Device Binding for Adaptive Multimodal Interfaces

Srihathai Prammanee

Submitted for the Degree of Doctor of Philosophy from the University of Surrey

Centre for Communication System Research School of Electronics and Physical Sciences University of Surrey Guildford, Surrey GU2 7XH, UK

June 2007

© S. Prammanee 2007
Summary

Multimodal interaction is one of the taxonomies for Human-Computer Interaction (HCI). With the introduction of multimodal interactions, input/output information is becoming associated with the different human senses so that information can be presented in the most efficient and natural way. However, in mobile communication, a number of restrictions are still remnant. Mostly, these restrictions are caused by limitations of a mobile terminal's user interfaces.

This thesis introduces an architectural framework to facilitate multimodal interaction in a virtual-device environment. The framework developed is called the Multi Interface-Device Binding (MID-B) system. MID-B provides the functions and features to overcome the drawbacks of classic multimodal interaction. In the classical sense, multimodality uses a strategy that simultaneously utilises several modalities generally offered on a 'single' device. In contrast, the MID-B's mechanisms take multimodality out of the single-device scenario. In MID-B, a 'controller-device' is aware of the availability of various devices in the vicinity, each of which may host one or more user interfaces (modalities). The capabilities of those co-located devices, together with the context in which the user acts, are exploited to dynamically customise the interface services available. MID-B binds these devices into a virtual device to exploit their individual user interfaces (modalities) in a combined way.

This thesis describes the MID-B architecture and its mechanisms to collect the context information of 'devices' and 'users'. The thesis presents the methodologies to exploit that context information to dynamically adapt user interfaces.

SP, Guildford, June 2007

Key words: device context, service discovery, modality context, modality discovery, multimodal interface, multi-device binding, virtual device, Bluetooth, adaptive UI.
Acknowledgments

I would like to thank my supervisor, Dr. K. Moessner, for his guidance and suggestions on this doctoral work. Furthermore, thanks must go to B. Atingyena who tutored me English during the first year of the PhD course.

I would like to express my deepest gratitude to everyone who kindly supported me over the 9-month medical treatment, during the PhD course. Especially, my appreciation is shown to V. Vongpadungkiet, P. Phrutthithada and P. Kunama; they spent time nursing me during and after I was hospitalised. Their valuable support became the motivation for me to overcome the difficult time and to continue the PhD.

Moreover, I would like to thank Dr. V. Kueh, Dr. H. Nourizadeh, Dr. L. Huang and L. Hanzo for their friendship. Special thanks go to my lovely friends, Kelly and Mel, for sharing experiences in UK. They make my life in UK more cheerful.

Finally, I would like to thank to my family for their endless support.
## Contents

Summary..............................................................................................................................................ii  
Acknowledgments.............................................................................................................................iii  
Contents ..............................................................................................................................................iv  
List of Figures ....................................................................................................................................ix  
List of Tables ...................................................................................................................................xiii  
Glossary of Terms...............................................................................................................................xiv  
Chapter 1 ....................................................................................  1  
  1 Introduction .......................................................................................................................................1  
    1.1 Overview ..................................................................................................................................1  
    1.2 Motivation and Objective ........................................................................................................3  
    1.3 Contributions and major Achievements ..................................................................................4  
    1.4 Structure of Thesis ...................................................................................................................7  
Chapter 2 ..............................................................................................................................................9  
  2 Problems and Scope of this Research .........................................................................................9  
    2.1 Introduction ............................................................................................................................9  
    2.2 Traditional Multimodal Interfaces ..........................................................................................9  
        2.2.1 Developing Traditional Multimodal Interfaces ...............................................................9  
        2.2.2 Scope of Traditional Multimodal Interface Development ............................................11  
    2.3 Extending Multimodal Interaction into Ubiquitous Environments: Additional Challenges .................................................................................................................................11  
        2.3.1 Context Awareness .........................................................................................................12  
            2.3.1.1 Context Awareness Challenges ............................................................................12  
            2.3.1.2 Context Awareness in MID-B.............................................................................13  
        2.3.2 Service Discovery ..............................................................................................................15  
            2.3.2.1 Service Discovery Challenges ............................................................................15  
            2.3.2.2 Service Discovery in MID-B.............................................................................16  
        2.3.3 Interface Adaptation ..........................................................................................................16  
            2.3.3.1 Interface Adaptation Challenges ...........................................................................16
2.3.4 Presentation Adaptation and Connectivity Management .......................................................... 18
  2.3.4.1 Presentation Adaptation and Connectivity Management Challenges .................................. 18
  2.3.4.2 Presentation Adaptation and Connectivity Management in MID-B .................................. 19
2.4 Summary ........................................................................................................................................ 19

Chapter 3 ............................................................................................................................................... 21
3.1 Introduction .................................................................................................................................... 21
3.2 SDP ServiceSearchAttribute Transaction ...................................................................................... 21
  3.2.1 Services in Bluetooth ................................................................................................................ 21
    3.2.1.1 Service and Service Class ................................................................................................. 21
    3.2.1.2 Service Record .................................................................................................................. 22
    3.2.1.3 Data Element Descriptor ................................................................................................. 22
  3.2.2 Bluetooth Profile ....................................................................................................................... 24
  3.2.3 ServiceSearchAttribute Transaction .......................................................................................... 25
    3.2.3.1 ServiceSearchAttribute Request PDU ............................................................................... 25
    3.2.3.2 ServiceSearchAttribute Response PDU ........................................................................... 26
3.3 Multimodal Interaction ..................................................................................................................... 26
  3.3.1 Human-Centred Technology ...................................................................................................... 26
  3.3.2 A Theoretical Framework for Multiple Modalities .................................................................... 27
    3.3.2.1 Sequential Combination ................................................................................................... 28
      3.3.2.1.1 Transfer Action .......................................................................................................... 28
      3.3.2.1.2 Specialisation Action ................................................................................................. 28
      3.3.2.1.3 Equivalence Action .................................................................................................. 28
    3.3.2.2 Coordinated-and-Simultaneous Combination .................................................................... 29
      3.3.2.2.1 Redundancy Action .................................................................................................. 29
      3.3.2.2.2 Complementary Action ............................................................................................. 30
    3.3.2.3 Uncoordinated-and-Simultaneous Combination ................................................................. 30
3.3.3 Classification of Multimodal Dialogue .......................................................................................... 30
  3.3.3.1 Auditory Modality ................................................................................................................ 30
  3.3.3.2 Visual Modality .................................................................................................................... 31
  3.3.3.3 Tangible Modality ................................................................................................................ 32
  3.3.3.4 Olfactory and Gustatory Modalities .................................................................................... 33
3.3.4 Research on Multi-Device Multimodality ..................................................................................... 33
3.4 Summary ......................................................................................................................................... 34
5.2.2 Three-Party Model applied in MID-B ................................................................. 70
5.3 Discovery in MID-B .............................................................................................................. 71
5.4 IP-Based Discovery .............................................................................................................. 73
  5.4.1 Implementation Stack of IP-Based Discovery ............................................................. 73
  5.4.2 Detailed Implementation Stack of IP-Based Discovery ............................................... 75
  5.4.3 Proof-Of-Concept .......................................................................................................... 79
    5.4.3.1 Service Discovery in the 'ue_c' Library ................................................................. 80
    5.4.3.2 Service Discovery in the 'ue_i' Library ................................................................. 86
    5.4.3.3 Service Discovery in the 'msb' Library ................................................................. 89
5.5 Bluetooth-Based Discovery ................................................................................................... 92
  5.5.1 Implementation Stack of Bluetooth-Based Discovery ................................................. 92
  5.5.2 Emulation........................................................................................................................ 93
5.6 Summary .............................................................................................................................. 100

Chapter 6 .......................................................................................................................................... 102
6 Interface and Modality Binding ............................................................................................... 102
  6.1 Introduction ........................................................................................................................ 102
  6.2 Modality Service Tree (MST) ............................................................................................. 103
    6.2.1 Theoretical Approach to the MST .............................................................................. 103
    6.2.2 Dependencies between MSTs ...................................................................................... 111
      6.2.2.1 Theoretical Concept of Logical Relations between UE_C and UE_I ........................ 111
      6.2.2.2 The Theoretical Concept of Cluster Relations between the UE_C And the UE_I ................................................................. 114
    6.2.3 MST Search Mechanism .............................................................................................. 116
    6.2.4 Implementation ........................................................................................................... 119
  6.3 Resource Handling ............................................................................................................... 123
    6.3.1 Interface-Equipment Table and Interface Equipment Handler .................................... 125
    6.3.2 Tables Of Input & Output Modalities and Modality Handler .................................... 125
  6.4 Interface and Modality Binding .................................................................................... 130
    6.4.1 Theoretical Approach to Interface and Modality Binding ......................................... 130
    6.4.2 Binding Mechanism ...................................................................................................... 133
      6.4.2.1 Sequence of Interface and Modality Binding ....................................................... 136
      6.4.2.2 Basic Algorithms for the Binding Mechanism ..................................................... 140
        6.4.2.2.1 Function ‘checkInterfaceEq()’ ............................................................... 144
        6.4.2.2.2 Funciton ‘consultToSelectAmbiguousIEs()’ .............................................. 146
        6.4.2.2.3 Function ‘getConnectivity()’ ................................................................... 147
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Taxonomy of human-computer interaction topic</td>
<td>5</td>
</tr>
<tr>
<td>1-2</td>
<td>Multimodality in mobile environments</td>
<td>5</td>
</tr>
<tr>
<td>2-1</td>
<td>Information flow for multimodal UI</td>
<td>10</td>
</tr>
<tr>
<td>2-2</td>
<td>Context awareness architecture</td>
<td>12</td>
</tr>
<tr>
<td>2-3</td>
<td>Interface adaptation</td>
<td>17</td>
</tr>
<tr>
<td>2-4</td>
<td>Two agents involved in interface adaptation</td>
<td>18</td>
</tr>
<tr>
<td>3-1</td>
<td>Hierarchical structure of the Bluetooth profiles</td>
<td>24</td>
</tr>
<tr>
<td>3-2</td>
<td>Generic structure of the SDP PDU</td>
<td>25</td>
</tr>
<tr>
<td>3-3</td>
<td>Grouping of modality combination</td>
<td>27</td>
</tr>
<tr>
<td>4-1</td>
<td>MID-B Overview</td>
<td>36</td>
</tr>
<tr>
<td>4-2</td>
<td>Idealised waterfall model</td>
<td>37</td>
</tr>
<tr>
<td>4-3</td>
<td>System level requirements</td>
<td>39</td>
</tr>
<tr>
<td>4-4</td>
<td>Functional level requirements</td>
<td>39</td>
</tr>
<tr>
<td>4-5</td>
<td>Hierarchy of requirements of the MID-B system</td>
<td>40</td>
</tr>
<tr>
<td>4-6</td>
<td>Basic transition-diagram of FSM</td>
<td>43</td>
</tr>
<tr>
<td>4-7</td>
<td>Deterministic FSM</td>
<td>44</td>
</tr>
<tr>
<td>4-8</td>
<td>Non-deterministic FSM</td>
<td>44</td>
</tr>
<tr>
<td>4-9</td>
<td>Set of terminals</td>
<td>45</td>
</tr>
<tr>
<td>4-10</td>
<td>Example showing how to use the special tags</td>
<td>47</td>
</tr>
<tr>
<td>4-11</td>
<td>Application profile</td>
<td>48</td>
</tr>
<tr>
<td>4-12</td>
<td>Device profile</td>
<td>49</td>
</tr>
<tr>
<td>4-13</td>
<td>End-user’s information</td>
<td>51</td>
</tr>
<tr>
<td>4-14</td>
<td>Learning for affected modalities</td>
<td>52</td>
</tr>
<tr>
<td>4-15</td>
<td>Content-router profile</td>
<td>53</td>
</tr>
<tr>
<td>4-16</td>
<td>Using the MSB for a feedback on the selected modality</td>
<td>54</td>
</tr>
<tr>
<td>4-17</td>
<td>Core functions of the MSB</td>
<td>54</td>
</tr>
<tr>
<td>4-18</td>
<td>Virtual-device environment</td>
<td>55</td>
</tr>
<tr>
<td>4-19</td>
<td>Logical prototype of the MID-B system</td>
<td>56</td>
</tr>
<tr>
<td>4-20</td>
<td>The two mechanisms of Interaction adaptivity: trading and binding</td>
<td>57</td>
</tr>
<tr>
<td>4-21</td>
<td>Sequence diagram of the MID-B prototype</td>
<td>58</td>
</tr>
<tr>
<td>4-22</td>
<td>FSM of the MID-B system</td>
<td>60</td>
</tr>
<tr>
<td>4-23</td>
<td>Discovery and binding phase and data transmission phase</td>
<td>61</td>
</tr>
<tr>
<td>4-24</td>
<td>FSM of the UE_C</td>
<td>63</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4-25</td>
<td>FSM of the UE_I</td>
<td>66</td>
</tr>
<tr>
<td>5-1</td>
<td>Three-party model</td>
<td>69</td>
</tr>
<tr>
<td>5-2</td>
<td>Registration process of the three-party model</td>
<td>69</td>
</tr>
<tr>
<td>5-3</td>
<td>General use case of the three-party model</td>
<td>70</td>
</tr>
<tr>
<td>5-4</td>
<td>Discovery, registration and modality-stream delivery in the three-party model</td>
<td>71</td>
</tr>
<tr>
<td>5-5</td>
<td>Phases of multimodal service delivering in the MID-B system</td>
<td>73</td>
</tr>
<tr>
<td>5-6</td>
<td>Implementation stack of the IP-based discovery</td>
<td>73</td>
</tr>
<tr>
<td>5-7</td>
<td>Comparing the TCP/IP suite and the implementation stack of the IP-based discovery</td>
<td>74</td>
</tr>
<tr>
<td>5-8</td>
<td>Extensive implementation stack of the IP-based discovery</td>
<td>76</td>
</tr>
<tr>
<td>5-9</td>
<td>Generic Modality Service Tree (MST)</td>
<td>78</td>
</tr>
<tr>
<td>5-10</td>
<td>Guideline for the implementation of the IP-based discovery</td>
<td>80</td>
</tr>
<tr>
<td>5-11</td>
<td>Class and packages in ‘ue_c’ library (scope on service discovery)</td>
<td>81</td>
</tr>
<tr>
<td>5-12</td>
<td>Packages in ‘discovery’ library</td>
<td>81</td>
</tr>
<tr>
<td>5-13</td>
<td>Classes in ‘multicast’ package</td>
<td>82</td>
</tr>
<tr>
<td>5-14</td>
<td>AcknowledgementAddress object</td>
<td>82</td>
</tr>
<tr>
<td>5-15</td>
<td>Packages in ‘serviceSearchAttributeTransaction’ library</td>
<td>83</td>
</tr>
<tr>
<td>5-16</td>
<td>Classed in ‘request’ package</td>
<td>84</td>
</tr>
<tr>
<td>5-17</td>
<td>Class in ‘dataElementDescriptor’ package</td>
<td>84</td>
</tr>
<tr>
<td>5-18</td>
<td>Classes in ‘response’ package</td>
<td>85</td>
</tr>
<tr>
<td>5-19</td>
<td>Class in ‘mstTransaction’ package</td>
<td>85</td>
</tr>
<tr>
<td>5-20</td>
<td>Class and packages in ‘ue_i’ library (scope on service discovery)</td>
<td>86</td>
</tr>
<tr>
<td>5-21</td>
<td>Classes and packages in ‘mid_bSetup’ library</td>
<td>87</td>
</tr>
<tr>
<td>5-22</td>
<td>Classes in ‘requestReceiver’ package</td>
<td>88</td>
</tr>
<tr>
<td>5-23</td>
<td>Classes in ‘response’ package</td>
<td>88</td>
</tr>
<tr>
<td>5-24</td>
<td>Classes in ‘dataElement’ package</td>
<td>89</td>
</tr>
<tr>
<td>5-25</td>
<td>Class in ‘mstUtil’ package</td>
<td>89</td>
</tr>
<tr>
<td>5-26</td>
<td>Class diagram of device profile and other associated classes</td>
<td>90</td>
</tr>
<tr>
<td>5-27</td>
<td>Device handler</td>
<td>91</td>
</tr>
<tr>
<td>5-28</td>
<td>Implementation stack of Bluetooth-based discovery</td>
<td>92</td>
</tr>
<tr>
<td>5-29</td>
<td>Class diagram for UE_1</td>
<td>93</td>
</tr>
<tr>
<td>5-30</td>
<td>Sequence diagram of the UE_1 routines</td>
<td>94</td>
</tr>
<tr>
<td>5-31</td>
<td>Class diagram for UE_C implementation</td>
<td>95</td>
</tr>
<tr>
<td>5-32</td>
<td>ServiceDiscoveryList</td>
<td>95</td>
</tr>
<tr>
<td>5-33</td>
<td>Sequence diagram of the UE_C routines</td>
<td>97</td>
</tr>
<tr>
<td>5-34</td>
<td>(a) – (h): Emulation of the Bluetooth-based discovery</td>
<td>100</td>
</tr>
<tr>
<td>6-1</td>
<td>Structure of the MST</td>
<td>106</td>
</tr>
</tbody>
</table>
List of Figures

Figure 6-2: MST representing a single interface equipment ........................................................... 107
Figure 6-3: MST representing a composite device ........................................................................ 107
Figure 6-4: Top-down search mechanism ..................................................................................... 117
Figure 6-5: Bottom-up search mechanism .................................................................................... 118
Figure 6-6: Libraries that execute MST and Dependencies between Different MSTs .......... 119
Figure 6-7: Classes and package that are implemented in ‘structure’ library ............................... 120
Figure 6-8: Classes implemented in ‘relationBetweenLevel’ package ......................................... 120
Figure 6-9: Packages in ‘dependency’ library ................................................................................ 121
Figure 6-10: Classes implemented in ‘logical’ package ................................................................. 122
Figure 6-11: Classes implemented in ‘cluster’ package ................................................................ 123
Figure 6-12: MST forest ................................................................................................................ 123
Figure 6-13: Interface-equipment table and interface equipment handler .................................... 125
Figure 6-14: Class defining an entry of input and output modality tables ................................... 125
Figure 6-15: Modality handler ....................................................................................................... 126
Figure 6-16: Internal mechanism of ‘modifyNegativeModality’ function ................................. 127
Figure 6-17: Internal mechanism of ‘operateRelevantIEQ’ function ............................................ 129
Figure 6-18: Sequence to examine user physical challenges ......................................................... 130
Figure 6-19: Exploiting the binding function ................................................................................ 133
Figure 6-20: Three components for interface and modality binding ............................................ 134
Figure 6-21: Dependencies and associations between classes of the three components .......... 134
Figure 6-22: Result delivered from the binding mechanism ......................................................... 135
Figure 6-23: Sequence of interface and modality binding ............................................................ 137
Figure 6-24: Phase to initiate the binding mechanism ................................................................. 137
Figure 6-25: Phase to operate a binding mechanism ................................................................. 139
Figure 6-26: Top-level flowchart: binding mechanism ................................................................. 141
Figure 6-27: Sequence diagram: selecting an appropriate modality ............................................ 143
Figure 6-28: The function ‘checklnterfaceEqO’ ............................................................................ 145
Figure 6-29: The function consultToSelectAmbiguousIEs() ....................................................... 147
Figure 6-30: Connectivity information ......................................................................................... 147
Figure 6-31: Function getConnectivity (1) .................................................................................. 150
Figure 6-32: Function getConnectivity (2) .................................................................................. 151
Figure 6-33: Function getConnectivity (3) .................................................................................. 152
Figure 6-34: Function getConnectivity (4) .................................................................................. 153
Figure 7-1: The contributions of the thesis over four functionalities ............................................ 157
Figure A2-1: SDP ServiceSearchAttribute request (1) ............................................................... 178
Figure A2-2: SDP ServiceSearchAttribute response (1) ............................................................. 180
Figure A2-3: SDP ServiceSearchAttribute request (2) ............................................................... 182
Figure A2-4: SDP ServiceSearchAttribute response (2) .................................................................182
Figure A.3-1: The MST representing smart phone ........................................................................183
Figure A3-2: The MST representing a laptop .................................................................................184
Figure A4-1: MST representing a printer .........................................................................................186
Figure A5-1: MST representing a desktop ......................................................................................187
List of Tables

Table 3-1: Type descriptor .................................................................................................................23
Table 3-2: Size descriptor ...................................................................................................................23
Table 4-1: State description of the MID-B’s FSM ............................................................................62
Table 6-1: Information how to reroute a data content stream to an UE_1 ......................................149
# Glossary of Terms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHR</td>
<td>Automatic Handwriting Recognition</td>
</tr>
<tr>
<td>ASR</td>
<td>Automatic Speech Recognition</td>
</tr>
<tr>
<td>CC/PP</td>
<td>Composite Capability/Preferences Profile</td>
</tr>
<tr>
<td>CDC</td>
<td>Connected Device Configuration</td>
</tr>
<tr>
<td>CLDC</td>
<td>Connected Limited Device Configuration</td>
</tr>
<tr>
<td>FSM</td>
<td>Finite State Machine</td>
</tr>
<tr>
<td>GAP</td>
<td>Generic Access Profile</td>
</tr>
<tr>
<td>GOEP</td>
<td>Generic Object Exchange Profile</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphic User Interface</td>
</tr>
<tr>
<td>HCC</td>
<td>Human-Centred Computing</td>
</tr>
<tr>
<td>HCI</td>
<td>Human Computer Interaction</td>
</tr>
<tr>
<td>HCS</td>
<td>Human-Centred System</td>
</tr>
<tr>
<td>ISO</td>
<td>International Organisation for Standardisation</td>
</tr>
<tr>
<td>JSR</td>
<td>Java Specification Request</td>
</tr>
<tr>
<td>MIDP</td>
<td>Mobile Information Device Profile</td>
</tr>
<tr>
<td>OBEX</td>
<td>OBject EXchange</td>
</tr>
<tr>
<td>PDA</td>
<td>Personal Digital Assistant</td>
</tr>
<tr>
<td>PDU</td>
<td>Protocol Data Unit</td>
</tr>
<tr>
<td>RMI</td>
<td>Remote Method Innovation</td>
</tr>
<tr>
<td>SDAP</td>
<td>Service Discovery Application Profile</td>
</tr>
<tr>
<td>SDP</td>
<td>Service Discovery Protocol</td>
</tr>
<tr>
<td>SPP</td>
<td>Serial Port Profile</td>
</tr>
<tr>
<td>TCD</td>
<td>Technology-Centred Design</td>
</tr>
<tr>
<td>TTS</td>
<td>Text-To-Speech</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>TUI</td>
<td>Tangible User Interface</td>
</tr>
<tr>
<td>UAN</td>
<td>User Action Notation</td>
</tr>
<tr>
<td>UAProl</td>
<td>User Agent Profile</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>UUID</td>
<td>Universally Unique Identifier</td>
</tr>
</tbody>
</table>
Chapter 1

1 Introduction

1.1 Overview

With every new generation of mobile terminals, be it cell phones, handheld computers, wristwatch PDAs, or gaming consoles, the range of features constantly increases. This needs towards the actual technical capabilities, such as processing power, operating system functionality and internal memory. The types of mobile applications and the styles how content is presented become increasingly diverse. Furthermore, multimodal input (using pointing and keypad as input modalities) has been available for some time [1]. However, the user interface-capabilities of mobile devices did remain restricted to the small displays and limited audio input/output capabilities and, in some cases, pointing devices (such as a stylus on a touch screen). On the other hand, the environment in which mobile communication usually takes place becomes ever more crowded with devices offering a wide range of user interfaces and modalities. Such surrounding devices may include interactive screens, loud speakers or desktop peripheral devices, such as keyboard, mouse or joystick. Assuming a mobile user entering the physical range of interface devices that host modalities (display/ audio systems, etc), these ambient devices could be ‘bound’ into an overall to dynamically form a multimodal user interface. Following such a concept, whenever suitable devices are available, the user interface can be tailored to the needs of the user as well as to the requirements of the application. Actual user interaction can be performed through both the user’s personal mobile device and through other public (surrounding) devices. This shared use of modalities will on one hand enhance the user experience and add more possibilities for multimodal user interaction, on the other hand it raises a number of challenges. One of the aims of this thesis is to provide an analysis of the implications that such dynamic multimodal systems may cause. The work also discusses the means and mechanisms required for self-adaptive multimodal interaction for mobile applications. For use in this thesis the terms self-adaptive, multimodal, interaction, mobile application and distributed are defined as follows:

- Self-adaptive: Adaptive as used in this thesis refers to the dynamic adjustment of an application’s behaviour. Also, adaptive means the automatic configuration of the existing modality services, in accordance with the available physical resources and the user environment. Adaptation is a consequence of changes in the composition of the
environment (i.e. changes in the availability of devices and user interfaces). Self-adaptation means that any changes should be transparent to the use. Making user interfaces self-adaptable increases the user-friendliness and usability of mobile terminals and applications.

- Multimodal: ‘Modal’ combines the concepts of modality and mode. Nigay and Coutaz [2] defined that a ‘modality’ is as a type of communication channel by which data is exchanged. The ‘mode’ characterises how information able to be interpreted in order to extract or to convey the meaning of content. ‘Multimodality’ indicates that potentially more than one channel and modes are used to convey content. In general, multimodality and multimedia relate to multi-dimensional sensory content exchanged. In addition, a multimodal system integrates input/output information to different human senses, mostly visual and audible multimodality, and capitalises on the interdependences between the human senses and uses them to present information in the most efficient and natural way [3].

- Interaction: Human–computer interaction (HCI) is the implementation of interactive computing systems for human use. During the evolution of HCI, the focus of the user-interface (UI) design extended from the notion of ‘interface’ into ‘interaction revision’ [4]. While ‘interface’ means the set of methods that can be used by a service, ‘interaction revision’ refers to the procedure that determines which interface technology fits in with the human needs, rather than how the interface looks like. The prevailing HCI introduces multimodal interaction; the basic idea is motivated by natural human communication that mostly makes use of the different modalities available.

- Mobile application: Mobile applications are software components hosted and often executed on portable devices. Dissimilar to the realm of stationary devices and applications, the mobile application development must consider the particular capabilities, compatibilities and interface features of mobile devices. According to findings of experiment on the mobile user experiences [5], the five main factors influencing mobile application development are the user himself (skill, emotion and etc.), the product (aesthetic characteristic, mobility and etc.), the context of use (time, accompanying person and etc.), social factors (time pressure and etc.) and cultural factors (sex, language and etc.).

- Distributed: The term ‘distributed’ incorporates, in the context of work, a number of modality services available within the wireless environment. The distributed environment as specified in this thesis is characterised by simultaneous interactions of a heterogeneous input/output modality services located in different physical interface devices.
Chapter 1. Introduction

1.2 Motivation and Objective

Historically, multimodality was mainly a computer science exercise, this was included in areas as diverse as cognitive science, human-computer interaction (HCI), interactive learning environments and software engineering [6-9]. In principle, multimodal interaction was developed from conventional interface use. It was constructed of multiple parallel input/output streams available with semantic information. At first, multimodal developments aimed at automatic speech recognition (ASR) and automatic handwriting recognition (AHR), allowing spoken commands and hand-writing to be used as alternative inputs. This removed the restriction to standard keyboards and mouse or touch-pad interface. Integrating vision-based technologies, such as, facial expression, manual gesturing, and interpretation of gaze, the development moved towards more sophisticated means for recognition of human’s activities. With a combination of human activity recognitions and the standard means of input/output interfaces, more robust multimodal interaction was achieved. Further literature on the evolution of recognition technologies is summarised in [10]. In general, early multimodal developments rather presented multimodal interactive services mainly designed for desktop applications. Taking advantages of the available features and capabilities ensures the access to the service and enhanced rendering of application content for an end-user. However for mobile environments, the development must consider the restrictions in term of 1) physical features of mobile terminals and 2) the continuously changing conditions and any surrounding noise effecting stable qualities of interactive services [11, 12]. To date, most research aimed towards applying the principles of (conventional) multimodal interfaces in current ‘hardwired’ single devices to the implementation in less powerful portable devices. Detailed understanding of dynamic adaptability and context-awareness in mobile environment facilitates the use of multimodal interaction in such cases [13-15]. A presence-aware application, which liberates the user from manually updating presence status and making decision for communication means, must be extended into mobile scenarios. For example, the Mobile Presence Agent (MPA) of the Presence-Enabled Mobile Service (PEMS) [16] has been proposed as a mechanism to monitor the information on device status and user presence. The MPA integrates information within collaborative environments via Web services and SMS for synthesis of a user’s availability. In short, the multimodal interaction was originally designed for stationary terminals with rather powerful capabilities; consequently, for mobile scenario, adaptivity and context-awareness also need to be facilitated.

