioritise the service adaptation policies when the user's network resources are limited (as was discussed in 3.5.1.5).
6. Case Study and Prototype Evaluation

6.1. Introduction

The variety of terminals that access the Internet has been increasing in step with advances in access network technologies. The increase in terminal and network access heterogeneity poses the challenge of finding suitable means, methods and technologies for a uniform, efficient service and content delivery. This challenge has partially been addressed by Web technologies that allow the delivery of web-based content services to relatively powerful terminals such as laptops and personal computers attached to broadband, wired and wireless networks.

The situation is different in the case of content and service delivery to the mobile user, which represents a rapidly growing slice of the telco consumer market. In fact, as the cost of owning Internet-enabled mobile terminals decreases, mobile users are bound to become the predominant category. Terminal manufacturers are also pushing towards the mobile market, while network operators and service providers seek to inflate their revenues through new, appealing, advanced services that were unimaginable just few years ago. On the other hand, it is also becoming common for consumers to own different terminals with varying functionalities and capabilities such as laptops, smart phones etc.

The resultant of these forces is a pressing need to provide new service delivery solutions that allow consumers to access a plethora of web-based content services whilst on-the-move irrespective of the terminal capabilities and access network, ideally expecting a homogeneous user experience and uncompromised service functionality. This may be particularly difficult to achieve from the content service provider's point of view as creating content specifically for each and every available terminal type, so as to provide comparable user experience and service functionality, may increase the complexity of managing and maintaining the website.

Some current content service providers typically ignore such requirements and set their service access requirements to cater to only specific terminal types, e.g. computers with Internet-browsing capability or WAP mobile terminals. This practice, however, can only be viewed as a temporary solution to the problem because the diversity of terminal capabilities will only continue to increase in the future.

Automated service adaptation and VHE concepts lack at the moment the necessary level of support for the service and content provider. In order to succeed in offering manageable, advanced services, they will need a framework that can facilitate the various phases of
service engineering from service conception to its deployment and management. This chapter introduces a novel approach to automated web content adaptation, addressing the important requirements of terminal independence and ease of manageability, taking the provider's standpoint. The chapter presents an evaluation of a policy-managed content adaptation server prototype built around the latest technologies for service engineering which works in tandem with a policy-based management engine. Through an example content adaptation scenario and prototype, the chapter demonstrates the impact that policy-based management can have on simplifying and speeding up the development of adaptable services.

The case study and prototype includes the design and prototyping of a generic policy-managed content adaptation server architecture developed around XML-based technologies to provide adaptable web content to user terminals capabilities. The technological choices in its development were driven by industrially-endorsed protocols, specifications and recommendations in the context of 3G. For instance an implementation of a JSR-188 CC/PP processor needed to dynamically obtain terminal capability information was undertaken for the prototype. This chapter also presents an empirical evaluation of the benefits of a policy-based approach, namely the added flexibility, ease of programmability and correctness, against the inevitable overheads associated with the system complexity. The case study and the development of the prototype were motivated by three factors: Firstly, the author is not aware of any current framework for XML content publishing that incorporates the Java CC/PP API (JSR-188) specification. The prototype pioneers this approach while also demonstrating its feasibility and drawbacks. Secondly, although content adaptation using CC/PP has been possible since its introduction in 2001, nobody has yet assessed the actual performance degradation induced by the CC/PP adaptation process on the web server (this is a critical aspect that impacts scalability). This chapter examines and attempts to quantify the loss in web server performance when adapting content to CC/PP terminals so as to provide a yardstick for future research in this area. Finally, the framework explores the use of policies as a rule-based notation in which to manage the adaptation process. Research in policy-based management has been mainly associated with network and network security management over the last few years while this case study illustrates that policy-based management can also be beneficially applied to the management of adaptable services.

6.2. Scenario Deployment of the Content Adaptation Server Prototype

Figure 6.1 shows a scenario featuring the deployment of the OSA's PBM service capability by a network operator for the management of adaptable content for 3rd party content service providers. In this scenario, the network operator uses 3rd party content providers to offer a
customised content portal as a VHE service for its customers. In release 6 of the OSA interfaces, the PBM functionality is represented by two different service capability features: Policy Provisioning and Policy Evaluation. The content service provider uses a policy management tool to interact with the Policy Provisioning Service Capability Server (PP-SCS) via the OSA gateway to define, create, modify, and delete policies for content adaptation in the policy repository housed in the network operator's domain. The Policy Evaluation Service Capability Server (PE-SCS) is used by the content provider to evaluate policies and generate policy events into the system so as to control the adaptation strategy and the provisioning of the content. The PE-SCS is particularly important when new policies are added, deleted, or modified in the policy repository as it is used to evaluate if there is a need to assert a change in policy for the current managed system. The content adaptation capability servers are responsible for interpreting and enforcing the policies when performing content adaptation, as well as retrieving policies from the policy repository when there is a change in the content adaptation and provisioning strategy. The result of enforcing the appropriate policy is that the adaptation capability server always chooses the most appropriate content, as defined by the adaptation strategy, from the content provider and delivers it to the mobile user.

Figure 6.1 Scenario deployment of policy-based management in a 3G/B3G network

6.3. The Building Blocks of Automated Content Adaptation to Terminal Capabilities

New XML-based content frameworks, such as the Apache Cocoon Project, represent the required evolution in adaptable web content provisioning and utilise the inherent flexibility of XML to make content reusable and easily adaptable by logically separating the actual
6. Case Study and Prototype Evaluation

'content' from its 'presentation' [87][88][89]. Furthermore, the CC/PP framework provides terminals with a vocabulary to communicate their exact capabilities so as to allow web servers to use this information when adapting the web content.

XML-based frameworks represent, together with the CC/PP framework, the fundamental building blocks for content adaptation systems. However, a significant part of the adaptation process consists of the logic which decides how to best adapt the content in cases in which there are multiple possible solutions. This process can be interpreted as matching the content's operating and accessibility requirement profile to that of the terminal capability profile [20]. Butler suggests using a rule-based approach in a CC/PP-aware style-sheet to transform XML-formatted content into its final, adapted form through eXtensible Stylesheet Language Transformation (XSLT), i.e. transformations dependent on the values of CC/PP terminal capability attributes [90].

A rule-based approach is an appropriate solution to this process but performing the matching algorithm solely through a stylesheet has limitations on flexibility because the intelligence of the adaptation is neither distributed nor partitioned. In addition, Butler's work adopts a simple 'if (condition) – then (action)' rule as the way of controlling adaptation. Although such an ad hoc approach is still technically viable, it is not suitable for managing adaptation in the case in which different, conflicting conditions arise. Per contra, a policy-based approach incorporates a well-defined information model as well as specific means for detecting and resolving conflicting conditions or rules [7]. For instance, multiple CC/PP capability attribute conditions can be defined in a scalable manner and adaptation actions can be easily managed and effectuated towards an appropriate adaptation strategy.

6.3.1 Mechanisms for Obtaining Device Capabilities

In the framework prototype, the wireless HTTP (W-HTTP) protocol was used in combination with the UAProf vocabulary. W-HTTP was chosen as the protocol with which the terminal embeds CC/PP headers as it required the least complexity to achieve the desired requirement (i.e. maintain compatibility with UAProf).

6.4. Description of the Policy-Managed Content Adaptation Server Prototype

6.4.1 Design Issues and Motivations

There are two major Java-based frameworks that are available for producing adapted content on-the-fly: the Apache Jakarta Struts and the Cocoon project [88][87]. The former is based on a Model-View-Controller design pattern and to a large extent involves Java Servlets, Java Server Pages technologies to provide the adapted content. The Apache Cocoon Project is primarily a pipeline-designed XML-based framework that uses the Simple API for XML
(SAX) and XSLT to parse and adapt XML content appropriately in an extensible manner [91]. These two frameworks share the common aim of clearly separating the adaptation management domain from the content authoring domain in order to facilitate the development of the actual service.

The framework prototype builds on the aims and requirements of the two aforementioned content provisioning frameworks looking at the novel approach of policy-based management. The prototype was designed from the ground-up, addressing in particular the requirement of separating web site management from content management. The design was orientated to be XML-driven in order to pursue flexibility in delivering and provisioning content through the many XML-based specifications available for content formatting. For example, the XML content can be transformed to Extensible HyperText Markup Language (XHTML) for web browsers, Wireless Markup Language (WML) for WAP devices, VoiceXML for voice-enabled web browsers and voice-driven content browsing, and PDF through the use of Extensible Stylesheet Formatting Objects (XSL-FO). XSLT plays no part in the content layout adaptation process if the content is binary in nature (e.g. a video).

Figure 6.2 The Framework Prototype within the Content Adaptation Capability Server

Figure 6.2 shows a high-level view of the proposed framework that was also prototyped for experimental purposes on a Java Servlet based server - the Apache Tomcat Web Server [92]. All the blue-coloured components in Figure 6.2 have been developed from scratch and are compliant with their related state-of-the-art specifications, with the exception of XSLT ‘markup transformation’ component that was readily available [93]. The functionality and the role that the various components play in providing the adapted content service are described below.
6.4.2 The Controller Servlet

The controller Servlet serves as the entry point for a web request to the framework. It is accessed by a consumer's terminal via the network by a Uniform Request Locator (URL) and represents one or more web resources that contain the adaptable content service. Each of these web resources, i.e. content service, has an associated XML content skeleton. Figure 6.3 shows an example of such a skeleton for a 'film review content' service.

```
<Doc>
  <Headline>Reviews of upcoming films</Headline>
  <Review>
    <AdaptComponent>http://localhost:8080/films/SilenceOfTheLambs/</AdaptComponent>
  </Review>
  <Picture>
    <AdaptComponent>http://localhost:8080/files/pics/SilenceOfTheLambs/</AdaptComponent>
  </Picture>
</Doc>
```

*Figure 6.3 Example of an XML Content Skeleton*

The content skeleton serves to declare the structure of the web content in a flexible and extensible manner through the use of XML. It may contain adaptable parts, which are represented by the presence of AdaptComponent tags, and non-adaptable parts. In figure 3, the Picture and the Review portions of the content service are adaptable while the Headline portion of the content remains constant.

The information within each set of AdaptComponent tags represents the location, which is defined as a URL, of the component responsible for the adaptation of the portion of the content skeleton in which the AdaptComponent tag is declared. The Controller Servlet’s main functionality is to manage the adaptation process of the content as a whole by referring to the content skeleton. It leaves the adaptation of adaptable parts of the content skeleton to the component declared within the AdaptComponent tag, which is actually the Policy Execution Logic (PEL) component of the framework. For example, the PEL accessible at URL http://localhost:8080/films/SilenceOfTheLambs is responsible for adapting the Review portion of the content service in Figure 6.3.

The Controller Servlet reads the XML content skeleton when it is initialised and separates the adaptable part of the skeleton from the non-adaptable part after parsing the XML content skeleton. It does this so as to maintain a list of relevant Policy Execution Logic (PEL) components to invoke through the declared URLs when controlling the adaptation process. More importantly, in the Controller Servlet has to include information about the terminal's
capabilities when invoking the PEL so as to allow the PEL to use this information when adapting the content.

Upon receiving the adapted content back from the PEL, the controller Servlet replaces the *AdaptComponent* tags portion in the XML skeleton with the adapted content. It does this, in turn, for every adaptable content portion of the XML skeleton until all adaptable content portions have been appropriately adapted. The final document is then transformed through XSLT for the appropriate markup language and before being relayed to the user's terminal.

```xml
<Doc>
  <Headline>Reviews of upcoming films</Headline>
  <Review>
    Silence of the Lambs is a thriller about an imprisoned cannibalistic psychopath who helps a detective solve murders
  </Review>
  <Picture>
    http://www.filmreviews.com/SilenceOfTheLambs/small.jpg
  </Picture>
</Doc>
```

**Figure 6.4 Example of a Partially Adapted XML Content Skeleton**

Figure 6.4 shows an adapted XML content skeleton after receiving content from the PEL in charge of adapting the Review portion of skeleton, which results in a short textual description of the movie. The other PEL in charge of adapting the Picture portion of the skeleton returned a link to a small version of the picture. The number of Controller Servlets varies in the framework as each Controller Servlet can serve more than one adaptable web resource.

Figure 6.5 shows a UML state diagram highlighting the various states a Controller Servlet goes through when it is deployed. The first state is the initialising state, which happens when a Controller Servlet is loaded by the server. A Servlet will access its initialisation parameters and its configuration files when it starts up in a server. The Controller Servlet gets the locations of the XML skeleton and the XSL stylesheets for the different terminal browser user agents on the server. It then reads the XML skeleton does the following: it obtains a list of IP addresses for the Policy Execution Logic components representing each adaptable web resource; and it reads in the static non-adaptable portion of the XML skeleton and stores it into a buffer for further use in the adaptation process. The final part of the initialisation stage involves the creation of a *Transformer* object for each XSL stylesheet for the purpose of XSL transformation via XSLT transformer. The Controller Servlet is then ready to receive HTTP *Get* requests from user terminals via the W-HTTP protocol.
The next state of the Controller Servlet, **Start Processing HTTP Request**, occurs after it has received a HTTP Get request from a user terminal. The Controller Servlet then examines the user agent header of the HTTP request as it uses this information to apply the appropriate stylesheet for XSL transformation before delivering the content to the user later on in the state chart. It then simultaneously forwards, in separate running threads, the UAProf headers containing the user's terminal capabilities information to each Policy Execution Logic components stated in the XML skeleton. The Controller Servlet receives the individual adapted content from the Policy Execution Logic components. The Controller Servlet then constructs the adapted full XML document using both the static non-adaptable data in the XML skeleton, and the adapted content from the various Policy Execution Logic components. The Controller Servlet sends the adapted XML document to the user.
components. It then proceeds to the next state which is to transform this XML document into the appropriate markup language by sending to the XSL Transformer object which represents the user terminal's browser user agent. After the XML document is transformed, the Controller Servlet then returns the fully adapted content back to the user terminal as a HTTP response.

**6.4.3 Policy Execution Logic Component**

The Policy Execution Logic component (PEL) is mainly responsible for adapting the adaptable portions of the XML content skeleton and delivering the adapted content back to the Controller Servlet. The PEL interface was implemented as a Java Servlet in our framework prototype and receives requests for adaptation from the Controller Servlet through the W-HTTP protocol. It is also possible to implement the PEL interface in other ways (e.g. distributed objects, Simple Object Access Protocol agents etc.) while also making the necessary amendments to the contents of the AdaptComponent tag of the XML skeleton (e.g. object reference for a distributed object implementation).

The PEL discovers information about the terminal capability by using the JSR-188 compliant processor (described further in 6.4.4) to construct an object representation of the terminal capability profile through using the UAProf or CC/PP information contained within the request.

Each PEL component maintains a reference to all the possible adapted content that it can deliver back to the controller Servlet. This may include textual, pictorial, and other multimedia content. The PEL must then decide which 'content' should be delivered on the basis of the terminal capabilities (obtained from the JSR-188 compliant processor) making a selection from a list of all the 'content' it is in charge of provisioning. The decision process is performed with the aid of adaptation policies that are stored in a repository. In our framework, policies are used solely for managing the adaptation process by controlling the PEL's content selection decision.

While each PEL Servlet is in charge of adapting a single web resource, a single policy can be made to be reusable for many PELs. This means that a group of PEL Servlets can possibly share the same adaptation policy and strategy. A programmer can thus input the location of the policy to the PEL Servlet's deployment description file, and the PEL Servlet will start processing the policy in the Servlet initialisation phase. The policy associated with the PEL will then continue to govern its behaviour for the period of its lifetime (i.e. until the Servlet is stopped and redeployed).

With respect to scalability, it is important to note that a PEL is associated with one adaptable resource, which itself can involve many static content formats and files used for
the adaptation. Thus, a PEL is not required for every single static media content file located in the content provider’s server.

The final responsibility that the PEL has within the framework is that it is responsible for interacting with the Policy Evaluation SCS and the policy repository. These interactions are necessary when so that the PEL can be informed when policies which influence its adaptation decision are created, deleted, or modified, and it can then proceed to retrieve the new policy from the policy repository.