Adaptivity of multimodal interactions for mobile environments can overcome some of the constraints of mobile device’s physical properties. However, there are some limitations remaining. Research into the multimodal interfaces and context-awareness technologies identified the lack of suitable physical access/render media on a mobile device [17]. The work documented in this thesis allows the utilisation of external interface-devices, which dynamically appear in the
service vicinity. Due to mobility, the idea incorporates, dependent on availability, dynamic extension of the range of interface-devices. Dynamic extension allows arbitrary change of the available modalities, which may influence the overall multimodal interactions. Although, similar or same modality types may be hosted on different devices, the main idea is to exploit complementary interface capabilities, rather than redundancy features. Abilities of the similar modalities may vary from one device to another, and one may fit better for different application’s content characteristics. The mechanisms investigated in this thesis allow a user to communicate (interact) via miscellaneous modalities that are offered by a personal mobile terminal and extended through external interfaces that are dynamically bound into the users system. In addition, the thesis introduces mechanisms to achieve self-adaptive interfaces in mobile environments by monitoring the availability of interface-devices that may affect the system deployment. The thesis makes the internal behaviour of multimodal interfaces to adapt themselves to either the following conditions:

- Mobility effects may lead to changes in the users (network) environment, while the current interface configuration should be maintained. The system needs to buffer the incoming data streams, and then alters the existing application by amending the implementation of data content delivered to the most suitable interface.

- Due to the changes in the accessibility of user interfaces, devices may change or connections between devices may become corrupted. A re-allocation scheme is required to adapt to the changes in the system. The scheme re-allocates modalities according to the new physical resources and manages the system behaviour.

1.3 Contributions and major Achievements

An architecture that enables the dynamic shaping of the human-computer interface for distributed networked (interface) devices in mobile communications is the core achievement presented in this thesis. The framework developed is called the Multi Interface-Devices Binding system (MID-B) of which the structural design extends the concept of multimodality into the mobile environment. MID-B allows dynamic use of available interface devices (e.g. display, sound system, etc.). Following the taxonomy of the HCI topic introduced by [18], the subtopic where the MID-B system applies is 'Requirements gathering analysis.'
Chapter 1. Introduction

The thesis summarises four added functionalities to accomplish the implementation of a (multimodal) user-interface for mobile and distributed environment.

1. Detection of user’s environment context (i.e. mobility awareness, situation awareness) and maintenance of a level of quality of services in scenarios with arbitrary context changes.

2. Device and modality discovery; discovering types and capabilities of external modalities.

3. Device and interface binding mechanism; amalgamating the selected devices and modalities into one overall user interface as well as reacting, in real time, to changes of connectivity among the available devices.

4. User content adaptation, transparently transforming the format of the data content from one modality to another.

Figure 1-1: Taxonomy of human-computer interaction

Figure 1-2: Multimodality in mobile environments
Mechanisms have been investigated and new functions are developed to facilitate these four major functionalities.

- **Extension to Bluetooth SDP.** The functionality of Bluetooth SDP transaction already provides some basic service descriptions. However, it lacks the detailed descriptions of modality services and their characteristics. This work proposes some extension, called Multimodality (MM) transaction. The MM transaction is added to provide information of modality types, characteristics, dependencies and device connectivity. It is implemented based on two protocol scenario: a) pure Bluetooth stack and b) an IP network connection, such as WLAN.

- **Multimodal Service Base (MSB).** While the Bluetooth SDP implements its service discovery following a two-party model (request/response architecture), the proposed extension introduces a third software-based party, named “Multimodal Service Base” (MSB). The MSB provides basic management of modalities. It hosts a registry for the available device-and-modality characteristics. In the SDP, every time a client needs services from a server, it has to send a request to the server. In contrary to the SDP, the MSB maintain the service information in the registry; this reduces traffic occurring at request/response transaction.

- **Modality Service Tree (MST).** The generic MST is the unifying database that describes the relations between two service levels. Device information is captured in a hierarchical structure that clearly explains the multi-layered details of services. The structure is defined as a rooted tree in which a special node is singled out describing a “user interface device”. The children along the root node have a designated order that exhibits different services categories.

- **Service search algorithm.** In general, the MST’s nodes are elements of service types. The linking between different levels of nodes forms the dependency of a pair of specific services. The service search algorithm is an object-oriented approach employing dependency information to explore the features within a node holding the target service.

- **Modality relationships.** The relationship states the semantic relation between modalities offered by two different devices. In the MID-B system, the MSB maintains the state information of relationship between a user’s mobile device and discovered public devices.

- **Binding theory.** Not only aim at modality integration, the binding theory developed distinguishes types of binding as a) binding for modality regardless of capabilities, b) binding of interface equipment, and c) binding of modality considering its capabilities and
d) binding of user device. Individual binding types are operated at different stages of the MID-B system.

1.4 Structure of Thesis

This thesis is organised as follows.

Chapter 2 describes the MID-B system. The system extends the conventional multimodal interface scenario into mobile environments. The chapter defines the scope of the work and states how the system exploits the existing contributions multimodality functions. Further, the specifications associated with the four main functionalities (shown in Figure 1-2) are provided. A discussion of related works is also included.

Chapter 3 presents how MID-B extends the state of the art. The background information used for the actual implementation is provided. Since part of MID-B includes the extension of the Bluetooth SDP, the chapter describes the state of art on service functions in Bluetooth, the Bluetooth profiles, and the ServiceSearchAttribute transaction\(^1\). In addition, the chapter presents an overview of the multimodal interaction, covering the human-centred technologies, the theoretical framework of mobility combinations, the multimodal dialogue classification, and a review of works enabling multimodal multi-device operations.

Chapter 4 describes the system requirements of the MID-B framework. MID-B at the high level of abstraction and the architecture’s components are discussed. The states of the MID-B system are illustrated in a Finite State Machine (FSM). In addition, the interactions between different components are explained.

Chapter 5 discusses the novel service discovery mechanisms enabling dynamic modality discovery. The mechanisms extend the functionality of the SDP transaction by introducing a three-party model and creating the Multimodality transaction. The mechanism is developed based on an IP stack as well as for Bluetooth. The chapter describes both mechanisms.

Chapter 6 proposes the mechanism for interface and modality binding. The mechanism appears in the Modality Service Base (MSB). In the chapter, the device context, which is captured as Modality Service Tree (MST), is presented. Furthermore, the service search algorithm is discussed and the theoretical approach to express the dependencies between different MST is included. The binding function is as well introduced; it shows how the MST is exploited for

---

\(^1\) This is one kind of SDP transactions. The other two are the ServiceSearch transaction and the AttributeSearch transaction.
multimodal multi-device interface adaptation. At the end of the chapter, the basic algorithm of the binding function is presented.

Chapter 7 provides conclusions of the main achievements and lists areas where future work could be based on to further extend the MID-B system.
Chapter 2

2 Problems and Scope of this Research

2.1 Introduction

The thesis presents mechanisms for self-adaptive configuration of multimodal interactions for mobile applications in distributed environments. Such configuration involves several aspects including both traditional multimodal interfaces development and ubiquitous networks. The related research area is very broad; this chapter identifies the specific topics that have to be considered for the work reported in this thesis. Rather than describing traditional multimodal interface definition, the intension of this thesis is to deliver a new methodology bringing multimodality, together with the topics within the area of ubiquitous networking. In this chapter, existing approaches dealing with both topics are briefly described. The advantages, disadvantages and problems caused by the inadequate design methods of the existing approaches are discussed. The suggested solution and the explicit scope of the work are put into the concept of current research work in the related areas.

2.2 Traditional Multimodal Interfaces

2.2.1 Developing Traditional Multimodal Interfaces

Unlike a classical user interface (UI), multimodal UI takes advantage of the features of several modalities to enhance human-computer communication. The communication takes place in more natural way. In general, a multimodal UI facilitates an advanced bi-directional interaction between a human and machine(s). As shown in Figure 2-1, the communication takes place either from output devices aimed at human sensory organs (i.e. towards human perception) or from human output channels aimed at input devices (i.e. towards machine perception). Normally, an input device hosts a modality linked to a human sense [19]. For example, these devices could be camera (vision/sight), microphone (audio/hearing), haptic sensor (haptic/touch), olfactory sensor (olfactory/smell), and gustatory sensor (gustatory/taste). Besides, there are still some devices that can be mapped to the human sensory perceptions only indirectly. This includes keyboard, mouse, writing tablet, and etc. [19].
The methodology to develop the multimodal UI includes four steps [20]. The first step is to identify the available means of recognition, for example Automatic Speech Recognition (ASR) and Automatic Hand-writing Recognition (AHR). Subsequently, the second step is to identify a reference object. Martin [2] defines the reference object as a concrete target of interest, for instance ‘building’, ‘car’, ‘desk’ or ‘table’. The multimodal UI lets the same object be concurrently referred to by several recognition methods, such as speech ‘and’ gesture. Then, the third step is to specify the processing functions for each recognition method. Different recognition methods require different internal processing. For instance, the speech recognition needs the processes of the language modelling and acoustic modelling [21]. Finally, the last step is to specify the types of cooperation between modalities derived from the available recognition methods. ‘Cooperation’ is the function of how to integrate information. Six types of modality co-operations have been proposed as following [22]:

- ‘transfer’, information created by one modality is used as an input by another,
- ‘equivalence’, information can be alternatively generated by either of modalities,
- ‘specialisation’, a specific information is always attained by the same modality,
- ‘redundancy’, the same information can be produced by several modalities,
- ‘complementarity’, different information from diverse modalities has to be merged.
Chapter 2. Problems and Scope of this Research

• 'concurrency', independent information is simultaneously generated by assorted modalities, but merging of this information is not required.

In addition to allowing concurrent use of different modalities, the multimodal UI facilitates automatic personalisation. This allows interaction be customised and adapted, according to a particular user's personal features, physical challenges and skills. Therefore, user identification becomes a necessity. This can be achieved through several approaches, such as speaker identifier or biometric characteristic [23]. Two commonly used models of the user identification functions are the 'open-set' and the 'closed-set' [24]. In the open-set model, any unknown person will fail authorisation to access an application. In contrast, a reject scenario is not defined for the closed-set model. Thus, an application based on the closed-set model gives access rights to an unregistered person, and any interaction executes regardless of a personal profile.

2.2.2 Scope of Traditional Multimodal Interface Development

This work does not intend to introduce another implementation of a traditional multimodal interface. Yet, it is based on the assumptions made in other related research projects. In term of the user-interface design, this includes [25-30]. Regarding user identification, [24] and [31-34] are reviewed. The work presented here considers open-set user identification. That is, a multimodal UI is carried out 'with' applying a user profile. By using the user profile, information about user physical abilities are represented as a set of 'fixed-value contexts'. This is then combined with a collection of 'unfixed-value contexts', which corresponds to current environment constraints. The combination of these two context types promises to facilitate multimodal UIs that meet a particular user's requirements. The further details on the context information are described in the following section.

2.3 Extending Multimodal Interaction into Ubiquitous Environments: Additional Challenges

The ubiquitous environment is characterised by its intrinsic service distribution and its dynamic system behaviour. It requires mechanism capable to handle erratically changing resources [35-38]. The MIRS (Multimodal Interaction and Rendering System) [39] suggests three involved subjects to be available to implement a multimodal UI in the ubiquitous environment. These are 'device and modality selection', 'UI description', and 'UI transformation'. However, the suggestion from [39] considers only rendering UI on a single device. It cannot be directly applies in a virtual-device environment where a number of modalities from different devices are logically integrated.
This thesis assumes that for the development of the multimodality in the ubiquitous environment, four aspects ought to be considered. These are:

- a context-awareness, through sensing a service vicinity,
- service discovery; gathering a set of modality services provided by within the service vicinity,
- user-interface adaptation; dynamically binding of distributed modalities into an overall system, then locating the selected modality that matches the desired capabilities, and
- presentation adaptation and connectivity management; automatically transforming the data format to the potential modality computed by the user-interface adaptation. It also includes the functions to choose a communication link between currently-used and prospective interface devices.

2.3.1 Context Awareness

2.3.1.1 Context Awareness Challenges

As defined in [40], context is “the set of environmental states and settings that either determines an application’s behaviour or in which an application event occurs and is interesting to the user”. Generally, context may influence on applications performance. Two main classes of context features determine how the context influences the execution of an application [40, 41]. The first class defines ‘internal’ (or active) context. The internal context is placed inside the application’s domain and strongly influences the application behaviour. The second class characterises an ‘external’ (passive) context that contains associated, but not significant, information. A framework accommodating abilities of sensing the contexts is called a ‘context-awareness’ framework, which is structured in two distinct systems, i.e. ‘ informational’ and ‘operational’ systems [42], see Figure 2-2. Data within the informational system is described in a basic notation without using any expressive scheme. The captured raw data is then forwarded to
Chapter 2. Problems and Scope of this Research

other part at the operational system for further manipulation and modelling. As introduced in the
Figure 2-2, the context data is modelled as user, platform, and environment model. Applying this
on mobile applications, particularly on their interaction tasks, raises significant requirements
forwarded. In literatures [41, 43-48] advocate such a conceptual context modelling for ubiquitous
environment. Context modelling results in meaningful structures that encapsulate all relevant
attributes and constraints of the actual situation. Also, it helps to identify any correlations between
the meaningful structures. As shown in the Figure 2-2, the structure of context is generally formed
in the three aforementioned models [49, 50]. The user model is “a knowledge source containing
explicit assumptions about all aspects of the user that may be relevant to the dialogue behaviour
of the system” [51]. Vrieze [52] presents the evidence how user models impact personalised
interfaces. Further, the platform model contains a specification of software and hardware
properties. For instance, Julien [48] differentiates the specification view of the device constrains
into four main types, i.e. network constraints (connectivity abilities), host constraints (logical
properties, such as the device’s ID and general offered service types), agent constraints (particular
details of the specific services on the host) and data constraints (restrictions of individual content
item). Finally, the environment model represents information about the surroundings that could
have an effect on the user-application interactions.

2.3.1.2 Context Awareness in MID-B

The MID-B framework is constructed as a context-sensible system equipped with capabilities to
capture information on the availability and accessibility of modalities. In this thesis, the domain of
the internal (active) context is characterised by the usage of a ‘platform’ model and a ‘user’
model. The environment context is grouped into the category of external-context. Some example
of environment context are given in [53, 54].

The platform model plays the role of the ‘primary’ internal context for MID-B services. In the
sense, a user device, for example, a printer, a mobile terminal and a PC, are considered as one
platform. This thesis examines several approaches to express the device and to describe its
features. To describe an input interface device, the contributions from [55, 56] classify five
desired attributes. These attributes include a list of modalities, expression, roles, bounded or
infinite, and relative or absolutes. The ‘modality’ identifies the explicit mode of interaction.
Different types of modality will be explained in Chapter 3. ‘Expression’ describes the details of
the modality’s operations and manipulations. ‘Roles’ are the semantic descriptions of functions
capturing the input device controls an interested object. The next attribute is either ‘bounded’ or
‘infinite’. This attribute indicates whether a service provided by the input modality is restricted or
not. The last attribute indicates whether the modality deals with a relative input-value or an
absolute input-value. The relative input-value needs another input-value to be combined. Absolute
input-value gives a significant by itself. However, the device description suggested by \[55, 56\] has shortcomings. Firstly, it excludes detailed information of output interface devices. And secondly, it does not deliver an unambiguous description of the relationship between dynamic modality services on multiple devices.

The registry data model described in \[57\] is more advanced. It provides a service-oriented component-based architecture to flexibly configure interfaces. Thereby, components are distinguished as "device" and "service". Supporting frequent changes, newly discovered components can be added to the registry database at any time. The database is defined as a hierarchical ontology to express the service available within a device. It maintains and updates a list of available services and devices. Despite supporting frequent changes, there are some problems; the link between different levels within the hierarchical database only states a 'subset-of' of relations between devices and services. Information on device and service capabilities is not yet taken into account.

UAProf (WAP User Agent Profile) \[58\], introduced by the WAP Forum, is a domain-specific ontology capturing classes of device capabilities and preference information. It is built atop the Composite Capability/Preferences Profile (CC/PP) \[59\]. UAProf describes a set of components considering hardware platform, software platform, network characteristics, user agent, WAP characteristics and Push characteristics. The IST SIMPLICITY project \[60\] manipulates the UAProf and the CC/PP for use in a context-aware in mobile communication system. Although UAProf provides a description of the device's technical-capabilities and device-independent options for web access \[61, 62\], it presents an incomplete modality expression and restricted features for interface adaptation \[63\]. As a result, standalone UAProf will not be able to maintain the context description used for the dynamic interface and modality binding system. For MID-B, however, a device model must be expressive enough to provide the details of modality capabilities in a dynamically changing environment. And, it must facilitate a rapid and efficient mechanism for modality search. Hence, this thesis introduces an extensive device model, which includes selected UAProf's components and a range of additional new components. The thesis augurs that device descriptions should be formulated in a hierarchical structure as this enables the overall detail of devices to be visualised in a rather simple management. Unlike the registry data model in \[57\], the descriptions of the here proposed hierarchy are rather detailed. The model provides the information of service capabilities and limitations as well as the means to mathematically express the link between service levels. In addition, the connection between levels is directional; this indicates the flow and control direction (i.e. input/output modalities).

Acting as secondary internal context, the user context description includes personal data as well as social information \[64\]. Accordingly, the user model for MID-B comprises the user profile and the user's social relation. The user profile maintains the personal information, such as education,
physical abilities, and etc.; whereas the user’s social relation maintains various kinds of (social) connections between the user and others. With the assistance of the user model, the content and service presentation can be tailored to a particular user considering different levels of expertise and social links.

2.3.2 Service Discovery

2.3.2.1 Service Discovery Challenges

The Sun’s Jini technology is a service centric mechanism [65] that collects the functions of distributed services for the execution of a particular task. Jini is composed of three sub-protocols, i.e. a ‘Discovery’, a ‘Join’ and a ‘Lookup’. It also contains and maintains a lookup table that keeps a service object together with its attributes [66]. Once a device is plugged in, Discovery and Join protocol pair is executed to include the different kinds of the distributed services into the Jini network. Jini’s Discovery protocol looks for a service matching a set of defined requirements. Then, the discovered service must create a service object and its attributes must register with the lookup table. This is carried out by the Join protocol. The Lookup protocol, providing the Jini lookup service within the Jini network, will be responsible to maintain the tracking of services. The Jini network is limited by its software dependency. Its implementation is based on Java and its protocols rely on Remote Method Innovation (RMI) [67] to facilitate the interactions between service provider and lookup service. The traditional Jini methods seem to be not suitable in mobile computing. Particularly as the J2ME (Java 2 Micro Edition for Mobile Solution) does not foresee RMI as part of its configurations (CDC/CLDC) or its profiles (MIDP, etc). Although there have been attempts to extend Jini’s capabilities into wireless computing, the size of Jini is too large for execution on mobile terminals with their memory and power restrictions [68].

Unlike Jini, the Bluetooth Service Discovery Protocol (SDP) is hardware dependent and does not require any supplement protocol. The SDP, commonly appearing in mobile applications, provides a discovery solution for wireless ad-hoc networks. It enables a device (“client”) to discover the existence of services and attributes provided by other Bluetooth-enabled devices (“servers”). Bluetooth SDP uses a request/response-based transaction where by any transaction consists of a request and an associated response PDUs (Protocol Data Unit). The Bluetooth specification [69] classifies this mechanism as ‘ServiceSearch’ and ‘ServiceAttribute’ transactions. ServiceSearch and ServiceAttribute transactions are usually coupled. ServiceSearch first locates the specific service records corresponding to a service search pattern, and then the ServiceAttribute retrieves the attribute values from the specific service records. The two transactions can be combined into a single one, i.e. the ServiceSearchAttribute transaction. The requested attributes are then retrieved in one single transaction, reducing the total singling traffic.
Any Bluetooth device contains the service records of all available devices; this records consist of a list of all service attributes, including the Bluetooth profiles. Bluetooth profiles facilitate data exchanged between devices, via a specified communication channel (such as LAN, serial port, and etc) [70]. Although the SDP service record provides some service attributes' information, it lacks records on modality services and capabilities. This sets one of the challenges in this thesis, i.e. how to discover all information desired and required to facilitate dynamic modality adaptation.

2.3.2.2 Service Discovery in MID-B

Jini and Bluetooth SDP are capable to announce and discover services and devices within dynamically changing and distributed networks. However, their service descriptions do not include all information needed for an adaptive interface-device binding mechanism. There are following important deficiencies:

- Inadequate details about characteristics of available modalities in discovered devices.
- Insufficient information regarding the relation between different modalities that may be available within a discovered device.
- No information provided about dependencies between discovered devices.

Based on the analysis of the existing approaches to service discovery, the thesis proposes a mechanism that integrates the benefits of current service discovery technologies with a new multimodal interface discovery function. As shown in the previous section, Sun's Jini technology is not suitable for small and restricted terminals. The mechanisms developed for multimodal interface discovery therefore concentrate on an extension to the Bluetooth protocols. Bluetooth is already widely used in mobile terminals and it is relatively stable in ad-hoc networks. The thesis introduces mechanisms that extend the standard ServiceSearchAttribute transaction of the SDP Bluetooth protocol by appending a Multimodality transaction (MM transaction). The MM transaction also uses the request/response part and it itemises the multimodal services available, i.e. modality/ies type, characteristics, dependencies, and device connectivity features.

2.3.3 Interface Adaptation

2.3.3.1 Interface Adaptation Challenges

Interface adaptation is a capability to dynamically construct a Human-Computer Interaction (HCI) by utilising different interfaces and modalities, either separately or combined, depending on availability. Early approaches of interface adaptation included the LOTOS notation, launched by the International Organisation for Standardisation (ISO), and the UAN (User Action Notation), introduced by the Virginia Polytechnic Institute [71]. The LOTOS notation expresses the entire
user interaction as a set of small interaction processes. Each process has temporal relations, representing a collection of external observable actions [72]. UAN is a task and user-oriented notation capturing the collaborations of the user and a sequence of interactions [73]. Despite the features of these early approaches, they merely introduce a chronological method invoked by the service, but only functioning with single device. There is no attention paid to difference in device characteristics and personalised interfaces.

In contrast to the early mechanisms, modern interface adaptations offer the additional features to transparently access the frequently changing interfaces and modalities. This is due to a generic middleware that abstracts the interfaces and hides the inherent complexity caused by frequent interface changes, see Figure 2-3. In response to the changes, the middleware dynamically adjusts and reformats to an appropriate configuration and maintains the service delivery at the level satisfying the user’s preferences. Interface adaptation relatively correlate with the context models, as explained in [74-78]. The behaviour of the adapted UI is a result of the interpretation of the context information. Figure 2-4 shows two core agents that are involved, i.e. a user agent and an interface agent [79]. In the user agent, the initial user model facilitates personalised interaction, while the interface agent handles diverse interface components that are used to fulfil the context requirements of an active application.

**Figure 2-3: Interface adaptation**
Chapter 2. Problems and Scope of this Research

2.3.3.2 Interface Adaptation in MID-B

Currently, there is no standard for interface adaptation and the way to implement such adaptation varies for each framework proposed. Design of an interface adaptation is coordinated by the framework, based on the context model mentioned in Section 2.3.1. The context model characterises coherent structures of the context information, and the adaptation takes these structures to implement the UI configuration. According to the scope of context awareness, the adaptation mechanism proposed in this thesis is built on the implementation of the user and the platform (or device) model, yet it also must be aware of the environment context.

2.3.4 Presentation Adaptation and Connectivity Management

2.3.4.1 Presentation Adaptation and Connectivity Management Challenges

Adaptable user interfaces make the content rendering of applications dynamic. In dynamic multi-device scenarios, the range of physical interface equipments often changes. Such a case, a system needs two additional support mechanisms. These are the presentation adaptation and connectivity management. Presentation adaptation (PA) is responsible to transform the source UI’s format to any other specialised format. PA maintains the generic user-interface (UI) descriptions and presentation languages, such as VoiceXML, WML and HTML. The transformation could take place between dissimilar user devices; the UI’s could be physically positioned. Hence, connectivity management is needed to establish and maintain the communication path between the different UIs involved. Related methodologies and implementations available for the UI transformation and the various routing protocols for ad hoc networks are described in [39, 80-83].
Chapter 2. Problems and Scope of this Research

2.3.4.2 Presentation Adaptation and Connectivity Management in MID-B

Within the scope of MID-B, the abstraction of the presentation adaptation follows the methodology defined in the IST MobiLife project [84]. The project defines that presentation adaptation gathers sets of adaptation agents. In each agent, the access and render formats, such as HTML for visual modalities and VoiceXML for audio modality, are detailed. Moreover, a link to a URL of the UI-adapter agent is usually provided.

For the connectivity management, the criteria for connection selection help to maintain seamless connectivity between source and target modalities. For this thesis, connectivity is regarded as physical and logical connectivity.

- Physical connectivity connects two user devices based on the selected network topology, using the available underlying technologies (for example, radio link, wired, wireless, and etc.)

- Logical connectivity is the abstract notion of a link. If the source and target modalities are hosted in the same device, only a logical link is used.

2.4 Summary

This chapter examines the range of topics related to mobile multimodality and future refines the scope of the thesis. First, the development of traditional multimodal interfaces is reviewed and state of the art is discussed, setting the background for the work. Next, the chapter addresses additional challenges that have to be overcome to facilitate multimodality in ubiquitous networks and multi-device environments. These additional challenges include context-awareness, service discovery, interface adaptation, as well as presentation adaptation & connectivity management. The advantages and disadvantages of the existing approaches are discussed.

Concerning context-awareness, the internal (or active) context used in this thesis is characterised by user and platform (device) models. Any environment information is captured as external (or passive) context. According to the definitions made by the MobiLife project, the user model embraces a user profile (including the user physical abilities) and information about the user's social relations. Next, the chapter reviews state of the art related to platform modelling. None of existing approaches provides sufficient expressiveness as are required for dynamic modality provisions. The thesis introduces an extended device (platform) model which includes some selected UAProf components and a range of new components. The standardised service discovery protocols, i.e. Sun’s Jini and Bluetooth SDP, have also been studied. While Jini is not suitable for wireless communications, the mechanism of Bluetooth SDP are. The thesis extends Bluetooth
Chapter 2. Problems and Scope of this Research

SDP ServiceSearchAttribute transaction by appending it with an extension for Multimodality transaction (MM transaction). The MM transaction provides scripts of the multimodal services available, i.e. modality/ies type, characteristics, dependencies, and device connectivity. Then, the chapter addresses the framework for the interface adaptation, this will (in a subsequent chapter) be implemented using afore described context model. Finally, presentation adaptation and connectivity management are discussed. Following the definition of MobiLife, adaptation is seen as a set of adaptation agents. Each agent provides details of agent type, source modality type, available formats, target modality type and a list of target formats. Regarding connectivity management, the chapter explains the two categories of links, i.e. physical and logical connectivity. While this chapter provides an overview of the related areas and pinpoints the scope the further work, the following chapter looks at the state of the art for the actual implementation details.
Chapter 3

3 Background and Related Work

3.1 Introduction

In this chapter, the technological and methodical knowledge required for the work documented in this thesis are explained. Section 3.2 presents the Bluetooth SDP ServiceSearchAttribute transaction, including the details of the services in Bluetooth and these of the Bluetooth profile. The function of the multimodal interaction, which is the fundamental element of the MID-B system, is described in the section 3.3. This section covers human-centred technology, the theoretical framework of combinations in MID-B, and classifications for multimodal dialogues. A brief literature review regarding multimodal multi-device approaches is included. Finally, the section 3.4 concludes the chapter.

3.2 SDP ServiceSearchAttribute Transaction

Often, a client is looking not only for types of available services, but for additional information required to evaluate its service that meets the applications requirements. Generally, the service identifier provides adequate information, but Bluetooth has been developed to provide search mechanisms using both the service identifier and information about the desired attributes of the services. To quicken the discovery process and still obtain a high level of details, the SDP ServiceSearchAttribute transaction has been defined. It combines capabilities of the service search with these of attribute searching.

3.2.1 Services in Bluetooth

3.2.1.1 Service and Service Class

A service specified in the SDP layer of the Bluetooth stack is defined as "any entity that can provide information, perform an action, or control a resource on behalf of another entity" [85]. Because the SDP has no central control scheme, it does not include a service registry mechanism. This however causes some complexity to the identification of individual service if several services are hosted on one device. To handle services, Bluetooth defines two distinct terms, i.e. the
Chapter 3. Background and Related Work

'services' and the 'service classes' [86]. Services are actually specific instances of service classes. Each service consists of collection of service attributes that are required for a particular service class. Uniquely, the service class is identified by a 128-bit Universally Unique Identifier (UUID), which guarantees the uniqueness across space and all time, without the used for central authority [87].

3.2.1.2 Service Record

Containing the information of services, a set of Bluetooth service records is a database of which a unique key-value corresponds to a list of service attributes. Each service attribute characterises a 'single' feature of service. The feature includes, for instance, a service name, a textual description of the service, and a method for service connection. The attribute is said to be universal if and only if its definition is unanimously known to all service records. However, the list of attributes is not limited to the universal definitions, a service provider can define their additional attributes in the 'reserved attribute' fields. Literally, the service attribute is comprised of an attribute ID and an attribute value. Represented by a 16-bit unsigned integer, the attribute ID distinguishes the service attributes located within a service record as well as it determines the meaning of the varied-length attribute values. In case that an attribute ID is allocated to the universal attribute, the associated value must match with the universal attribute's definition. According to the Bluetooth specification, sixteen universal service attributes are defined [69], shown in Appendix A1. In every service record, at least two attributes are required, i.e. the ServiceRecordHandle (attribute ID 0x0000) and the ServiceClassIDList (attribute ID 0x0001), while the remaining fields are optional.

3.2.1.3 Data Element Descriptor

The representation format of attribute values can be either an integer, a data element sequence (a list of data), an UUID, an URL, or a string. A variety of the representation formats can cause difficulties to distinguish the variable-size attributes in the attribute list. Therefore, the attributes are generally formed as 'data element'. Data element starts with a 1-byte data element descriptor, followed by the actual data. To clarify the exact format and the precise length of the data, the data element descriptor is split into 1) a type descriptor occupying the first five bits, and 2) a size descriptor contained in the next three bits. Regarding the type descriptor, there are nine possible data types which are distinctively symbolised by numbers between 0-8, shown in Table 3.1.
### Table 3-1: Type descriptor

<table>
<thead>
<tr>
<th>Type Descriptor Value</th>
<th>Type Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>null type</td>
</tr>
<tr>
<td>1</td>
<td>Unsigned integer</td>
</tr>
<tr>
<td>2</td>
<td>Singed integer</td>
</tr>
<tr>
<td>3</td>
<td>UUID</td>
</tr>
<tr>
<td>4</td>
<td>string</td>
</tr>
<tr>
<td>5</td>
<td>Boolean</td>
</tr>
<tr>
<td>6</td>
<td>Data element sequence</td>
</tr>
<tr>
<td>7</td>
<td>Data element alternative (one of the data in the sequence must be chosen)</td>
</tr>
<tr>
<td>8</td>
<td>URL</td>
</tr>
<tr>
<td>9-31</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

The data element size descriptor expresses the actual data length in bytes, it is composed of a 3-bit size index (placed in the last three bits of the data element descriptor) and additional 0, 8, 16 or 32 bits. The mechanism of the size descriptor encoding is presented in Table 3.2.

### Table 3-2: Size descriptor

<table>
<thead>
<tr>
<th>Size Index</th>
<th>Additional Bits</th>
<th>Data Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1 or 0 byte if the type descriptor value is 0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2 bytes</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>4 bytes</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>8 bytes</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>16 bytes</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>The data size is contained in the next 1 byte</td>
</tr>
<tr>
<td>6</td>
<td>16</td>
<td>The data size is contained in the next 2 bytes</td>
</tr>
<tr>
<td>7</td>
<td>32</td>
<td>The data size is contained in the next 4 bytes</td>
</tr>
</tbody>
</table>
3.2.2 Bluetooth Profile

The Bluetooth profile characterises the functionality provided by a Bluetooth device. It ensures inter-operability among Bluetooth devices provided by different manufactures. To interact between Bluetooth devices, all devices must include the basic profile(s). For example, to connect a PAD to the Internet via a Bluetooth phone, both the PDA and the phone have to implement the Dial-Up Networking Profile. Another example, the phone and a Bluetooth headset require the Headset Profile in order to allow the headset to act as a remote audio interface. Technically, the Bluetooth SIG (Special Interest Group) defines four basic profiles, i.e. the Generic Access Profile (GAP), the Serial Port Profile (SPP), the Service Discovery Application Profile (SDAP) and the Generic Object Exchange Profile (GOEP) [88].

![Hierarchical structure of the Bluetooth profiles](image)

- The most common profile, the GAP, exists in every Bluetooth-enabled device for basic connection establishment, for example discovery, link configuration, and security management.

- The SPP is responsible to setup virtual ports between two devices and to connect them via Bluetooth.

- Directly interacting with the SDP layer of the Bluetooth stack, the SDAP facilitates the fundamental operations needed for the service searching within the vicinity.

- The GOEP defines all necessary utilities for the OBject EXchange (OBEX) usage model, such as file transfer, synchronisation and object push.
Details of these profiles can be found in [70, 88].

Basically, the profiles are designed as building blocks with a hierarchical dependency. Figure 3-1 illustrates the hierarchy of the basic profile structure. In the hierarchy, lower profiles form the basic for the implementations of the higher profile(s). For instance, the GOEP is created on top of the SPP, which is dependent on GAP.

3.2.3 ServiceSearchAttribute Transaction

The Protocol Data Unit (PDU) of the SDP contains messages exchanged in the SDP transaction. The PDU is structured as a 5-byte header and a flexible-length parameter field. For every SDP PDU, the header is established in the same way, while the content in the parameter field varies as defined by the PDU ID.

<table>
<thead>
<tr>
<th>0</th>
<th>8</th>
<th>16</th>
<th>24</th>
<th>32</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDU ID</td>
<td>Transaction ID</td>
<td>Parameter length</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3-2: Generic structure of the SDP PDU

The header construction, shared by the diverse SDP PDU types, encloses three consecutive fields.

- 1-byte PDU ID identifies the meaning of the messages in the parameter field.
- 2-byte transaction ID is uniquely assigned for every SDP request PDU, and subsequently copied into the associated response PDU.
- 2-byte parameter length indicates the entire length in byte of the parameter field.

The complete specification of the parameter fields allocated for each PDU ID is described in [4]. This section provides the explanations of the parameter fields of the SDP PDUs selected for the MID-B system (i.e. the SDP ServiceSearchAttribute request PDU and SDP ServiceSearchAttribute response PDU). An example of ServiceSearchAttribute transaction is provided in Appendix A2. The example shows how to form the request PDU and the corresponding response PDU. Within the Appendix A2, the way of how to use the data element descriptor is also shown.

3.2.3.1 ServiceSearchAttribute Request PDU

PDU ID 0x06 represents the SDP ServiceSearchAttribute request PDU. Messages carried in the parameter fields are characterised by four parameters, i.e. ServiceSearchPattern, MaximumAttributeByteCount, AttributeIDList, and ContinuationState.
• ServiceSearchPattern is a sequence of UUIDs representing patterns of the service record search.

• MaximumAttributeByteCount suggests maximum size in byte of the AttributeLists parameter contained in the parameter field of the associated response PDU.

• AttributeIDList is made up of a sequence of the interesting attribute IDs.

• ContinuationState specifies whether the request PDU starts a new transaction or continues from a previous request. It begins with an 8-bit count that indicates the length in byte of the following continuation-state information. If the count is set to 0, the continuation-state information does not exist and the PDU initiates a new transaction.

3.2.3.2 ServiceSearchAttribute Response PDU

The PDU ID 0x07 signifies the SDP ServiceSearchAttribute response PDU, of which the parameter field includes AttributeListByteCount, AttributeLists and ContinuationState.

• AttributeListByteCount indicates the number of bytes in the following AttributeLists parameter.

• AttributeList is created by a number of attributes obtained from the service records that correspond to the ServiceSearchPattern contained in the matching request PDU.

• ContinuationState describes whether the PDU is a partial or complete response. Similar to the request PDU, the first 8-bits correspond to the number of bytes of the following continuation-state information. If it is set to 0, there is no continuation-state information. The transaction then can be terminated after the client receives this response PDU.

3.3 Multimodal Interaction

3.3.1 Human-Centred Technology

Forming user interface (UI) for applications relies on the specification of a physical device; this follows the traditional Technology-Centred Design (TCD). TCD takes the set of functionalities that are available to support the application and presentation level. It maps the user interface and interactive dialogue to the physical device and physical interfaces [89]. Unfortunately, TCD only considers the presentation level and the hardware technology side. Thus, there is no flexibility to match the pattern of the interaction to the needs of single user and circumstances. Users have to adapt themselves to the interface offered. Enhanced user interfaces often follow a reference framework as described in [90-92]. Interactive dialogues enable the Human-Centred Systems
Chapter 3. Background and Related Work

(HCS) rather than only TCD. HCS introduces cognition and collaboration across common people's behaviours and activities; applied to UIs allows a human-centred design. HCS also encloses psychological processes to understand a user's behaviour; this is called the Human-Centred Computing (HCC). In relation to Human-Computer Interaction (HCI), HCC is the advancement of HCI. In [93], NASA exemplifies the needs of a human-centred model for multimodal interface design. To summarise, HCC provides a process to understand a humans' behavioural patterns. The user behaviour model allows the personalisation of multimodal dialogues [94]. Moreover, HCC also introduces proactive systems that understand and respond to cultural and social contexts [95, 96]. Multiculturalism takes an important role in the proactive system, and cultural differences can be incorporated for the interaction design. For instance, in [97], mobile terminal manufactures envision how the unique cultural background influences the behaviour of mobile consumers. As the reference argues, there are noticeable differences between US and Korean users' behaviours. Voice-mail service is more favourable in USA, while most Korean users prefer SMS. Accordingly, the physical interface of phones is differed between the markets. For example, while in US one of the hotkey is assigned to the voice-mail function, in Korea, the same key is used to start the messaging application.

### 3.3.2 A Theoretical Framework for Multiple Modalities

Following the principle of HCS, data content is rendered and accessed relative to available sensory perceptions. The interaction is driven in collaboration with the usage of the available modality services. Collaboration can therefore be defined physical action on modality combinations. The (collaboration) action considers information obtained from diverse I/O equipments that host the available modalities. As stated in Section 2.2.1, the (collaboration) action could behave in six possible ways; i.e. transfer, specialisation, equivalence, redundancy, complementarity and concurrency.

![Modalities Combination Diagram](image)

**Figure 3-3: Grouping of modality combination**

Alternatively, Niklfeld [98] proposes the categories for modality combinations, i.e. a sequential combination, a coordinated-and-simultaneous combination, and an uncoordinated-and-
Chapter 3. Background and Related Work

simultaneous combination. Each category deals with the absence and presence of modalities as well as it determines which and how chunks of multimodal information will be exploited. The following sub-sections group the modality-combining actions into these categories, and present the logical notation formally expressing the dependencies between the action and its desired parameter(s). Originally formulated in [99], these notations provide the groundwork to implement the multi interface-device binding. Figure 3-3 illustrates the grouping of modality combinations.

3.3.2.1 Sequential Combination

The sequential combination characterises a chronological use of modalities. It allows a number of modalities to be active, but only a single modality can function at a time. In essence, the actions of the sequential combination use one modality after another. Three types of combining actions, i.e. transfer, specialisation and equivalence actions, are grouped within the sequential combination.

Let a modality M receive and produce a set of chunks of information $E(M)$ and $S(M)$, respectively. $E(M)$ and $S(M)$ will be used in the rest of this section.

3.3.2.1.1 Transfer Action

Modalities $M_1$ and $M_2$ liaise on the transfer action if some of the received information at $M_2$ are generated and sent by $M_1$. Denoting $tr$ as a transfer operator, the notation of the transfer action can be expressed as:

$$ transfer(M_1, M_2, tr) : tr(S(M_2)) \subseteq E(M_2) \quad (3.1) $$

3.3.2.1.2 Specialisation Action

Given, $I$ being a set of specific chunks of information produced by a particular modality $M_i$. Let $M_i$ be any modality in the system, excluding $M_i$.

The specialisation action restricts that $I$ must be produced only by modality $M_i$. If $M_i$ needs the information contained in set $I$, it has to request the information from the modality $M_i$.

$$ specification(M_i, I, \{M_i\}) : I = S(M_i), \forall M_i, I \subseteq S(M_i) \quad (3.2) $$

3.3.2.1.3 Equivalence Action

A chunk of information, $I$, is alternatively produced by 'either' a modality $M_1$ or a modality $M_2$. Assuming, $eq$ is an equivalence operator that considers several influential facets in a modality determination, such as user experiences and environmental conditions. The notation of the equivalence action is expressed as:
Chapter 3. Background and Related Work

equivalence(M_1, M_2, I, eq):
\( \forall i \in I, \exists e_1 \in S(M_1), \exists e_2 \in S(M_2), i = eq((M_1, e_1), (M_2, e_2)) \)  

3.3.2.2 Coordinated-and-Simultaneous Combination

The coordinated-and-simultaneous combination supports parallel use of modalities. Chunks of information simultaneously treated by assorted modalities are in sync to form a joint event, based on time stamp. For example, from the observations on the use of an interactive map [100], the majority of the users preferred coordination between multiple modalities. 95% of the participants preferred an incorporation of speech and gesture modalities, while only 5% preferred on pen-only interaction. Yet, a speech-only interaction was not appreciated at all. Indeed, the coordinated-and-simultaneous combination is time-sensitive and requires semantic data fusion and fission. For instance in [101], a basic application of “Copy and Paste” maintains the Parallel Control Agent (PCA) and the Language Agent (LA) to relate the speech modality to simple mouse clicking. In such cases, one modality could replicate the significance of information generated by another. Using the “Copy and Paste” example, a user could utter “quit” and click a quit button. Therefore, this combination necessitates an additional attribute to differentiate whether the chunks of multimodal information are ‘redundant’ or ‘complementary’.

For the redundancy and complementary actions consider:

'I' being a set of chunks of information,
're_att' being the redundancy attribute,
'crit' being a criterion to interpret the value of the 're_att',
's_1' being the information produced by a modality M_1, and
's_2' being the information produced by a modality M_2.

3.3.2.2.1 Redundancy Action

The redundancy action allows a range of modalities to simultaneously produce the same information. Both modalities M_1 and M_2 provide a chunk of redundant information, if the meaning of s_1 coincides with that of s_2. Assuming, re represents a redundancy operator. re combines a pair of s_1 and s_2 if the values of re_att on the M_1 and M_2 are identical and crit is true. The notation of the redundancy action is expressed as:

\[ \text{redundancy}(M_1, M_2, I, \text{re} \_ \text{att}, \text{crit}) : \forall i \in I, \exists s_1 \in S(M_1), \exists s_2 \in S(M_2), i = \text{re}(s_1, s_2, \text{crit}) \]  

3.3.2.2.1 Redundancy Action

The redundancy action allows a range of modalities to simultaneously produce the same information. Both modalities M_1 and M_2 provide a chunk of redundant information, if the meaning of s_1 coincides with that of s_2. Assuming, re represents a redundancy operator. re combines a pair of s_1 and s_2 if the values of re_att on the M_1 and M_2 are identical and crit is true. The notation of the redundancy action is expressed as:

\[ \text{redundancy}(M_1, M_2, I, \text{re} \_ \text{att}, \text{crit}) : \forall i \in I, \exists s_1 \in S(M_1), \exists s_2 \in S(M_2), i = \text{re}(s_1, s_2, \text{crit}) \]
Chapter 3. Background and Related Work

3.3.2.2 Complementary Action

Specifically, the complementary action is the synergistic action of $M_1$ and $M_2$. The values of $s_1$ and $s_2$ are highly inter-reliant and synchronised. Let $co$ be a complementary operator. Dissimilar to the redundancy operator, $co$ unites $s_1$ and $s_2$ if the values of $re\_att$ are not the same and crit is true. The notation of the complementary action is expressed as:

$$complementary(M_1, M_2, I, re\_att, crit) : \forall i \in I, \exists s_1 \in S(M_1), \exists s_2 \in S(M_2),$$

$$re\_att(s_1) \neq re\_att(s_2) \land i = co(s_1, s_2, crit)$$

3.3.2.3 Uncoordinated-and-Simultaneous Combination

The uncoordinated-and-simultaneous combination manages the concurrency action. The concurrency action considers the fact that although a multimodal interaction consents to simultaneous use of different modalities, simultaneity does not always imply synchronism. The chunks of available multimodal information could be uncoordinated and independent. In contradiction of the complementary action, the information is self-reliant. A mechanism to merge the information is not required. The chunks of information are processed in isolation. Because a dependency between the chunks of multimodal information does not exist, the combination notation of the concurrency action does not have to be defined [99].

3.3.3 Classification of Multimodal Dialogue

Design of human-machine dialogue is part of the artificial-language development [102]. Traditionally, the basic mode of input and output, such as mouse clicking and Graphic User Interface (GUI), utilises a collection of (computer) peripherals. A dialogue following such conventional interaction is commonly operated using a single-modality, each modality used has a discrete role in the communication. In contrast, multimodal interfaces exploit combinations and distinct natures of the various modalities. The scope of multimodal interfaces can include I/O modalities that use all human senses, i.e. hearing, seeing, touching, smelling, and tasting [103].

3.3.3.1 Auditory Modality

In the most basic way, the sense of hearing (auditory modality) does not confine a user to looking in a definite direction. A warning tones are one of the most basic form of acoustic information [104]. In addition, this modality involves linguistic-based technologies that comprise the studies on a Speech Application Programming Interface (Speech API) [105, 106], speech identification [107], dialogue modelling [108] and emotional content in speech [109]. In detail, the speech API entails speech recognition and speech synthesis that control over a speech input and a speech output, respectively. Also known as Automatic Speech Recognition (ASR), speech recognition is
"the ability to converse freely with a machine represents the ultimate challenge to our understanding of the production and perception processes involved in human speech communication" [110]. For the speech input's point of view, prosodic information provides extra knowledge concerning emotion of a user. Emotional knowledge is generally computed by means of additional acoustic features, such as energy, pitch and speaking rate [111]. On the other hand, the speech synthesis results in a natural speech output. So far, the existing technicality of speech synthesis only covers a text-to-speech (TTS) conversion. Tippy [112] presents the definition of the TTS as “the process of converting text into human recognizable speech based on language and other vocal requirements”. Speech can be modelled as either an ‘ordinary conversation’ or a ‘verbal shadowing’ [108]. The ordinary conversation conveys a specific message to carry out a precise task; whereas the verbal shadowing represents a subsidiary sound that is a commonly generated alongside the ordinary conversation.

### 3.3.3.2 Visual Modality

The visual modality corresponds to the sense of seeing. Mostly, the multimodal research pays particular attention to implementations of vision-based recognition, rather than a visual output. Still, the traditional GUI is used for the visual output. The Automatic Handwriting Recognition (AHR), widely integrated in PDAs, is the typical example of vision-based recognition. The AHR detects the trajectory of the pen tip [113]. With the introduction of the Human-Centred Computing (HCC), vision-based recognition moves forward to observe and to estimate a user’s position and posture. It detects physiological activities of a human-being, including head and body movement, facial expression, hand pointing, and eye gaze [114]. Literally, the physiological activity is a body gesturing that represents a particular intention. Movement pattern analysis has been presented in [115, 116].

While the auditory modality accounts for the linguistic expressions, gestures relate to body-language expressions. The combination of linguistics and body language provides possibilities for rich collaborative interaction. A dialogue simultaneously allowing verbal and visual inputs enables exact addressing of objects in the workspace. Tse [117] implements a multimodal multiplayer gaming environment for two commercial single player games (i.e. ‘Warcraft III’ and ‘The Sims’). The command dialogues in these games are coordinated between hand posture and speech, such as uttering ‘move here’ or ‘build a farm here’ while pointing to a location. If a body-language expression is able to completely substitute speech, it is called as a ‘manifesting action’ [108]. ‘Warcraft III’ users can make a command input by using a sign language mimicking real world military commands. In addition to large-scale body movement, be it gesture or hand posture, small movements, like eye gaze, are taken into consideration as visual input to confirm a user’s attention. Obviously, the gaze is captured by a gaze awareness function, which returns
information in three different ways [118]. The ‘mature’ gaze indicates whether other participants are looking at the user. The ‘partial’ gaze indicates the direction of the gaze (up, down, left or right). And finally, the ‘full’ gaze provides information about an object at which a user is looking. Gaze-awareness is classified as ‘diagnostic’ and ‘interactive’ [119]. A diagnostic application does not consider gaze movement as an input modality method. Gaze movement is analysed offline assessment, impacts on the overall input is given, rather than a particular step only. In contrast, for an interactive application, the gaze is supposed to function as a pointing device in most applications. However, any application of this type requires a real-time gaze detector, any interactive dialogue and output presentation must instantaneously act in response to the user’s gaze.

3.3.3.3 Tangible Modality

By definition, tangible refers to a concrete object that can be actually seen and touched. The Tangible User Interface (TUI) allows for a communication between user and computer via everyday physical objects. The concept excludes the traditional computer peripherals. The TUI facilitates manipulation of palpable materials located within an observed area. Indeed, the logical view of the TUI is the extension of the ubiquitous computing vision. That is, the TUI focuses on embedded physical devices as primary means for information access and management [120]. The key task of TUI is to convert the physical information into digital representation. Given by [121], the digital presentation, such as audio/visual, is “computationally mediated displays that are perceptually observed in the world, but are not physically embodies, and thus intangible in form.” Clearly, a modality grouped in the TUI, for example haptic, involves the sense of touch. The types of touch perception are ‘kinaesthesia’ and ‘tactile’ [122]. The kinaesthesia is a sense detecting bodily movement of muscles, tendons, and joints, whereas tactile relates to the receptors in the skin. [122] also suggests six different procedures contributing to the haptic modality. There are lateral motion (such as texture), pressure (such as density/hardness), enclosure (such as global shape/volume), static contact (such as temperature), unsupported holding (such as weigh) and contour following (such as shape). The demonstration in [120] illustrates how a haptic dialogue is exploited to a stress-reduced application. A pen, for instance, is assigned as an object to measure the stress. The act of gripping the pen tightly implies that a user could be under stress. The stress level can be examined by how tight the grip is or how much pressure the user exerts on the tip when writing. Alternative example of the tangible modality is provided by [123], a link of URLs will be automatically added to a personal URL favourite list by touching, on a screen at the position of the URL. The index finger could be uniquely identified by a finger print. The ATELIER project [124] yields an interface adaptation that transforms the standard GUI sketching tool to the tangible interface. The ATELIER system is setup by a table with a semi-transparent
Chapter 3. Background and Related Work

white glass plate, a simple web camera under the plate, and the colour cubes. Instead of dragging a mouse, the sketch in the ALELIER system is made by arranging the cubes on the table.

3.3.3.4 Olfactory and Gustatory Modalities

Other human sensations, smell and taste are closely linked [125]. Generally, smell and taste are hardly considered as means for communication, they are generally excluded from the early version of human-computer interaction (HCI). However, to increase the diversity of HCI techniques, some research now puts effort in the development of modalities supporting the senses of smell and taste, such as [126, 127]. Both references envision how to implement the olfactory and gustatory modalities. Alternative to auditory and visual modalities, the AROMA project [128] uses smell for rendering a message notification. Although the experiment shows that the performance of the olfactory modality is at a lower level than the other senses formats, it introduces less disruption to user concentration. In the same way, the Olfoto project [129] makes a comparison between text-based and smell-based labels used for digital photo searching. The text-based search is more effective, but in some cases, the results do show almost comparable behaviour for the two search approaches. In general, the gustatory modality is still underexplored, there is no explicit approach to combine the gustatory modality with others. So far, the gustatory modality is operated as a singular input system, like the multi-sensor electronic tongue system (ET) [130] used to recognition to distinguish synthetic and alimentary grain ethanol by taste only.

3.3.4 Research on Multi-Device Multimodality

Not only the single modalities are investigated, but also research into multimodal approaches has been undertaken. For example, healthcare application in [131] illustrates a multiple-device deployment supporting local mobility in a hospital. The application facilitates dynamic tailoring of user interfaces using multiple devices for accessing a patient’s records. The Context-Plane defined in the Nightingale project [132] discovers services and devices in a user area. Then, it processes a multimodal dialogue through a range of dynamic devices. Service and device selection is implemented by prioritising the discovered components. Another example, the Multiple-Device Environment (MDE) [133] controls, via a 2D iconic tool, how an application presents its contents; it can facilitate the relocation of the media stream to a remote display. The Device-Capability-On-Demand (DCOD) framework [134] enables multiple device cooperation. It uses a language describing these devices’ capabilities, called the Virtual Device Description Language (VDDL). The VDDL provides an instruction on how to select and split a service across interface devices. In general, distributed interface devices are organised autonomously, but function cooperatively [135]. A high-level specification automatically synthesising the UI for
multiple devices is shown in [136]. The specification suggests four processes needed for UI synthesis. These processes are 1) specifying user interface (UI) domains, 2) creating a finite state machine of the systems, 3) assembling the information and communicative operation of the UI domains in accordance with each state, and 4) rendering the corresponding UI. Further, [137] presents a method to simultaneously render an UI across multiple devices and modalities. The method extends the standard User Interface Description Language (UIDL), which is a XML-based single authoring language. Similarly, Scooby [138] is a Java-based language able to bind the distributed compositions of interface services in a pervasive environment. Scooby defines sets of binding policies and binding variables that provide a connection between a main service and the remote services sited on other devices. To transform the UI description between different interface devices, an interface layout management is needed [139]. The Device-Independent UIML (DI-UIML) [140] enables an automatic layout adjustment for multi-device UIs. It is built atop the User Interface Mark-up Language (UIML), and adds an abstraction of UI layout constraints.

3.4 Summary

This chapter presented the background knowledge needed for this thesis. The chapter presented the Bluetooth SDP ServiceSearchAttribute transaction, which offers the mechanisms for service search based on both service identifier and required service attributes. Indeed, the service information is maintained in a set of Bluetooth service records, each record comprises a unique identifier and a corresponding list of service attributes. In theory, the ServiceSearchAttribute transaction is a client-server mechanism; the client sends the ServiceSearchAttribute request PDU and the server replies with the corresponding ServiceSearchAttribute response PDU. The formats of the request and response PDUs share a header structure, while information in the parameter field is defined by the types of PDUs. The chapter then examines the work on the multimodal interaction. Conceptually, the multimodal user interface evolves into the Human-Centred System (HCS) which encloses the psychologically process called the Human-Centred Computing (HCC). In principle, HCC is an analysis of user behaviour and of the cultural and social contexts. The chapter continues with the revision on the multimodal combinations. Multimodal interaction is a combination of the distinct modalities. The combination can be broken down into three categories: sequential combination, coordinated-and-simultaneous combination, and uncoordinated-and-simultaneous combination. In the sequential combination, only one modality can provide its service at a time. The other options allow simultaneous manipulation of multiple modalities. While coordinated-and-simultaneous combination offers synchronism, uncoordinated-and-simultaneous does not. Multimodal dialogue is stimulated by human sensational perceptions, i.e. hearing, seeing, touching, smelling, and tasting. This chapter also presented the different kinds
of modalities that are equivalent to the human senses. Modalities are grouped as: auditory, visual, tangible, olfactory, and gustatory. Finally, this chapter presented a review of the research on the multimodal multi-device approaches. In the next chapter, the detailed contributions of the thesis are presented, together with the details of how the background techniques are exploited within the MID-B system.
Chapter 4

4 The Multi Interface-Device Binding (MID-B) System

4.1 Introduction

The core contribution of this thesis is the Multi Interface-Device Binding (MID-B) system, which enhances a multimodal interaction into a virtual-device environment. MID-B is built of a set of support services as well as on the capability to use a wide range of context information. The architecture of MID-B is based on networked interface-devices within a mobile environment. The architecture provides the ability of interaction services to automatically and dynamically adapt themselves according to context changes. This will help to overcome the drawbacks of classic multimodal interaction, which multimodality uses a strategy to simultaneously exploit modalities offered on a single device. The MID-B's mechanism gets multimodality out of the single-device scenario. The approach exploits the multiple devices and their available user interface for 'single-users'. This is, distributed interface-devices are considered as one virtual device with many...
interfaces and modalities available for a user. In MID-B, a 'controller-device' is aware of the availability of various devices within the vicinity, each of which may host one or more user interfaces (modalities). The capabilities of each co-located device, together with the context in which the user acts, are exploited to dynamically customise the interface services available.