Figure 6.6 shows the internal design of the entire Policy Execution Logic. The Policy Execution Logic Interface acts as a service access interface for the Controller Servlet. It receives W-HTTP requests (with the UAProf headers regarding terminal capabilities) and is expected to provide the adapted content back to the Controller Servlet. In the prototype implementation, the Policy Execution Logic Interface was implemented as Java HTTP Servlet. The PEL Servlet loads policy objects, which represents the policies that is applicable in its domain, and loads them into a Policy Decision Logic. These policy objects include policy groups and policy rules at the higher level, and the policy conditions and policy actions specified within these groups or rules. As the policy action involves obtaining an adapted content, it requires access to the content source at the content provider. It accesses and obtains the content source through the use of a Data Access Object, which is a design pattern [109]. The Data Access Object encapsulates access to the content source and provides the policy object with a standardised interface regardless of the content source type. For example, the content source could be a database, a flat file, a remote object or a web server. The Data Access Object provides a Transfer Object to the policy code which represents the adapted content. In our prototype implementation, the Transfer Object was implemented as a Java String object. Finally, the Policy Decision Logic is in charge of making the decision of the appropriate policy to apply when given a set of terminal capabilities. It returns the chosen adapted content to the Policy Execution Logic Servlet.
Figure 6.6 Design of the Policy Execution Logic

Figure 6.7 Sequence diagram of the Policy Execution Logic Initialisation

Figure 6.7 shows an example of the initialisation sequence of the Policy Execution Logic Interface (which is a Java Servlet object in our prototype) involving a single policy rule with one policy condition and one policy action:

2. The Policy Execution Logic Servlet creates a PolicyCondition object and inserts it into the PolicyRule object.
4. On insertion of the PolicyAction object, the PolicyRule object creates a new HttpTextContentDataAccessObject. It does this to gain access to the adapted content represented in the policy action, and to store that adapted content locally so that it can deliver the content instantaneously during the policy enforcement process. The
performance penalty in obtaining the adapted content is therefore incurred in the creation process of the policy rule instead.

5. The HttpTextContentDataAccessObject sends a HTTP request to the content provider for the content specified in the policy action.

6. The HttpTextContentDataAccessObject receives the content from the content provider.

7. The HttpTextContentDataAccessObject stores the content as a Java String object and returns it to the PolicyRule object.

8. The Policy Execution Logic Servlet then adds the newly created PolicyRule into the Policy Decision Logic.

9. The Policy Decision Logic initiates an internal process to load the PolicyRule into the decision process on receipt of this request. This is important as policy rules have to be evaluated in order of priority and the addition of a policy rule may affect the current decision sequence in the policy decision process.

10. The Policy Decision Logic sends an indication to the Policy Execution Logic Servlet that the addition of the PolicyRule object was successful.

Figure 6.8 shows the entire state diagram of the Policy Execution Logic Servlet. The initialisation state was described in detail in Figure 6.7. After its initialisation, the Policy Execution Logic Servlet is ready to receive requests from the Controller Servlet (see section 6.4.2) for adapted content. The Policy Execution Logic Servlet starts processing HTTP requests by taking the HTTP headers associated with the terminal capabilities and passing these parameters to the JSR-188 CC/PP processor. The JSR-188 CC/PP processor returns a Profile object which represents the terminal capabilities. The Policy Execution Logic Servlet then passes this Profile object to the Policy Decision Logic component which then decides which the appropriate content to be delivered to the user by comparing the policy conditions and the user's terminal capabilities. The Policy Decision Logic component then passes the chosen content back to the Policy Execution Logic Servlet, which then passes it to the Controller Servlet through a HTTP response. The implementation Java codes for the important policy-related classes are provided in Appendix C.
6.4.4 The JSR-188 Compliant Terminal Profile Processor

Before the specification of JSR-188, development of tools for processing CC/PP and UAProf vocabularies in industry were often done independently. As such, programmers often interpreted the specifications differently resulting in the situation whereby code that used a particular CC/PP processing API could not interoperate with other APIs. Consequently, evaluating the performance of CC/PP processing could not be done reliably as there was often no basis on which to make an objective evaluation. The development of the JSR-188 specification eradicates this problem as it provides a universal API that is independent of the underlying CC/PP processing logic for web developers to use. Table 6.1 shows the major differences between DELI (version 0.9.7) [63], the most well-known pre-JSR-188 processor, and a JSR-188 compliant processor.
Table 6.1 Comparison between DELI and JSR-188 Processors

<table>
<thead>
<tr>
<th>Internal Profile Representation</th>
<th>DELI APIs</th>
<th>JSR-188 APIs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Flattens the profile to the level of CC/PP attributes. Ignores the CC/PP</td>
<td>Profile hierarchy is maintained at all times. An object representing a CC/PP</td>
</tr>
<tr>
<td></td>
<td>component to which the attribute belonged. Objects representing CC/PP</td>
<td>attribute must have a reference to an enclosing CC/PP component object as well as the enclosing profile object.</td>
</tr>
<tr>
<td></td>
<td>attributes only have a reference to the enclosing profile object.</td>
<td></td>
</tr>
<tr>
<td>CC/PP and UAProf Attribute datatypes</td>
<td>Implementation-specific</td>
<td>Datatypes are well-defined through abstract classes and interfaces.</td>
</tr>
</tbody>
</table>

A JSR-188 compliant processor that uses a SAX parser to parse UAProf profiles was developed for the prototype framework as there were no reference implementations available at the time of prototype development. SAX was chosen because it was less memory intensive than Document Object Model (DOM) based parsers when parsing XML documents [94], making the prototype framework's JSR-188 processor more scalable when deployed in a web server for processing device profiles. Apache Xerces was chosen to be the SAX parser used in our JSR-188 processor, as this is the only currently available parser that supports the latest XML and Java JAXP specifications at the time of the evaluation [95]. The main responsibility of the JSR-188 compliant processor in the framework is to provide CC/PP processing functionality to other components within the framework. Only one processor is required to serve the whole framework.

Profile attribute datatypes are also well-defined in JSR-188 and provide more structure and flexibility if compared to the DELI APIs. In DELI, an attribute is represented as a Java Object class in a Java Vector object, which makes it generic but extensible. The datatypes defined for profile attributes in JSR-188 are:

a. BooleanAttribute
b. DimensionAttribute – This datatype is used mainly for representing dimensions, and is used for only 3 attributes in UAProf: ScreenSize, PixelAspectRatio, and ScreenSizeChar
c. IntegerAttribute
d. LiteralAttribute – The datatype is used to represent an XML Literal
e. RationalAttribute – This datatype represents a rational number (i.e. a ratio)
f. SetAttribute – This datatype represents a container that holds zero or more other datatypes. Set Attributes usually act as a container for multiple LiteralAttribute datatypes.
g. **SequencedAttribute** – The datatype is used to represent an ordered list of other datatypes. It is significant and different from the other datatypes because it can be used to indicate a preference. There are only 2 sequenced attributes in UAProf – CcppAccept-Language, and Push-Accept-Language – that are used to indicate a preference for the language of the service in a ‘pull’, and ‘push’ scenario respectively.

The datatypes of CC/PP attributes are well-defined in JSR-188 processors, which aid the definition and interpretation of policies in the framework.

Figure 6.9 shows the class diagram of the more important classes in the implemented JSR-188 compliant processor for the framework. The CcppProfileFactory extends the ProfileFactory class specified by the JSR-188 specification and is used to create a Profile object representing the terminal’s capabilities. The CcppProfileFactory creates a Profile object (represented in the figure as a CcppProfile class) through the use of a SAX parser. A CcppHandler class was developed to be used in conjunction with the SAX parser by the CcppProfileFactory to parse CC/PP files into Profile objects.

The CcppProfileFactory allows for the caching of terminal capability profiles to maximise efficiency when delivering parsed CC/PP files. The ProfileCache class was developed to be used by the CcppProfileFactory to store CcppProfile objects that has already been created. This class allows the CcppProfileFactory insert and retrieved stored CcppProfile classes, therefore avoiding the computational task of reparsing a UAProf file with a SAX parser. The Profile objects are stored internally in the ProfileCache class using a Java HashMap object. The key used to obtain the Profile objects from the HashMap is the values of the terminal capability related headers a W-HTTP request.

Finally, the ProfileComposite class was developed as an aggregator of multiple device profiles which forms a single terminal UAProf. The W-HTTP protocol allows multiple terminal capability headers to be represented through the use of x-wap-profile and the x-wap-profile-diff headers. For example, a user may connect a large computer monitor to a handheld 3G smart phone and may wish to declare the screen resolution of the computer monitor instead. The ProfileComposite class was developed to deal with such situations where multiple individual device profiles form the overall device profile. It resolves terminal capability attributes through the UAProf specifications on profile resolution for that particular attribute (i.e. locked, override, or append). The Java implementation codes for these classes are provided in Appendix D.
6.4.5 Markup Transformation via XSL

In Figure 6.2, the Markup Transformation via XSLT component is used mainly to transform a document received from the controller Servlet into another (more appropriate) document and to return the transformed document back to the controller Servlet. The transformation is usually done to convert content into an appropriate markup for the consumer's terminal (e.g., XHTML, WML etc.). Figure 6.10 shows an example of possible XSL stylesheets to transform the XML skeleton in Figure 6.4 for a XHTML-capable browser. Figure 6.11 shows the transformed XHTML document which will be sent to a client with a XHTML-capable browser.

```xml
<?xml version="1.0" encoding="UTF-8"?>
```
6. Case Study and Prototype Evaluation

6.4.6 The Adaptable Content Creation Process

This subsection elaborates on the entities, roles and responsibilities involved in the process of producing adaptable content within our framework. The framework allows for a partitioning of responsibility and this subsection highlights the differences, if any, with the current production scenarios. They main players include the content author, the content adaptation strategist, the web resource editor (or information architect), and stylesheet and markup editor.

Content authors are the group whose responsibilities are least affected by the deployment of the proposed framework. These include photographers, graphic designers, freelance writers etc. Their responsibilities have not changed at all as they are still required to produce the raw service content. This may have to be further processed in order to generate formats that suit different terminal capability. These changes may, for instance, include reducing the length of textual content, converting pictures to other picture formats (e.g. jpeg to gif), and resizing the resolution of the picture. The person in charge of this is the content adaptation strategist. S/he must plan in advance the number of terminal types that the content services can serve and their associated capabilities. For example, if the service is destined to
terminals having small screen sizes as well as terminals with normal screen sizes, then pictures and videos must be duplicated and scaled down to the appropriate resolutions.

The choice of a threshold value to differentiate between a terminal with small screen size and that with normal size belongs to the content adaptation strategist who must co-ordinate his/her decisions with the adaptation policy editor. The latter is in charge of writing policies regarding the adaptable web resource to be delivered by the PDA component. Up to this point, the only person that needs technical knowledge about programming is, therefore, the policy editor.

The next person involved in the provisioning of the content service is the web resource editor. This entity is responsible for determining the choice of content to be placed in the XML content skeleton, and is therefore the person who is most influential in constructing the content service. The web resource editor also decides on the content structure of the service, discriminating the adaptable portions of the service from the static ones. Another appropriate term more commonly used in current literature to describe this position of responsibility is information architect [96].

The web resource editor liaises with the stylesheet and markup editor on the creation of the XML content skeleton. The latter editor is a person with technical knowledge of XSL and the XSLT transformation process and authors the stylesheet and the XML content skeleton as well as selecting the domain-specific XML tags to be used in the XML content skeleton.

6.5. Policy Conflicts Concerning Adaptation Decisions

Policy conflicts generally occur in policy-based management when a given set of conditions may be applicable to at least two different policy rules. For the case of our framework, this would mean that the PDA component would be left with more than a single choice of optimised content, thus rendering the adaptation process ineffective. The example below shows a possible occurring set of conflicting conditions involving two separate policy rules, based on the policy notation specified on page 31 of [75].

| Rule 1: if BitsPerPixel MATCH [2..32] && ImageCapable MATCH true |
| Rule 2: if BitsPerPixel MATCH [2..8] && ColorCapable MATCH true |

The first rule is valid when the terminal display has a capability, in bits per pixel, comprised within the range of 2 to 32 bits (inclusive) and has also the ability to display images (on top of text). The second rule is valid when the terminal display has a capability, in bits per pixel, comprised within the range of 2 to 8 bits (inclusive) and is also colour capable. A policy conflict might therefore occur when the service is accessed by a terminal
characterised by a resolution of 8 bits per pixel and capable of displaying colour images. In this case both rules would be evaluated to be true.

Fortunately, the PCIMe defines two policy decision strategies that will solve such policy conflicts. The \textit{FirstMatching} strategy involves placing a priority on a policy rule. The policy rules are then evaluated according to their priorities and the PDA component uses the first rule whose policy conditions match that of the terminal capabilities. \textit{Per contra}, the \textit{AllMatching} strategy involves evaluating all policy rules and effecting their associated actions only if all the rule's policy conditions are true. The \textit{AllMatching} strategy is not applicable to our framework and was therefore not considered.

At first glance, the \textit{FirstMatching} strategy seems to be a solution for resolving adaptation policy conflicts in our framework. However, a UAProf device profile need not declare every single device capability attribute available in the UAProf vocabulary, and thus this generates another problem evaluating policy conditions. This problem can be illustrated by using, again, the two rules defined above and assuming that \textit{Rule 1} has a higher priority than \textit{Rule 2}. If a device profile does not contain any information about its ability to display images (refer \textit{Rule 1}), then the PDA will have no way of evaluating if policy \textit{Rule 1} is true, thus compromising the integrity of its policy-based decision. The solution to this problem is to assign a default condition evaluation outcome for policy \textit{Rule 1} that the PDA can fall back on when encountering such situations. The CIM guidelines state that it is acceptable to add properties to an existing CIM-derived class when extending a CIM-compliant model. Therefore, the new property \textit{DefaultPolicyResolution} was added to the \textit{PolicyCondition} abstract class in the PCIM and PCIME. The datatype of the value associated to this new property is a Boolean attribute whose value represents the default outcome of evaluating the policy conditions. There might also be a situation where none of the policy rules defined for the policy is applicable because none of their conditions are satisfied. In our framework, this means that the web resource cannot be adapted to or is not applicable for the consumer's terminal (e.g. the terminal cannot display the Russian characters needed for accessing a content service in Russia). It is usual practice to define a \textit{default} policy rule to be used if such a situation occurs. If a default policy rule is not defined, the PDA Servlet sends back empty content to the controller Servlet in our framework. The latter then realises there is no content match and deletes the portion of the content service from the XML content skeleton appropriately.
6.6. Experimental Results and Analysis

6.6.1 Statistical Analysis of the UAProf profile

Currently, there have been 6 different versions of the UAProf vocabulary since the year 2000, with a total of 91 definable attributes. There are two significant benefits if a statistical analysis of UAProf profiles was available to policy and content authors. Firstly, they can know what is the most often used UAProf capability declared by terminal manufacturers and can therefore plan the adaptable service around those capabilities. Secondly, the policy author can know the average number of attributes declared in a profile and can use that knowledge to plan the number of policy conditions to use in the adaptation. Knowledge of the structure and capability distribution of the average terminal can also be obtained from such statistics and can be useful for choosing a terminal profile for our experiments in a non-biased and informed manner. A survey of 98 different UAProf profiles available for download at [97] was conducted to discover the average statistics of a UAProf profile. Table 6.2 shows the results of our statistical analysis.

Table 6.2 Statistical Analysis of UAProf Profiles

<table>
<thead>
<tr>
<th>Occurrence</th>
<th>Locked</th>
<th>Append</th>
<th>Override</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percentage of Occurrence</td>
<td>73.2</td>
<td>18.2</td>
<td>8.6</td>
</tr>
<tr>
<td>Average per profile</td>
<td>21.60</td>
<td>5.38</td>
<td>2.53</td>
</tr>
<tr>
<td>Median (per profile)</td>
<td>21</td>
<td>5.5</td>
<td>2.5</td>
</tr>
<tr>
<td>Maximum Number of attributes (per profile)</td>
<td>60</td>
<td>18</td>
<td>13</td>
</tr>
</tbody>
</table>

The total number of attributes that was scanned was 2892. The above table shows that the average UAProf profile has (rounded to the nearest whole number) 22 locked attributes, 5 Append attributes, and 3 override attributes. Using the 98 different UAProf profiles that was surveyed, a list highlighting the popularity of specific UAProf attributes was obtained. Figure 6.12 below shows the ranking of UAProf attributes according to the number of occurrences that appeared in the statistical study.