The development of MID-B follows the four steps of waterfall model for software engineering [141]. These steps include requirement analysis, preliminary design, detailed design, and integration & test. In detail, the requirement analysis was used to determine the functional scope of the MID-B workspace. Then, the preliminary design step introduced the model of the MID-B as well as it defined the system's main components. All remaining functionality was defined during the 'detailed design' step. In depth information of the MID-B implementation, together with the components' roles, was provided. Finally, the step of integration & test merged the set of the MID-B functions and evaluated the functionality of the system. In practice, the four-step waterfall model is iterative. That is, the higher step provides its outcomes to the lower step, while the lower step provides feedback to possibly request some changes in the preceding definitions.

The remainder of this chapter describes the MID-B architecture and its mechanisms to collect and exploit device and user context information to dynamically adapt the user interfaces. In the chapter, the first two steps of the system development process, i.e. 1) the requirements analysis and 2) the preliminary design that provides the overall architecture of the MID-B system, are described. The chapter is organised as follows. Section 4.2 presents a detailed description of the MID-B requirements. The potential problems are analysed in terms of the objectives that the MID-B system has to achieve. To specify the problems, this section dissects the whole system into a set of functionalities, which includes several support functions. The functions include the processes for collecting, binding, selecting, setting and/or amending the interaction and the combinations of the possible modalities and devices. Section 4.3 explains the MID-B architecture at a high level of abstraction. In the first of three subsections, an introduction to the Finite State
Machine (FSM) is given. The second subsection describes how the components are integrated in the architecture and how they are interconnected. Indeed, they collaborate to facilitate the multimodal services in a multiple-device environment. The third subsection presents the logical prototype of the MID-B system by using the Finite State Machine (FSM). At this point, the interaction between different components is demonstrated. Finally, Section 4.4 concludes the chapter.

4.2 System Requirements

Prior to designing and implementing the MID-B system, a review of the system requirements is needed to capture the system objectives. In practice, the review includes an examination of the MID-B’s expected behaviours and an observation of the system’s needs. This thesis determines sets of requirements from different perspectives; this leads to an understanding of the system impacts at different levels of attention. This principle is founded on the basic assumption that the definition of system goals is atomic. Therefore, dividing the system into constituent parts enables the system’s behaviours to be thoroughly investigated. The MID-B requirements are elicited in three levels, i.e. the system level, the functional level, and the implementation level. At the system level and the functional level, the requirements investigation corresponds to the main objective and the core functionalities of the MID-B. The detail of practical implementation is not considered here. In contrast, the requirements analysed at the implementation level are detailing the factual needs of the MID-B.

4.2.1 System Level Requirements

At the system level, the ‘goal-oriented requirements’ are captured. The primary interest at this level is what mechanisms of the MID-B’s front-end service will provide. Mainly, the MID-B system provides mechanisms to make decisions for the adaptation of interfaces as well as to determine whether a service will be delivered via several collaborating modalities and/or multiple devices. It envisions a ‘physical-independent’ framework that dynamically unites distributed interface devices into a single ‘virtual-device’. The operation of the system is designed based on a dynamic and wireless networking environment. Due to mobility, devices providing user interfaces and modalities may or may not be available for the whole duration of a communication. The MID-B system exploits the fact that the number and types of accessible interfaces may change over time. Hence, the system needs to follow with its internal behaviour and the way a service is delivered to the randomly changing interface capabilities. The system has to handle the interface changes in two different ways:
• Adjusting existing service behaviour. A user may change an accessing device, and thereby an accessing modality may have to be changed.

• Dynamically deploying a system configuration. Any change of user context, such as location, may trigger a change for the internal system configuration.

### System Level Requirements

Goal-oriented requirements: Envision a ‘physical-independent’ framework that dynamically unites multiple distributed devices into a ‘virtual-device’ environment

<table>
<thead>
<tr>
<th>Required mechanisms</th>
</tr>
</thead>
<tbody>
<tr>
<td>A mechanism to make a decision for the adaptation of interface service</td>
</tr>
<tr>
<td>A mechanism to determine whether a service will be delivered via several collaborating modalities and/or multiple devices.</td>
</tr>
</tbody>
</table>

![Figure 4-3: System level requirements](image)

### 4.2.2 Functional Level Requirements

The framework (MID-B) requires a number of support functionalities. Literally, the requirements at the functional level correspond to the scope specified in Chapter 2. The MID-B system is multidisciplinary; the range of function required includes:

• service and modality discovery,

• context-awareness,

• device binding & user-interface (modality) adaptation, and

• presentation-adaptation & connectivity-management.

![Figure 4-4: Functional level requirements](image)
4.2.3 Implementation Level Requirements

The requirements on the implementation level are how to implement the required functionalities. The internal mechanisms and essential tasks of these functionalities are shown in Figure 4-5.

<table>
<thead>
<tr>
<th>Goal-Oriented Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Envision a ‘physical-independent’ framework that dynamically unites multiple distributed devices into a powerful ‘virtual-device’ environment</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Functionality-Oriented Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service and Modality Discovery</td>
</tr>
<tr>
<td>• Implementation-Oriented Requirements</td>
</tr>
<tr>
<td>• A basic mechanism of device discovery</td>
</tr>
<tr>
<td>• A driving mechanism of service discovery</td>
</tr>
<tr>
<td>Context-Awareness</td>
</tr>
<tr>
<td>• Implementation-Oriented Requirements</td>
</tr>
<tr>
<td>• A scheme describing a device and embedded services</td>
</tr>
<tr>
<td>• A scheme describing the user information</td>
</tr>
<tr>
<td>Device Binding &amp; User-interface (Modality) Adaptation</td>
</tr>
<tr>
<td>• Implementation-Oriented Requirements</td>
</tr>
<tr>
<td>• The technique to trace a service site</td>
</tr>
<tr>
<td>• A set of rules to bind distributed interface into a virtual-device environment</td>
</tr>
<tr>
<td>Presentation-Adaptation &amp; Connectivity-Management</td>
</tr>
<tr>
<td>• Implementation-Oriented Requirements</td>
</tr>
<tr>
<td>• A mechanism to reconfigure an interaction session</td>
</tr>
<tr>
<td>• A mechanism to network physical devices</td>
</tr>
</tbody>
</table>

Figure 4-5: Hierarchy of requirements of the MID-B system

4.2.3.1 Implementation Requirements for Service and Modality Discovery

Service discovery needs the joint mechanisms of device and service discoveries. Device discovery allows a core device to identify co-located devices. The implementation of service discovery requires an extensive mechanism to observe the users’ changing condition. In this case, the condition implies the dynamic availability and accessibility of current interaction means, including all information regarding the actual interfaces together with capabilities and limitations.

4.2.3.2 Implementation Requirements for Context-Awareness

Context-awareness entails description of internal contexts, i.e. device context and user context. Basically, the device cannot be described by simple identifier, but a semantically rich explanation of embedded services is actually needed. In addition to the semantic expression of the device context, a well-defined description of the user context is also required. The potential service must be in line with the user’s preferences. Especially, selecting an interface service hinges upon the understating of user skills, including the user physical challenges and abilities.
4.2.3.3 Implementation Requirements for Device Binding & User-Interface (Modality) Adaptation

User-interface adaptation ensures the suitability of interface services. It needs methods covering 1) tracing of service, and 2) handling of binding rules. Initially, the tracing method locates the modalities and devices in the environment. Usually, the co-located modalities are limited by the device’s capabilities. Such a case, the tracing approach must provide the ability to search for modalities and devices in diverse conditions and parameters. Examples for such searches include:

- Search a set of devices hosting a given modality, regardless of the modality capabilities.
- Search a set of exact devices hosting a given modality, based on specific capabilities of the modality.
- Search a modality, regardless of the service’s capabilities, on a particular device.
- Search an exact modality, with specific capabilities, on a particular device.
- Search a modality, regardless of the service’s capabilities, when a device’s address is not identified.
- Search an exact modality, with specific capabilities, when a device’s address is not identified.

After the tracing procedure, the precise location of a modality is found, a reason why this modality should be selected is still missing. This may result in improper service delivery, hence rules of when and how to bind a set of interface devices are needed. Despite multiple devices are available in a service area, not all of their modalities may be accessible. Accordingly, the underlying rules must specify the accessibility state of devices and modalities; this includes consideration of information such as location, ownership, security policy, etc. In a complex environment with (feature) rich physical interface devices, the rules exploit the relational dependency between modalities that are offered by a number of devices. Assuming, ‘A’ and ‘B’ are interface devices. Let \( M_A \) and \( M_B \) be a set of modalities provided by A and B, respectively. The rational dependency must express the semantic relation of the available modalities as one of the five following cases.

- The set of modalities hosted on devices A and B are identical.

\[
M_A = M_B \tag{4.1}
\]

- Devices A and B provide the same set of modalities, their capabilities may differ.

\[
M_A \approx M_B \tag{4.2}
\]
4.2.3.4 Implementation Requirements for Presentation-Adaptation & Connectivity-Management

The functionality of presentation adaptation and connectivity management reconfigures the interaction session and changes the set of interface hosting devices. To reconfigure an interaction and its associated process, the presentation plan must be amended. Hence, a mechanism converting the original media formats to the format of the target modalities is needed. In some case, the interface for the target modality may not be available on the same device as the original media, but it may be located in a neighbouring device. These two devices can then be networked in several ways, depending on the type of available connections. The type and quality of connection may raise additional requirements to the system; this includes the need for media stream re-routing, etc. In the MID-B, the connectivity management incorporates following tasks:

- to decide whether the presentation of content should be on the same device or rerouted to another, and
- to decide on which networking technology should be used to connect the content source with the (virtual device) user interface.

4.3 System Architecture

The MID-B architecture is designed in relation to the MID-B requirements. This section describes the preliminary design of the MID-B architecture. Abstractions of the problem-solving strategy and the prototype are presented.

4.3.1 Theoretical Background – The Finite State Machine (FSM)

The formal analysis of the MID-B prototype uses the means of a Finite State Machine (FSM). The FSM is a visual-aid to depict automata theory. The basic FSM, represented as a transition diagram
shown in Figure 4-6, is made of a set of system states and a set of events [142]. Typically, the FSM diagram consists of labelled states (depicted as circles) and directional labelled-connections (for system events). In general, the system state encompasses properties describing the system’s behaviour at that time [143]. And, the state transaction is normally triggered by an action of the events that may occur at a state. For every FSM, there is at least one initial state. The initial state is typically the first state after initiation of a system.

![Figure 4-6: Basic transition-diagram of FSM](image)

The FSM is a high-level mathematical model of system incidents. Theoretically, the model of FSM can be defined as a ‘Moore machine’ [144] and a ‘Mealy machine’ [145]. In the Moore machine, the state's output relies only upon current states of system. Dissimilarly, the output of the Mealy machine depends on both of the input events and the current states. Because of introducing a smaller size of states, the model based on Mealy machine is selected in this thesis. Formally, the Mealy FSM is defined as a 6-tuple \((I, O, S, S_0, F, G)\):

\[
I = \{i | i \in I\} \text{ is a finite set of input events,}
\]

\[
O = \{o | o \in O\} \text{ is a finite set of output events,}
\]

\[
S = \{s | s \in S\} \text{ is a finite set of internal states of the system,}
\]

\[
S_0 \subseteq S \text{ is a finite set of initial states,}
\]

\[
F: S \times I \rightarrow S \text{ is the relation between the pairs of a current state and an input event for the next states, that is } F \subseteq I \times S \times S.
\]

\[
G: S \times I \rightarrow O \text{ is the relation between the pairs of a current state and an input event for the corresponding output, that is } G \subseteq I \times S \times O.
\]

With the FSM definition, relevant specifications are identified.

- At every state \(S\), a set of outputs \((O)\) is generated. If \(O\) is an empty set, this state \(S\) is considered as an end-state.

- The FSM is said to be deterministic if the set of \(S_0\) contains one element and both of \(F\) and \(G\) are functions. In mathematics, the relation of \(F\) becomes a function if the pairs of input event and current state \((I, S)\) are uniquely related to the next state \((S)\). Similarly, the relation of \(G\) is a function if \((I, S)\) uniquely specifies the output \((O)\).
Chapter 4. The Multi Interface-Device Binding (MID-B) System

4.3.2 System Components

The MID-B architecture consists of a set of components with distinctive properties. The defined components must be amenable to the requirement specification. Three categories of the architectural components are identified, i.e. a set of ‘terminals’, a set of ‘profiles’, and an ‘interaction manager’. Firstly, the set of terminals is the collaborative physical devices encompassing interfaces between an end-user and an active application. Secondly, the set of profiles is all about databases maintaining context information related to the MID-B execution. Finally, the interaction manager uses the context to customise the interface service as required by the contexts.

4.3.2.1 Set of Terminals

In an on-demand multi-device scenario, the MID-B system takes advantages of the modalities of co-located (public) devices to deliver an enhanced interaction. In the system, the devices are defined as:

- a private device, called the User Equipment Core Device (UE_C), and
- none, one or many public device(s), called the User Equipment Interface Device (UE_I).

Characteristically, the UE_C is a user access device, always carried by the end-user. Generally, it is a portable (private) device with diverse limitations in its user interface and interaction means, etc. The UE_C may not be able to deal with advanced user interface (UI) technology. Hence,
supplementary interfaces and modalities offered by other surrounding devices can be used to enhance the interface capability. When bound into the virtual device, the surrounding devices, the UE_Is, will act as the content accessing/rendering devices in the locality. A typical UE_I may host a microphone, screen, speaker, etc. From the UE_I point of view, any (user) interface is generally described by a name, a communication channel, a list of attributes like connectivity and user privilege information. For example, a monitor is a visual output device providing a resolution of maximum 1024 by 168 pixels at 85 Hz frame rate, connected with fixed wire, and not bound in an adaptive user interface for a blind person. Another example would be a printer providing printing service which is able to offer colour printing at 120 by 720 dpi, and only employees in a particular department are able to use this printer. With in the MID-B architecture, the UE_C takes an action as the system’s master device that temporarily manipulates/uses the UE_I(s). The manipulation could be either manually controlled by a user or automatically managed.

On the whole, the UE_C and the UE_I(s) shape a virtual-device. A service and interaction to the end-user could be distributed between user interfaces on the UE_C and those offered by the UE_I(s). Yet, the inherent details of the devices are hidden from the end-user.

In relation to the specifications of the UE_C and the UE_I, the MID-B architecture is based on three assumptions:
• A UE_C belongs to a specific end-user. The address of the UE_C explicitly identifies the end-user, but not vice versa. That is because the end-user may own a number of portable devices.

• There is only one UE_C integrated into the architecture at a time. Although the end-user carries several devices at the same time, only one UE_C is allowed for each individual service session.

• All devices encompass wireless discoverability, for example the Bluetooth SDP. However, their underlying network connectivity, such as WAN, LAN, WLAN or cellular network, to transfer data content is transparent to the system.

4.3.2.2 Profiles in MID-B

A profile represents an instance record of a system condition at a given situation. A set of profiles is often described using XML-based ontology. The XML document provides a hierarchically organised database [146] that allows simple extensibility. Actually, there must be an agreement on the terminology for the representation format of XML. This thesis exploits the XML tags defined in the UAProf ontology [58]. Within the profiles, the context elements are captured by one of these tags. UAProf tags include <Bag>, <li>, <sequence>, and <component>. The context enclosed by the <Bag> tag can have multiple elements. The <li> tags captures the context values and the order of appearance is unimportant, see example in Figure 4-10. In other words, the ordering presents no difference to the context description. In contrast, the order of context values under the <sequence> tag captures prioritised data. The first appearing element of context values holds the most significance, and the important of the following values decrease gradually. Finally, the <component> tag indicates compound context information. All elements of a context description under this tag are requested to complement each other and provide a compound context. Figure 4-10 presents an example of the XML-based format used to maintain profile information in the MID-B system.
Maintaining all available context information in a large centralised record is impractical. Therefore, discrete context information is maintained in separate profiles. The MID-B uses five different profiles. They include an application profile, a device profile, a user profile, an affected-interface profile, and a content-router profile. Possibly pre-defined profiles (as provided by MobiLife [147], including a context-broker profile and presentation-adaptation) are also used.

### 4.3.2.2.1 Application Profile

The application profile represents the characteristics of a mobile UI hosted on the UE_C. It can be visualised as shown in the Figure 4-11.
Chapter 4. The Multi Interface-Device Binding (MID-B) System

The application profile is based on the definition provided by the MobiLife [147]. Within the profile, two request types are enclosed. The first type inquires static information about the communication channel (input/output); this request type describes a method delivering a mobile media content, such as a conventional text, audio, video, etc. Inside each method, attributes about how to access the application are maintained. The attributes are grouped as ‘mandatory’ and ‘optional’ sets. The attributes in the mandatory set, i.e. source, source type and format, must be enclosed in every application profile. In principle, a value of the ‘source’ attribute can be either the ‘location’ where the content is stored or the ‘content’ itself. In the first case, the associated source-type attribute will indicate the kind of link pointing to the content location, e.g. URI and URL. Otherwise, the value of the source-type attribute in the second case is set to “Data”. The last attribute contained in the mandatory set is the format of the content. This suggests an extension of the file storing the content, such as .wav, .mp3, .html, and .xhtml. Attributes contained inside the optional set vary corresponding to the request ID. For example, given that the request ID is conventional text, a ‘character set’ will become an attribute in the profile. ‘System bit rate’ and ‘resolution’ are attributes for a video application.

As shown in the Figure 4-11, the right branch represents the second request type; that is it captures the multimedia settings. This request describes the ways to normalise the media delivery. This request contains

- a set of main parameters, which indicates the type of application (e.g. audio and video),
Chapter 4. The Multi Interface-Device Binding (MID-B) System

- a set of sub parameters, which holds the media elements that need to be adjusted for the particular type of application (e.g. volume for audio delivery and colour for video or still image delivery), and
- the value of the sub parameters (e.g. 4 and true).

### 4.3.2.2.2 Device Profile

The device profile is stored in UE_C and UE_I libraries. It offers a coherent description of the features of a manufactured device. General information of hardware characteristic, such as major and minor classes, CPU, vendor, and model, directly relates to all functions and capabilities of the device. Such information is maintained on the top of other attributes. In addition, the device profile consists of four distinct attribute sets containing information of the Modality Service Tree (MST), software characteristics, communication technologies, and dynamic-value attributes.

<table>
<thead>
<tr>
<th>Device Profile – General Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>(e.g. Device major and minor classes, CPU, vendor, device model)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attributes Relevant to the MST</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Interface and modality services</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attributes Relevant to the MST</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Interface and modality services</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attributes Relevant to Offered Communication Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Interface for exchanging a file, such as USB</td>
</tr>
<tr>
<td>• Wireless connectivity, such as Bluetooth</td>
</tr>
<tr>
<td>• Wired connectivity, such as LAN</td>
</tr>
<tr>
<td>• Cellular band and support, such as GSM 900</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attributes Relevant to Software Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>• System software, such as OS type, OS version, OS vendor and etc.</td>
</tr>
<tr>
<td>• Application software, such as Language, file viewer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attributes Relevant to Dynamic Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Battery</td>
</tr>
<tr>
<td>• Free memory</td>
</tr>
<tr>
<td>• Etc.</td>
</tr>
</tbody>
</table>

Figure 4-12: Device profile

The MST has been defined and is used to explain the attributes of provided interface and modality services at various levels of details and abstraction. Chapter 5 and Chapter 6 will thoroughly explain the principles and hierarchical structure of the MST. Next, the set of attributes relevant to software characteristics presents the embedded features enabling the device to execute a particular task. These features are considered as the system software and the application software. The system software includes, for example, OS type, OS version, Os vendor, and etc. The application software provides the functional features of the application. The attributes regarding
available communication technologies specify details of 1) interface for exchanging a file, 2) wireless connectivity, 3) wired connectivity, and 4) cellular support functions. A device can have one or several accessible means of communication. Yet, it is not necessary that they are activated concurrently. Finally, the device profile also documents attributes of which a value is changeable. It keeps updating the modification of dynamic attributes, for example remaining battery lifetime.

4.3.2.2.3 User Profile

The user profile hosts the context information of a user. As stated in Chapter 2, the end-user information encloses personal data and connection between other users. Consequently, the user profile implements two distinct categories of data. The first concerns the information of the user profile itself. And, the second involves the user’s social situation. A tree structure, shown in Figure 4-13, is used to capture the end-user information. The left branch of the tree shows four attributes that strongly influence the operational component handling an interaction adaptation (represented in italic). In a normal case, several devices are allowed to enlist in a set of personal devices. For example, the user may own a PDA and two mobile phones. However, at one time, only one device can act as UE_C. Furthermore, the user profile includes the set describing possible physical restrictions. This set primarily maintains the ordered-pair of disability types and affected sensing organs. Let \( P \) be the set of physical challenges.

\[
P = \sum(t, A) \tag{4.6}
\]

\( 't' \) represents a physical disability type, such as colour blindness, paralysis and partial paralysis. \( A = \{ a \mid a \in A \} \) is a finite set\(^2\) of affected sensing organs, such as eyes and hands. Inside the set ‘A’, a negative value is assigned to each element. The negative value indicates whether the focused (sensing) organ is completely or partially disabled. This assists in the interaction adaptation to prioritise the provided interface and modality services. The last two attributes sited on the left branch specify the sets of positive preferred input/output modalities. A static set of attributes of user preferences is explored. This makes the end-user’s instantaneous expectation understood in order to deliver an adequate interaction session.

\(^2\) A finite set is a set that has an exact number of elements.
Chapter 4. The Multi Interface-Device Binding (MID-B) System

User Context

User Profile

- Set of personal devices
- First name
- Last name
- Gender
- Home address
  - House number
  - Street
  - City
  - Country
  - Postcode
- Affiliation
  - House number
  - Street
  - City
  - Country
  - Postcode
- Email
- A set of jabbers
- Birthday
  - Date
  - Month
  - Year
- Set of native languages
- Set of physical challenges
  - Type (ex. color blindness and partial paralysis)
  - A set of affected (sensing) organs
    - Type of sensing organs (ex. eyes and hands)
  - Negative value
- Set of preferred input modalities
- Set of preferred output modalities

Social Relation

- Relation ID
  (e.g. Friend, kid and colleague)
  - First name
  - Last name
  - ID
  - Accuracy

Figure 4-13: End-user’s information

At the other side of the tree, the social relation is implemented based on the specification of a user context in the MobiLife [64]. Let S be the social relation.

$$S = \sum (r, P)$$

(4.7)

‘r’ corresponds to the kind of relations, such as friend, kid, relative, colleague. P = \{p | p \in P\} is a finite set of people grouped in this relation ‘r’. According to [64], general information of people is provided, such as first name, last name, ID, and ‘accuracy’ that is indicative of how precise the provided information is.
4.3.2.2.4 Affected-Interface Profile

In principle, the affected-interface profile suggests the relations between a human's physical sensing organ and a list of affected interface services. The profile maintains information as 4-tuple. Let $A$ be the profile of affected modalities.

$$A = (d, O, c, I) \quad (4.8)$$

'd' indicates by how interfaces are affected (negative or positive). $O = \{o \mid o \in O\}$ is a finite set of physical (sensing) organs, such as eyes and skin. 'c' indicates a human communication channel associated with the sensing organs. For instance, eyes associates with an input channel. $I = \{i \mid i \in I\}$ is a finite set of interfaces perhaps affected by any disability of the sensing organ $O$. In practice, this profile works in conjunction with the user profile. In the user profile, the end-user's physical abilities are captured. Together with the profile of the affected-interface, the interface adaptation is able to learn which underlying modalities will be affected and then derives a recommendation for these affected modalities.

![Figure 4-14: Learning for affected modalities](image)

**Figure 4-14: Learning for affected modalities**

4.3.2.2.5 Content-Router Profile

Although discoverability is limited by the network protocols, the data transmission between the UE_C and the UE_I is not limited. The MID-B system allows data/content to be sent either via an ad-hoc network or through an infrastructure network, such as WLAN or LAN. For the second case, the content-router profile is used. Such a profile retains a matrix of an UE_I and a router. The profile consists of router records which are uniquely identified by a router ID. Within each record, the router URL and a list of supported devices are provided.
4.3.2.2.6 Other Relevant Profiles

The profiles of context-broker and presentation-adaptation introduced by MobiLife [147, 148] are exploited in the MID-B system. The brief definitions of these profiles are provided in this section, the details are provide in [147, 148].

Context-broker profile

As specified in the thesis's scope, environment context, such as location and temperature, is categorised as an external (passive) context acquired by a sensor. This context information is maintained by the corresponding context brokers. In essence, the use of the context broker profile informs the interaction management about the precise place to discover an external context value.

Presentation-adaptation profile

The presentation adaptation profile represents the available adaptation agents that are responsible to transform a media format to fit to the available user interfaces. Each adaptation agent describes the possible types of converted modalities derived from the original modality. It consists of five following attributes:

- Type of the agent (e.g. Text-to-Speech (TTS))
- Original type of modality (e.g. a conventional text)
- A list of original modality formats (e.g. plain text, HTML and xHTML)
- Target type of modality (e.g. audio)
- A list of target modality formats (e.g. .mp3 and/or .wav)
4.3.2.3 Interaction Manager

The interaction manager, titled as the Multimodal Service Base (MSB), is the logical component which organises the data flows among assorted input/output modalities. In general, the MID-B system assists a mobile application to offer a multimodal-interface service, without implementing the complete support mechanisms within the application. Rather, the application uses the MSB for selection and information of chosen modality.

![Feedback on the selected modality](image)

In essence, the MSB is considered as the central part of MID-B; it implements three main tasks, as shown in Figure 4-17.

- **Internal-Context Management**
  - Task: Handle the device and user profiles.
    - Device profile
    - User profile

- **The Execution Function**
  - Tasks:
    - Reconfigure the structure of modalities
    - Combine the profiles
    - Context-router profile
    - Context-broker profile
    - Affected-modality profile

- **The Description of Interfaces And Modalities**
  - Tasks:
    - Process all contexts of application.
      - Adaptation profile
      - Presentation-adaptation profile
      - Embedded profile

![Core functions of the MSB](image)

- The internal-context management maintains the information of internal (active) context. As specified in Chapter 2, the category of internal context includes the context of a
platform (device) and a user. So, the internal context management handles the profiles of device(s) and the user.

- The description of interfaces and modalities processes all involving context of the application. It maintains the presentation adaptation profile as well as captures the functional interfaces and modalities from the application profile.

- The execution function integrates the central algorithms that implement the modality management. In order to make a final decision on the most appropriate interaction means, it consolidates and interprets the profiles of content routers, context brokers and affected modalities.

### 4.3.3 System Prototype

The MID-B prototype defines the interoperations of the components, which together shape a virtual-device environment. In the virtual-device environment, a running application (executed on the UE_C) could be rendered or accessed through the co-located UE_I(s).
4.3.3.1 Logical Prototype

The logical prototype of the MID-B is illustrated in the Figure 4-19. Although the MSB is located in the UE_C, the UE_C and the MSB are logically separated as physical-device domain and computing-element domain, respectively. Indeed, the process of interaction adaptation is mostly taking place at the MSB where all profile components are consolidated.

As stated in the previous section, the MSB contains the internal-context management and identifies where the profile of an end-user is forwarded to. Then, it interprets the information of user-system relation. The user-system relation refers to all critical impacts on system behaviour that are derived by the personal facts. The internal-context management also has the responsibility of maintaining the records of the coexisting devices. Profiles describing the UE_C and all UE_I(s) are forwarded into the internal-context management functions. In order to being computed, the on-demanded information of these internal context descriptions is sent to the execution function.

At the interface and modality description, the profile of a running application is obtained from the UE_C and the profile of presentation adaptations is maintained. By the cooperation of these profiles, a list of 'alternative' modalities is investigated. Unfortunately, this function does not yet include mechanisms to determine the most appropriate modality. Then the mapping of the original modality and the alternative modalities are forwarded to the execution function.

Figure 4-19: Logical prototype of the MID-B system
In the execution function, two modules are included, i.e. the multimodal dialogue management and the connectivity management. The multimodal dialogue management integrates the approach of interaction adaptivity. The entire adaptivity is divided into two mechanisms, as shown in Figure 4-20. These are the trading mechanism and the binding mechanism. Each mechanism individually deals with different context information. Due to the volatility of the configuration of a (virtual) user interface, the ‘trading mechanism’ monitors information of all devices and services that are currently in reach. Possibly, several user devices in the neighbourhood could offer equivalent services. Hence, the trading mechanism implicates a cooperating strategy to identify the precise service and device location. The strategy involves the ‘service search algorithm’ and the ‘theoretical descriptions of logical relation between different devices’. Initially, the service search algorithm exploits the Modality Service Tree (MST), which hierarchically describes the semantic dependencies of and between services within a single user device. The methodology described facilitates a tree-search of the means (interfaces/modalities) available for the user interaction. These are the interface services and modality services, together with their features and capabilities. Then, the theoretical descriptions of logical relations between different devices are included comprising the algebraic expressions that epitomise a semantic analysis of modalities hosted by two devices. The tactics followed by the pair-devices’ logical relation description limits the iterations required to evaluate the capabilities of devices located in the system.

The ‘binding mechanism’ processes and consolidates all context information with the recommendation derived during the trading sequence. The binding mechanism declares two ways to classify the context information.

- Classification based on changeability of the context values.
Chapter 4. The Multi Interface-Device Binding (MID-B) System

- The static context describes unalterable values. This may be a user’s physical challenges. The same information is used for all the service session and navigated once at the beginning of the binding mechanism.

- Temporal information for example dynamism of surrounding conditions, is captured in has variable values and requires the availability of trading mechanism.

• Classification based on how the context information is represented.
  - Numerical information
  - Non-numerical information

To make use of the consolidated context information, the MID-B system implements the ‘binding function’ which is responsible for arbitration of service session. Once the distributed resources (user interfaces) attached to the system, initial usage permission is assigned. Afterwards, the binding function applies its inference rules, based on historical interface usage, user preferences, and profiles, to prioritise the resources. The resource with the highest priority is most likely to control the interaction session. After the selection of modalities has been completed (and the suitable interfaces are bound into the dynamic multimodal user interface), the application’s content stream will be subsequently transformed, by the connectivity management. The connectivity management contains the content-router profile. It controls the transport media and method of transmission. Its main task is to facilitate dynamic and transparent communication between the UE_C and the UE_Is.

Figure 4-21: Sequence diagram of the MID-B prototype
4.3.3.2 The MID-B Finite State Machine

This section presents the logic of the prototype using a formal description that captures the MID-B’s functions at a high-level. The formal introduction to the MID-B prototype uses the theory of the Mealy Finite State Machine (Mealy FSM). According to the Mealy FSM, the sequences of the MID-B events are precisely and unambiguously modelled. After initiation, the system is idle. That is, the UE_C does not yet function; none of the UE_Is is recognised. From the service point of view, an interaction session is not yet triggered. Once the UE_C interacts with a service, the system is in alert state. The core task of the alert state is to identify an end-user. The relations between the end-user and the system and between the end-user and other relevant people are investigated. Next, the system announces itself by discovering coexisting devices (UE_Is). The system is always aware of instantaneously joining and leaving devices. Thus, the process of announcing the services is iterative. The information of devices currently in the system neighbourhood is updated. Subsequent to the system announcement, the UE_C gathers a list of services, together with attributes and features, from all available UE_Is. Literally, the use of services from the UE_C and the UE_I must be reasonable, so the system consults the MSB for device and modality selection. The discovered UE_Is negotiate whether they can be used for the service session and can be bound into a virtual-device environment. If a selected modality is on the UE_I, the system enters the states of ‘UE_I-established’. The physical connection between the UE_C and the UE_I is created. That is, a virtual-device environment embraces both of the UE_C and the UE_I. Alternatively, if the selected modality is hosted on the UE_C, the system goes into the state of ‘UE_C-established’. A logical link between a current modality and the target modality is generated. Under the circumstances, the created virtual-device environment encloses only the UE_C. Once all binding are completed, the user is able to use the service through the chosen (and connected) modalities and interfaces. The interaction session is arranged through the virtual-device environment. At any point, there could be any change in the execution context. The system listens for changes and re-negotiates new corresponding actions. For instance, when the UE_C-UE_I connection is unstable; the connection becomes terminated. Then, the system will search for the alternative interaction means. After the interaction session is finished, the session is closed and the system goes back to the initial point (Idle state).
The described functions can be composed as a Finite State Machine (FSM) as shown in Figure 4-22. In the Section 4.3.1, FSM (MID-B) = (S, S₀, I, O, F, G). S is a finite set of the internal states forming the entire MID-B system. The MID-B environment includes eight states, i.e. S = {idle, alert, service-announcing, service-gathering, device-negotiating, UE_C-established, UE_I-established, and interaction-performing}. Within the MID-B system, only one initial state is allowed. That is S₀ = {idle}. I is a finite set of the input event in the MID-B system. It controls the direction of the MID-B behaviour. In the Figure 4-22, the set of input events is represented by the second sentence, following ‘/’, of the labelled arrows, such as ‘The UE_C interacts with a system’ and ‘The selected modality is on the UE_I’. In the FSM tuple, the output set is characterised by ‘O’. The sentences prior to ‘/’ indicate a set of relevant outputs produced by the state. If the output derived from a state is symbolised by ‘-’, the state does not provide the corresponding output. The next constituent of the FSM’s tuple is ‘F’ which is a finite set of the state transitions within the MID-B system. This describes the relation mapping between the current state, the input event triggering the current state, and the next state, i.e. F: S × I → S. And finally, G is a finite set of the output relations within the MID-B system. This describes the relation between the current state and the expected output, i.e. G: S × I → O. Importantly, the state transitions and the output relations must be amenable to the model of the MID-B functions. They are represented by the
labelled arrows. As can be seen in the illustration of the FSM of the MID-B environment, the modelled functions of the MID-B are deterministic because there is only one initial state – the idle state – allowed in the system. Moreover, there can be unique transition and single output from a current state with a particular input event.

In the FSM, the manipulation of the UE_C-To/From-UE_I communication is split into two consecutive phases. Firstly, the MID-B system processes a style of the interface discovery & binding phase. And, the second is the phase of data transmission. In the phase of interface discovery & binding, “signalling stream” is transmitted. The phase operates the discovery protocol for ascertaining knowledge of modality and device connectivity. It maintains and implements the states of alert, service-announcement, service-gathering and device-negotiating. As specified by name, the data transmission phase conveys the “data stream”. This data stream is the actual user content for an application from/to a connected interface device. With non-restricted device connectivity, the media stream (i.e. video) may be re-routed to public displays via wired networks that offer more real time guarantees. The data transmission phase involves the states of UE_I-established, UE_C-established and the interaction-performance.

Figure 4-23: Discovery and binding phase and data transmission phase
<table>
<thead>
<tr>
<th>System State</th>
<th>Description</th>
<th>Involved Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>The idle state</td>
<td>Service session is not yet started.</td>
<td>None of components is involved.</td>
</tr>
<tr>
<td>The alert state</td>
<td>The service session is begun. The relations between the end-user and the system and between the end-user and other relevant people are identified.</td>
<td>UE_C, User profile and MSB</td>
</tr>
<tr>
<td>The service-announcing state</td>
<td>The UE_C discovers coexisting devices</td>
<td>UE_C and UE_I</td>
</tr>
<tr>
<td>The service-gathering state</td>
<td>The UE_C collects the service information of the coexisting devices.</td>
<td>UE_C, UE_I and device profile.</td>
</tr>
<tr>
<td>The device-negotiating state</td>
<td>Modality selection is operated at the MSB. Fundamentally, the selection is associated with all the profile information.</td>
<td>MSB and all profile components.</td>
</tr>
<tr>
<td>The UE_I-established state</td>
<td>A virtual-device environment is created. The environment encloses the UE_C and the UE_I.</td>
<td>UE_C and UE_I</td>
</tr>
<tr>
<td>The UE_C-established state</td>
<td>A virtual-device environment is created. The environment encloses only the UE_C.</td>
<td>UE_C</td>
</tr>
<tr>
<td>The interaction-performing state</td>
<td>The service session is arranged through the virtual-device environment.</td>
<td>UE_C and/or UE_I</td>
</tr>
</tbody>
</table>
4.3.3.3 UE_C and UE_I Finite State Machines

The virtual-device environment of the MID-B is based on the dynamic use and collaboration of coexisting interface devices. This section presents the FSMs that capture the actions of a UE_C and a UE_I. Practically, the event-driven operations between the UE_C and the UE_I provide the internal shared states of the entire system. That is, a set of states and system transitions assigned for the entire system’s FSM are corresponding to the task-and-action-mapping between the UE_C and UE_I.

Modelling the actions of the UE_C

![Figure 4-24: FSM of the UE_C](image)

The UE_C’s FSM is shown in Figure 4-24. Firstly, the UE_C is in an idle state. At this state, the UE_C profile is documented. The UE_C starts the service session with a process of device discovery. Iteratively, the device discovery mechanism recurs in a fixed-time frame. Principally, the mechanism sends an announcement of the UE_C. In return, the UE_C will receive acknowledgements from all neighbouring devices. Once an acknowledgement is received, the internal state, entitled the ‘address-check’ state, is required to determine whether the device is re-discovered or new. In practice, the MID-B system implements the ‘availability tag’ for each entry.
of the discovered-device database. The tag is modified every time the device is temporarily removed or re-discovered in order to indicate whether the device is currently located around the UE_C. The address-check state holds the core responsibility to explore the discovered-device database and to examine the tag. If the device is unknown, the UE_C enters into a state of service discovery to query the information of interface and modality provided by the discovered device. Otherwise, the device was discovered previously, but temporarily removed from the environment. In case, the device’s detail have been available in the database of discovered devices, the state of service discovery is not necessary. The concept of changing the tag’s value benefits the UE_C not to re-enter the state of service discovery. Subsequent to either the states of address check or service discovery, depending on the above described condition, the UE_C forwards the profiles of the device and application to the MSB. The UE_C then wait until the MSB finishes calculating the selected interaction means. As soon as the information of the selected interaction means is received, the UE_C is bound into a virtual-device environment. The UE_C may make a logical connection between modalities hosted by the UE_C itself. Or else, the modality between the UE_C and the UE_I is physically linked. In a faulty case, if the states of establishment fail, the UE_C’s action returns to the wait state. It returns its physical resources and waits until new selected interaction means are restored. Otherwise, for a normal case, the UE_C enters into the user/interface state where the actual interaction is performed. In this state, the UE_C becomes a user access device controlling other devices. Although the functioning modality may not be hosted on the UE_C, the UE_C is still up for providing the content data stream. During the interaction service is given, an occurring fault may interrupt the service. In this case, the UE_C steps back to the wait state. It then returns its physical resources, and listens to a new command. After the service session is finished, the session is closed and the UE_C goes back to the idle state.

Modelling the actions of the UE_I

The UE_I’s FSM is shown in Figure 4-25. As usual, the initial state of the UE_I is the idle state. Dissimilar to the idle state of the UE_C where the UE_C profile is documented and delivered, the UE_I is not aware of the MID-B service yet. None of functionality is processed. In other words, the UE_I is completely outside the MID-B environment. When the UE_I receives an announcement from the UE_C, it enters the device discovery state. It now recognises the MID-B session, and sends back an acknowledgement. Within a fixed-period of time, if a request for service information is obtained, the UE_I generates and transmits the corresponding response. Otherwise, the UE_I waits for a command. In fact, the internal performances at the states of device and service discoveries are in an opposite direction compared with those of the UE_C. In principle, the UE_C functions as a ‘the master’ device requesting the UE_I’s information, whereas the UE_I becomes a ‘slave’ device responding to the request. Prior to being integrated
into the MID-B environment, the UE_I is in hibernation. At the hibernation state, the UE_I is integrated to the MID-B system, but not functioning. At this state, it does not provide any service. Literally, the wait state of the UE_C and the hibernation state of the UE_I, both, force the devices' actions to be suspended. To re-triger the action, the service instruction computed by the MSB is expected. Although these two states share a common concept, their internal executions are different. The UE_C’s wait state happens because the MID-B session has failed, while the hibernation state takes place since none of its modalities is used in the present service session. The UE_I will leave the hibernation state once at least one of its modalities is selected for a service session. Subsequent to the hibernation state, the UE_I enters an ‘establishment’ state where it becomes physically bound with the UE_C. It then becomes a member of the virtual-device environment. At this point, the UE_I begins the user/interface state and continues to operate the concrete user interface service. Comparing to the UE_C, the UE_I’s user/interface state is reached differently. Independent if and how the UE_C’s modality is exploited, the UE_C must enter to the user/interface state for providing the data content service. In contrast, the UE_I reaches the user/interface state ‘if and only if’ at least one of its modalities is functioning. During operating the service, there could be any reason forcing the physical resource of interfaces and modalities to be retired. The UE_I’s action is suspended. So, the UE_I will revisit the hibernation state. At the moment, the UE_I physical resource is dormant (not occupied). However, the physical resource can be re-enabled at any time. The interval over of the hibernation state is determined by when the modality on the UE_I will be re-selected. So, the UE_I resumes to the service session. After the service session is successfully finished, the session is closed and the UE_I goes back to the idle state. Yet, the UE_I can leave the service vicinity during any system state. If this happen, the UE_I will return to the idle state.
Chapter 4. The Multi Interface-Device Binding (MID-B) System

4.4 Summary

In short, the MID-B extends the conventional multimodal interaction by introducing the multiple-device and single-user scenario. It provides the mechanism for self-adaptive interface binding. The system allows awareness of arbitrary changes of distributed interface devices. Subsequently, it provides a flexible solution to respond to such changes. The system is developed following the steps of idealised waterfall model. These are 1) a requirements analysis, 2) a preliminary design, 3) a detailed design, and 4) an integration and test. The steps of the requirements analysis and the preliminary design are presented in this chapter. During the step of requirement analysis, the potential problems and the possibilities of the MID-B behaviours are thoroughly analysed and examined. This chapter presents a concept of 'levelled' requirements that elicit the MID-B requirements in different levels of attention, i.e. the system level, the functional level, and the implementation level. At the system level, the goal-oriented requirements are captured. Overall, the system requires the methodologies for adjusting the internal behaviours of the system and dynamically deploying the system configuration. Next, the functional level specifies the

Figure 4-25: FSM of the UE_1
functionalities necessary for MID-B execution. These are service discovery, context-awareness, interface adaptation, and presentation adaptation & connectivity management. Finally, the implementation-oriented requirements for these required functionalities are exposed.

Further, MID-B's preliminary design is provided. In fact, the system is defined as a set of components that collaboratively implement the MID-B functions. In the MID-B system, the architectural component are categorised as following.

- The set of terminals, characterising physical devices encapsulated in the virtual-device environment. The system defines the UE_C as a referencing master device, and the UE_I as a slave device.
- The set of profiles, characterising databases of context information used in the MID-B system. Principally, they involve information of a user, an application, devices, affected interface, a context broker, a content router, and a presentation adapter.
- The interaction manager, characterising the middleware agent controlling the configuration of the virtual-device environment. It encloses the fundamental tasks to enable and disable the modalities on the involving physical devices.

The behaviour of the collaborative components affects the overall system. This chapter also provides a logical prototype to describe how these components are incorporated. In addition, the MID-B system is formally analysed in terms of events, status and actions. The Mealy FSM is employed for such an analysis. The key concept of the FSM is reviewed. Indeed, the state space of the MID-B is implemented on top of the event-driven operations of the embedded physical devices. The end of this chapter, the FSMs of the UE_C and the UE_I are presented. While this chapter presents the overall MID-B, more details of required functionalities are provided in the following chapters. The next chapter describes the mechanism for service discovery defined for MID-B.
5 Device and Service Discovery

5.1 Introduction

Following the principles of the ubiquitous and mobile computing, MID-B uses available devices to form a virtual-device in which several interface devices are cooperated and federated. Implementing a configuration of this virtual-device environment requires the capabilities to discover and control the access and use of user interfaces that are hosted on distributed devices. The aim of this chapter is to explain the mechanisms and implications of the service discovery function within MID-B. Considering the complexity of distributed environments, this constitutes a core task to enable the UE_C to identify distributed UE_Is. The UE_C can then find UE_I's interface and modality services to bind them into the virtual device.

This chapter describes the approach of device and service discovery defined for MID-B, it is organised as follows. Section 5.2 describes the three-party model, in MID-B; the underlying prototype of the service discovery is defined following a three-party model. This model is an extension of the Bluetooth SDP architecture. The UE_C, UE_I and MSB are integrated in this model. Section 5.3 gives an introduction to the discovery process of MID-B. Depending on the types of networks, discovery is implemented in two groups, one is IP-based and the other is Bluetooth-based. Section 5.4 explains the mechanisms of IP-based discovery, while Section 5.5 provides the details of the Bluetooth-based service and modality discovery. Finally, Section 5.6 provides the summary of the chapter.

5.2 Three-Party Model

5.2.1 Architectural Design

In general, a discovery mechanism locates the range of resources and services within a limited physical environment. Extending this general mechanism, MID-B introduces a “three-party model”, which expands the existing discovery mechanisms with features to retrieve and evaluate modality characteristics. Its architectural implementation is the extension of the Service Discovery Protocol (SDP) transaction of Bluetooth. The SDP transaction implements its service
discovery following two-party model (client-server architecture) using unicast communication. A complete transaction is made up of a service request PDU,\(^3\) sent by a client, and a matching response PDU, generated by a server. An introduction to the SDP transaction is provided in Chapter 3.

\[
\text{Client (UE}_C\text{) } \quad \text{Unicast communication (client/server scenario)} \quad \text{UE}_I \text{(Server)}
\]

\[
\text{Logical dependencies of discovery interfaces and modalities}
\]

\[
\text{Interaction Management (Multimodal Service Base, MSB)}
\]

**Figure 5-1: Three-party model**

**Figure 5-2: Registration process of the three-party model**

In a volatile environment, a server can leave a client's neighbourhood at anytime. Once a server re-enters to the neighbourhood, the request PDU must be issued to rediscover the server's service offerings. This results in an increment of unessential network traffic. The simple client-server based architecture is not scalable. To reduce traffic, MID-B introduces a three-party model, which adds an interaction management as a third component. The interaction management hosts information on the logical dependencies between a client and server. Also, it hosts a registry of all discovered servers. Each entry of the registry implements an 'availability tag' used to indicate

\(^3\) Protocol Data Unit
whether an associated server is still within service vicinity. The tag of all discovered servers is set to ‘true’. Given that the server temporary withdraws itself from the client’s area, the interaction management will change the value of the tag to ‘false’. Once the server returns, the tag will be set back to ‘true’. The service information of the returning server is then retrieved from the registry. Thus, a re-evaluation of a server’s services is not required.

5.2.2 Three-Party Model applied in MID-B

Figure 5-3: General use case of the three-party model

Figure 5-3 presents the general use case of all the components required in the three-party model. In relation to the components defined in MID-B, the three-party model is composed of the ‘private interface device’ (UE_C), the ‘public interface device(s)’ (UE_I(s)), and the ‘interaction management’ (MSB). The mechanism is fairly straightforward. Once initialising, the UE_C acts as a client. In general, it is a resource-limited device interested in using interface and modality services offered by other co-located UE_I(s). The UE_I is a networked resource acting as a server. A user, carrying a UE_C, moves into a new virtual-device environment in where a number of UE_I(s) are located. The MID-B proposes a protocol for unicast communication between the UE_C and the UE_I. This protocol implements a range of message exchanges comprehensible by every device situated within the virtual-device environment. Such a case, the UE_C issues a message asking for the service information, i.e. a service request PDU. On receipt of the request PDU, the UE_I replies with a service response PDU. Subsequently, all information captured by the response PDU is unified and stored in the MSB. In the MSB, a basic management of modalities is
implemented. The main responsibilities of the MSB are to control the availability tag and to manage the dependencies between the discovered interface devices and modalities. In return, the application retrieves (from the MSB) information on modality available within the vicinity. The UE_C then binds the chosen UE Js and their modalities and reroutes the modality streams accordingly. The here described mechanism is illustrated as a sequence diagram shown in Figure 5-4.

![Sequence Diagram](image)

**Figure 5-4: Discovery, registration and modality-stream delivery in the three-party model**

### 5.3 Discovery in MID-B

The three-party model defines the discovery protocol with the capability of bridging the gap caused by the divergence of UE_I platforms, using a meta-information services. All devices, regardless of their actual implementation specifics, are able to comprehend the representation description of services from other devices. In MID-B, the term ‘service’ captures the entities supporting the operation of an interaction. The term entity directly corresponds to the hardware-based and the software-based services. Generally, hardware-based entities are characterised by concrete devices, including their embedded physical interface equipments. On the other hand, the software-based entity focuses on a widespread set of interfaces and modalities. Because the meaning of entity is twofold, the communication between the UE_C and the UE_I has a double-layered architecture that is built on the definition of a ‘basic’ protocol for device discovery in combination with an ‘extensive’ service discovery protocol.
In principle, device discovery senses the dynamical and arbitrary availability of resources. In MID-B, the mechanism for device discovery is called ‘User Equipment Discovery’ (UED). To prepare a system for any possibly required interface adaptation, UED tracks any changes of states and availabilities of public interface devices, i.e. UE_I(s). UED facilitates ad-hoc service discovery. It allows an UE_C to identify UE_Is and to physically establish a connection needed for implementing a supporting router. The implementation of UED concentrates on a small-sized local network. Allowing manipulation in the real world, the implementation exploits existing technologies. Two distinct scenarios are applied, i.e. the Bluetooth-based UED and the IP-based UED. The Bluetooth-based implementation uses the Baseband layer of the Bluetooth stack as access control method. Devices integrated in the system must be Bluetooth enabled, which together with the specified protocol allows a device to automatically register and deregister from MID-B.

Although the introduced three-party model is an extension to the SDP transaction, UED allows some freedom for the choice of the actual transport. Due to the fact that IP-based technologies are broadly used in current communications, the second scenario of the UED uses IP-based discovery. Rather than only using the specific connection media of Bluetooth, the IP-based UED makes UE_C and UE_I(s) independent of the Bluetooth stack. Bridging the gap in a transparent manner, discovery is not confined to a specific physical layer protocol. The IP-based implementation eases the integration of IP network technologies, such as IEEE 802.3 (Ethernet) and IEEE 802.11 (Wireless LAN), into a MID-B environment. Regardless of a Baseband of Bluetooth stack, the IP-based UED employs a transport layer of classic protocol stack to convey a set of discovery signals. The IP address is used as an identifier for the devices.

After completion of UED, service discovery announces the available modalities. Basically, the service discovery mechanism of MID-B accommodates the underlying SDP (i.e. the Service Discovery Protocol of Bluetooth). Yet, the conventional transaction presented by the SDP is not fully enabled to discover the details of modality capabilities and features. Hence, MID-B introduces an extension for a Multimodality transaction. The combination of the conventional SDP transaction and the Multimodality transaction extension (MM) facilitates the investigations of general-service capabilities and modality-service descriptions, respectively. Similar to UED, the mechanism for service discovery is implemented in two scenarios, i.e. Bluetooth-based and IP-based. In practice, the implementation of service discovery abides by the corresponding UED implementation. Meaning, the Bluetooth-based service discovery always occurs subsequent to the Bluetooth-based UED. In contrast, the IP-based service discovery follows the IP-based UED. Figure 5-5 shows the phases of multimodal service delivering in MID-B. As normal, a service discovery is employed after UED, but before establishing a connection.
5.4 IP-Based Discovery

5.4.1 Implementation Stack of IP-Based Discovery

Figure 5-6: Implementation stack of the IP-based discovery
Figure 5-6 illustrates an implementation of the IP-based discovery. It consists of two parts, i.e. a ‘single layer for physical media’ and a ‘double layers for the implementation stack’. The layer for physical media, simply called as ‘physical layer’, defines a specification of physical connections of an interface device. Actually, the MID-B system makes use of existing data links defined in standardised communications. Supporting the IP-based discovery, IEEE 802.3 (Ethernet) and IEEE 802.11 (Wireless LAN) are used as examples for physical connections.

The implementation stack above the physical layer is an aggregation of two control layers; each of which possesses specific rules and control information to process the required functions and tasks. As a whole, the implementation stack hosts a signalling & control layer and an application control layer. The signalling and control layer is responsible to transmit the control messages and signals used during discovery. At this layer, a number of paths are established. These paths convey UED’s signals, SDP PDUs, and modality information. Next, the application control layer involves all the programming of a MID-B application. Because the IP network is not ad-hoc by nature, the application layer is enabled to introduce ad-hoc behaviour to an environment. Between the physical layer and the actual implementation stack, the interface component is developed to join these two parts. The interface component obtains and forwards all control information, such as the networked devices’ addresses and the generated discovery signals, to/from the implementation stack from/to the physical layer.

Figure 5-7: Comparing the TCP/IP suite and the implementation stack of the IP-based discovery
Obviously, the implementation of IP-based discovery is constructed with reference to the classic protocol suite of the TCP/IP stack\(^4\). Figure 5-7 presents a comparison between the TCP/IP suite against the introduced implementation stack. Also, the comparison is shown against the reference model of the 7-layer OSI stack\(^5\). A ‘network access’ of the TCP/IP suite and the ‘physical layer’ of the implementation stack realise IP-based networks. The ‘interface component’ takes action in partial functionality of the ‘Internet Layer’ of the TCP/IP suite. The tasks of the interface component are to obtain a logical host identifier (i.e. an IP address) of network devices, and to pass it to the discovery signals. The ‘signalling and control’ layer uses sockets and ports, which are implemented in the ‘transport layer’ of the TCP/IP stack, to establish the path over UE_C and UE_I. Similar to the ‘application layer’ of the TCP/IP, the ‘application control layer’ handles the user interactions. It also implements a socket API to serve the communication in the signalling and control layer. The following section presents the implementation approach.

5.4.2 Detailed Implementation Stack of IP-Based Discovery

The implementation stack relates to different phases of multimodal service delivery in MID-B (see Figure 5-5). Combining Figure 5-5 and Figure 5-6, Figure 5-8 gives the illustration of the extensive implementation stack for IP-based discovery. The implementation stack has four phases, i.e. UED, SDP transaction, MM transaction, and data transmission.

\(^4\) TCP/IP stands for Transmission Control Protocol/Internet Protocol. It is the suite of communications protocols used to connect hosts on the Internet.

\(^5\) The 7-layer OSI is a layered, abstract description for communications and computer network protocol design.
The UED phase

UED is a semi-structural discovery which applies the principle of un-structural discovery in combination with the concept of structural discovery. In relation to the un-structural discovery, UED exploits the fact that a UE_C generally has no knowledge about the number of UE_Is and the UE_Is do not have the route information to the UE_C. Thus, the total number of threads sending an announcement signal cannot be known. To make a communication request, the UE_C has to announce itself by flooding the network with an announcement signal containing an improvised plan. In practice, the announcement signal is encapsulated in a datagram called the ‘UE_C-Announcing Datagram’ (UAD). To establish a route to transmit UAD, the signalling and control layer instantiates the communicating socket and port. Technically, all devices must have an agreement on an explicit port number. For example, port number ‘6666’ is used for the UED session. UE_C publishes the UAD via port 6666. At UE_I side, the port 6666 handles the UAD reception. However, the pure un-structural discovery, i.e. the flooding solution, consumes heavy resources and increases the signalling overhead traffics. Supporting more efficient discovery, a scheme using structural addresses is combined into the session. The structural address scheme
borrows a concept from multicast transmission. For instance, The MID-B group address '239.1.2.4' is assigned. Multicast address is specified in [149]. All UE_Is must implement the defined group address. After being flooded by UE_C, the UAD will be correctly delivered to only those devices included in the group. Upon the receipt of UAD, UE_I(s) is aware of a MID-B service and has knowledge of the UE_C’s IP address. Subsequently, UE_I(s) generates an acknowledgment which includes their address information. The UE_I then makes a unicast communication via the port 6666 for sending the acknowledgment back to the UE_C.