By referring to the statistical data on the distribution of attribute datatypes, the Sony-Ericsson T200A UAProf profile was the choice of the terminal profile when assessing the framework as it had the a similar number of attributes as the average of the 98 different UAProf profiles surveyed. The Ericsson T200A UAProf profile is shown in appendix B.
6. Case Study and Prototype Evaluation

6.6.2 Empirical Analysis of the Policy Framework

The performance of the prototype framework was assessed through a local area network test bed, ensuring that network delay had negligible influence on the results. The server used had an Intel Pentium 4, 2.53 GHz processor with 512Mbytes of memory. Apache Tomcat version 4.1.27 was used to support the framework, with a Java 1.4.1_05 Virtual Machine. The server was deployed using the ‘server’ option of the Java virtual machine to improve performance.

The experiments were planned as follows: firstly, in order to assess the performance of a non-policy-based version of the framework, the effects of using just Java Servlets with the developed CC/PP processor for the delivery of adaptable web content was observed. The experiments were then repeated with a policy-enabled system, gradually increasing the number of adaptable components from 0 to 4. Throughout the experiments, the same policy consisting of two rules was used, each of which contained a SimplePolicyCondition. Experimental conditions were designed to be CPU-bound rather than disk-bound by constructing relatively simple adaptable web content. The Apache JMeter (version 1.9.1) [98] was used to measure the response times and throughput, and the results were obtained through measuring 2000 web requests. Figure 6.13 shows the physical setup of the experiments. An IBM T40 Laptop with 768 Mb of physical RAM and a Pentium M 1.3 processor simulated the user terminal. The Apache JMeter was run on the laptop to measure the server response time and throughput. The Apache JMeter allowed the customisation of the HTTP request headers, and an x-wap-profile header and its associated value of a Sony-Ericsson T200A UAPof profile was inserted into the request headers. The IBM T40 Laptop was connected to the server via an Ethernet hub and no other devices were connected to the router to ensure the absence of external interfering traffic. The server acted as a content adaptation capability server as described in the case study in section 6.2. The server ran the Apache Tomcat Servlet engine, the JSR-188 CC/PP processor, the Policy Execution Logic, the XSLT component, and the Controller Servlet.
Figure 6.13 shows the average response times of the prototype framework, while Figure 6.15 shows the throughput of the server in requests served per minute. In both diagrams a change in slope which corresponds to the activation of the policy-based engine was observed, generating an increase in the performance degradation rate. This reflects the cost
paid for the additional flexibility, programmability and manageability offered by the policy-based approach.

6.7. Discussion and Summary

As the number of terminal capabilities increases in the future, the options for flexible service provisioning become greater and adaptable service provisioning represent an increasingly important requirement for consumers in the future. It is fairly straightforward to expect a loss in server performance when providing such services, but a scientific and empirical approach to quantifying this loss is certainly needed. This chapter explored the applicability of a policy-based framework to provision adaptable services, highlighting the added flexibility in managing the adaptation process. The chapter described how policies can be modelled, defined and interpreted to control the adaptation, as well as the methods that can be used to avoid policy conflicts. It also described guidelines on what terminal attributes to adapt to and how much adaptation is necessary by analysing statistically the popularity of specific device attributes, and the number of device attributes declared in terminal profiles. Finally, experiments to quantify the loss in server performance against the adaptability of a web content service were performed with the policy-managed prototype server framework.

6.7.1 Performance of the Prototype Implementation

Some lessons were learnt on performance optimisation during the implementation of the prototype. Firstly, it is important that the JSR-188 CC/PP processor caches processed Profile objects representing the UAProf terminal capability files that it has parsed in the past. The JSR-188 specification currently only recommends that the processed Profile objects be cache. If the Profile objects are not cached, then the JSR-188 CC/PP processor will have to process the UAProf terminal capabilities files from the repository for every single HTTP request that it receives from the user. This in turn will severely cripple the performance of the prototype. This thesis strongly recommends that profile caching for the purposes of real-time adaptation of content be made obligatory.

Secondly, it is important that the prototype gathers and caches all the available adaptable content in the server during initialisation. The benefits of doing this are twofold: This minimises the time taken for the prototype to deliver the adapted content to the user as opposed to fetching it from the content source (at the content provider) at every HTTP request received. This means that the response time to serve a HTTP request is reduced and the overall prototype system throughput is increased. The other benefit of caching the content source is to minimise the amount of network traffic between the prototype and the content provider as there is no need to fetch the appropriate content at every HTTP request.
6. Case Study and Prototype Evaluation

received. This reduces the likelihood of a bottleneck occurring because of network congestion between the prototype and the content provider.

Though the experiments do show a loss in server performance when dealing with content adaptation, this loss can be easily countered in the future by using more powerful servers, and techniques such as load-balancing and those mentioned in this subsection. In this respect, policy-based service management is certainly a promising approach toward increasing the level of automation in the deployment of advanced services.

6.7.2 Difficulties of working with the UAPerf specification

During the statistical analysis of UAPerf profiles, we observed the following:

- Not all advertised UAPerfs are available. This makes it impossible for a CC/PP processor to obtain the terminal capabilities.
- UAPerf profiles can contain schema or data errors which can cause parsing to fail.
- The UAPerf document itself does not contain the user agents of the devices it might apply to in the schema. These user agents can contain key information about the applications that the user terminal can operate, e.g. Microsoft word documents, PDF documents etc. The only way to obtain this information is by looking at the user-agent and user-agent-accept headers in the HTTP request itself.
- UAPerf headers can often be plain wrong. There is no way to ensure that the UAPerf profile that the headers state is for the correct user terminal.

Furthermore, not all terminals may declare their terminal capabilities at all, i.e. they do not use the UAPerf specification. A workaround to this problem is to use a unique user agent identifier in the HTTP headers to state the device, and for network operators to create a UAPerf profile on behalf of the terminal manufacturer. Another workaround to this problem is to let the user state the terminal capabilities in the user profile, and to automate the creation of a personalised UAPerf profile on behalf of the user. This personalised UAPerf can then be stored in the network operator's profile repository.
7. Conclusions and Future Work

7.1. Summary of Contributions

This thesis has proposed the deployment of a policy-based management framework within a 3G network for the management of context-aware services. An outline of the requirements of deploying a policy-based management framework was presented. The main requirements included the exploration of appropriate scenarios in which policy-based management would be beneficial to context-aware services, and the analysis of existing business models that facilitates the policy-based management of context-aware services to be financially viable to all business roles and entities involved in 3G service provisioning. One criticism of other related service adaptation frameworks is that they do not place enough emphasis on fitting the service adaptation framework within a viable and feasible business model. These frameworks typically result in the definition of superficial business roles such as the “Service provision platform operator” in the IST MOBIVAS project and the “VHE provider” in the IST VESPER project. On the contrary, the proposed framework highlighted in this thesis considered both the UMTS forum’s business models and the business roles specified in 3GPP’s VHE specification, and ensured compliance with both.

This thesis presents a convincing argument on how policy-based management can be beneficial to managing the adaptation process of a context service. It showed how multi-dimensional adaptation of the service to different numerous aspects of the user context can be performed. Furthermore, it illustrates how pro-active adaptation strategies can be achieved by influencing the decision on the network QoS requested from the network to suit the service operating requirements or user’s preferences.

The design of a policy-based management framework which works in tandem with the OSA policy management API was one of the key contributions in this thesis. 3GPP did not provide any binding or mapping specifications of the OSA policy management API to the underlying 3G network. The bulk of the research in this thesis was to propose a suitable mapping between the OSA API and the underlying network entities while maintaining complete compliance to the plethora of 3GPP specifications dealing with network functionalities and protocols. A comparison of the benefits and disadvantages was also made between the approach of using the OSA mobility and terminal capabilities APIs for context gathering, and the approach of using the OSA policy management API to perform the same tasks via policies.
Policy information models to represent aspects of the user context are required to allow policy administrators to create and apply policies via the OSA policy management APIs. This thesis has proposed policy information models dealing with the user’s location, terminal capabilities, and service access status as no existing models were specified by the 3GPP. The policy information models were designed to be compliant with the various underlying 3GPP specifications such as the LCS and the Go interface.

Finally, the design of a case study and the prototyping of a policy-managed content adaptation framework were performed as a proof of concept. A statistical analysis of UAProf terminal attributes was undertaken to provide future guidelines on the creation of policy conditions for terminal capabilities. The prototype was evaluated in a testbed and its performance was analysed to determine the operational costs of utilising a policy-managed adaptation process for services adaptable to the user’s terminal capabilities. Lessons learnt from the implementation of the prototype were also presented and discussed in the thesis.

7.2. Discussion on Future Work

7.2.1 Improving Scalability and the Performance of Policy Enforcement

The analysis of the performance of the prototype in chapter 6 showed that there is performance degradation when the adaptation process was managed through policies. It is therefore imperative to minimise the performance degradation if policy-based management of context-aware service is to take off. One method of increasing performance is to create efficient policies at the policy authoring process. This thesis showed that if the statistical analysis of UAProf terminal capabilities was considered, policy authors could avoid using terminal capabilities attributes which are not often declared by terminals as policy conditions. This decreases the number of “false” policy conditions that the policy decision function will have evaluate thus increasing performance and scalability. The additional benefit of using this approach is a more streamlined content adaptation operational strategy. Future work can be undertaken to discover the effect of efficient policy authoring on location and network QoS through the use of statistical analysis on policy conditions.

It is also possible to examine the effects of computational load balancing will have on the scalability of policy enforcement. For example, one could examine the deployment of a policy-based management framework on top of a grid-computing infrastructure.

7.2.2 Authorisation Policies for Location-Aware Services

With the introduction of the Presence Service in 3G networks in the near future, access to location information becomes a more sensitive issue. The Presence Service allows a user to authorise other users to monitor his/her status (of the willingness to communicate - a
concept similar to that provided by an Internet messaging service such as Yahoo messenger) and other information such as the user's location [99]. Privacy of a user's information is key point for the Presence Service requirements, and the user has the ability to manage the access rules to manage the privacy.

Policy-based management has been applied to the authorisation of network QoS reservation, and is therefore very suitable for managing the authorisation and access to a user's location. Possible future work in this area could include the design of a policy information model to create such authorisation policies.

7.2.3 Managing the Service Aggregation and Composition Process

Service aggregation and composition frameworks have grown increasingly popular over the last few years with the introduction of standards such as W3C's Web Services. 3GPP has also planned the introduction of a Service Broker API in its OSA framework for release 7 [100]. Interesting future work in this area could be to examine the effects of aggregating different context-aware services and managing the adaptation of this aggregated context-aware service through the use of policies. Sections 7.2.3.1 and 7.2.3.2 provide preliminary discussions on this area for the scenario of aggregating location-aware services.

7.2.3.1 Accommodating Different Types of Aggregated Location-Aware Services

There are three methods of aggregating composite location aware services. Any proposed framework must accommodate these three types of aggregated location-aware services, which are:

1. Aggregating services of different service categories for a particular location – ClL1 aggregated service. For example, an aggregated service is charged with finding a list of restaurants in the area with automated on-line reservation capabilities, a list of car parks in the area including the available capacity at the moment, and a traffic update on the roads in the area as well. This example combines three different independent service categories to provide an aggregated location-aware service.

2. Aggregating services of the same service category over a range of locations – C1Ln aggregated service. For example, a user may want to take a bus from his current location to another location but there is no single bus route or operator that covers this trip. He/She uses an aggregated service that determines his/her current location, contacts different bus operators' timetables and routes to coordinate an inter-change within a reasonable transit period, and displays to the user the location of the bus stops to board the bus at each transit point via a map. An additional benefit provided to the user by such a service is that a composite service's dynamic information and functionalities, such as bus route detours and
cancellations, are incorporated into the aggregated service thus alleviating the user of such concerns. In this situation, the aggregated service may be seen as a single location-aware service whose locality is the union of each composite location-aware service’s locality.

3. **Aggregating services of different service categories over a range of locations – CnLn aggregated service.** This category of aggregated service combines the characteristics of the previous two. The locality of a CnLn aggregated service is also the union of each composite location-aware service’s locality.

### 7.2.3.2 Accuracy of the User’s Location

There are various methods available to ascertain a user’s location in current cellular networks. These methods vary widely in the accuracy of determining the location, and the environmental characteristics that are required for a specific positioning method and technology to be available and accurate. The latter usually affects the results of the former in most location-tracking technologies. For example, the Global Positioning System (GPS) requires the user to be mainly outdoors so as to receive a Line of Sight (LOS) signal from the Low Earth Orbit satellites and its accuracy increases as the number of satellites over the user’s location increases. While many positioning methods and technologies do exist, e.g. Round-Trip Time, Angle of Arrival, Reference Node Based Positioning, 3GPP has standardised three methods for the Location Services (LCS) functionality in 3G mobile networks. They are Cell ID based positioning, Observed Time Difference of Arrival (OTDOA) positioning (which is based on Time Difference of Arrival positioning), and Assisted GPS positioning (which essentially is GPS positioning). OTDOA works well indoors and provide reasonable accuracy when more than two neighbouring base stations are used in determining the user’s location. GPS is the most accurate but requires LOS between the user and three or more satellites to perform well. Cell ID based positioning is the least accurate and its accuracy is inversely proportionate to the size of the cell that the user is in.

It can be easily deduced that the accuracy of the user’s location is a major factor when provisioning location-aware services. Inaccurate location positioning of the user can result in providing the user with services that he/she does not want or need, thus decreasing user satisfaction and service reputability. The importance of this factor increases when various location-aware services are aggregated together. Although all aggregated services would suffer to some extent from positioning inaccuracy, a CnL1 aggregated service would be more likely to be affected than a CnLn or a CnLc aggregated service as its overall service locality is smaller than that of the other two (assuming that the area of each composite service locality is equal and exclusive) – a small inaccuracy would constitute a greater percentage of error over a small area than a large area. However, this may not be the case for
an aggregated service with functionalities that has strict positioning accuracy requirements regardless of aggregation type, e.g. a service involving the location of the nearest hospital or 24-hours emergency medical clinic. Therefore, the requirement for positioning accuracy from an aggregated service depends on both the type of aggregated service, and the aggregated service functionality. More importantly, if service aggregation is performed on the fly, then there must be a way to resolve the individual location-aware service's positioning accuracy requirements against that of the aggregated location-aware service's requirements. Policy-based management could prove to be a promising solution in tackling this challenge.
References

[4] 3GPP technical specification TS 22.127 v 6.80, Service Requirements for the Open Services Access (OSA); Stage 1 (Release 6).
[5] 3GPP technical report TR 22.121 v 5.3.1, Service Aspects: The Virtual Home Environment (Release 5).
References


[34] IST MAGNET project (IST-507102), Draft user requirements for PN to drive the definition of a valid architecture, deliverable D1.1.1.a, March 2004.


References


[41] 3GPP technical specification TS 23.198 v6.0.0, Open Service Access (OSA); Stage 2 (Release 6).


[45] 3GPP Technical Specification TS 23.207 v6.5.0, End-to-end Quality of Service (QoS) concept and architecture (Release 6).


[51] 3GPP Technical Specification TS 29.198-6 v6.5.0, Open Service Access (OSA); Application Programming Interface (API); Part 6: Mobility Service Capability Feature (SCF), (Release 6).


[59] OMA-LIF-MLP-V3_1-20040316-C.


[61] Open Mobile Alliance, User Agent Profile 1.1, Candidate Version dated 12 December 2002, OMA-WAP-UAProf-v1.1-20021212-c.


References

[65] 3GPP Technical Specification TS 22.240 v 6.5.0, Service requirements for the Generic User Profile (GUP); Stage 1 (Release 6).
[68] 3GPP Technical Specification TS 22.228 v 6.11.0, Service requirements for the Internet Protocol (IP) multimedia core network subsystem (IMS); Stage 1 (Release 6).
[99] 3GPP Technical Specification TS 22.141 v 6.5.0, Presence Service; Stage 1 (Release 6).
[100] 3GPP Technical Specification TS 29.198-16 v 7.0.0, Open Service Access (OSA); Application Programming Interface (API); Part 16: Service Broker Service Capability Feature (SCF) (Release 7).