During the UED session, UE_C periodically announces UAD over the group. Because UED is periodical, there could be the same UE_I(s) rediscovered. As defined in the three-party model, the knowledge of the rediscovered UE_I(s) is managed by implementing the 'availability tag' in the registry database. At the application control layer, the registry is constructed and organised as a hash table. UE_C uses the hash table and the tag to perform following optional phase.

- If UE_I has not existed in the hash table (i.e. it is a new discovered device), process the phases of the SDP and MM transactions.
- If UE_I has existed in the hash table and the availability tag is false (i.e. UE_I reattaches to a service area), reset the tag to true and skip the phases of the SDP transaction and the MM transaction.
- If UE_I has existed in the hash table and the availability tag is true (i.e. the UE_I has still been available), go to the data transmission phase.

The phase of the SDP transaction

Unlike UED, which is a flooding-and-multicast-based communication, the SDP transaction phase is node-by-node based. The unicast routing between the UE_C and the UE_I is established. This phase utilises the SDP PDUs as the message bearer. Because the classic TCP/IP communication does not include any functionality dealing with the SDP PDUs, the main interest of this phase is to provide UE_C and UE_I the abilities of managing, generating, receiving, and comprehending the SDP PDUs. The implementation of such functions takes place at the application control layer. The signalling and control layer is responsible to transmit the PDUs via the established port. Regarding the standard SDP, to minimise the duration of service discovery, the ServiceSearchAttribute transaction of Bluetooth SDP is used. The PDUs exchanged in the transaction is described in Chapter 3.

Multicast refers to the delivery of information to a group of destinations. It is the communication between a single sender and multiple receivers on a network.
Chapter 5. Device and Service Discovery

The phase of MM transaction

The extension of the Multimodality transaction (MM transaction) is taking place following the SDP transaction. Similarly, it performs its unicast transmission via a specifically assigned port. The MM transaction delivers the information that is needed for determining on which interface device is suitable for an application/service. Unlike the SDP transaction, the MM transaction uses one way communication (from UE_I to UE_C only). UE_C does not issue a request. Rather, UE_I sends the interface and modality information straightaway after the SDP transaction finishes. In contrast to the SDP response, the modality information is not encapsulated in a PUD. But, it is formed as an object that accommodates an abstract syntax tree. The syntax tree provides an extensive script about modality and interface description. It has formed part of this work and is called Modality Service Tree (MST).

Because a device generally provides multiple layers of details about services, the MST captures context information of devices using a three-tier hierarchy. The generic MST is illustrated in Figure 5-9.

![Generic Modality Service Tree (MST)](image)

Figure 5-9: Generic Modality Service Tree (MST)

At the vertical dimension, four categories of service layers are defined. The top layer identifies a device; it indicates whether the device is either a UE_C or a UE_I. The set of embedded physical interface equipments are described in the second layer. Subsequently, the third layer describes a set of interface services provided by the physical interface equipments. In reality, within a mobile multimodal scenario, one interface service can be offered by more than one physical interface equipment. In a case, the Un-Coordinated Interface Service Operator (UCO) and Synchronising
Interface Service Operator (SIO) are employed. Accessing and rendering the interface service with the UCO ‘may’ take place through several physical interface equipments, which work independently. With the SIO, the interface service is simultaneously controlled by different physical interface equipments. The service will operate only if all the required equipments are available. Next, at the lowest layer of the MST, the precise information of modality services related to a particular interface service is presented.

Two neighbouring layers in the vertical dimension are linked by a labelled relation, symbolised by a directional arrow. The relation details the direction of the media (modality) flow between elements. Providing the arrowhead points to a focused modality, the involving elements provide an ‘input’ channel. That is, data flows to that element to be executed and interpreted. If the arrow points away, the flow direction is ‘output’. The output channel indicates the element contributes data to a user. The arrow with double heads indicates that an elements hosts both ‘input and output’ channels.

The horizontal dimension describes the type of services provided in each layer. For instance, at the second layer, a set of physical interface equipments could include screen, loudspeaker, handset and keypad.

The depth dimension provides knowledge on ‘how’ the services implement an interaction. Using the depth dimension provides information about the capabilities and limitations of modality services.

**The phase of data transmission**

Main tasks of the phase of data transmission are to carry out the interface and modality binding mechanism and to transmit a modality stream from a UE_C to a UE_I. This phase occurs after the discovery has finished. In most case, the data transmission phase involves the application control layer. The mechanism and implementation of this phase will be detailed in the next chapter.

**5.4.3 Proof-Of-Concept**

Due to a number of advantages provided by the design and programming features of Java, the proof-of-concept is implemented based on Java. The nature of Java programming is object-oriented; this, together with the virtual machine, promises portability and platform independence.

In MID-B, one of the most compelling reasons to choose java is its networking capability; it supports the common set of network programming APIs. In java, ‘package’ and ‘library’ are similar in a way that they group components. However, their definitions are slightly different. While the packages collect related classes and objects, the library combines a set of other packages or smaller libraries. In practice, organising MID-B into libraries and packages allows a structured implementation and system definition.
In relation to the three-party model introduced in Section 5.2, MID-B implements three independent libraries carrying out the functionalities of UE_I, UE_C and MSB. While the use case diagram shown in Figure 5-4 illustrates the general requirements of these functionalities, their implementations are described in this section. Figure 5-10 presents a general guideline for IP-based discovery in MID-B. Each framework is separately implemented in distinct libraries. Based on the client/server scheme, ‘ue_c’ library and ‘ue_i’ library integrate all packages and classes facilitating the behaviours of a client and a server, respectively. Next, ‘msb’ library supports the functionality of the MSB component. Focusing on the service discovery, the MSB component implements the device registry. The MSB is in fact defined as a logical software-based component located inside a private device (UE_C); hence, msb library is logically merged into ue_c library. In addition, the ‘utility’ library is implemented to supply the supporting utilities required for the IP-based discovery mechanism. This library is imported into the aforementioned three libraries.

5.4.3.1 Service Discovery in the ‘ue_c’ Library

Figure 5-11 illustrates the high-level diagram for the IP-based discovery on UE_C side. The diagram depicts the breakdown of packages and shows the dependencies among them. In the ‘ue_c’ library, a core class of the system is accommodated, i.e. ‘UE_C’ class. The UE_C class is the only class interacting with a user. It provides a common UI that allow a user to exploit and control services offered by MID-B. UE_C class realises a ‘discovery’ library which is formed...
Chapter 5. Device and Service Discovery

according to the extended stack described in the previous section. To accomplish the phases of UED, SDP transaction and MM transaction, the ‘discovery’ library is nested into the three principal packages, i.e. ‘multicast’, ‘serviceSearchAttributeTransaction’, and ‘mstTransaction’.

Figure 5-11: Class and packages in ‘ue_c’ library (scope on service discovery)

Figure 5-12: Packages in ‘discovery’ library
The ‘multicast’ package

As shown in Figure 5-13, ‘multicast’ package contains three classes, i.e. ‘UECmulticast’, ‘AcknowledgmentReceiver’, and ‘AcknowledgmentAddress’. Declared as a public class, UECmulticast is called by the UE_C class. Inside UECmulticast, the discoverDevice() function announces the appearance of a UE_C and it returns a set of the currently discovered UE_I(s). In practice, discoverDevice() initiates the communication socket and port. It then generates and floods the UAD. To receive the UE_I acknowledgement(s), the AcknowledgmentReceiver class is called. This class implements multithreads for simultaneously obtaining the message(s) of acknowledgements from different UE_Is. Finally, AcknowledgmentAddress is instantiated as a composite object, containing the host address and the IP address.

Figure 5-13: Classes in ‘multicast’ package

Figure 5-14: AcknowledgementAddress object
Assembling all related classes in a package facilitates a level of visibility and accessibility. In the multicast package, all attributes (both variables and constants) are declared as either private or protected. They become invisible to other non-related classes. Accessing the value of these attributes is done by the 'get' operation, which must be declared as public. In general, the get operation grants permission to obtain the attribute value, but not to modify it. For instance, the 'knowledgeVector' attribute is declared in the 'AcknowledgmentReceiver' class as a private attribute. Asking for its value, any off-package class must call the 'getKnowledgeVector()' operation. Literally, the 'get' operation will be used though the rest of the thesis.

The 'serviceSearchAttributeTransaction' package

Importantly, the IP-based discovery has to guarantee that devices (UE_C and UE_I(s)) can emulate the SDP ServiceSearchAttribute transaction. The tasks creating, transmitting, and receiving the SDP PDUs are implemented in the 'serviceSearchAttributeTransaction' package. Figure 5-15 shows a number of smaller implemented sub-packages contained in this package. There are 'dataElementDescriptor', 'request', and 'response'. Since the serviceSearchAttributeTransaction package mostly involves other smaller packages, it can be (technically) considered as a library.

Figure 5-16 shows classes implemented in ‘request’ package. As suggested by its name, the ‘request’ package deals with generating and transmitting a SDP request PDU. A set of utilities from the ‘utility’ library is imported. In ‘RequestPDU’ class, ‘createRequestPDU()’ operation controls the creation of a SDP request PDU. This operation takes the transaction ID of the latest
generated PDU as parameter; it then produces a new PDU increasing the transaction ID. ‘RequestPDU’ class associates with ‘SDPRequestParameter’ and ‘SDPHeader’. The composite aggregation, symbolised by a connection with a symbol ‘♦’, is used to depict that the request PDU is made of a header and a set of request parameters. The ‘<thread>> Transmitter’ class implements multithreads that simultaneously send the generated SDP request PDU to discovered UE_I(s).

In ‘dataElementDescriptor’ package, Figure 5-17, ‘UE_CSizeIndex’ class implements a size descriptor of a SDP PDU. The size descriptor has been presented in Chapter 3. This class provides the mechanism for conversions from a size index to an actual data size in byte, and vice versa.

Figure 5-16: Classed in ‘request’ package

Figure 5-17: Class in ‘dataElementDescriptor’ package

Figure 5-18 shows the classed implemented in ‘response’ package. ‘UecPDUReceiver’ provides threads handling the process to receive a SDP response PDU(s). On receipt of the response PDU, the ‘PDUResponseInterpreter’ class provides a callback to extract the message of that response.
PDU. 'BluetoothProfile' implements another multithread to learn the information of the Bluetooth profiles; this information is contained in the parameter of the response PDU. This class subsequently organizes that information into a file.

Figure 5-18: Classes in 'response' package

The 'mstTransaction' package

Figure 5-19: Class in 'mstTransaction' package
The ‘mstTransaction’ package is a single-class package. ‘DeviceDescriptionReceiver’ implements the multiple threads to control the receiving of the MST object from different UE_Is.

5.4.3.2 Service Discovery in the ‘ue_i’ Library

In the ‘ue_i’ library, a number of packages corresponding to UE_I routines are implemented, shown in Figure 5-20. Compared to the ue_c library, ue_i library is less complex. It contains a ‘UE_I’ class as the main class. Enabling public interface devices to connect to a MID-B environment, the UE_I class imports the ‘mid_bSetup’ library and the ‘mstUtil’ package.

The ‘mid_bSetup’ library

Figure 5-21 documents dependability of the ‘mid_bSetup’ library. This library contains two classes i.e. ‘UelServiceSetupSession’ and ‘UelAcknowledge’. The ‘UelServiceSetupSession’ class implements a non-return-value operation, called ‘setup()’, to handle the UED phase. This operation is responsible to join a UE_I to the MID-B group address. The operation returns an acknowledgement to a corresponding UE_C. To efficiently transmitting the acknowledgement, setup() calls a thread from ‘UelAcknowledge’.

In addition, mid_bSetup library encompasses a smaller library called ‘serviceSearchAttribute’. In serviceSearchAttribute, the SDP transaction is managed. Three packages are enclosed in this library.
Figure 5-21: Classes and packages in ‘mid_bSetup’ library

The ‘requestReceiver’ package, illustrated in Figure 5-22, contains a number of classes responsible to receive a SDP request PDU, and to extract the information gained from the parameters of the request PDU. Second, a ‘response’ package hosts classes to generate and to transmit the corresponding response PDU(s). The dependencies of these classes are illustrated in Figure 5-23. Basically, the response package is constructed following the standard definition of the SDP response PDU. The maximum number of bytes of the attribute-list parameter in each response PDU must not exceed the limit specified in the MaximumsAttributeByteCount parameter of request PDU. The excess information must be continued in a the second PDU (see explanation and example in Appendix A2.). Accordingly, in a single SDP transaction, one request PDU could correspond to more than one response PDUs. In a case, the ‘AttributeTracking’ class declares ‘overLimit’ to indicate whether the transaction needs the continuing response PDU(s). If yes, AttributeTracking also maintains the information of the service record ID and attribute ID used in the latest PDU. The actual content of the attributes list in the responds parameters is generally prefixed by the element descriptor, ‘dataElement’ package, shown in Figure 5-24, provides the facilities to generate the element-descriptor.
Figure 5-22: Classes in ‘requestReceiver’ package

Figure 5-23: Classes in ‘response’ package
Chapter 5. Device and Service Discovery

5.4.3.3 Service Discovery in the ‘msb’ Library

To fulfil the three-party model mechanism, an ‘msb’ library is generated. Note that, the msb library is actually multifaceted. It engages in, for example, service discovery, profile management, and interface binding. However, classes and packages that are unrelated to the service discovery mechanism will be left out in this chapter; they are provided in the following chapter.

Focusing on the service discovery mechanism, ‘msb’ library handles the registration of devices located in a MID-B environment. Recalled the Figure 5-10, the <<merge>> dependency links
ue_c library to msb library. Meaning, ue_c library is expanded by adding the registry features implemented in the msb library. Logically separating between the core ue_c library and the msb library simplifies the MID-B implementation. That is, during designing and implementing, a developer could concentrate either only at the ue_c routine, or registration feature, or both.

![Class diagram of device profile and other associated classes](image)

**Figure 5-26: Class diagram of device profile and other associated classes**

In msb library, a hash table reporting changes of available UE_Is is maintained. Literally, each entry of the table represents a UE_I’s profile. Figure 5-26 shows a class diagram of device profile and associated classes. Each entry in the device table is identified by an ‘address’ attribute. As described in Section 4.3.2.2.2, a device profile contains the attributes ‘device class’, ‘hardware’ and ‘software’ characteristics, ‘communication technologies’, and ‘dynamic-value’. Besides, an ‘availabilityTag’ attribute suggests whether a device is currently present nearby the UE_C. Yet, it does not indicate whether the present device is usable. This is indicated by ‘usableTag’ attribute. Manipulation of the two attributes gives advantages. Although a UE_I is currently located in the service vicinity, there could be some reasons that make this UE_I temporarily unusable. For example, given that an application is highly confidential, only the UE_C is allowed to run the application. In a case, even if the availabilityTag is set true, the usableTag is set to false. The ‘logicalRelationCode’ attribute indicates the semantic dependency between the modalities offered by UE_I and UE_C. The semantic dependency will be described in the following chapter.
To manage the context information of devices, a device handler assembles all the device profiles into the implemented hash table. Literally, MID-B defines two types of devices, i.e. UE_C and UE_I. So, the device handler maintains a UE_C profile and a device table storing UE_I details. Shown in Figure 5-27, the ‘DeviceHandler’ class contains operations that enable the management of both the UE_C information and the device table. ‘registerUE_C()’ is used for subscribing a specific core device to a system. To disable the UE_C, ‘makeUE_CDisable()’ sets the usableTag of the UE_C to false. Such a case, all services on the UE_C must be disabled too. ‘makeUE_CDisable()’ calls a function provided in ‘ModalityHandler’ (described in Section 6.3.3) to modify records of all modalities that belong to the UE_C.

For the UE_I, ‘registerDevice()’ adds new discovered UE_I to the device table. In some situation, an interaction should not be operated on any public device; all UE_Is needs to go into the hibernation state. This will be handled by ‘makeAllUEIHibernated()’. ‘makeAllUEIHibernated()’ calls a private operation ‘modifyStatusToAllUEI()’ to change usableTag of all entries in the device table. Also, a function from ‘ModalityHandler’ (described in Section 6.3.3) is called to disable all modalities provided by the UE_Is. Because a UE_I could be temporarily withdrawn from an environment, ‘findAvailableUEIs()’ searches the device table for the entries of which availableTab is true. It returns a set of UE_Is that are currently present. In addition, ‘DeviceHandler’ implements a private operation ‘changeAvailabilityTag’ to modify a value in the availabilityTag in a specific entry of the table.
5.5 Bluetooth-Based Discovery

5.5.1 Implementation Stack of Bluetooth-Based Discovery

![Implementation stack of Bluetooth-based discovery](image)

Figure 5-28: Implementation stack of Bluetooth-based discovery

Figure 5-28 illustrates an implementation stack for Bluetooth-based discovery. For the Bluetooth-based discovery, most mechanisms are provided by common language APIs. Targeting small terminals, Java APIs for Bluetooth wireless technology \(^7\) [150] and Java 2 Micro Edition (J2ME) \(^8\) [151] are used.

With Bluetooth, the UED enables a UE_C to dynamically discover UE_I(s) by a method called 'inquiry'. Both the UE_C and the UE_I(s) have to specify a mode of inquiry to perform. The mode could be either 'general' or 'limited'. UE_C starts the inquiry by issuing an inquiry request. UE_I(s) responds to this inquiry request, depending on its discovery mode. Considering Bluetooth allows up to seven slave devices connected to a master device, the maximum number of UE_Is is limited to seven.

\(^7\) Java Specification Request – 82 (JSR-82)

\(^8\) Java Specification Request – 118 (JSR-118)
After the UED, the SDP transaction is carried out. This has been specified at the SDP layer of the generalised Bluetooth stack. This transaction is implemented by using the SDP APIs provided in JSR-82.

Next, the extension MM transaction is implemented at the MST layer. In MID-B, the MST layer is added to the same layer as the SDP. A message of the MST is transmitted via a serial port profile implemented at the L2CAP layer. Note that, a serial port is selected as communication media since it is guaranteed in every Bluetooth-enabled device.

5.5.2 Emulation

Figure 5-29: Class diagram for UE_I

Figure 5-29 illustrates a class diagram for UE_I. Based on J2ME, a core MIDlet suite\(^9\) is implemented in the ‘UserInterfaceDevice’ class. An ‘ErrorMessage’ provides a number of callbacks generating an error notification. In ‘SettingInquiryMode’, the mode of inquiry is assigned.

Figure 5-30 depicts a sequence of the UE_I routines. The sequence is logically separated into three parts. In the first part, the UED and the SDP transactions are carried out. This part is implemented using standardised APIs (Java JSR-82). In this part, ‘UE_IForm’ is initialised to retrieve a Bluetooth address of a local device (UE_I in this case). Then, UserInterfaceDevice passes this address and the instance of UE_IForm to ‘UEIConnectionService’. MID-B requires sufficient information about ‘Bluetooth profiles’, hence, UEIConnectionService implements

\(^9\) Generally, J2ME application follows the MIDlet lifecycle, including ‘pause’, ‘active’ and ‘destroyed’.
multithreads to initiate a set of Bluetooth profiles, and then adds these profiles into UE_I's service records (N.B. Bluetooth profiles correspond to the attribute ID 0x09). Each thread waits for a connection being requested by a UE_C. Once the connection is opened, a service search of the SDP transaction is automatically carried out and the service records are sent to the UE_C.

Figure 5-30: Sequence diagram of the UE_I routines

The second part of the diagram shows the additional MM transaction. 'handleAcceptAndOpen()' collaborates with 'handleStreamsOpen()' in managing a serial port connection. The MST is transmitted via this serial port. A function 'handleReceivedMessage()' deals with a method to obtain a request message asking for the MST. Then, 'generateAndSendMSTResponse()' generates and writes the MST to the open serial port.
The last part of this sequence is to close the connection. If UserInterfaceDevice obtains a notification that the UE_C has left, 'handleClose()' terminates the connection.

At the UE_C side, Figure 5-31 presents the class diagram. Similar to the UE_I, 'UserInterfaceDevice' encompasses the core MIDlet suite. The 'ErrorMessage' is responsible to generate error notifications and 'SettingInquiryMode' assigns an inquiry mode to the UE_C.

Figure 5-31: Class diagram for UE_C implementation

<table>
<thead>
<tr>
<th>ServiceDiscoveryList</th>
</tr>
</thead>
<tbody>
<tr>
<td>discoveryAgent: DiscoveryAgent</td>
</tr>
<tr>
<td>inquiryAccessCode: int</td>
</tr>
<tr>
<td>matchingServiceRecords: Hashtable</td>
</tr>
<tr>
<td>numConnectionsAlreadyOpen: int = 0</td>
</tr>
<tr>
<td>sdTransMax: int = 10 {readOnly}</td>
</tr>
<tr>
<td>uuid: UUID</td>
</tr>
</tbody>
</table>

+ deviceDiscovered(DeviceClass, RemoteDevice): void
+ displayUE_ITable(): void
+ getNumberUE_II(): int
+ inquiryCompleted(int): void
+ run(): void
+ searchService(): void
+ ServiceDiscoveryList(int, String, UserInterfaceDevice)
+ serviceSearchCompleted(int, int): void

Figure 5-32: ServiceDiscoveryList
As a client, the UE_C requires functions to discover UE_Is and locate their services. These functions are defined in a ‘ServiceDiscoveryList’, shown in Figure 5-32. ServiceDiscoveryList sets the maximum number of SDP transactions that the UE_C can handle at a time. To accomplish the three-party model, this class includes the functionality of the MSB. The attribute ‘matchingServiceRecords’ is declared as a hash table; it maintains a list of discovered UE_Is and the matching service records.

Inside ServiceDiscoveryList, an instance of the ‘DiscoveryAgent’ is declared to operate device and service discovery. DiscoveryAgent, specified in JSR-82, defines a number of functions to start an inquiry (‘discoveryAgent.startInquiry()’) and to perform a service search (‘discoveryAgent.searchService()’).

In addition, ServiceDiscoveryList implements ‘DiscoveryListener’ interface. Again, ‘DiscoveryListener’ interface is specified in JSR-82. With DiscoveryListener, four operations are inherited, i.e. ‘deviceDiscovered()’, ‘inquiryComplete()’, ‘serviceDiscovery()’, and ‘serviceSearchComplete()’. In MID-B, deviceDiscovered() is responsible to add all discovered UE_I(s) to a temporary table, regardless of whether a UE_I is rediscovered. Once device discovery ends, inquiryComplete() modifies a display to show a list of UE_I(s) stored in that temporary table. Subsequently, serviceDiscovered() matches these discovered UE_I(s) with the service record(s) maintained in the declared ‘matchingServiceRecords’. The matching is carried out in one of three ways.

- There is no service record matching with a discovered UE_I. The new record(s) of this UE_I is then added in matchingServiceRecords.
- There is no discovered UE_I matching with some records in matchingServiceRecords. This indicates that the matching UE_I may be temporarily withdrawn from the service area. Hence, those records are tagged out.
- UE_I matches with some service record(s). The information about UE_I services has been maintained in a system.

After all, ‘serviceSearchComplete()’ notifies that all the UE_Is’ service records are located.

10 The javax.bluetooth.DiscoveryAgent provides methods to perform device and service discovery. A local device must have only one DiscoveryAgent object. This object must be retrieved by a call to getDiscoveryAgent() on the LocalDevice object.
Chapter 5. Device and Service Discovery

Figure 5-33: Sequence diagram of the UE_C routines

Figure 5-33 depicts a sequence of the UE_C routines. The sequence is divided into three main parts. UED and SDP transaction are operated during the first part. This part involves ServiceDiscoveryList described above. Next, the MM transaction is operated in the second part. To accomplish the MM transaction, a function ‘serviceDiscoveryListOpen()’ opens the serial port connections that correspond to the service records in matchingServiceRecords. Then, an instance of the ‘UE_CForm’ is initialised to display the activities currently taking place. UE_CForm also establishes the MST connections to the discovered UE_Is. ‘UE_CConnectionHandler controls the serial port to where the MST will be received. On receipt of the MST, ‘handleReceivingMST()’ files the obtained MSTs.

Under the condition that the UE_I has left the UE_C’ area, ‘handleClose()’ closes the connection to that particular UE_I.

The following Figure 5-34 shows an example emulation of the Bluetooth-based discovery. The UE_C form and the UE_I emulations are illustrated.
Chapter 5. Device and Service Discovery

(a) At the UED, a UE_C discovers surrounding UE_I(s). However, there could be a possibility that there is no UE_I.

UE_I (003131127835)  UE_I (003130127835)

UE_C

UE_I (003131127835)  UE_I (003130127835)

UE_I Identification

(b) Two devices are added to be emulated as UE_Is. They are discovered by the UE_C.

(c) The UE_C makes a connection to the first UE_I.
(d) The UE_C makes a connection to the second UE_I.

(e) The MST request has been sent to both of the UE_Is.

(f) The UE_C has received the MSTs from both of the UE_Is.
Chapter 5. Device and Service Discovery

One of the UE_Is leaves the service vicinity. The affected connection is closed.

Since the UE_C is generally a portable device carried by a user, it may leave the UE_I at anytime. At the UE_I, a function is called to terminate the connection.

Figure 5-34 (a) – (h): Emulation of the Bluetooth-based discovery

5.6 Summary

This chapter presents the service discovery mechanisms defined for MID-B. In this chapter the three-party model is introduced. It expands the mechanism of the SDP transaction with the capabilities of the modality searches. The three-party model reduces the network traffic by introducing the UE_I registry-table and the availability tag. Rather than rediscovering service information, the information of any re-discovered UE_Is will be retrieved from the registry table. This chapter continues by explaining the phases of multimodal service delivering in the MID-B system. These include the basic protocol of User Equipment Discovery (UED), the underlying SDP transaction of Bluetooth, and the extension of the Multimodality transaction (MM transaction).
The implementation of discovery is carried out in two scenarios, i.e. IP-based and Bluetooth-based discovery. The IP-based discovery allows a degree of freedom in physical connection media. The Bluetooth-based discovery is implemented based on the Bluetooth stack, i.e. devices must be Bluetooth-enabled. This chapter illustrates how the UE_C discovers UE_Is and their modality services. The next chapter will present how to consolidate all of the context information and explain the mechanism deciding whether the discovered services of the UE_Is comply with the requirements.
Chapter 6

6 Interface and Modality Binding

6.1 Introduction

The interface and modality binding facilitates the customisation and adaptation of interaction services. It maximises the features of services provided in an environment as well as it controls the way to use interaction means. Binding integrates and interprets the heterogeneous information of different contexts. The actual content can then be transformed according to the user and device contexts. In the MID-B environment, due to the binding, UE_C and UE_I(s) are used to form a virtual multimodal interface. For a virtual multimodal interface, a semantic expression of modalities hosted on diverse physical devices is required. Such an expression makes use of the Modality Service Tree (MST), which delivers the description of dependencies between different modalities.

This chapter describes the mechanisms of interface and modality binding. Device and service discovery has been introduced in previous chapters; this chapter combines the remaining functionalities needed for MID-B. The chapter explains how MID-B exploits the discovered information to accomplish interface and modality binding. While a brief introduction of the MST has been presented in the previous chapter, Section 6.2 describes the Modality Service Tree (MST) in detail. The section starts with introducing the theoretical approach of the MST. It then details the dependencies between different MSTs. Then, it describes a method for service and modality information within the MST. At the end of the section, an implementation of the MST is presented. Section 6.3 presents the resource handling mechanism. Several ‘devices’ are involved in a MID-B environment. This makes the system rather complex. MID-B needs an effective way to organise the context information collected from these devices. It requires a maintenance mechanism to handle service information of arbitrarily available devices. While the MST manages the modality context of each user’s device, the resource handling is the mechanism to organise the context information for the overall system. The overall system is visualised as a ‘MST forest’. Resource information captured within the MST forest is maintained in a number of hash tables. Section 6.4 introduces a mechanism and implementation for interface and modality binding. For multiple physical devices, a binding function unites the capabilities of cross-device services. A theoretical approach to the interface and modality binding is described subsequently.
Following, the section also presents the basic algorithm how to manipulate contexts of user, device, and etc. Section 6.5 presents the summary of the chapter.