### Appendix A UAPref Terminal Capability Attributes

**Table A.1 UAPref Terminal Hardware Capabilities**

<table>
<thead>
<tr>
<th>Component: HardwarePlatform</th>
<th>Attribute</th>
<th>Description</th>
<th>Resolution</th>
<th>Datatype</th>
<th>Example(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BitsPerPixel</td>
<td>The number of bits of colour or grayscale information per pixel, related to</td>
<td>Override</td>
<td>Number</td>
<td>&quot;2&quot;, &quot;8&quot;</td>
</tr>
<tr>
<td></td>
<td>BluetoothProfile</td>
<td>Supported Bluetooth profiles as defined in the Bluetooth specification</td>
<td>Locked</td>
<td>Literal (bag)</td>
<td>&quot;dialup&quot;, &quot;lanAccess&quot;</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>Name and model number of the device CPU.</td>
<td>Locked</td>
<td>Literal</td>
<td>&quot;Pentium III&quot;, &quot;PowerPC</td>
</tr>
<tr>
<td></td>
<td>ColorCapable</td>
<td>Indicates whether the device's display supports colour. &quot;Yes&quot; means colour</td>
<td>Override</td>
<td>Boolean</td>
<td>&quot;Yes&quot;, &quot;No&quot;</td>
</tr>
<tr>
<td></td>
<td>ImageCapable</td>
<td>Indicates whether the device supports the display of images. If the value is</td>
<td>Locked</td>
<td>Boolean</td>
<td>&quot;Yes&quot;, &quot;No&quot;</td>
</tr>
<tr>
<td></td>
<td>InputCharSet</td>
<td>List of character sets supported by the device for text entry. Property's</td>
<td>Append</td>
<td>Literal (bag)</td>
<td>&quot;US-ASCII&quot;, &quot;ISO-8859-1&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>value is a list of character sets, where each item in the list is a</td>
<td></td>
<td></td>
<td>&quot;Shift_JIS&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>character set name, as registered with IANA [78].</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Keyboard</td>
<td>Type of keyboard supported by the device, as an indicator of ease of text</td>
<td>Locked</td>
<td>Literal</td>
<td>&quot;Disambiguating&quot;, &quot;Qwerty&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>entry.</td>
<td></td>
<td></td>
<td>&quot;PhoneKeypad&quot;</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>Model number assigned to the terminal device by the vendor or manufacturer.</td>
<td>Locked</td>
<td>Literal</td>
<td>&quot;Mustang GT&quot;, &quot;Q30&quot;</td>
</tr>
<tr>
<td></td>
<td>NumberOfSoftKeys</td>
<td>Number of soft keys available on the device.</td>
<td>Locked</td>
<td>Number</td>
<td>&quot;3&quot;, &quot;2&quot;</td>
</tr>
</tbody>
</table>
### Appendix A. UAProf Terminal Capability Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OutputCharSet</strong></td>
<td>List of character sets supported by the device for output to the display. Property value is a list of character sets, where each item in the list is a character set name, as registered with IANA [78].</td>
<td>Append</td>
<td>Literal (bag) &quot;US-ASCII&quot;, &quot;ISO-8859-1&quot;, &quot;Shift_JIS&quot;</td>
</tr>
<tr>
<td><strong>PixelAspectRatio</strong></td>
<td>Ratio of pixel width to pixel height.</td>
<td>Locked</td>
<td>Dimension &quot;1x2&quot;</td>
</tr>
<tr>
<td><strong>PointingResolution</strong></td>
<td>Type of resolution of the pointing accessory supported by the device.</td>
<td>Locked</td>
<td>Literal &quot;Character&quot;, &quot;Line&quot;, &quot;Pixel&quot;</td>
</tr>
<tr>
<td><strong>ScreenSize</strong></td>
<td>The size of the device's screen in units of pixels, composed of the screen width and the screen height.</td>
<td>Locked</td>
<td>Dimension &quot;160x160&quot;, &quot;640x480&quot;</td>
</tr>
<tr>
<td><strong>ScreenSizeChar</strong></td>
<td>Size of the device's screen in units of characters, composed of the screen width and screen height. The device's standard font should be used to determine this property's value. (Number of characters per row) x (Number of rows). In calculating this attribute use the largest character in the device's default font.</td>
<td>Locked</td>
<td>Dimension &quot;12x4&quot;, &quot;16x8&quot;</td>
</tr>
<tr>
<td><strong>SoundOutputCapable</strong></td>
<td>Indicates whether the device supports sound output through an external speaker, headphone jack, or other sound output mechanism.</td>
<td>Locked</td>
<td>Boolean &quot;Yes&quot;, &quot;No&quot;</td>
</tr>
<tr>
<td><strong>StandardFontProportional</strong></td>
<td>Indicates whether the device's standard font is proportional.</td>
<td>Locked</td>
<td>Boolean &quot;Yes&quot;, &quot;No&quot;</td>
</tr>
<tr>
<td><strong>TextInputCapable</strong></td>
<td>Indicates whether the device supports alphanumeric text entry. &quot;Yes&quot; means the device supports entry of both letters and digits. &quot;No&quot; means the device supports only entry of digits.</td>
<td>Locked</td>
<td>Boolean &quot;Yes&quot;, &quot;No&quot;</td>
</tr>
<tr>
<td><strong>Vendor</strong></td>
<td>Name of the vendor manufacturing the terminal device.</td>
<td>Locked</td>
<td>Literal &quot;Ford&quot;, &quot;Lexus&quot;</td>
</tr>
</tbody>
</table>
Appendix A. UAProf Terminal Capability Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Resolution</th>
<th>Datatype</th>
<th>Example(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VoiceInputCapable</td>
<td>Indicates whether the device supports any form of voice input, including speech recognition. This includes voice-enabled browsers.</td>
<td>Locked</td>
<td>Boolean</td>
<td>&quot;Yes&quot;, &quot;No&quot;</td>
</tr>
</tbody>
</table>

Table A.2 UAProf Terminal Software Capabilities

<table>
<thead>
<tr>
<th>Component: SoftwarePlatform</th>
<th>Attribute</th>
<th>Description</th>
<th>Resolution</th>
<th>Datatype</th>
<th>Example(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AcceptDownloadableSoftware</td>
<td>Indicates the user's preference on whether to accept downloadable software.</td>
<td>Locked</td>
<td>Boolean</td>
<td>&quot;Yes&quot;, &quot;No&quot;</td>
</tr>
<tr>
<td></td>
<td>AudioInputEncoder</td>
<td>List of audio input encoders supported by the device.</td>
<td>Append</td>
<td>Literal (bag)</td>
<td>&quot;G.711&quot;</td>
</tr>
<tr>
<td></td>
<td>CLIPPlatform</td>
<td>The list of standard Common Language Infrastructure platforms and profiles installed in the device. Each item in the list is a name token describing the name and edition of the CLI platform specification including the name of the profile specification.</td>
<td>Append</td>
<td>Literal (bag)</td>
<td>&quot;Standard CLI 2002/Compact&quot;, &quot;Standard CLI 2002/Kernel&quot;</td>
</tr>
<tr>
<td></td>
<td>CcppAccept</td>
<td>List of content types the device supports. Property value is a list of MIME types, where each item in the list is a content type descriptor as specified by RFC 2045 [79].</td>
<td>Append</td>
<td>Literal (bag)</td>
<td>&quot;text/html&quot;, &quot;text/plain&quot;, &quot;text/html&quot;, &quot;image/gif&quot;</td>
</tr>
<tr>
<td></td>
<td>CcppAccept-Charset</td>
<td>List of character sets the device supports. Property value is a list of character sets, where each item in the list is a character set name registered with IANA [78].</td>
<td>Append</td>
<td>Literal (bag)</td>
<td>&quot;US-ASCII&quot;, &quot;ISO-8859-1&quot;, &quot;Shift_JIS&quot;</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Constraint</td>
<td>Value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------</td>
<td>----------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accept-Encoding</td>
<td>List of transfer encodings the device supports. Property value is a list of transfer encodings, where each item in the list is a transfer encoding name as specified by RFC 2045 and registered with IANA [79].</td>
<td>Append</td>
<td>&quot;base64&quot;, &quot;quoted-printable&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accept-Language</td>
<td>List of preferred document languages. If a resource is available in more than one natural language, the server can use this property to determine which version of the resource to send to the device. The first item in the list should be considered the user's first choice, the second the second choice, and so on. Property value is a list of natural languages, where each item in the list is the name of a language as defined by RFC 3066 [80].</td>
<td>Append</td>
<td>&quot;zh-CN&quot;, &quot;en&quot;, &quot;fr&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DownloadableSoftwareSupport</td>
<td>List of executable content types which the device supports and which it is willing to accept from the network. The property value is a list of MIME types, where each item in the list is a content type descriptor as specified by RFC 2045 [79].</td>
<td>Locked</td>
<td>&quot;application/x-msdos-exe&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Email-URI-Schemes</td>
<td>List of URI schemes the device supports for accessing e-mail. Property value is a list of URI schemes, where each item in the list is a URI scheme as defined in RFC 2396 [81].</td>
<td>Override</td>
<td>&quot;pop&quot;, &quot;imap&quot;, &quot;http&quot;, &quot;https&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Type</td>
<td>Value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------</td>
<td>--------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JVMVersion</td>
<td>List of the Java virtual machines installed on the device. Each item in the list is a name token describing the vendor and version of the VM.</td>
<td>Append</td>
<td>Literal (bag) &quot;SunJRE/1.2&quot;, &quot;MS JVM/1.0&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JavaEnabled</td>
<td>Indicates whether the device supports a Java virtual machine.</td>
<td>Locked</td>
<td>Boolean &quot;Yes&quot;, &quot;No&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JavaPackage</td>
<td>(from J2EE Client Provisioning) Details about optional packages installed on the device over and above those that are part of the Java profile, and the versions of these additional packages.</td>
<td>Append</td>
<td>Literal (bag) &quot;com.acme.reg.exp/1.1&quot;, &quot;com.acme.helper/3.0&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JavaPlatform</td>
<td>The list of JAVA platforms and profiles installed in the device. Each item in the list is a name token describing compatibility with the name and version of the java platform specification or the name and version of the profile specification name (if profile is included in the device). See [82] for more information.</td>
<td>Append</td>
<td>Literal (bag) &quot;PersonalJava&quot;, &quot;CLDC&quot;, &quot;MIDP&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JavaProtocol</td>
<td>(from J2EE Client Provisioning) Details about protocols supported by the device over and above those that are part of the standard Java profile indicated and the versions of these additional protocols.</td>
<td>Append</td>
<td>Literal (bag) &quot;sms/1.0&quot;, &quot;file/1.0&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>MexeClassmarks</strong></td>
<td>List of MExE classmarks supported by the device. Value &quot;1&quot; means the MExE device supports WAP, value &quot;2&quot; means that MExE device supports Personal Java, value &quot;3&quot; means that MExE device supports MIDP applications and value &quot;4&quot; means the device supports the CLI Platform.</td>
<td>Locked</td>
<td>Literal (bag)</td>
<td>&quot;1&quot;, &quot;3&quot;</td>
<td></td>
</tr>
<tr>
<td><strong>MexeSecureDomains</strong></td>
<td>Indicates whether the device's supports MExE security domains. &quot;Yes&quot; means that security domains are supported in accordance with MExE specifications identified by the MexeSpec attribute. &quot;No&quot; means that security domains are not supported and the device has only untrusted domain (area).</td>
<td>Locked</td>
<td>Boolean</td>
<td>&quot;Yes&quot;, &quot;No&quot;</td>
<td></td>
</tr>
<tr>
<td><strong>MexeSpec</strong></td>
<td>Class mark specialisation. Refers to the first two digits of the version of the MExE Stage 2 spec.</td>
<td>Locked</td>
<td>Literal</td>
<td>&quot;7.02&quot;</td>
<td></td>
</tr>
<tr>
<td><strong>OSName</strong></td>
<td>Name of the device's operating system.</td>
<td>Locked</td>
<td>Literal</td>
<td>&quot;Mac OS&quot;, &quot;Windows NT&quot;</td>
<td></td>
</tr>
<tr>
<td><strong>OSVendor</strong></td>
<td>Vendor of the device's operating system.</td>
<td>Locked</td>
<td>Literal</td>
<td>&quot;Apple&quot;, &quot;Microsoft&quot;</td>
<td></td>
</tr>
<tr>
<td><strong>OSVersion</strong></td>
<td>Version of the device's operating system.</td>
<td>Locked</td>
<td>Literal</td>
<td>&quot;6.0&quot;, &quot;4.5&quot;</td>
<td></td>
</tr>
<tr>
<td><strong>RecipientAppAgent</strong></td>
<td>User agent associated with the current request. Value should match the name of one of the components in the profile. A component name is specified by the ID attribute on the prf:Component element containing the properties of that component.</td>
<td>Locked</td>
<td>Literal</td>
<td>&quot;BrowserMail&quot;</td>
<td></td>
</tr>
</tbody>
</table>
### Appendix A. UAProf Terminal Capability Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Resolution</th>
<th>Datatype</th>
<th>Example(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SoftwareNumber</td>
<td>Version of the device-specific software (firmware) to which the device's low-level software conforms.</td>
<td>Locked</td>
<td>Literal</td>
<td>&quot;2&quot;</td>
</tr>
<tr>
<td>VideoInputEncoder</td>
<td>List of video input encoders supported by the device.</td>
<td>Append</td>
<td>(bag)</td>
<td>&quot;MPEG-1&quot;, &quot;MPEG-2&quot;, &quot;H.261&quot;</td>
</tr>
</tbody>
</table>

### Table A.3 UAProf Terminal Network Capabilities

<table>
<thead>
<tr>
<th>Component: NetworkCharacteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
</tr>
<tr>
<td>CurrentBearerService</td>
</tr>
<tr>
<td>SecuritySupport</td>
</tr>
<tr>
<td>SupportedBearers</td>
</tr>
<tr>
<td>SupportedBluetoothVersion</td>
</tr>
</tbody>
</table>

### Table A.4 UAProf Terminal Browser User Agent Capabilities

<table>
<thead>
<tr>
<th>Component: BrowserUA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
</tr>
<tr>
<td>BrowserName</td>
</tr>
<tr>
<td>BrowserVersion</td>
</tr>
</tbody>
</table>
### Appendix A. UAProf Terminal Capability Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default Data Type</th>
<th>Property Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DownloadableBrowserApps</td>
<td>List of executable content types which the browser supports and which it is willing to accept from the network. The property value is a list of MIME types, where each item in the list is a content type descriptor as specified by RFC 2045 [79].</td>
<td>Append (bag)</td>
<td>&quot;application/x-java-vm/java-applet&quot;</td>
</tr>
<tr>
<td>FramesCapable</td>
<td>Indicates whether the browser is capable of displaying frames.</td>
<td>Override Boolean</td>
<td>&quot;Yes&quot;, &quot;No&quot;</td>
</tr>
<tr>
<td>Html1Version</td>
<td>Version of HyperText Markup Language (HTML) supported by the browser.</td>
<td>Locked Literal</td>
<td>&quot;2.0&quot;, &quot;3.2&quot;, &quot;4.0&quot;</td>
</tr>
<tr>
<td>JavaAppletEnabled</td>
<td>Indicates whether the browser supports Java applets.</td>
<td>Locked Boolean</td>
<td>&quot;Yes&quot;, &quot;No&quot;</td>
</tr>
<tr>
<td>JavaScriptEnabled</td>
<td>Indicates whether the browser supports JavaScript.</td>
<td>Locked Boolean</td>
<td>&quot;Yes&quot;, &quot;No&quot;</td>
</tr>
<tr>
<td>JavaScriptVersion</td>
<td>Version of the JavaScript language supported by the browser.</td>
<td>Locked Literal</td>
<td>&quot;1.4&quot;</td>
</tr>
<tr>
<td>PreferenceForFrames</td>
<td>Indicates the user's preference for receiving HTML content that contains frames.</td>
<td>Locked Boolean</td>
<td>&quot;Yes&quot;, &quot;No&quot;</td>
</tr>
<tr>
<td>TablesCapable</td>
<td>Indicates whether the browser is capable of displaying tables.</td>
<td>Locked Boolean</td>
<td>&quot;Yes&quot;, &quot;No&quot;</td>
</tr>
</tbody>
</table>
### Table A.5 UAProf Terminal WAP Characteristics