6.2 Modality Service Tree (MST)

Every interface device, i.e. UE_C and UE_I, encloses a generic hierarchy describing the details of their inherent interfaces and modality services. The hierarchy is captured as ‘Modality Service Tree (MST)’. In principle, MST describes device characteristics and the context in which a device can be used. It thoroughly explains the interfacing hardware and describes the constraints of the services imposed by this hardware. Basically, MST exploits the fact that the interaction means can be specified and produced at and from various levels of abstractions. Hence, MST deals with different levels of interaction means as well as maintains the mapping between these levels. The concept and theory is straightforward to support the MID-B execution. The binding function then can use the information from MST at any level of the hierarchy. In chapter 5, the definition of a three-tier hierarchy for MST has been presented, followed by the introduction of the theoretical approach to the MST. In this section, MST explains the MST in a mathematical way.

6.2.1 Theoretical Approach to the MST

A number of research groups in Human Computer Interaction (HCI) have established how mathematical methods can be used to describe and analyse sophisticated HCI mechanisms. For example, PUMA [152] introduced generic specifications to describe users and devices, based on cognitive psychology and higher-order logic (HOL). In addition, PUMA showed how to use these specifications to express the properties of a system. In [153], Thimbleby and Gow suggested how to express user interfaces and user behaviours by using mathematical theorems, and subsequently used the proof of these theorems to develop interface services. Furthermore, they advertised the advantages of algebraic and mathematical theorems by claiming that: “We can arrange for functions to account for detailed user behaviour at a fine level”. Jeschke [154] recommended that mathematics can express “granularity” and “structure” of complex models. That is, algebraic computing minimises the encumbrances of routine calculation as well as offers the ability to a system to promptly familiarise with a new concept and method.

Taking advantage of mathematical methods, this thesis uses graph theory and a set-oriented design to construct the theoretical approach to multimodal user interface services. In a theoretical approach, the characteristics of modalities are expressed using Discrete Mathematics [155]. A number of definitions of graph theory are employed. In MID-B, the MST's properties are inherited from the theories of ‘bipartite graph’ and ‘labelled graph’. A bipartite graph uses a set of graph vertices that can be decomposed into two disjoint sets; two graph vertices of the same set do
not share a graph edge. On the other hand, a labelled graph contains the set of labels associated with a graph edge and/or graph vertices. These theories mean for MST concept:

- MST primarily expresses the relations linking two elements, for example the relation between 'interface equipment' and 'interface service'.
- None of the relations in the same MST can coincide with others.

Further, the complete MST is extending the graph configuration forming a more meaningful structure using the 'rose tree definition'. The rose tree structure, introduced by Lambert Meertens [156], allows the construction of a multi-branch tree. It is a means to define a unifying and abstract datatype. Therefore, it is suited to capture the complexity of user interface features that may be offered by a complex user interface device. The latter extension forming the rose tree allows the MST to be more flexible, expressive, and representative.

The thesis uses the general elements of axioms and definitions to express the modality hierarchy and to describe the MST rules.

**Axiom 6.1 Generic modality hierarchy**

A modality hierarchy is defined as a rooted tree in which a special node is singled out to describe the type of an interface device, i.e. UE_C or UE_L. The children along the root node have a designated order that can be referred to explicitly.

**Definition 6.1 Modality hierarchy rules**

A set of modality hierarchy rules is defined in relation to the three-tier hierarchy explained in Chapter 5. The rules are specified as follows:

1. Tree Level: Level or order represents a set of particular services in a hierarchical structure. MST consists of four levels. These are 1) user equipment device (i.e. root-level), 2) physical interface equipment (i.e. middle-level), 3) interface service (i.e. middle-level), and 4) modality service (i.e. leaf-level).

2. Root node, level 1: The root node represents a user device (i.e. UE_C or UE_L).

---

11 A rose tree is a tree whose internal nodes have an arbitrary number of children.

12 Similarly, an axiom and a definition are the statement taken to be true without any proof. However, the axiom generally refers to a short self-evident statement, whereas the definition is a precise statement clearly delineating and naming a concept by relating it to previously defined concept or un-defined concept.
3. Level-2 nodes: The nodes at level two correspond to a set of interface equipments provided by the device.

4. Level-3 nodes: The nodes located at level three correspond to a set of interface services offered by the device. In [157] defines "a user interface is formed of all the aspects of a computer system of which the user is aware and uses to communicate with the system".

5. Leaf nodes: The leaf nodes characterise the set of modality services of the device. A modality is "the component performing interface tasks pertinent to their particular interface modality (e.g. voice, pen, visual display)" [158].

6. Dynamic number of nodes: The number of nodes at each level is dynamic, except of those on the root level.

7. Labelled relation: Two services in two succeeding orders (levels) are linked by a labelled relation, which is represented by a directional arrow (Referring to the Figure 5-9).

8. Interface Service Operator: A level-3 node 'may' contain an 'Interface Service Operator', i.e. an 'Un-Coordinated Interface Service Operator' and a 'Synchronizing Interface Service Operator'. The symbol ☐ is used for a Un-Coordinated Interface Service Operator. The interface service may be operated by several interface equipments; however, those interface equipments work independently. The symbol ☒ represents a 'Synchronising Interface Service Operator'. The interface service may be synchronically operated by different interface equipments, i.e. the service will function if all the required interface equipments are available.

9. Modality channel: Modality channel, i.e. output channel and input channel, defines the direction of a modality service, i.e. the direction in which the modality stream (user in/output) flows.

**Definition 6.2 Modality Service Tree (MST)**

A MST structure is designed following Axiom 6.1 and Definition 6.1. MST forms the relations between two labelled services offered by a user equipment device. That is,

---

13 See example in Appendix A3.
Chapter 6. Interface and Modality Binding

\[ \text{MST} = \Sigma \text{R}(S_k, S_j) \]  \hspace{1cm} (6.1)

Where \( \text{R} \) is a relation between two services ordered \( k \) and \( j \), and \( |k| - |j| = 1 \). The following definition 6.7, 6.8 and 6.12 further define the different types of the relation \( \text{R} \). Let ‘\( S \)’ be a finite non-empty set of services available within a hierarchy.

\[ S = \{ S_1, S_2, S_3, S_4 \} \]  \hspace{1cm} (6.2)

\( S_1 \) is the root level, i.e. a user equipment device.

\( S_2 = \{ i_1, i_2, \ldots, i_m \} \) is a finite non-empty set of interface equipments.

\( S_3 = \{ (x, y) : x \text{ is a type of the interface service and } y \text{ is an interface service operator} \} \).

\( S_4 = \{ (x, y) : x \text{ is a type of modality service and } y \text{ is a modality channel} \} \).

![Figure 6-1: Structure of the MST](image)

From now, the set \( S \) will be referred to throughout the rest of the thesis.

**Definition 6.3 Disjoint set**

MST must be able to be decomposed into two disjoint sets. Assuming ‘\( M \)’ is a modality service element, ‘\( I \)’ is an interface service element, and \( \text{R} \) is a relation between \( M \) and \( I \).

\[ \text{R}(M_1, I_1) = \text{R}(M_2, I_2) \text{ iff } M_1 = M_2 \land I_1 = I_2 \]  \hspace{1cm} (6.3)

**Definition 6.4 MST for single interface equipment**

A single interface equipment represents either a stand alone device or a device component. A stand alone device can provide a modality service without the necessity to be plugged-in to another device (e.g. a printer). A device component is a pluggable computer peripheral like a

---

14 See example in Appendix A4.
mouse, a headphone and a keyboard which cannot perform tasks on its own. For a representation (in the MST) of single interface equipment, interface equipment ($S_2$) is a mirror of a user equipment device ($S_1$).

$$|S_2| = 1 \land S_2 = S_1 \leftrightarrow \text{A single interface equipment} \quad (6.4)$$

![Figure 6-2: MST representing a single interface equipment](image)

**Definition 6.5 MST for composite device**

A composite user device, such as a PC, accommodates a number of physical plugged-in components. Literally, every component is considered as one user equipment device (with single interface equipment), and hosts an individual MST. A MST representing a composite device requires an additional feature to describe 1) a physical connectivity between the device and its components, and 2) the logical connectivity between the device and the interface equipment provided by the attached components. This additional information is captured in level 1 of the MST as shown in Figure 6-3.

![Figure 6-3: MST representing a composite device](image)

15 See example in Appendix A5.
Definition 6.6 MST datatype

Since a structure of interface and modality services hosted on different devices is arbitrary, MST’s internal nodes have a random number of children. Besides, the number of nodes at each service level varies from one MST to another. To process the MST’s nodes, ‘MST datatype’ specifies a linear linked list, of which the head is set for a reference node. The list tracks from the reference node to the next nodes in the tree branch until the corresponding modality-service node is reached. The MST datatype specifies the detail of services in a particular branch. Let \( \Omega \) be a reference node in the MST. The datatype is defined as:

\[
\Omega = \text{Service } S_k \left[ \sum (\text{Service } S_i[j]) \right] S_{i[n-1]} \leftarrow S_{i[0]} \text{ where } k<j<n
\] (6.5)

At each level, the datatype is derived as:

1) Device \( \Omega \) = DeviceType \( \Omega \) [Interface_Equipment \( \Omega \) [Interface_Service \( \Omega \)

   [Modality \( \Omega \)]]

2) Interface_Equipment \( \Omega \) = Interface_Equipment \( \Omega \) [Interface_Service \( \Omega \) [Modality \( \Omega \)]]

3) Interface_Service \( \Omega \) = Interface_Service \( \Omega \) [Modality \( \Omega \)]

4) Modality \( \Omega \) = Modality \( \Omega \) [ ]

Definition 6.7 SubEquipment relation

A SubEquipment relation describes a set of embedded physical interface equipments provided by a user device. Let \( \text{SE} \) be a SubEquipment relation from level one onto level two, \( S_1 \rightarrow_{onto} S_2 \).

\[
\text{SE(} \text{User Equipment Device} \text{)} \rightarrow S_2 = S_1.\text{SubEquipment}
\] (6.6)

By means of the Cartesian product, the relation (6.6) is re-derived as an ‘ordered pair’:

\[
\text{SE(} \text{User Equipment Device} \text{)} = \{(x, y) \in S_1 \times S_2 | S_2 = S_1.\text{SubEquipment}\}
\] (6.7)

Definition 6.8 InterfaceGive relation

Interface services are the elements of \( S_3 \). InterfaceGive relation characterises a relation between an interface and interface equipment at \( S_2 \) that provides this interface. Let \( \text{IG} \) be an InterfaceGive relation from level two onto level three, \( S_2 \rightarrow_{onto} S_3 \).

---

16 The linear linked list is a data structure in which each element contains a pointer to the next element.

17 In mathematical definition, function or relation is said to be ‘onto’ if every \( y \) in the Range \( Y \) have a mapping to \( x \) in Domain \( X \).
Chapter 6. Interface and Modality Binding

\[ \text{IG (User Equipment Device)} \rightarrow S_3 = S_2 . \text{InterfaceGive} \] (6.8)

By means of the Cartesian product, the relation (6.8) is re-derived as an ‘ordered pair’:

\[ \text{IG (User Equipment Device)} = \{(x, y) \in S_2 \times S_3 | S_3 = S_2 . \text{InterfaceGive}\} \] (6.9)

**Definition 6.9 Interface’s ancestor**

The interface’s ancestors correspond to a set of physical interface equipments, of which each element shares at least one interface service. Relation between this interface service and its ancestor set is derived by the inverted InterfaceGive relation, \( \text{IG}^{-1} \). The inversion of the relation (6.8) and (6.9) is obtained as:

\[ \text{IG}^{-1} (\text{User Equipment Device}) \rightarrow S_2 = S_3 . \text{InterfaceGive}^{-1} \] (6.10)

\[ \text{IG}^{-1} (\text{User Equipment Device}) = \{(y, x) \in S_3 \times S_2 | S_2 = S_3 . \text{InterfaceGive}^{-1}\} \] (6.11)

**Axiom 6.2 Interface service operator**

An interface service operator is used to indicate whether an interface service is derived by multiple physical interface equipments. Providing the size of an interface’s ancestor set of the interface service ‘i’ is greater than 1, ‘i’ is hosted on diverse equipment. In such a case, ‘i’ must enclose one interface service operator, either an Un-Coordinated interface service operator (\( \otimes \)) or a Synchronizing interface service operator (\( \oplus \)).

**Definition 6.10 Un-coordinated interface equipments relation**

An interface ‘x’ with an operator ‘\( \otimes \)’ indicates that the interface may be operated by any member of the interface equipment (IE) in the interface’s ancestor set.

\[ \otimes : S_3 \leftarrow S_2 , \text{that is } x \in S_3 \text{ if } \sum_{i \in I} IE_i \exists i \in I, IE \in S_2 \] (6.12)

Where ‘I’ is a finite index set \(^{18}\) and \( k \) is size of the interface’s ancestor set.

**Definition 6.11 Synchronising interface equipments relation**

An interface ‘x’ with an operation ‘\( \oplus \)’ indicates that the interface service ‘x’ requires all synchronised interface equipments (IE) in the interface’s ancestor set.

\[ \oplus : S_3 \leftarrow S_2 , \text{that is } x \in S_3 \text{ iff } \sum_{i \in I} IE_i \forall i \in I, IE \in S_2 \] (6.13)

Where ‘I’ is a finite index set and \( k \) is size of the interface’s ancestor set.

---

\(^{18}\) A index set is a set of which the members label (index) the members of another set.
Definition 6.12 ModalityOf relation

A ModalityOf relation represents a correlation between a modality service (S₄) and an interface service (S₃). Let MO be a ModalityOf relation from the level 4 onto the level 3 in MST, S₄ → S₃

\[
\text{MO(} \text{User Equipment Device} \text{)} \rightarrow S₃ = S₄.\text{ModalityOf} \tag{6.14}
\]

By means of the Cartesian product, the relation (6.14) is re-derived as an 'ordered pair':

\[
\text{MO(} \text{User Equipment Device} \text{)} = \{(x, y) \in S₄ \times S₃ \mid S₃ = S₄.\text{ModalityOf}\} \tag{6.15}
\]

Definition 6.13 Modality set

A set of modalities provided by a particular interface is derived by an inversion of ModalityOf Relation, \( MO^{-1} \). The inversion of the relation (6.4) and (6.15) are obtained as:

\[
\text{MO}^{-1}(\text{User Equipment Device}) \rightarrow S₄ = S₃.\text{ModalityOf}^{-1} \tag{6.16}
\]

\[
\text{MO}^{-1}(\text{User Equipment Device}) = \{(y, x) \in S₃ \times S₄ \mid S₄ = S₃.\text{ModalityOf}^{-1}\} \tag{6.17}
\]

Definition 6.14 Modality direction

A modality is a directional service (i.e. input and/or output). In MST, the information flow of modalities is defined by a directional arrow. Let I be a finite index set, \( I = \{1, 2\} \), \( M₁ \) be a set of input modalities, and \( M₂ \) be a set of output modalities. A Set of all modalities hosted on a device (S₄) is the union of sets of the input modalities and the output modalities.

\[
\therefore S₄ = \bigcup_{i \in I} M_i = \{x \mid \exists i \in I, x \in M_i\} \tag{6.18}
\]

Definition 6.15 Bi-directional modality

Assuming In/Out is a set of modalities offering both of input and output channels. In/Out is an intersection between \( M_i \).

\[
\therefore \text{In} / \text{Out} \subseteq S₄ = \bigcap_{i \in I} M_i = \{x \mid \forall i \in I, x \in M_i\} \tag{6.19}
\]
Chapter 6. Interface and Modality Binding

6.1 Modality function

Let function $M(\Omega)$ be a modality function of a device $\Omega$.

$$M(\Omega) = \Sigma [\lambda, \gamma]$$  \hspace{1cm} (6.20)

Where:

$\lambda \in S_4 = \{x, y\} | x$ is a type of modality service and $y$ is a modality channel}, and

$\gamma$ = Capabilities of a specific modality $x$.

6.2.2 Dependencies between MSTs

A MID-B environment can be characterised by a collection of MSTs. MID-B therefore corresponds to a union set between the representation MST of UE_C and this of UE_I(s). Using a set-oriented design, MID-B is defined as

$$MID-B = MST_{\text{UE,C}} \cup \Sigma MST_i$$  \hspace{1cm} (6.21)

Where, $\Sigma MST_i = \{MST_1, MST_2, MST_3, \ldots, MST_n\}$, and $n$ is the number of the UE_Is in the system. In the union set, the information of the constituent MSTs is used to define the dependencies between the UE_C and the UE_I(s). The dependencies can be expressed in both a logical relation and a cluster relation. The logical relation refers to the semantic dependencies between a number of interface and modality services provided by different representation MSTs. A number of theorems defined for the logical relation are presented in the Section 6.2.2.1. The cluster relation describes the dependencies within a sub group of UE_Is (and one UE_C) forming a closed group within the (radio) proximity. The theoretical explanation of the cluster relation is provided in Section 6.2.2.2.

6.2.2.1 Theoretical Concept of Logical Relations between UE_C and UE_I

Axiom 6.3 Semantic of interface equipments

Physical interface equipments of two devices are said to be semantically identical if the sets of interface services obtained from these equipments are identical.

19 Function and relation both give a mapping between the provided domain and range. However, they are slightly different. Let $A$ and $B$ be any set, and $F$ be a relation from $A$ to $B$. $F$ is said to be function if and only if $\forall x \in A, \forall y \in B, \forall z \in B : (x, y) \in f \land (x, z) \in f \rightarrow y = z$. That is, every $x$ in the domain $A$ has one-to-one mapping to $y$ in the range $B$. 

111
Definition 6.16 Semantic of the interface equipment ‘A’ between two user equipment devices

Assuming a device ‘x’ and a device ‘y’ have a number of physical interface equipments ‘A’ that:

\[ A = \{ a \mid a \in S_2 \text{ of } \text{MST}_x \cap S_2 \text{ of } \text{MST}_y \} \]  

(6.22)

An element \( a \in A \) claims a ‘semantic relation of interface equipment’ between the device x and the device y if the set of the interfaces provided from ‘a’ at the device x and that of the device y are coincided.

Theorem 6.1 Semantic of the interface equipment ‘A’ between the \( \text{UE}_C \) and the \( \text{UE}_I \)

Let:

\[ A = \{ a \mid a \in S_2 \text{ of } \text{MST}_{\text{UE}_C} \cap S_2 \text{ of } \text{MST}_{\text{UE}_I} \} \]

be a set of semantic interface equipment between the \( \text{UE}_C \) and the \( \text{UE}_I \),

B be a set of child nodes of set ‘A’ at the \( \text{MST}_{\text{UE}_C} \), and

C be a set of child node nodes of set ‘A’ at the \( \text{MST}_{\text{UE}_I} \).

\( \text{UE}_C \) and \( \text{UE}_I \) encompass a semantic of interface equipments on ‘A’ iff

\[ A \times (B \cap C) = A \times (B \cup C) \]  

(6.23)

Proof:

The assumption is that there is a set ‘A’, i.e. a set of semantic of interface equipments on A, between \( \text{MST}_{\text{UE}_C} \) and \( \text{MST}_{\text{UE}_I} \). Let \( IG_1 \) and \( IG_2 \) be an InterfaceGive relation on ‘A’ at the \( \text{MST}_{\text{UE}_C} \) and \( \text{MST}_{\text{UE}_I} \), respectively. According to the relation (6.8), \( IG_1 = A \times B \), and \( IG_2 = A \times C \). The set A is said to be semantic if \( IG_1 \) and \( IG_2 \) coincide, that is the union operation and the intersection operation do not give a difference. Thus, \( IG_1 \cap IG_2 = IG_1 \cup IG_2 \).

Case 1: Assuming, \( IG_1 \cap IG_2 = A \times (B \cap C) \)

Let \( (x, y) \in A \times (B \cap C) \) \( \leftrightarrow x \in A \land y \in (B \cap C) \)

\[ \leftrightarrow (x \in A \land x \in A) \land (y \in B \land y \in C) \]
\[ \leftrightarrow (x \in A \land y \in B) \land (x \in A \land y \in C) \]
\[ \leftrightarrow (x, y) \in A \times B \land (x, y) \in A \times C \]
\[ \leftrightarrow (x, y) \in (A \times B) \cap (A \times C) \]
\[ \leftrightarrow (x, y) \in IG_1 \cap IG_2 \]

---

20 See example in Appendix A6.
Case 2: Assuming, \( IG_1 \cup IG_2 = A \times (B \cup C) \)

Let \((x, y) \in A \times (B \cup C) \leftrightarrow x \in A \land y \in (B \cup C)\)

\[
\begin{align*}
\leftrightarrow (x \in A) \land (y \in B \lor y \in C) \\
\leftrightarrow (x \in A \land y \in B) \lor (x \in A \land y \in C) \\
\leftrightarrow (x, y) \in A \times B \lor (x, y) \in A \times C \\
\leftrightarrow (x, y) \in (A \times B) \cup (A \times C) \\
\leftrightarrow (x, y) \in IG_1 \cup IG_2
\end{align*}
\]

\( \therefore IG_1 \cup IG_2 = A \times (B \cup C) \)

\( \therefore \) Case 1 and Case 2, a semantic interface equipment on A between the UE_C and the UE_I provides an equational specification as \( A \times (B \cap C) = A \times (B \cup C) \). Accordingly, the equation (6.23) is true.

**Axiom 6.4 Semantic dependency of modality services**

Semantic dependency of modality services corresponds to a relation between ‘modality functions’ of \( M(c) \) and \( M(i) \). \( M(c) \) is a modality function of \( MST_{UE,C} \) and \( M(i) \) is a modality function of \( MST_{UE,I} \). MID-B categorises the dependency in five groups.

- Semantic modality relation, SM.
- Comprehensive semantic modality relation, CSM.
- Partial semantic modality relation – static, PSM-S.
- Partial semantic modality relation – dynamic, PSM-D.
- Non-semantic modality relation, NSM.

**Definition 6.17 Semantic modality relation, SM**

SM implies that \( MST_{UE,C} \) and \( MST_{UE,I} \) provide the same set of modalities (yet may have different capabilities). The SM is derived as:

\[
SM : MST_{UE,C} \rightarrow MST_{UE,I} \iff \forall i \in M(i), \forall c \in M(c) : \forall i \forall c \in M(i) = M(c) \quad (6.24)
\]

**Definition 6.18 Comprehensive semantic modality relation, CRM**

CSM extends the definition of SM. \( MST_{UE,I} \) encompasses the comprehensive semantic modality relation if and only if it provides exactly the same modality services with the \( MST_{UE,C} \). The CRM relation is very strong as it involves all the data representation dimensions of the MSTs. The CSM
consider modality features and capabilities. MST\textsubscript{UE.C} and MST\textsubscript{UE.I} must be identical; S\textsubscript{1} to S\textsubscript{4} in these MSTs are carried in the same structure.

\[ CSM : MST_{UE \_C} \rightarrow MST_{UE \_I} \iff \forall i \in M(i), \forall c \in M(c) : \forall i \forall c \in M(i) = M(c) \quad (6.25) \]

**Definition 6.19 Partial semantic modality relation – Static, PSM-S**

PSM-S corresponds to that a MST\textsubscript{UE.I} provides 'some' modalities similar to MST\textsubscript{UE.C}, and 'does not' introduce any new one.

\[ PSM - S : MST_{UE \_C} \rightarrow MST_{UE \_I} \iff \forall i \in M(i), \forall c \in M(c) : \forall \exists c \in M(i) \cap M(c) = 0 \quad (6.26) \]

**Definition 6.20 Partial semantic modality relation – Dynamic, PSM-D**

PSM-D corresponds to that a MST\textsubscript{UE.I} provides 'some' modalities similar to MST\textsubscript{UE.C}, and 'does introduce' at least one new modality.

\[ PSM - D : MST_{UE \_C} \rightarrow MST_{UE \_I} \iff \forall i \in M(i), \forall c \in M(c) : \exists \exists c \in M(i) \cap M(c) = 0 \quad (6.27) \]

**Definition 6.21 Non-semantic modality relation, NSM**

NSM corresponds to that a MST\textsubscript{UE.I} introduces a new set of modality which is not included in the M(c).

\[ NSM : MST_{UE \_C} \rightarrow MST_{UE \_I} \iff (SM \land PSM - S \land PSM - R) \land (\forall i \in M(i), \forall c \in M(c) : M(i) \cap M(c) = 0) \quad (6.28) \]

### 6.2.2.2 The Theoretical Concept of Cluster Relations between the UE.C And the UE.I

**Definition 6.22 UE.C set**

MID-B works on the assumption that there is one portal device active at a time. The assumption gives technical meaning to the system as that a UE.C must be functioning for the entire service life cycle. Let UE.C be a set of user core device. The set UE.C has to satisfy the following conditions.

1) UE.C \neq \emptyset

2) The cardinality of UE.C equal to 1, |UE.C| = 1.
Axiom 6.5 Candidate UE_I

A candidate UE_I refers to any discovered user device, regardless of whether the device is still available in the service area.

Function 6.2 Function of the candidate UE_Is

Let F be the function of candidate UE_Is. F: discovered device \(\rightarrow\) candidate UE_I. The function F corresponds that a discovered device will be considered as a candidate UE_I when it is filed in a registry table located at the MSB. Let Candidate_UE_I be a set of candidate UE_Is, and R be a set of data inside the table. Thus,

\[
\text{Candidate\_UE\_I} = \{a : a \in \text{Candidate\_UE\_I} \subseteq R \}
\]

Axiom 6.6 UE_I

A candidate UE_I will be considered as a UE_I if it is reasonably physically close to the UE_C. This is indicated by an availability tag of which the value is ‘true’ (See Section 5.2).

Function 6.3 UE_I function

Let F be a UE_I function, and UEI be the set of UE_Is. Given, F: candidate UE_I \(\rightarrow\) UE_I. The function F corresponds to

\[
\text{UEI} = \{a : \forall a \in \text{UEI} \subseteq \text{Candidate\_UE\_I} \}
\]

Case 1: If a registry table is empty, the UEI is also empty.

Case 2: If any device in the registry hosts the availability tag equal to ‘true’, the device is considered as a UE_I.

Case 3: If the case 1 and the case 2 fail, the set of UE_I is empty.

Axiom 6.6 Non-reachable UE_I

A non-reachable UE_I is a candidate UE_I which is not reasonably close to the UE_C. It is indicated by an availability tag, of which the value is ‘false’ (See Section 5.2).

Function 6.4 Function of non-reachable UE_Is

Let F be the function of non-reachable-UE_I, and \(\overline{\text{UEI}}\) be the set of non-reachable UE_Is. Given, F: candidate UE_I \(\rightarrow\) \(\overline{\text{UEI}}\). The function F corresponds to the function (6.31).
Chapter 6. Interface and Modality Binding

\[
UEI = \left\{ \phi : \text{Candidate}_{\text{UE}_I} \cap \phi = \phi \right\}
\]

\[
UEI = \left\{ \phi : \text{Candidate}_{\text{UE}_I} \cap UEI = R \right\}
\]

\[
a : (a \in UEI \subseteq \text{Candidate}_{\text{UE}_I}) \land (a \in \text{Candidate}_{\text{UE}_I})
\]

Case 1: If a registry table is empty, the \( \overline{UEI} \) is also empty.

Case 2: If all devices in the registry are reachable, the \( \overline{UEI} \) is also empty.

Case 3: Some of the devices in registry host the availability tag equal to 'true', the remaining devices are so considered as the non-reachable \( \text{UE}_I \)s.

**Definition 6.23 Set of clusters**

A cluster set shows the physical correlation between an \( \text{UE}_C \) and \( \text{UE}_I \)s. Let \( C \) be the cluster set, therefore

\[
C = \{ a | a \in \text{UE}_I \cup \text{UE}_C \}
\]

**Theorem 6.2 Cluster set is not empty**

In a cluster, there is at least one interface device functioning. Let \( C \) be a cluster set, then \( \phi \in C \).

Proof:

Definition 6.23 says \( C = \{ a | a \in \text{UE}_I \cup \text{UE}_C \} \).

Due to Definition 6.22, the \( \text{UE}_C \) is a finite set with cardinality 1. Thus, \( \text{UE}_C \) cannot be an empty set.

Case 1: \( \text{UE}_I \) is empty.

Providing \( \text{UE}_I = \phi \), \( |\text{Cluster}| = |\phi \cup \text{UE}_C| = |\text{UE}_C| = 1 \).

Case 2: \( \text{UE}_I \) is not empty.

Providing \( \text{UE}_I \neq \phi \), \( |\text{Cluster}| = |\text{UE}_I \cup \text{UE}_C| > 1 \).