<table>
<thead>
<tr>
<th>Component: WapCharacteristics</th>
<th>Attribute</th>
<th>Description</th>
<th>Resolution</th>
<th>Datatype</th>
<th>Example(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DrmClass</td>
<td>DRM Conformance Class as defined in OMA-Download-DRM-v1_0 [84]</td>
<td>Locked</td>
<td>Literal (bag)</td>
<td>&quot;ForwardLock&quot;, &quot;CombinedDelivery&quot;, &quot;SeparateDelivery&quot;</td>
<td></td>
</tr>
<tr>
<td>DrmConstraints</td>
<td>DRM permission constraints as defined in OMA-Download-DRMREL-v1_0 [85]. The datetime and interval constraints depend on having a secure clock in the terminal.</td>
<td>Locked</td>
<td>Literal (bag)</td>
<td>&quot;datetime&quot;, &quot;interval&quot;</td>
<td></td>
</tr>
<tr>
<td>OmaDownload</td>
<td>Supports OMA Download as defined in OMA-Download-OTA-v1_0 [86]</td>
<td>Locked</td>
<td>Boolean</td>
<td>&quot;Yes&quot;, &quot;No&quot;</td>
<td></td>
</tr>
<tr>
<td>SupportedPictogramSet</td>
<td>Pictogram classes supported by the device as defined in &quot;WAP Pictogram specification&quot;.</td>
<td>Append</td>
<td>Literal (bag)</td>
<td>&quot;core&quot;, &quot;core/operation&quot;, &quot;human&quot;</td>
<td></td>
</tr>
<tr>
<td>WapDeviceClass</td>
<td>Classification of the device based on capabilities as identified in the WAP 1.1 specifications. Current values are &quot;A&quot;, &quot;B&quot; and &quot;C&quot;.</td>
<td>Locked</td>
<td>Literal</td>
<td>&quot;A&quot;</td>
<td></td>
</tr>
</tbody>
</table>
### Appendix A. UAProf Terminal Capability Attributes

#### WapVersion
- Version of WAP supported.
- Locked
- Literal
- "1.1", "1.2.1", "2.0"

#### WmlDeckSize
- Maximum size of a WML deck that can be downloaded to the device. This may be an estimate of the maximum size if the true maximum size is not known. Value is number of bytes.
- Locked
- Number
- "4096"

#### WmlScriptLibraries
- List of mandatory and optional libraries supported in the device's WMLScript VM.
- Locked
- Literal (bag)
- "Lang", "Float", "String", "URL", "WMLBrowser", "Dialogs", "PSTOR"

#### WmlScriptVersion
- List of WMLScript versions supported by the device. Property value is a list of version numbers, where each item in the list is a version string conforming to Version.
- Append
- Literal (bag)
- "1.1", "1.2"

#### WmlVersion
- List of WML language versions supported by the device. Property value is a list of version numbers, where each item in the list is a version string conforming to Version.
- Append
- Literal (bag)
- "1.1", "2.0"

#### WtaVersion
- Version of WTA user agent.
- Locked
- Literal
- "1.1"

#### WtaLibraries
- List of WTAI network common and network specific libraries supported by the device. Property value is a list of WTA library names, where each item in the list is a library name as specified by "WAP WTAI" and its addendums. Any future addendums to "WAP WTAI" should be reflected in the values of this property.
- Locked
- Literal (bag)
- "WTAVoiceCall", "WTAAddText", "WTAAddPhoneBook", "WTAAddCallLog", "WTAAddMisc", "WTAAddGSM", "WTAAddI136", "WTAAddPDC"

### Table A.6 UAProf Terminal Push Characteristics

<table>
<thead>
<tr>
<th>Component: PushCharacteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
</tr>
<tr>
<td>Push-Accept</td>
</tr>
<tr>
<td>Attribute</td>
</tr>
<tr>
<td>-------------------</td>
</tr>
<tr>
<td>Push-Accept-AppID</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Push-Accept-Charset</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Push-Accept-Encoding</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Push-Accept-Language</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Push-MaxPushReq</td>
</tr>
<tr>
<td>Push-MsgSize</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Appendix B  UAProf of Ericsson T200A

<?xml version="1.0" encoding="UTF-8"?>
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:prf="http://www.wapforum.org/UAPROF/ccppschema-20010330#"
xmlns:rfdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
.rdf:Description ID="Profile">
<prf:component>
  <rdf:Description ID="HardwarePlatform">
    <prf:ScreenSize>101x67</prf:ScreenSize>
    <prf:Model>T200R101</prf:Model>
    <prf:InputCharSet>
      <rdf:Bag>
        <rdf:li>ISO-8859-1</rdf:li>
        <rdf:li>UTF-8</rdf:li>
      </rdf:Bag>
    </prf:InputCharSet>
    <prf:ScreenSizeChar>15x6</prf:ScreenSizeChar>
    <prf:BitsPerPixel>2</prf:BitsPerPixel>
    <prf:ColorCapable>No</prf:ColorCapable>
    <prf:TextInputCapable>Yes</prf:TextInputCapable>
    <prf:ImageCapable>Yes</prf:ImageCapable>
    <prf:Keyboard>PhoneKeypad</prf:Keyboard>
    <prf:NumberOfSoftKeys>0</prf:NumberOfSoftKeys>
    <prf:Vendor>Sony Ericsson Mobile Communications</prf:Vendor>
    <prf:OutputCharSet>
      <rdf:Bag>
        <rdf:li>ISO-8859-1</rdf:li>
        <rdf:li>UTF-8</rdf:li>
      </rdf:Bag>
    </prf:OutputCharSet>
    <prf:SoundOutputCapable>Yes</prf:SoundOutputCapable>
    <prf:StandardFontProportional>Yes</prf:StandardFontProportional>
    <prf:PixelAspectRatio>1x1</prf:PixelAspectRatio>
  </rdf:Description>
</prf:component>

<prf:component>
  <rdf:Description ID="SoftwarePlatform">
    <prf:AcceptDownloadableSoftware>No</prf:AcceptDownloadableSoftware>
  </rdf:Description>
</prf:component>

<prf:component>
  <rdf:Description ID="NetworkCharacteristics">
    <prf:SecuritySupport>WTLS class 1/2/3/signText</prf:SecuritySupport>
    <prf:SupportedBearers>
      <rdf:Bag>
        <rdf:li>TwoWaySMS</rdf:li>
        <rdf:li>CSD</rdf:li>
        <rdf:li>GPRS</rdf:li>
      </rdf:Bag>
    </prf:SupportedBearers>
  </rdf:Description>
</prf:component>

<prf:component>
  <rdf:Description ID="BrowserUA">
    <prf:BrowserName>Sony Ericsson</prf:BrowserName>
    <prf:CcppAccept>
      <rdf:Bag>
        <rdf:li>application/vnd.wap.wmlc</rdf:li>
        <rdf:li>application/vnd.wap.wbxml</rdf:li>
        <rdf:li>application/vnd.wap.wmlscript</rdf:li>
        <rdf:li>application/vnd.wap.multipart.mixed</rdf:li>
        <rdf:li>text/x-vCard</rdf:li>
        <rdf:li>text/x-vCalendar</rdf:li>
        <rdf:li>text/x-vCalendar</rdf:li>
        <rdf:li>image/vnd.wap.wbmp</rdf:li>
        <rdf:li>image/gif</rdf:li>
        <rdf:li>application/vnd.wap.wtls-ca-certificate</rdf:li>
        <rdf:li>application/vnd.wap.sic</rdf:li>
        <rdf:li>application/vnd.wap.slc</rdf:li>
        <rdf:li>application/vnd.wap.coc</rdf:li>
      </rdf:Bag>
    </prf:CcppAccept>
    <prf:CcppAccept-Charset>
      <rdf:Bag>
        <rdf:li>ISO-8859-1</rdf:li>
        <rdf:li>UTF-8</rdf:li>
      </rdf:Bag>
    </prf:CcppAccept-Charset>
  </rdf:Description>
</prf:component>
</RDF>
<rdf:Bag>
  <rdf:li>ISO-8859-1</rdf:li>
  <rdf:li>UTF-8</rdf:li>
</rdf:Bag>
</prf:CcppAccept-Charset>
</prf:CcppAccept-Encoding>
</rdf:Description>
</prf:component>
<pf:component>
<pf:component>
</rdf:Description>
</prf:component>
</rdf:Description>
</rdf:RDF>
Appendix C  Implementation Code of Important Policy Related Classes

```java
/*
 * HttpHeadersDataAccessObject.java
 * Created on September 20, 2003, 1:31 PM
 */
package uk.ac.surrey.Policy;
import java.net.*;
import java.io.*;
import java.lang.Exception;

/**
 * @author Alvin Yew
 */
public class HttpHeadersDataAccessObject {
    private String fullContent;
    /*
     * Creates a new instance of HttpHeadersDataAccessObject
     */
    public HttpHeadersDataAccessObject(String urlIn) throws Exception {
        fullContent = new String();
        try {
            URL url = new URL(urlIn);
            BufferedReader in = new BufferedReader(new InputStreamReader(url.openStream()));
            String str;
            while ((str = in.readLine()) != null) {
                fullContent += (str + "\n");
            }
            in.close();
        } catch (MalformedURLException e) {
            throw new Exception(e);
        } catch (IOException e) {
            throw new Exception(e);
        }
        public String getContent() {
            return fullContent;
        }
    }
}
```
Appendix C. Implementation Code of Important Policy Related Classes

/*
 * PolicyDecisionLogic.java
 * Created on November 19, 2003, 3:27 PM
 */
package uk.ac.surrey.PolicyDecisionLogic;
import uk.ac.surrey.Policy.*;
import java.util.Set;
import java.util.List;
import java.util.Iterator;
import java.util.TreeMap;
import java.util.HashMap;
import java.util.ArrayList;
*/
/*
 * @author Alvin Yew
 */
public class PolicyDecisionLogic {
    private String profileAttribute;
    private PolicyValue value;
    private TreeMap policies;
    private HashMap policyNames;
    private PolicySet[] readiedPolicies;
    private boolean decisionStrategy;

    /** Creates a new instance of PolicyDecisionLogic */
    /*
     * decisionStrategy parameter represents the decision strategy to evaluate
     * policies. False = first matching strategy. True = all matching strategy.
     */
    public PolicyDecisionLogic(boolean decisionStrategyIn) {
        policies = new TreeMap();
        policyNames = new HashMap();
        readiedPolicies = null;
        decisionStrategy = decisionStrategyIn;
    }

    public boolean addPolicySetIntoDecision(PolicySet policyIn, int priorityIn) {
        if (policyIn != null && priorityIn >= 0) {
            Integer tempint = new Integer(priorityIn);
            if (!this.policies.containsKey(tempint)) {
                this.policies.put(tempint, policyIn);
                this.policyNames.put(policyIn.getCommonName(), tempInt);
                return this.loadDecisionLogic();
            } else {
                return false;
            }
        } else {
            return false;
        }
    }

    public boolean replaceExistingPolicySet(PolicySet policyIn, int priorityIn) {
        if (policyIn != null && priorityIn >= 0) {
            Integer tempint = new Integer(priorityIn);
            this.policies.put(tempInt, policyIn);
            this.policyNames.put(policyIn.getCommonName(), tempInt);
            return true;
        } else {
            return false;
        }
    }

    public boolean removePolicy(String policyCommonName) {
        if (this.policyNames.containsKey(policyCommonName)) {
            return false;
        }
    }
}

133
Appendix C. Implementation Code of Important Policy Related Classes

```java
Integer temp = (Integer) this.policyNames.get(policyCommonName);
if (this.policies.containsKey(temp)) {
    this.policies.remove(temp);
    this.policyNames.remove(policyCommonName);
    return this.loadDecisionLogic();
} else {
    return false;
}
else {
    return false;
}

public PolicySet[] getPoliciesInOrderOfHighestToLowestPriority() {
    Set temp = this.policies.entrySet();
    PolicySet[] tempResult = new PolicySet[temp.size()];
    int i = temp.size() - 1;
    for (Iterator it = temp.iterator(); it.hasNext();)
    {
        PolicySet tempPolicy = (PolicySet) it.next();
        tempResult[i] = tempPolicy;
        i = i - 1;
    }
    return tempResult;
}

public boolean loadDecisionLogic() {
    this.readiedPolicies = this.getPoliciesInOrderOfHighestToLowestPriority();
    if (this.readiedPolicies != null && this.readiedPolicies.length > 0) {
        return true;
    } else {
        return false;
    }
}

public String[] evaluate(Profile profile) {
    boolean temp = false;
    ArrayList tempArray = new ArrayList();
    for (int i = 0; i < this.readiedPolicies.length && !temp; i++) {
        PolicySet ps = readiedPolicies[i];
        boolean temp2 = ps.evaluate(profile);
        if (temp2) {
            tempArray.add(ps.getAction());
            if (!decisionStrategy) {
                temp = true;
            }
        }
    }
    String[] tempResult = (String[]) tempArray.toArray();
    return tempResult;
}
```
package uk.ac.surrey.Policy;

import javax.ccpp.Profile;
import java.util.ArrayList;
import java.util.TreeMap;
import java.util.HashMap;
import java.util.Set;
import java.util.Iterator;

public class PolicyGroup extends PolicySet {
    private String commonName;
    private PolicySet[] policySet;
    private int size = 0;
    private Integer priority;
    private String[] actionResult;
    private boolean sequenceOverride = true;
    private boolean decisionStrategy = false;
    private TreeMap policies;
    private HashMap policyNames;

    public PolicyGroup(String name) {
        this.commonName = name;
        this.priority = null;
        this.actionResult = new String[2];
        this.policies = new TreeMap();
        this.policyNames = new HashMap();
    }

    public void setPriority(String priorityValue) {
        this.priority = Integer.getInteger(priorityValue);
    }

    public void setDecisionStrategy(boolean decisionStrategy) {
        this.decisionStrategy = decisionStrategy;
    }

    public void setSequenceOverride(boolean overBool) {
        this.sequenceOverride = overBool;
    }

    public boolean hasSequence() {
        if (sequenceOverride)
            return true;
        else {
            boolean result = false;
            for (int i = 0; i < policySet.length && ! result; i++) {
                PolicySet temp = policySet[i];
                result = temp.hasSequence();
            }
        return result;
    }

    public boolean hasPriority() {
        return (this.priority != null);
    }

    public int getPriority() {
        return this.priority.intValue();
    }

    public void addPolicySet(PolicySet set) {
        int temp = set.getPriority();
        if (set != null && temp >= 0) {
            Integer tempint = new Integer(temp);
            if (this.policies.containsKey(tempint)) {
                this.policies.put(tempint, set);
            }
        }
    }
}
Appendix C. Implementation Code of Important Policy Related Classes

```java
this.policyNames.put(set.getCommonName(), tempInt);

Set temp2 = this.policies.entrySet();
PolicySet[] tempResult = new PolicySet[temp2.size()];
int i = temp2.size() - 1;
for (Iterator it = temp2.iterator(); it.hasNext();)
    PolicySet tempPolicy = (PolicySet) it.next();
    tempResult[i] = tempPolicy;
    i = i - 1;
this.policySet = tempResult;

public boolean removePolicy(String policyCommonName) {
    if (this.policyNames.containsKey(policyCommonName)) {
        Integer temp = (Integer) this.policyNames.get(policyCommonName);
        if (this.policies.containsKey(temp)) {
            this.policies.remove(temp);
            this.policyNames.remove(policyCommonName);
        } else { return false; }
    } else { return false; }
    return true;
}

public PolicySet[] getPolicySet() {
    return this.policySet;
}

public boolean evaluate(Profile profile) {
    boolean temp = false;
    boolean finalTemp = false;
    ArrayList tempArray = new ArrayList();
    for (int i = 0; i < policySet.length && !temp; i++) {
        PolicySet ps = policySet[i];
        boolean temp2 = ps.evaluate(profile);
        if (temp2) {
            tempArray.add(ps.getAction());
            if (!finalTemp) {
                if (!decisionStrategy) {
                    temp = true;
                    if (!finalTemp) {
                        finalTemp = true;
                    }
                }
                this.actionResult = (String[]) tempArray.toArray();
                return finalTemp;
            }
        }
    }
    return temp;
}

public String[] getAction() {
    return this.actionResult;
}
```
Appendix C. Implementation Code of Important Policy Related Classes

```java
void implementationMethod()
{
    // Implementation code goes here
}
```
package uk.ac.surrey.Policy;
import java.util.ArrayList;
import org.csapi.jr.se.policy.provisioning.*;
import javax.ccpp.Profile;
/*
* Created on 18-Sep-2003
*/
/**
* Author: Alvin Yew
* To change the template for this generated type comment go to
* Window>Preferences>Java>Code Generation&Code and Comments
*/
public class PolicyRule extends PolicySet {
    /*
    * DNF = 1, CNF = 2
    */
    public int conditionListType = 1;
    public int enabled = 1;
    public boolean mandatory = true;
    public int sequencedActions = 1;
    public int executionStrategy = 1;
    private Integer priority;
    private String commonName;
    private ArrayList conditions;
    private boolean sequenceOverride = true;
    private String content;
    public PolicyRule(String name) {
        this.commonName = name;
        this.conditions = new ArrayList();
    }
    public PolicySet[] getPolicySet() {
        PolicySet[] result = {this};
        return result;
    }
    public void setPriority(String priorityValue) {
        priority = new Integer(priorityValue);
    }
    public void setSequenceOverride(boolean overBool) {
        this.sequenceOverride = overBool;
    }
    public void setSequencedActions(int sequencedActions) {
        this.sequencedActions = sequencedActions;
    }
    /*
    * The property ExecutionStrategy defines the execution strategy to be
    * used upon the sequenced actions aggregated by this PolicyRule. (An
    * equivalent ExecutionStrategy property is also defined for the
    * CompoundPolicyAction class, to provide the same indication for the
    * sequenced actions aggregated by a CompoundPolicyAction.) This
    * document defines three execution strategies:
    * Do Until Success - execute actions according to predefined order,
    * until successful execution of a single action.
    * Do All - execute ALL actions which are part of the modeled
    * set, according to their predefined order.
    * Continue doing this, even if one or more of the
    * actions fails.
    * Do Until Failure - execute actions according to predefined order,
    * until the first failure in execution of a single
    * sub-action.
    * The property definition is as follows:
    * NAME ExecutionStrategy
    * DESCRIPTION An enumeration indicating how to interpret the
    * action ordering for the actions aggregated by this
    * PolicyRule.
    * SYNTAX uint16 ([ENUM, {1=Do Until Success, 2=Do All, 3=Do
    * Until Failure}]
    * DEFAULT VALUE Do All (2)
    */
    public void setExecutionStrategy(int executionStrategy) {
        this.executionStrategy = executionStrategy;
    }
public boolean hasSequence() {
    if (sequenceOverride) {
        return true;
    } else {
        boolean result = false;
        for (int i = 0; i < this.conditions.size() && !result; i++) {
            PolicyCondition cond = (PolicyCondition) conditions.get(i);
            result = cond.hasSequence();
        }
        return result;
    }
}

public boolean hasPriority() {
    return (priority != null);
}

public int getPriority() {
    return priority.intValue();
}

public void addCondition(PolicyCondition condition) {
    this.conditions.add(condition);
}

public boolean evaluate(Profile profile) {
    if (this.conditionListType == 1) {
        boolean temp = false;
        for (int i = 0; i < conditions.size() && !temp; i++) {
            PolicyCondition tempCond = (PolicyCondition) conditions.get(i);
            temp = tempCond.evaluate(profile);
        }
        return temp;
    } else {
        boolean temp = false;
        for (int i = 0; i < conditions.size() && temp; i++) {
            PolicyCondition tempCond = (PolicyCondition) conditions.get(i);
            temp = tempCond.evaluate(profile);
        }
        return temp;
    }
}

public void addAction(String[] contentIn) {
    String tempStr = new String();
    boolean temp = false;
    for (int i = 0; i < contentIn.length && !temp; i++) {
        try {
            HttpTextContentDataAccessObject dao = new HttpTextContentDataAccessObject(contentIn[i]);
            tempStr += dao.getContent();
            if (this.executionStrategy == 1) {
                temp = true;
            }
        } catch (Exception e) {
            if (this.executionStrategy == 3) {
                temp = true;
            }
        }
    }
    this.content = tempStr;
}

public String[] getAction() {
    String[] result = { content };
    return result;
}
Appendix C. Implementation Code of Important Policy Related Classes

/ * Created on 18-Sep-2003 */
package uk.ac.surrey.Policy;
import javax.ccpp.*;
import javax.ccpp.uaprof.Dimension;
import java.util.Set;
import java.util.List;
import java.util.Iterator;
/**
 * ©author Alvin Yew */
public class SimplePolicyCondition extends PolicyCondition {
    private PolicyValue value;
    private String profileAttribute;
    private int groupNumber;
    private boolean conditionNegated;
    private CcppTypeMap map;
    public SimplePolicyCondition(
        int gNumber,
        boolean conNegated) {
        this.groupNumber = gNumber;
        this.conditionNegated = conNegated;
        this.map = CcppTypeMap.getInstance();
    }
    public boolean hasSequence() {  
        if (this.profileAttribute == null ||
            map.getCcppType(this.profileAttribute).equals("Sequence")) {  
            return false;
        } else {
            return true;
        }
    }
    public int getGroupNumber() {  
        return this.groupNumber;
    }
    public void setCondition(String attribute, PolicyValue value) {  
        this.value = value;
        this.profileAttribute = attribute;
    }
    public boolean evaluate(Profile profile) {  
        Attribute att = profile.getAttribute(profileAttribute);
        if (att == null) {  
            return false;
        }  
        String type = map.getCcppType(profileAttribute);
        if (type.equals("Boolean")) {  
            Boolean bool = (Boolean) att.getValue();
            PolicyBooleanValue bool2 = (PolicyBooleanValue) this.value;
            boolean result = bool.booleanValue() == bool2.getValue();
            return this.conditionNegated ? !result : result;
        }
        if (type.equals("Literal")) {  
            String temp = (String) att.getValue();
            PolicyStringValue str = (PolicyStringValue) this.value;
            List valueList = str.getList();
            boolean result = false;
            for (Iterator it = valueList.iterator();  
                 it.hasNext() && !result;  
            ) {  
                result = temp.equals((String) it.next());
            }
            return this.conditionNegated ? !result : result;
        }
        if (type.equals("Bag")) {  
            Set setAtt = (Set) att.getValue();
            PolicyStringValue str = (PolicyStringValue) this.value;
            List valueList = str.getList();
        }
    }

    public boolean hasSequence() {  
        if (this.profileAttribute == null ||
            map.getCcppType(this.profileAttribute).equals("Sequence")) {  
            return false;
        } else {
            return true;
        }
    }
    public int getGroupNumber() {  
        return this.groupNumber;
    }
    public void setCondition(String attribute, PolicyValue value) {  
        this.value = value;
        this.profileAttribute = attribute;
    }
    public boolean evaluate(Profile profile) {  
        Attribute att = profile.getAttribute(profileAttribute);
        if (att == null) {  
            return false;
        }  
        String type = map.getCcppType(profileAttribute);
        if (type.equals("Boolean")) {  
            Boolean bool = (Boolean) att.getValue();
            PolicyBooleanValue bool2 = (PolicyBooleanValue) this.value;
            boolean result = bool.booleanValue() == bool2.getValue();
            return this.conditionNegated ? !result : result;
        }
        if (type.equals("Literal")) {  
            String temp = (String) att.getValue();
            PolicyStringValue str = (PolicyStringValue) this.value;
            List valueList = str.getList();
            boolean result = false;
            for (Iterator it = valueList.iterator();  
                 it.hasNext() && !result;  
            ) {  
                result = temp.equals((String) it.next());
            }
            return this.conditionNegated ? !result : result;
        }
        if (type.equals("Bag")) {  
            Set setAtt = (Set) att.getValue();
            boolean tempBool = false;
            PolicyStringValue str = (PolicyStringValue) this.value;
            List valueList = str.getList();
        }
    }

    public boolean hasSequence() {  
        if (this.profileAttribute == null ||
            map.getCcppType(this.profileAttribute).equals("Sequence")) {  
            return false;
        } else {
            return true;
        }
    }
    public int getGroupNumber() {  
        return this.groupNumber;
    }
    public void setCondition(String attribute, PolicyValue value) {  
        this.value = value;
        this.profileAttribute = attribute;
    }
    public boolean evaluate(Profile profile) {  
        Attribute att = profile.getAttribute(profileAttribute);
        if (att == null) {  
            return false;
        }  
        String type = map.getCcppType(profileAttribute);
        if (type.equals("Boolean")) {  
            Boolean bool = (Boolean) att.getValue();
            PolicyBooleanValue bool2 = (PolicyBooleanValue) this.value;
            boolean result = bool.booleanValue() == bool2.getValue();
            return this.conditionNegated ? !result : result;
        }
        if (type.equals("Literal")) {  
            String temp = (String) att.getValue();
            PolicyStringValue str = (PolicyStringValue) this.value;
            List valueList = str.getList();
            boolean result = false;
            for (Iterator it = valueList.iterator();  
                 it.hasNext() && !result;  
            ) {  
                result = temp.equals((String) it.next());
            }
            return this.conditionNegated ? !result : result;
        }
        if (type.equals("Bag")) {  
            Set setAtt = (Set) att.getValue();
            boolean tempBool = false;
            PolicyStringValue str = (PolicyStringValue) this.value;
            List valueList = str.getList();
        }
    }

    public boolean hasSequence() {  
        if (this.profileAttribute == null ||
            map.getCcppType(this.profileAttribute).equals("Sequence")) {  
            return false;
        } else {
            return true;
        }
    }
    public int getGroupNumber() {  
        return this.groupNumber;
    }
    public void setCondition(String attribute, PolicyValue value) {  
        this.value = value;
        this.profileAttribute = attribute;
    }
    public boolean evaluate(Profile profile) {  
        Attribute att = profile.getAttribute(profileAttribute);
        if (att == null) {  
            return false;
        }  
        String type = map.getCcppType(profileAttribute);
        if (type.equals("Boolean")) {  
            Boolean bool = (Boolean) att.getValue();
            PolicyBooleanValue bool2 = (PolicyBooleanValue) this.value;
            boolean result = bool.booleanValue() == bool2.getValue();
            return this.conditionNegated ? !result : result;
        }
        if (type.equals("Literal")) {  
            String temp = (String) att.getValue();
            PolicyStringValue str = (PolicyStringValue) this.value;
            List valueList = str.getList();
            boolean result = false;
            for (Iterator it = valueList.iterator();  
                 it.hasNext() && !result;  
            ) {  
                result = temp.equals((String) it.next());
            }
            return this.conditionNegated ? !result : result;
        }
        if (type.equals("Bag")) {  
            Set setAtt = (Set) att.getValue();
            boolean tempBool = false;
            PolicyStringValue str = (PolicyStringValue) this.value;
            List valueList = str.getList();
        }
    }
}

140
for (Iterator it = setAtt.iterator(); it.hasNext() && !tempBool) {
    String temp = (String) it.next();
    tempBool = this.compareLiteral(temp);
}
return this.conditionNegated ? !tempBool : tempBool;
}

if (type.equals("Sequence")) {
    List listAtt = (List) att.getValue();
    boolean tempBool = false;
    for (Iterator it = listAtt.iterator(); it.hasNext() && !tempBool;)
    { String temp = (String) it.next();
      tempBool = this.compareLiteral(temp);
    }
    return this.conditionNegated ? !tempBool : tempBool;
}

if (type.equals("Number")) {
    PolicyIntegerValue numVal = (PolicyIntegerValue) this.value;
    Integer profileInt = (Integer) att.getValue();
    List numList = numVal.getList();
    for (Iterator it = numList.iterator(); it.hasNext()) {
        String tempStr = (String) it.next();
        int indexDot = tempStr.indexOf("..");
        if (indexDot == -1) {
            int iValue = Integer.parseInt(tempStr);
            if (iValue == profileInt.intValue()) {
                return this.conditionNegated ? false : true;
            } else {
                /* this is a range */
                int iValueLow = Integer.parseInt(tempStr.substring(0, indexDot));
                String sValueHigh = tempStr.substring(indexDot + 2, tempStr.length());
                if (sValueHigh.equals("INFINITY")) {
                    if (profileInt.intValue() >= iValueLow) {
                        return this.conditionNegated ? false : true;
                    } else {
                        boolean resolve1 = (profileInt.intValue() >= iValueLow);
                        boolean resolve2 =
                            (profileInt.intValue() <= Integer.parseInt(sValueHigh));
                        if (resolve1 && resolve2) {
                            return this.conditionNegated ? false : true;
                        }
                    }
                }
            }
        }
    }
}

if (type.equals("Dimension")) {
    PolicyIntegerValue pintValue = (PolicyIntegerValue) this.value;
    List tempList = pintValue.getList();
    int width;
    int height;
    Dimension temp = (Dimension) att.getValue();
    width = temp.getWidth();
    height = temp.getHeight();
    int iValueHLow;
    int iValueHHigh;
    int iValueWLow;
    int iValueWHigh;
    String sValueWHigh;
    String sValueHHigh;
    String sValueW;
    String sValueH;
    if (tempList.size() >= 2) {
        String sValueW = (String) tempList.get(0);
        String sValueH = (String) tempList.get(1);
        /* if the width is a range */
        if (sValueW.indexOf("..") != -1) {
            // Code for range in width
        } else {
            // Code for exact value in width
        }
        String sValueH = (String) tempList.get(1);
        /* if the height is a range */
        if (sValueH.indexOf("..") != -1) {
            // Code for range in height
        } else {
            // Code for exact value in height
        }
    }
}
int tempInt = sValueW.indexOf("..");
iValueWLow =
    Integer.parseInt(sValueW.substring(0, tempInt));
sValueWHigh =
    sValueW.substring(tempInt + 2, sValueW.length());

if (sValueWHigh.equals("INFINITY")) {
    iValueWHigh = Integer.parseInt(sValueWHigh);
    if (width != iValueWLow || width != iValueWHigh) {
        if (width > iValueWLow && width < Integer.parseInt(sValueWHigh)) {
            return this.conditionNegated ? false : true;
        } else if (width < iValueWLow || width > iValueWLow) {
            return this.conditionNegated ? true : false;
        } else {
            /* this means we have to resolve solely on height */
            int indexDot = sValueH.indexOf("..");
            if (indexDot == -1) {
                iValueH = Integer.parseInt(sValueH);
                return this.conditionNegated ? (iValueH != height) : (iValueH == height);
            } else {
                /* this is a range */
                iValueHLow =
                    Integer.parseInt(
                        sValueH.substring(0, indexDot));
                sValueHHigh =
                    sValueH.substring(indexDot + 2, sValueH.length());
                if (sValueHHigh.equals("INFINITY")) {
                    return this.conditionNegated ? (height >= iValueHLow) : (height >= iValueHLow);
                } else {
                    boolean resolve1 = (height >= iValueHLow);
                    boolean resolve2 =
                        (height <= Integer.parseInt(sValueHHigh));
                    return this.conditionNegated ? !(resolve1 && resolve2) : (resolve1 && resolve2);
                }
            }
        }
    } else {
        if (sValueWHigh.equals("INFINITY")) {
            if (width > iValueWLow) {
                return this.conditionNegated ? false : true;
            } else if (width < iValueWLow) {
                return this.conditionNegated ? true : false;
            } else {
                /* this means we have to resolve solely on height */
                int indexDot = sValueH.indexOf("..");
                if (indexDot == -1) {
                    iValueH = Integer.parseInt(sValueH);
                    return this.conditionNegated ? (iValueH != height) : (iValueH == height);
                } else {
                    /* this is a range */
                    iValueHLow =
                        Integer.parseInt(
                            sValueH.substring(0, indexDot));
                    sValueHHigh =
                        sValueH.substring(indexDot + 2, sValueH.length());
                    if (sValueHHigh.equals("INFINITY")) {
                        return this.conditionNegated ? (height >= iValueHLow) : (height >= iValueHLow);
                    } else {
                        boolean resolve1 = (height >= iValueHLow);
                        boolean resolve2 =
                            (height <= Integer.parseInt(sValueHHigh));
                        return this.conditionNegated ? !(resolve1 && resolve2) : (resolve1 && resolve2);
                    }
                }
            }
        }
    }
}
Appendix C. Implementation Code of Important Policy Related Classes

```java
private boolean compareLiteral(String variableValue) {
    return variableValue.equals(value);
}
```
package uk.ac.surrey.Policy;
import java.util.ArrayList;
import javax.ccpp.Profile;

/*
 * Created on 18-Sep-2003
 */
/**
 * @author Alvin Yew
 */
public class CompoundPolicyCondition extends PolicyCondition {
    public String commonName;
    public int groupNumber;
    public boolean conditionNegated;
    public int conditionListType = 1;
    private ArrayList conditions;
    public CompoundPolicyCondition(String name, int gNumber, boolean condNegated,
            int condListType) {
        this.commonName = name;
        this.groupNumber = gNumber;
        this.conditionNegated = condNegated;
        this.conditionListType = condListType;
    }
    public int getGroupName() {
        return this.groupNumber;
    }
    public boolean evaluate(Profile profile) {
        if (this.conditionListType == 1) {
            boolean temp = false;
            for (int i = 0; i < conditions.size() && !temp; i++) {
                PolicyCondition tempCond = (PolicyCondition)
                        conditions.get(i);
                temp = tempCond.evaluate(profile);
            }
        } else {
            boolean temp = false;
            for (int i = 0; i < conditions.size() && temp; i++) {
                PolicyCondition tempCond = (PolicyCondition)
                        conditions.get(i);
                temp = tempCond.evaluate(profile);
            }
        }
        return conditionNegated ? !temp : temp;
    }
    public void addPolicyCondition(SimplePolicyCondition condition) {
        this.conditions.add(condition);
    }
    /* (non-Javadoc)
     * @see PolicyCondition#hasSequence()
     */
    public boolean hasSequence() {
        boolean result = false;
        for (int i = 0; i < this.conditions.size() && !result; i++) {
            PolicyCondition cond = (PolicyCondition) conditions.get(i);
            result = cond.hasSequence();
        }
        return result;
    }
}
Appendix D  Implementation Code of Important JSR-188 Processor Classes

```java
package uk.ac.surrey.ccsr.ccpp;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;
import javax.ccpp.Attribute;
import javax.ccpp.Component;
import javax.ccpp.Profile;
import javax.ccpp.ProfileDescription;

/**
 * A concrete Implementation of the <code>Profile</code> Interface.
 * ©author Alvin Yew, University of Surrey
 */
public class CcppProfile implements Profile {
    private HashMap listOfComponents;
    private String name;
    private Set[] componentAttributesSet;

    protected CcppProfile() {
        this.name = "";
        this.listOfComponents = new HashMap();
    }

    /**
     * Constructor for the class.
     * @param name the value that is given in the rdf:Description about= or ID=
     */
    protected CcppProfile(String name) {
        this.name = name;
        this.listOfComponents = new HashMap();
    }

    protected CcppProfile(CcppProfile profile) {
        this.name = "";
        this.componentAttributesSet = null;
        this.listOfComponents = new HashMap(profile.listOfComponents);
    }

    /**
     * Gets the <code>Attribute</code> object associated with the specific CC/PP property.
     * @param attributeName the local name of the CC/PP attribute, eg ColorCapable.
     * @return the <code>Attribute</code> object that represents the CC/PP property.
     */
    public Attribute getAttribute(String attributeName) {
        // Implementation details here
    }
}
```
Set temp = this.getComponents();
Attribute attributeResult = null;
Iterator i = temp.iterator();
boolean flag = false;
if (attributeName != null) {
    for (; i.hasNext() &amp; !flag;) {
        Component tempComponent = (Component) i.next();
        attributeResult = (Attribute) tempComponent.getAttribute(attributeName);
        if (attributeResult != null) {
            flag = true;
        }
    }
    return attributeResult;
}
/**
 * gets all the <code>Attribute</code> objects that is contained within this profile.
 * @return a Set containing all the <code>Attribute</code> objects.
 * @see javax.ccpp.Profile#getAttributes()
 */
public Set getAttributes() {
    Set temp = this.getComponents();
    Iterator i = temp.iterator();
    HashSet resultSet = new HashSet();
    for (; i.hasNext();)
    {
        Component tempComponent = (Component) i.next();
        resultSet.addAll(tempComponent.getAttributes());
    }
    return resultSet;
}
/**
 * gets a specific <code>Component</code> object that represents a component within the CC/PP profile.
 * @param localtype the local name for the component, eg HardwarePlatform.
 * @return the component requested.
 * @see javax.ccpp.Profile#getComponent(java.lang.String)
 */
public Component getComponent(String localtype) {
    if (this.listOfComponents.containsKey(localtype)) {
        return new CcppComponent((CcppComponent) this.listOfComponents.get(localtype));
    } else {
        return null;
    }
}
/**
 * gets all the components that are associated with this CC/PP profile instance.
 * @return a Set containing all the <code>Component</code> objects.
 * @see javax.ccpp.Profile#getComponents()
 * @see javax.ccpp.Component
 */
public Set getComponents() {
    return new HashSet(listOfComponents.values());
}
/**
 * have not done this yet so don't use it!
 * @see javax.ccpp.Profile#getDescription()
 */
public ProfileDescription getDescription() {
    // TODO Auto-generated method stub
    return null;
}
/**
 * inserts a <code>Component</code> object into this CC/PP profile instance.
 * @param localtype the local name of the component, ie the component type.
 * @param component the <code>Component</code> object to be inserted.
 * @see javax.ccpp.Component
 */
protected void setComponent(String localtype, Component component) {
    this.listOfComponents.put(localtype, component);
    //
    CcppComponent ccpp = (CcppComponent) component;
    //
    this.setComponentlist(ccpp.getSetOfAttributeNames());
    return;
}
 */

protected String stats() {
    int locked = 0;
    int append = 0;
    int override = 0;

    HashSet set = (HashSet) this.getComponents();
    for (Iterator it = set.iterator(); it.hasNext(); ) {
        CcppComponent comp = (CcppComponent) it.next();
        append += comp.getAppendSet().size();
        locked += comp.getLockedSet().size();
        override += comp.getOverrideSet().size();
    }

    return "Locked, Append, Override" + String.valueOf(locked) + " + " + String.valueOf(append) + " + " + String.valueOf(override);
} /*
package uk.ac.surrey.ccsr.ccpp;
import org.xml.sax.helpers.DefaultHandler;
import java.util.ArrayList;
import java.util.HashSet;
import javax.ccpp.Profile;
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;

/**
 * A SAX-implementation observer/DefaultHandler that handles the XML parsing of
 * the CC/PP RDF document and constructs a profile from it.
 */
class CcppHandler extends DefaultHandler {
    private boolean bagFlag;
    // a flag to indicate whether the current tag/literal is in a sequence
    private StringBuffer buffer;
    // a buffer to store the contents between XML tags
    private boolean componentFlag;
    // a flag to indicate that we are at a component level of the XML tree
    // (as opposite to being in a profile level). It is used when
    // <prf:component> occurs
    private boolean componentResourceFlag;
    // a temporary List object used to store the contents of a <prf:Seq>
    private String currentCcppProperty;
    // this class is used to represent the current CC/PP component being parsed
    private int internalState;
    // a flag used solely for the situation of a component resource using
    // the <rdf:type> tag to declare what CC/PP component type
    private boolean liTagFlag;
    // this represent the internal state of the parsing algorithm.
    private CcppProfile profile;
    // a flag to indicate whether the current tag/literal is in a bag
    private boolean seqFlag;
    // this class is used to hold the contents of a CC/PP property of type rdf:Seq
    private CcppComponent tempCcppComponent;
    // a string used to indicate the current CC/PP property
    // e.g. ColorCapable, InputCharSet etc.
    private String tempComponentName;
    // this class is used to find out the CC/PP property's type
    private CcppTypeMap typeMap;
    // this class is used to find out the CC/PP property's type
    private String tempProfileURI;
    // a string used to indicate whether we are in a <prf:li> part of the XML tree
    private HashSet tempSet;
    // a temporary Set object used to store the contents of a <prf:Bag>
    // this class is used to hold the contents of a CC/PP property of type rdf:Bag
    private ArrayList tempList;
    // this class is used to hold the contents of a CC/PP property of type rdf:Bag
    private String tempComponentURI;
    // a temporary List object used to store the contents of a <prf:ProfileURI>
    // this class is used to store the contents of a CC/PP property of type rdf:ProfileURI
    private CcppProfileFragment[] arrayOfPF;
    // this class is used to find out the CC/PP property's type
    private CcppComponent tempCcppComponent;
    private CcppProfile profile;
    private int internalState;
    private String tempComponentName;
    private String tempComponentURI;
    private ArrayList tempList;
    private HashSet tempSet;
    private CcppTypeMap typeMap;
    private CcppProfileFragment[] arrayOfPF;
    /**
     * Default constructor of the class.
     */
    public CcppHandler() {
        this.buffer = null;
        this.bagFlag = false;
        this.seqFlag = false;
        this.componentFlag = false;
        this.componentResourceFlag = false;
        this.liTagFlag = false;
        this.tempSet = new HashSet();
        this.tempList = new ArrayList();
        this.currentCcppProperty = null;
        this.tempComponentName = null;
        this.tempProfileURI = null;
        this.tempCcppComponent = null;
        this.internalState = -1;
    }
}
this.profile = null;
this.typeMap = CcppTypeMap.getInstance();
this.arrayOfPF = null;
}

protected CcppProfileFragment[] getProfileFragmentArray() {
    if (this.arrayOfPF.length > 0) {
        return this.arrayOfPF;
    } else {
        return null;
    }
}

protected void setProfileFragmentArray(CcppProfileFragment[] pf) {
    this.arrayOfPF = pf;
    return;
}

/**
 * observes the characters that are not in XML tags.
 * @throws org.xml.sax.SAXException
 * @see org.xml.sax.ContentHandler#characters(char[], int, int)
 */
public void characters(char[] arg0, int arg1, int arg2)
    throws SAXException {
    if (buffer != null) {
        buffer.append(arg0, arg1, arg2);
    }
}

/**
 * observes the end of the XML document.
 * @throws org.xml.sax.SAXException
 * @see org.xml.sax.ContentHandler#endDocument()
 */
public void endDocument() throws SAXException {
    this.bagFlag = false;
    this.tempSet.clear();
    this.tempList.clear();
    this.currentCcppProperty = null;
    this.internalState = -1;
    this.componentFlag = false;
    this.componentResourceFlag = false;
    this.liTagFlag = false;
    this.seqFlag = false;
    this.tempCcppComponent = null;
    this.tempComponentName = null;
    this.tempProfileURI = null;
}

/**
 * observes the end element within the XML document.
 * @throws org.xml.sax.SAXException
 * @see org.xml.sax.ContentHandler#endElement(java.lang.String,
 * java.lang.String, java.lang.String)
 */
public void endElement(String arg0, String arg1, String arg2)
    throws SAXException {
    if (arg0.compareTo("http://www.w3.org/1999/02/22-rdf-syntax-ns#") == 0
        && arg1.compareTo("RDF") == 0) {
        if (this.internalState == 0) {
            this.internalState = -1;
        } else if (
            arg0.compareTo("http://www.w3.org/1999/02/22-rdf-syntax-ns#") == 0
            && arg1.compareTo("Description") == 0) {
            if (this.componentFlag) {
                this.componentResourceFlag = false;
                if (!tempCcppComponent.getLocalName().equals("") {
                    this.componentResourceFlag = false;
                }
            }
        }
    }
}

// test condition 2
else if {
    arg0.compareTo("http://www.w3.org/1999/02/22-rdf-syntax-ns#") == 0
    && arg1.compareTo("Description") == 0) {
        if (this.componentFlag) {
            this.componentResourceFlag = false;
        if (!tempCcppComponent.getLocalName().equals("")) {
            this.componentResourceFlag = false;
        }
    }
}

149
Appendix D. Implementation Code of Important JSR-188 Processor Classes

```java
this.profile.setComponent(
    tempCcComponent.getLocalName(),
    tempCcComponent);
) else {
    // this rdf:Description is at profile level and not at component level
    this.internalState = 0;
}
// test condition 3
else if (arg2.compareTo("prf: component") == 0) {
    if (this.internalState == 2) {
        this.componentFlag = false;
        this.internalState = 1;
    }
} // test condition 5
else if {
    if (arg2.startsWith("prf:")
        && ((arg0.length() >= 51)
            && arg0.regionMatches(0, "http://www.wapforum.org/profiles/UAPROF/ccp膠heme-", 0, 51))
            && (arg0.compareTo("http://www.openmobilealliance.org/tech/profiles/UA PROF/ccp膠heme-20021212#")
            || (arg0.compareTo("http://www.openmobilealliance.org/tech/profiles/UA PROF/ccp膠heme-20030226#")))
            && (arg0.equals("http://www.openmobilealliance.org/tech/profiles/MMS/ccp膠heme-20010111#")
            || arg0.equals("http://www.wapforum.org/profiles/MMS/ccp膠heme-20010111#"))) {
        if (((this.currentCcProp.compare("HardwarePlatform") == 0
            || this.currentCcProp.compare("SoftwarePlatform") == 0
            || this.currentCcProp.compare("HapCharacteristics") == 0
            || this.currentCcProp.compare("BrowserUA") == 0
            || this.currentCcProp.compare("NetworkCharacteristics") == 0
            || this.currentCcProp.compare("PushCharacteristics") == 0
            && (this.internalState == 3))
            && (this.componentResourceFlag = false;
    if (!tempCcComponent.getLocalName().equals("")) {
        this.profile.setComponent(
            tempCcComponent.getLocalName(),
            tempCcComponent);
    )
    this.internalState = 2;
} else if (this.internalState == 4) {
    this.internalState = 3;
    if (bagFlag) {
        this.bagFlag = false;
        if (!this.tempSet.isEmpty0) {
            tempCcComponent.setSetAttribute(
                currentCcPropProperty,
            )} 
    } else { 
    }
```
new HashSet(tempSet);
    tempSet.clear();
}
        }
    }
}
        }
    }
}
// test condition 6
else if (
    argO.equals("http://www.w3.org/1999/02/22-rdf-syntax-ns#")
    && argl.equals("Bag") || argl.equals("Seq")
    && this.internalState == 4) {
}
// test condition 7
else if (
    argO.equals("http://www.w3.org/1999/02/22-rdf-syntax-ns#")
    && argl.equals("li")
    && (bagFlag || seqFlag)) {
    this.liTagFlag = false;
    if (!buffer.toString().startsWith("<!--")
        && !buffer.toString().length() != 0) {
        if (bagFlag) {
            tempSet.add(buffer.toString());
        } else {
            tempList.add(buffer.toString());
        }
    }
}
/**
 * gets the profile associated with this CC/PP RDF-XML document.
 * @return Profile
 * @see javax.ccpp.Profile
 */
Appendix D. Implementation Code of Important JSR-188 Processor Classes

```java
protected Profile getProfile() {
    return profile;
}

/**
 * observes the start of the XML document.
 * @throws org.xml.sax.SAXException
 * @see org.xml.sax.ContentHandler#startDocument()
 */
public void startDocument() throws SAXException {
}

/**
 * observes a start element within the XML document.
 * @throws org.xml.sax.SAXException
 * @see org.xml.sax.ContentHandler#startElement(String, String, String, org.xml.sax.Attributes)
 */
public void startElement(String arg0, String arg1, String arg2, Attributes arg3)
    throws SAXException {
    // test condition 1 = <RDF> start tag
    if (arg0.compareTo("http://www.w3.org/1999/02/22-rdf-syntax-ns#") == 0
        && (arg1.compareTo("RDF") == 0)) {
        if (this.internalState == -1) {
            this.tempProfileURI = arg3.getValue("xmlns:prf");
            if (this.tempProfileURI == null) {
                this.tempProfileURI = "";
            } else {
                this.internalState = 0;
            }
        } else if (arg0.compareTo("http://www.w3.org/1999/02/22-rdf-syntax-ns#") == 0
            && arg1.compareTo("Description") == 0) {
            if (this.componentFlag) {
                this.componentResourceFlag = true;
                this.tempComponentName = arg3.getValue("ID");
                if (this.tempComponentName == null) {
                    this.tempComponentName = arg3.getValue("about");
                } else if (this.tempComponentName == null) {
                    this.tempComponentName = arg3.getValue("rdf:ID");
                } else if (this.tempComponentName == null) {
                    this.tempComponentName = arg3.getValue("rdf:about");
                } else {
                    this.internalState = 3;
                }
            } else {
                // this rdf:Description is at profile level and not at component
                this.internalState = 1;
                profile = new CcppProfile(tempProfileURI);
            }
        } else {
```
// test condition 3 = <prf:component> tag
else if (arg2.compareTo("prf:component") == 0) {
    if (this.internalState == 1) {
        this.componentFlag = true;
        this.internalState = 2;
    }
}

// test condition 5 = any other tag that starts with <prf:
// <prf:component> tag will not have an effect here
else if (arg2.startsWith("prf:")
    && (arg0.length() >= 51)
    && arg0.regionMatches(
        0,
        "http://www.wapforum.org/profiles/UAPROF/ccppschema-",
        0,
        51)
    || (arg0.length() >= 23)
    && arg0.regionMatches(
        0,
        "http://www.wapforum.org",
        0,
        23))
    || arg0.equals("http://www.openmobilealliance.org/tech/profiles/UAPROF/ccppschema-20021212#")
    || arg0.equals("http://www.openmobilealliance.org/tech/profiles/UAPROF/ccppschema-20030226#"))
{
    this.currentCcppProperty = arg1;
    if (this.currentCcppProperty.compareTo("HardwarePlatform") == 0
        || this.currentCcppProperty.compareTo("SoftwarePlatform") == 0
        || this.currentCcppProperty.compareTo("WapCharacteristics") == 0
        || this.currentCcppProperty.compareTo("BrowserUA") == 0
        || this.currentCcppProperty.compareTo("NetworkCharacteristics") == 0
        || this.currentCcppProperty.compareTo("PushCharacteristics") == 0
        || this.currentCcppProperty.compareTo("MmsCharacteristics") == 0
        && (this.internalState == 2))
    {
        this.componentResourceFlag = true;
        this.tempComponentName = arg3.getValue("ID");
        if (this.tempComponentName == null) {
            this.tempComponentName = arg3.getValue("about");
        }
        if (this.tempComponentName == null) {
            this.tempComponentName = arg3.getValue("rdf:ID");
        }
        if (this.tempComponentName == null) {
            this.tempComponentName = arg3.getValue("rdf:about");
        }
        tempComponentURI = arg0 + arg1;
        this.tempCcppComponent = new CcppComponent(
            this.tempComponentName,
            this.tempComponentURI,
            this.currentCcppProperty);
        this.internalState = 3;
    } else if (this.internalState == 3) {
        this.internalState = 4;
        buffer = new StringBuffer();
    }
}

// test condition 5.1 for MMS vocabs
else if (arg0.equals("http://www.openmobilealliance.org/tech/profiles/MMS/ccppschema-20011011#"))
{
Appendix D. Implementation Code of important JSR-188 Processor Classes

|| arg0.equals("http://www.wapforum.org/profiles/MMS/ccppschema-20010111#") {
  this.currentCcppProperty = arg1,
  if (this.currentCcppProperty.compareTo("MmsCharacteristics") == 0
  && (this.internalState == 2)) {
    this.componentResourceFlag = true;
    this.tempComponentName = arg3.getValue("ID");
    if (tempComponentName == null) {
      tempComponentName = arg3.getValue("about");
    }
    if (tempComponentName == null) {
      tempComponentName = arg3.getValue("rdf:ID");
    }
    if (tempComponentName == null) {
      tempComponentName = arg3.getValue("rdf:about");
    }
    tempComponentURI = arg0 + arg1;
    this.tempCcppComponent =
      new CcppComponent(
        this.tempComponentName,
        this.tempComponentURI,
        this.currentCcppProperty);
    this.internalState = 3;
  } else if (this.internalState == 3) {
    this.internalState = 4;
    buffer = new StringBuffer();
  }
} // test condition 6 = <rdf:Bag> tag or <rdf:Seq> tag
else if (arg0.equals("http://www.w3.org/1999/02/22-rdf-syntax-ns#")
  && (arg1.equals("Bag") || arg1.equals("Seq")
  && (this.internalState == 4)) {
  bagFlag = arg1.equals("Bag");
  seqFlag = !bagFlag;
  if (bagFlag) {
    tempSet = new HashSet();
  } else {
    tempList = new ArrayList();
  }
} // test condition 7 = <rdf:li> tag
else if (arg0.equals("http://www.w3.org/1999/02/22-rdf-syntax-ns#")
  && arg1.equals("li")
  && (bagFlag || seqFlag)) {
  this.liTagFlag = true;
  buffer = new StringBuffer();
}
Appendix D. Implementation Code of Important JSR-188 Processor Classes

/*
*  Created on 20-Aug-2003
*/
package uk.ac.surrey.ccsr.ccpp;
import javax.ccpp.Profile;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;
import javax.ccpp.Attribute;
import javax.ccpp.Component;
import javax.ccpp.ProfileDescription;

/**
*  ©author Alvin Yew
*/
public class ProfileComposite implements Profile {

    private CcppProfile[] arrayPF;
    private ArrayList lockedMap;
    private HashMap appendMap;
    private ArrayList overrideMap;
    private CcppTypeMap map;
    private HashSet lockedSet;
    private HashSet overrideSet;
    private HashMap listOfComponents;

    private static final String[] fullComponentArray = {
        "HardwarePlatform",
        "SoftwarePlatform",
        "MmsCharacteristics",
        "WapCharacteristics",
        "PushCharacteristics",
        "BrowserUA",
        "NetworkCharacteristics"};

    /* Constructor which takes in an array of CcppProfile Objects */
    protected ProfileComposite(CcppProfile[] array) {
        this.arrayPF = array;
        map = CcppTypeMap.getInstance();
        this.listOfComponents = new HashMap(12, 1.0f);
    }

    /* (non-Javadoc)
*  ©see javax.ccpp.Profile#getAttribute(java.lang.String)
*/
    public Attribute getAttribute(String name) {
        Set tempSet = this.listOfComponents.keySet();
        if (tempSet.size() > 3) {
            HashSet fullComponentSet = new HashSet(Arrays.asList(fullComponentArray));
            for (Iterator it = tempSet.iterator(); it.hasNext();)
                ((CcppComponent) this.listOfComponents.get(componentName))
                    .getAttribute(name);
            fullComponentSet.remove(componentName);
        } else {
            Iterator it = fullComponentSet.iterator();
            for (it.next();
                String currentComponentName = (String) it.next();
    
    
155
Appendix D. Implementation Code of important JSR-188Processor Classes

```java
currentComponent = (CcppComponent) this.getComponent(currentComponentName);
if (currentComponent != null && currentComponent.containsAttribute(name)) {
    return currentComponent.getAttribute(name);
} else {
    return null;
}

Set componentSet = this.getComponents();
for (Iterator it = componentSet.iterator(); it.hasNext();) {
    CcppComponent temp = (CcppComponent) it.next();
    if (temp != null && temp.containsAttribute(name)) {
        return temp.getAttribute(name);
    } else {
        return null;
    }
}

/* (non-Javadoc)
* @see javax.ccpp.Profile#getAttributes()
*/
public Set getAttributes() {
    Set componentSet = null;
    Set tempSet = this.listOfComponents.keySet();
    if (tempSet.size() != 7) {
        componentSet = this.getComponents();
    } else {
        componentSet = new HashSet(this.listOfComponents.values());
    }
    Set resultSet = new HashSet();
    for (Iterator it = componentSet.iterator(); it.hasNext();) {
        CcppComponent temp = (CcppComponent) it.next();
        if (temp != null) {
            resultSet.addAll(temp.getAttributes());
        }
    }
    return resultSet;
}

/* (non-Javadoc)
* @see javax.ccpp.Profile#getComponent(java.lang.String)
*/
public Component getComponent(String localtype) {
    if (localtype.equals("HardwarePlatform") ||
     localtype.equals("SoftwarePlatform") ||
     localtype.equals("BrowserUA") ||
     localtype.equals("NetworkCharacteristics") ||
     localtype.equals("WapCharacteristics") ||
     localtype.equals("PushCharacteristics") ||
     localtype.equals("MmsCharacteristics") ) {
        if (this.listOfComponents.containsKey(localtype)) {
            Object obj = this.listOfComponents.get(localtype);
            if (obj != null) {
                return (Component) obj;
            } else {
                return null;
            }
        } else {
            return null;
        }
    } else {
        // We create the component and add it to the listOfComponents
        // 1. we get the entrySet of each profile ....
        // actually we might as well start from the second profile
        // onwards since the first profile is definitely in.
        // requirement: what we want is to search each profile for
        // the attributes contained in each component.
        int index = 0;
        CcppComponent tempComponent = null;
        ...
```
for (boolean tempComponentFlag = true; 
(index < arrayPF.length) && tempComponentFlag; 
index++) {
    Component tempComp = 
        (Component) this.arrayPF[index].getComponent(localtype); 
    tempComponentFlag = (tempComp == null); 
    if (!tempComponentFlag) {
        tempComponent = 
            new CcppComponent((CcppComponent) tempComp); 
    }
    /*
    * this for loop only executes if the component is not null!
    */
    for (int i = index; i < arrayPF.length; i++) {
        CcppComponent current = 
            (CcppComponent) arrayPF[i].getComponent(localtype); 
        if (current != null) {
            Set currentSet = current.getSetOfAttributeNames();
            for (Iterator it = currentSet.iterator(); 
                it.hasNext();)
            {
                String currentAttributeName = (String) it.next();
                String currentResolution = 
                    map.getResolution(currentAttributeName);
                String type = map.getCcppType(currentAttributeName);
                if (!tempComponent 
                        .containsAttribute(currentAttributeName) 
                    || currentResolution.equals("Override")) 
                { 
                    tempComponent.setAttribute(
                        currentAttributeName,
                        current.getAttribute(currentAttributeName)); 
                } else if (currentResolution.equals("Append")) {
                    /*
                    * Now we append. Note that append attributes can
                    */
                    if (map 
                        .getCcppType(currentAttributeName) 
                        .equals("Bag"))
                    { 
                        CcppSetAttribute set = 
                            (CcppSetAttribute) tempComponent 
                            .getAttribute(currentAttributeName);
                        CcppSetAttribute anotherSet = 
                            (CcppSetAttribute) current 
                            .getAttribute(currentAttributeName);
                        try {
                            set.append(anotherSet);
                            tempComponent.setAttribute(
                                currentAttributeName, set);
                        } catch (Exception e) {
                            e.printStackTrace();
                        }
                    } else {
                        CcppSequenceAttribute seq = 
                            (CcppSequenceAttribute) tempComponent 
                            .getAttribute(currentAttributeName);
                        CcppSequenceAttribute anotherSeq = 
                            (CcppSequenceAttribute) current 
                            .getAttribute(currentAttributeName);
                        try {
                            seq.append(anotherSeq);
                        } catch (Exception e) {
                            e.printStackTrace();
                        }
                    }
                }
            }
        } 
    }
}
tempComponent.setAttribute(
    currentAttributeName,
    seq);
} catch (Exception e) {
    e.printStackTrace();
}
}

this.listOfComponents.put(localtype, tempComponent);
if (tempComponent != null) {
    return new CcppComponent(tempComponent);
} else {
    return null;
}

/* (non-Javadoc)
 * @see javax.ccpp.Profile#getComponents()
 */
public Set getComponents() {
    /*
     * 1. create an empty set
     * 2. call getComponent(String) for each component
     * 3. if it is not null, insert it into the set
     * 4. return the set
     */
    HashSet tempSet = new HashSet();
    if (this.listOfComponents.containsKey("HardwarePlatform")) {
        tempSet.add(this.listOfComponents.get("HardwarePlatform"));
    } else {
        tempSet.add(this.getComponent("HardwarePlatform"));
    }
    if (this.listOfComponents.containsKey("SoftwarePlatform")) {
        tempSet.add(this.listOfComponents.get("SoftwarePlatform"));
    } else {
        tempSet.add(this.listOfComponents.get("SoftwarePlatform"));
    }
    if (this.listOfComponents.containsKey("BrowserUA")) {
        tempSet.add(this.listOfComponents.get("BrowserUA"));
    } else {
        tempSet.add(this.getComponent("BrowserUA"));
    }
    if (this.listOfComponents.containsKey("NetworkCharacteristics")) {
        tempSet.add(this.listOfComponents.get("NetworkCharacteristics"));
    } else {
        tempSet.add(this.getComponent("NetworkCharacteristics"));
    }
    if (this.listOfComponents.containsKey("WapCharacteristics")) {
        tempSet.add(this.listOfComponents.get("WapCharacteristics"));
    } else {
        tempSet.add(this.getComponent("WapCharacteristics"));
    }
    if (this.listOfComponents.containsKey("PushCharacteristics")) {
        tempSet.add(this.listOfComponents.get("PushCharacteristics"));
    } else {
        tempSet.add(this.getComponent("PushCharacteristics"));
    }
    if (this.listOfComponents.containsKey("MmsCharacteristics")) {
        tempSet.add(this.listOfComponents.get("MmsCharacteristics"));
    } else {
        tempSet.add(this.getComponent("MmsCharacteristics"));
    }
    return tempSet;
}
Appendix D. Implementation Code of Important JSR-182Processor Classes

/* (non-Javadoc)
 * @see javax.ccpp.Profile#getDescription()
 * /
 * public ProfileDescription getDescription() {
 *    // TODO Auto-generated method stub
 *    return null;
 * }
 */
Appendix D. Implementation Code of Important JSR-188 Processor Classes

-package uk.ac.surrey.cosr.ccpp;
-import java.util.HashMap;
-
-class ProfileCache {
-     private static HashMap cacheNoValid;
-     private static HashMap weakValid;
-     private static HashMap strongValid;
-     private static ProfileCache instance;
-
-     protected static ProfileCache getInstance() {
-         if (instance == null) {
-             instance = new ProfileCache();
-         }
-         return instance;
-     }
-
-     protected void clearCache() {
-         ProfileCache.cacheNoValid.clear();
-         ProfileCache.weakValid.clear();
-         ProfileCache.strongValid.clear();
-     }
-
-     protected void addProfile(String ref, CcppProfile profile, int validation) {
-         if ((ref != null) && ref.startsWith("http://")) {
-             if (validation == 1) {
-                 ProfileCache.cacheNoValid.put(ref, profile);
-             } else if (validation == 2) {
-                 ProfileCache.weakValid.put(ref, profile);
-             } else if (validation == 3) {
-                 ProfileCache.strongValid.put(ref, profile);
-             }
-         }
-     }
-
-     protected CcppProfile getProfile(String ref, int validation) {
-         if ((ref != null) && ref.startsWith("http://")) {
-             if (validation == 1) {  
-                 return new CcppProfile ((CcppProfile) ProfileCache.cacheNoValid.get(ref));
-             } else if (validation == 2) {
-                 return new CcppProfile ((CcppProfile) ProfileCache.weakValid.get(ref));
-             } else if (validation == 3) {
-                 return new CcppProfile ((CcppProfile) ProfileCache.strongValid.get(ref));
-             } else {
-                 return null;
-             }
-         } else {
-             return null;
-         }
-     }
-
-     protected boolean containsProfile(String ref, int validation) {
-         if ((ref != null) && ref.startsWith("http://")) {
-             if (validation == 1) {
-                 return ProfileCache.cacheNoValid.containsKey(ref);
-             } else if (validation == 2) {
-                 return ProfileCache.weakValid.containsKey(ref);
-             } else if (validation == 3) {
-                 return ProfileCache.strongValid.containsKey(ref);
-             } else {
-                 // return false;  
-             }
-         }
-     }
- }
-
-* Created on 11-Aug-2003
-*
-©author Alvin Yew
-*
-
return false;
} else {
    return false;
}
}