\( \therefore \) Case 1 and Case 2, cardinality of the cluster set 'C' is equal to or greater than one. That is, there is at least one interface device operating in the cluster.

**6.2.3 MST Search Mechanism**

With Definition 6.2 and Equation (6.1), the MST specifies the relations between two consecutive service levels. That is, at any level of the hierarchy, the information at one level is associated with the service's properties at the next level. For example, the relations between modality service(s) \( (S_4) \) and the interface service(s) \( (S_3) \) are derived in Relation (6.14) and Relation (6.16). However, the explanation given by Definition 6.2 is insufficient to describe how two services that are not
located in two successive levels are related. The associations between $S_4 - S_1$, $S_4 - S_2$, and $S_3 - S_1$ cannot be identified. To solve this problem, MID-B introduces the ‘MST search mechanism’. Such a mechanism is able to explore the information provided in the MST at any level. The mechanism is carried out in two ways; these are the top-down and the bottom-up search mechanisms. Both of them are simple, understandable, but functional.

Figure 6-4: Top-down search mechanism

Figure 6-4 illustrates the mechanism of the top-down search. In principle, the top-down search mechanism discovers a range of interfaces and modalities provided by a reference node. In this mechanism, the reference node is characterised by either ‘a user device’ ($S_1$) or ‘interface equipment’ ($S_2$). ‘MST datatype’ described in Definition 6.6 is exploited. In principle, the MST datatype describes the path of a particular branch in the MST. The path is defined as a linked list, starting from the reference node to a modality node. The top-down search mechanism tracks the
path (i.e. the linked list) level by level. If the search does not yet reach an interesting element, the tracking moves down to the next level. Depending on an enquiry, the top-down search mechanism will stop at the level of the interface and/or modality services. At this point, the information of interface (S3) and modality (S4) corresponding to the particular path is discovered.

Possibly, for one reference node, there could be more than one corresponding paths (or linked lists). That is, the reference node could provide several types of interfaces and modalities. In such a case, the mechanism must check the number of paths. If one path is found, the reference node provides one interface and one modality. Otherwise, there are multiple interfaces and modalities provided. For the second case, the mechanism is iterative. When the mechanism finishes tracking one path, it continues to explore the next path. This is repeated until the last path is investigated. Due to such an iterative mechanism, the information of all interfaces and/or modalities associated with the reference node is thoroughly discovered.

The second search mechanism is a down-to-up search. In contrast to the previous mechanism, the bottom-up search explores a MST from a leaf node to a root node. The main task of this search is to discover a user device and (S1) and/or a set of interface equipments (S2) that are associated with a target modality (S4). Because MST datatype defines a linked list from a top level to a bottom level, it is not scalable for the bottom-up search mechanism. Rather, ModalityOf, InterfaceGive and SubEquipment relations are used. Figure 6-5 shows the manipulation of these relations against the bottom-up search mechanism.

Figure 6-5: Bottom-up search mechanism
Chapter 6. Interface and Modality Binding

The mechanism starts at a target modality (m) located in S4. As specified in Definition 6.12, ModalityOf relation simply defines a set of interface service(s) (S3) associated with the modality ‘m’. The mechanism firstly uses ModalityOf to obtain the information of interface services. Assuming, X = Σx_i is a set of interface services providing the modality ‘m’. Relation (6.14) can be rewritten as:

\[ X = m.\text{ModalityOf} \quad (6.33) \]

Assuming, Y = Σy_i is a set of interface equipments providing the interface(s) in the set ‘X’. The mechanism then applies Definition 6.9, i.e. Interface’s ancestor, to discover a set of interface equipments that are associated with the Set ‘X’. The Relation (6.10) can be rewritten as:

\[ Y = X.\text{InterfaceGive}^{-1} \quad (6.34) \]

\[ y_i = x_i.\text{InterfaceGive}^{-1} \quad (6.35) \]

The mechanism considers the fact that the relations between the MST levels are transitive. That is, if an interface equipment ‘y’ associates with the interface service ‘x’ that offers the modality ‘m’, ‘y’ also associates with ‘m’. Due to this fact, all interface equipments in the set ‘Y’ can render the modality ‘m’. In a similar way, to discover an actual user device providing the target modality ‘m’, ‘SubEquipment is computed. Let ‘d’ be a user device providing the set ‘Y’. According to Relation (6.6), the mechanism obtains that

\[ Y = d.\text{SubEquipment} \quad (6.36) \]

The mechanism computes the inversion of (6.37) to discover the information of the actual device. The inversion is derived as:

\[ d = Y.\text{SubEquipment}^{-1} \quad (6.37) \]

6.2.4 Implementation

Figure 6-6: Libraries that execute MST and Dependencies between Different MSTs
Figure 6-6 shows the implementation of the introduced MST theory. Mainly, the implementation involves ‘structure’ and ‘dependency’ libraries. The first library primarily carries out classes to describe information of a ‘single’ MST. The second computes the relations between the two representation MSTs, i.e. the MST of the UE_C and this of the UE_I.

The ‘structure’ library

![Diagram of MSTStructure and MSTDrawer classes]

Figure 6-7: Classes and package that are implemented in ‘structure’ library

Inside the ‘structure’ library, the ‘MSTStructure’ class defines the actual structure of the MST. The ‘MSTDrawer’ class implements multithreads and callbacks to comprehend the information of each MST. Mainly, these two classes correspond to the Definitions 6.1 to 6.5. Moreover, structure library contains ‘relationBetweenLevel’ package of which the class-members deal with the remaining definitions appearing in Section 6.2.1. Also, the ‘relationBetweenLevel’ package provides the MST search mechanisms described in Section 6.2.3.

![Diagram of relationBetweenLevel package]

Figure 6-8: Classes implemented in ‘relationBetweenLevel’ package
Figure 6-8 shows ‘relationBetweenLevel’ package. The ‘DataType’ class involves Definition 6.6. It contains a number of operations that take a reference node as a parameter. Each operation returns a linked list that represents the path in a particular branch. The path starts from the reference node (parameter) to the modality node.

DataType is associated with ‘MSTSearching’ and DependencyOfConsecutiveLevels. ‘MSTSearching’ class facilitates the search mechanisms described in the Section 6.2.3. Inside this class, the functions ‘getAllDeviceHaveModality()’ and the ‘getIEofModality()’ engage in the bottom-up search mechanism. On the other hand, ‘getModalityfromIE()’ performs the top-down search mechanism.

Next, the ‘DependencyOfConsecutiveLevels’ class describes the relations between two consecutive levels of the MST. This class is related to Definitions 6.8 to 6.13. Inside the class, the ‘getInterfaceGive()’ returns a set of interface services (S3) provided by a particular interface equipment (S2). Two operations, i.e. ‘getInvertIG()’ and getColIEfromOperator(), are similar in the sense that they return a set of interface equipments (S2) controlling a specific interface service (S3). Yet, they are slightly different. An internal algorithm of the ‘getInvertIG()’ is carried out irrespective of the Interface Service Operation, i.e. ⊙ and ⊕. Accordingly, the returned value does not show whether the interface equipments are uncoordinated or synchronised. On the contrary, the ‘getColIEfromOperator()’ uses the concepts of Interface Service Operation, described in Axiom 6.2 and Definitions 6.10 and 6.11. It takes the Interface Service Operation as a parameter, and then returns the value associated with that operation. The function ‘getModalityOfRelation()’ is related to the Definition 6.12, i.e. the ModalityOf relation. It returns a set of interface services (S3) providing a specific modality (S4). In the opposite direction, ‘getInvertModalityOf()’ is associated with Definition 6.13. It returns a set of modality services (S4) operated by an interface service (S3).

For a final class, i.e. ‘DirectionalModality’, corresponds to Definitions 6.14 and 6.15. Once any new device is discovered, the class adds the device’s input modalities into an unfixed-size array ‘inputModality’. Output modalities are added in an unfixed-size array ‘outputModality’.

The ‘dependency’ library
Figure 6-10: Classes implemented in ‘logical’ package

Figure 6-9 illustrates the ‘dependency’ library, which contains ‘logical’ and ‘cluster’ packages. Containing ‘SemanticOfIE’ and ‘SemanticDependency’ classes, a class diagram of the ‘logical’ package is presented in Figure 6-10. Inside ‘SemanticOfIE’ class, ‘getSemanticOfIE()’ and ‘isSemantic()’ operations exploit Theorem 6.1 to compute the semantic of interface equipments between the MST \(_{UE_C}\) and the MST \(_{UE_I}\). The first operation takes in a set ‘A’ and a UE \(_I\) as parameter. Literally, set ‘A’ is the subset of (or equal to) the union set between the interface equipments \((S_2)\) of the MST \(_{UE_C}\) and the MST \(_{UE_I}\). The operator calculates and returns the set of semantic interface equipment on A. ‘isSemantic()’ indicates whether the given interface equipment is semantically identical between UE \(_I\) and UE \(_C\).

Considering the second class, i.e. ‘SemanticDependency’ class, ‘computeSemanticDependency()’ computes type of semantic dependencies between a UE \(_C\) and a particular UE \(_I\). According to Axiom 6.4 and Definitions 6.17 to 6.21, five types of semantic dependencies are defined. If semantic dependency corresponds to PSM-D or NSM, the UE \(_I\) introduces a number of modalities, which are not provided by the UE \(_C\), to a system. Names of these modalities are added in ‘additionModalityNameSet’ attribute. In more detail, the ‘newModalityMap’ attribute maintains not only a modality name, but also the details about modality flow (i.e. input and/or output) and the list of UE \(_Is\) hosting the new modality. ‘retrieveAllModalityNameInSystem()’ gives a set of modalities provided by both UE \(_C\) and UE \(_I(s)\).
Chapter 6. Interface and Modality Binding

Figure 6-11: Classes implemented in ‘cluster’ package

The ‘cluster’ package, illustrated in Figure 6-11, implements the ‘PhysicalDevicesInMID’ class and the ‘ClusterRelation’ class. Firstly, ‘PhysicalDevicesInMID-B’ class declares a number of attributes maintaining the information of user devices in the MID-B environment. As specified in the Section 6.2.2.2, such devices could be categorised as UE_C, candidate UE_I, non-reachable UE_I or UE_I. Secondly, the ‘ClusterRelation’ class computes the cluster relations as described on Section 6.2.2.2.

6.3 Resource Handling

Figure 6-12: MST forest

A generalised MST mainly expresses the context information of a single device. For multiple devices, MID-B extends the principle of MST into ‘MST forest’. The extension applies Equation
(6.21), i.e. \( \text{MID-B} = \text{MST}_{\text{UE-C}} \cup \Sigma \text{MST}_t \). The MST forest adopts an abstract syntax tree of MST to describe the rich mixture of interface and modality characteristics provided by diverse devices.

Similar to the MST, the MST forest is composed of four levels. That is,

\[
\text{MST forest} = \{S_1, S_2, S_3, S_4\} \quad (6.38)
\]

\( S_1 = \text{UE}_C \cup \Sigma \text{UE}_I \) is a non-empty finite set of user devices available in a system.

\( S_2 = \Sigma S_2 \) is interface equipments provided by all of the user devices in \( S_1 \).

\( S_3 = \Sigma S_3 \) is interface services provided by all of the interface equipments in \( S_2 \).

\( S_4 = \Sigma S_4 \) is modality services provided by all of the interface services in \( S_3 \).

Extending the MST theory, at \( S_1 \) level, two distinct types of a connection between a \( \text{UE}_C \) and \( \text{UE}_I(s) \) are defined. Shown in Figure 6-12, the first connection is ‘physical connection’. It indicates that the MST forest is physically built of a number of actual user devices (\( S_1 \)). Secondly, ‘logical connection’ means that the MST forest ‘logically links’ together all the services provided by these physical user devices. The representation MSTs of \( \text{UE}_C \) and \( \text{UE}_I(s) \) are logically united.

Actually, MID-B considers the MST forest as a single virtual device. Resources available in this virtual device are captured at the different levels the MST forest. Initially, the MST forest contains a \( \text{MST}_{\text{UE-C}} \). Likewise; the virtual device includes resources provided by the \( \text{UE}_C \). Once a new device is discovered, its representation MST is added into the MST forest and resources of this discovered device are integrated into the virtual device. In such a case, resources are dynamic. To organise the dynamic information of these resources, MID-B contains a mechanism called ‘resource handling’.

Resource handling is interfacing with a discovery mechanism. Discovery is responsible to search for available and accessible devices. Indeed, these devices correspond to \( S_1 \) in the MST forest. According to Section 5.2, in the three-party model, discovered devices are registered into a ‘device table’. Afterwards, resource handling manages resources that are provided by these devices. In the MST forest, these resources are characterised by \( S_2 - S_4 \). A number of additional tables are implemented to maintain the resource information, including their status and precise identification. If the information needs to be modified, the modification is carried out in an entry to these tables. For instance, the tables facilitate a scheduler which triggers the usability of these resources. When a resource fails to render a service, its records will be tagged from the table.
6.3.1 Interface-Equipment Table and Interface Equipment Handler

A virtual device is embedded with several physical interface equipments that are accommodated on both UE_C and other devices recorded in a device table. Possibly the same interface equipment may be provided by multiple devices. An interface-equipment table is introduced to avoid confusion caused by such duplicate equipments. Such a table is associated to $S_2$ of the MST forest. ‘InterfaceEquipment’ class, shown in Figure 6-13, defines an entry structure for the interface-equipment table. The entry is composed of four attributes. First, a ‘hashcode’ represents identification of the entry. Second, ‘interfaceEquipmentName’ is the name of this interface equipment. Third, ‘deviceOwner’ specifies which user device in $S_1$ hosts this interface equipment. And, ‘isInUse’ indicates whether the interface equipment is currently in use.

Once a device has registered to a device table, new entry(ies) is/are created in the interface-equipment table. This is carried out via the ‘addEntries()’ operation, defined in ‘InterfaceEquipmentHandler’ class. According to Figure 6-13, InterfaceEquipmentHandler provides an operation to access the table, i.e. ‘getTable()’. ‘calculateHashCode()’ computes an identification of the table entry. In addition, this class contains a basic function to search an entry, i.e.‘findEntry()’. And, ‘modifyinUseTag()’ changes a value of an isInUse attribute.

![Figure 6-13: Interface-equipment table and interface equipment handler](image)

6.3.2 Tables Of Input & Output Modalities and Modality Handler

![Figure 6-14: Class defining an entry of input and output modality tables](image)
Information of discovered input and output modalities are stored in separate tables. However, these tables share their entry structure. Each entry contains seven attributes, as shown in Figure 6-14. Similar to the interface-equipment table, an entry of input and output modality tables is uniquely identified by its ‘hashcode’. The entry contains the name of a modality (‘modalityName’) and specifies the device hosting this modality (‘deviceOwner’). In addition, ‘negativeDegree’ shows how much a modality is preferred. A high value of negativeDegree indicates a less preferable modality. The value of this attribute is assigned in accordance with, for example, user physical challenges, user expertise, and context of the environment. ‘isNegativeDegreeLocked’ indicates whether the value of negativeDegree can be changed or not. Next, ‘availabilityTag’ exploits the same principle as in a device table. If a device holding a modality is withdrawn from the service vicinity, the ‘availabilityTag’ of all modalities provided by that device is set be false. The tag will be reset when the device re-attaches to the vicinity. Importantly, values of ‘availabilityTag’ in three tables, i.e. the tables of device, input modality, and output modality, must be synchronised. Another attribute, ‘isInUse’ suggests whether a modality is in use.

Figure 6-15: Modality handler

The tables of input and output modalities are maintained in a modality handler, shown in Figure 6-15. In ‘ModalityHandler’ class, fundamental operations for modality administration are provided. ‘addEntry()’ creates new a record of modality. Similar as for the interface equipment handler, ‘calculateHashcode()’ computes an identification of each record (i.e. table entry). ‘findEntry()’ searches a record of a modality in a particular table (i.e. either input or output).
However, a system may have no knowledge about the direction of a modality. Hence, ‘findTableStoringAModality()’ identifies a table associated with a given modality. Next, the ‘modifyModalitiesOfADevice()’ operation updates records of all modalities that belong to a particular device. This operation is useful when those records need to be updated with the same information. ‘modifyIsInUse’ changes a value of isInUse attribute.

If a modality provides less efficiency due to user’s physical restrictions, ‘modifyModalityAffectedByPhysicalChallenges()’ is invoked. This operation firstly modifies a value of negativeDegree. It then checks whether the negativeDegree is set to a maximum value. If yes, the operation then sets isNegativeDegreeLocked to true.

In addition, modality handler contains ‘NegativeModality’ class that gives essential operations to ensure that all negative modalities due to disability of physical (sensing) organ(s) are inspected. Inside this class, ‘modifyNegativeModality()’ exploits the user and affected-interface profiles to gain the information on negative modality(ies). Its mechanism is virtualised in Figures 6-16.

![Figure 6-16: Internal mechanism of ‘modifyNegativeModality’ function](image)
From Figure 6-16, the two profiles are imported. In the user profile, information about user's physical challenges is provided. This information is formed as Equation (4.6); i.e. \( P = \sum(t, A) \). According to Equation (4.6), a set of affected (sensing) organs for a particular user is obtained from ‘A’. In the affected-interface profile, information is maintained as a 4-tuple database as shown in Equation (4.8); i.e. affected-interface profile = \( \sum(d, O, c, I) \). ‘O’ represents a set of relevant physical (sensing) organs.

Let ‘a’ be an element in set A. If \( a_i \) is contained in set O, I (i.e. set of interfaces perhaps affected by any disability of the sensing organ O) is concerned. ModalityOf relation is applied to find a set of ‘possible’ affected modalities (\( M_i \)). \( M_i \) corresponds to all modalities provided by all interfaces in set I. In \( M_i \), both input and output modalities are still mixed. However, a system needs to ensure that only relevant modalities are taken into account. This is indicated by a modality direction. In general, modality direction is opposite to human perception. For example, a screen provides visual as information output modality, but from a human’s point of view, visual is an input perception. ‘c’ in the 4-tuple database indicates a human perception associated with the sensing organs in O. Therefore, a conversion of ‘c’ indicates an actual direction of modalities that will be affected. Assuming ‘\( \sim c \)’ represents a conversion of c. Let \( M_2 \) be a subset of \( M_1 \); \( M_2 \) is obtained by removing modalities of which the direction is not matching with \( \sim c \). Subsequently, by calling ModalityHandler.modifyModalityAffectedByPhysicalChallenges(), records of modalities in \( M_2 \) are modified.

Further, the principle of ‘Interface Service Operation’ is exploited. Literally, \( m \in M_2 \) may require multiple interface equipments. Assuming IE is the set of those required interface equipments. IE is computed by a function getCoIEfromOperator() in DependencyOfConsecutiveLevels. MID-B exploits the facts that ‘in addition to \( m \in M_2 \), other modalities provided by \( iee \in IE \) ‘may’ be affected as well (but not always)’. So, the set IE is passed into a private function operateRelevantIE(), which is responsible to discover those modalities and forward their information to a recommended component.
Figure 6-17: Internal mechanism of ‘operateRelevantIE()’ function

Figure 6-17 presents an internal mechanism of ‘operateRelevantIE()’. $M_{\text{mayAffected}}$ is declared as an empty set. Then, MSTSearching.getModalityfromIE() is called to discover a set of modalities provided by $\text{ie} \in \text{IE}$. Assuming, this modality set is called $M_a$. Generally, $M_a$ includes some elements which have existed in $M_2$. These duplicate elements need to be removed from $M_a$. After removing them, $M_a$ is added to $M_{\text{mayAffected}}$. As shown in Figure 6-17, this process is iterative until all elements in set IE have been examined.

Let $m$ be an element in $M_{\text{mayAffected}}$. The direction of $m_i$ is investigated. If $m_i$ is an input modality, information of $m_i$ is added to a list of negative inputs in a recommendation component. Otherwise, it will be added to a list of negative outputs. The recommendation component will be described in Section 6.4.
In short, the NegativeModality class helps the modality handler to control the preferences of modalities. This class deals with a user’s physical challenges. Any affected modality is tagged in the modality table. In addition, a set of ‘possible’ affected modalities is forwarded to a recommendation component and recorded in the list of negative modalities. A sequence to examine those user physical challenges, as described above, is concluded in Figure 6-18.

6.4 Interface and Modality Binding

6.4.1 Theoretical Approach to Interface and Modality Binding

MID-B defines interface and modality binding as “a mechanism that enables a UI to react to the changes of virtual capabilities provided by more than one physical device”. With the three components defined in MID-B, binding is mostly accommodated on the ‘MSB’, which provides an interaction management.

This section introduces the theoretical approach to interface and modality binding. This approach presents the logic to understand how to context information can be exploited. The approach formalises a ‘binding function’ by using the model of a context ‘space’ model [77]. In the context space model, an accepted range of a context value is specified. The most appropriate ‘interaction means’ is examined by comparing the context values with the accepted range. In MID-B, the appropriate ‘interaction means’ does not only consider an interface service. It rather is a compound group that is characterised by different levels of MST. That is, the group contains ‘a
device’, ‘interface equipment on that device’, and ‘a modality service’. The selection must comply with the accessibility and utilisation of all constituents of that group. Let $B(t, m)$ be the binding function.

$$B(t, m) = (A, A_v, A_r, E, s)$$  \hspace{1cm} (6.39)

t $\in$ T, when T is an enumeration set which contains the constituents inside the group of interaction means. $T = \{ \text{device, interface equipment, modality} \}$.

m is the identification of t.

A = $\{ A_1 \cup A_2 \mid A_1$ and $A_2$ are the sets representing hard-value and soft-value attributes (or contexts), respectively.$\}$ According to MID-B’s scope described in Chapter 2, set A is an ‘internal’ (active) context and placed inside an application’s domain.

$A_v = \{ A_{v1} \cup A_{v2} \mid A_{v1}$ and $A_{v2}$ are the sets of attribute values on $A_1$ and $A_2$, respectively.$\}$

$A_r = \{ A_{r1} \cup A_{r2} \mid A_{r1}$ and $A_{r2}$ are the sets of an accepted range on $A_1$ and $A_2$, respectively.$\}$

E = $\{ e \in E \mid E$ is a set of external contexts.$\}$. According to Section 2.3.1.2, in MID-B, an environment context is grouped into a category of external contexts. E will not be sensed, unless the values in $A_v$ are ambiguous.

$s$ indicates the status of a service provided by ‘(t, m)’, determining whether it is suitable under the conditions given by $A_v$.

In (6.39), an attribute in set A is categorised into hard-value and soft-value. Firstly, a hard-value attribute ($A_1$) is a critical and mandatory context. $A_1$ strongly affects a usability of a service provided. Its value ($A_{v1}$) is sensitive. Any attribute value that is outside the accepted range $A_{r1}$ causes a service provided by ‘(t, m)’ to be ‘immediately’ rejected. Then, ‘s’ is set to false. The condition is expressed as:

$$av_1 \in A_{v1}, ar_1 \in A_{r1}, \exists av_1 \forall ar_1, av_1 \not\in ar_1 \rightarrow s = false$$  \hspace{1cm} (6.40)

Secondly, a soft-value attribute ($A_2$) is optional. Providing some values are within the domain of $A_{r2}$, this increases the possibility that a service from (t, m) will be employed. In such a case, the employment of (t, m) can be confirmed by sensing the contexts in E. This condition is expressed as:

---

21 In Mathematics, an enumeration set defines a certain name space for a specific object.
Chapter 6. Interface and Modality Binding

\[ \exists a v_2 \exists a r_2, a v_2 \in a r_2 : \quad (6.41) \]

Providing both hard-value and software attributes are located in a domain of accepted ranges, a service provided by \((t, m)\) is trusted. \(E\) is not necessarily being sensed. This condition is expressed as:

\[ \exists a v_1, a v_2 \exists a r_1, a r_2 : \quad (6.42) \]

Attribute Format

Generally, an attribute is represented in numerical or non-numerical formats. Based on the two representation formats, two distinct ways to compare between an attribute value and its accepted range are applied. Regarding a numerical attribute, the generalised comparing operations are exploited. These operations include 'greater or equal' (\(\geq\)), 'greater' (\(>\)), 'smaller or equal' (\(\leq\)), 'smaller' (<), and 'equal' (=). In a real situation, often, information of context is formatted as non-numerical. The operations provided in set theory are used. These operations are 'not' (\(-\)), 'union' (\(\cup\)), and 'intersection' (\(\cap\)).

The 'not' operation is used for Boolean attributes. Let \(a v_i \in A v_i, a r_i \in A r_i\), and \(i \in I = \{1, 2\}\). if \(a v_i = -a r_i\), attribute value is outside the accepted range.

Considering the fact that a union set of two identical sets is also identical. Given that \((a v_i \cup a r_i) = a v_i = a r_i\), an attribute value is equal to an accepted range.

Next, an intersection operator gives a new set containing a shared region between \(a v_i\) and \(a r_i\). An empty set of interactions \((a v_i \cap a r_i = \emptyset)\) indicates that \(a v_i\) is not in an accepted range of \(a r_i\). The illustration of the binding function is presented in Figure 6-19.
Chapter 6. Interface and Modality Binding

6.4.2 Binding Mechanism

The Interface and modality binding is closely related to resource handling. In the resource handling, a number of tables are implemented to organise context information of a device and its resources. The binding then manipulates such information and responds to a change of I/O
interface widgets located in an environment. In the binding, three main components are invoked, shown in Figure 6-20. These components include binding mechanism, recommendation, and context control. In each component, a number of classes are implemented. Figure 6-21 depicts the dependencies and associations between classes of the three components.

![Figure 6-20: Three components for interface and modality binding](image)

**Figure 6-20: Three components for interface and modality binding**

![Figure 6-21: Dependencies and associations between classes of the three components](image)

**Figure 6-21: Dependencies and associations between classes of the three components**
In the ‘binding mechanism’ component, most binding tasks are carried out. This component delivers the final result of a binding procedure to the system. In principle, the result is gaining knowledge of three facets, as shown in Figure 6-22.

- A selected modality, \( m \in S_4 \) of MST.
- A set of selected interface equipments rendering the modality \( m \), \( IE \in S_2 \) of MST. Note that, the size of \( IE \) is greater than one if \( m \) requires at least two synchronising equipments, and
- Routing information. If \( ie \in IE \) is hosted on an \( UE_I \), the routing information provides the details of a selected connectivity method used to retransmit data content stream from a \( UE_C \) to the \( UE_I \).

If the binding mechanism encounters an uncertain situation, it consults a ‘recommendation’ component, where lists of preferences and negative modalities are maintained. Inside the recommendation component, the concept of ‘fake \( UE_C \)’ is introduced. A fake \( UE_C \) is “a \( UE_I \) that holds a CSM relation and the interaction services provided by this \( UE_I \) can substitute the services from the \( UE_C \)”. According to Definition 6.18, for the CSM relation, modalities (including their capabilities) hosted on the \( UE_I \) coincide with those of the \( UE_C \). Once the \( UE_C \) fails to deliver a user interface, it could be (logically) replaced by such an \( UE_I \). MID-B ‘temporarily’ considers this \( UE_I \) as core device. Under the circumstances, this replacement of the core device is considered as a ‘fake \( UE_C \)’. According to Figure 6-21, UECSwitch is an object defined in the recommendation component. It comprises of 1) a Boolean attribute indicating if any \( UE_I \) is functioning as a fake \( UE_C \), and 2) a string representing the address of that fake \( UE_C \).

In addition, the recommendation component suggests whether a system operates an ‘automatic’ or ‘manual’ selection. In the automatic selection, the result of a binding is given without user.
intervention. In exception that the result is still vague, a final decision will be made by a user. In contrast, in a manual selection, a user participates in every step of the binding.

In case that binding is facing a difficulty, a PopupAgent in the recommendation component is responsible to throw a warning or error message. For the manual selection, the PopupAgent also generates a warning message for the user.

For a final component, the context control provides an interface between the binding and context information. Through the context control component, the binding mechanism can access the profiles of an application and the context-brokers. In addition, this component is responsible to select and pass the information about a chosen connectivity method to the binding mechanism component.

The following section presents a sequence of interface and modality binding mechanism as well as shows how the three components are interfacing.

6.4.2.1 Sequence of Interface and Modality Binding

Figure 6-23 depicts a sequence of interface and modality binding. The sequence is divided into four phases, i.e. ‘initiating’, ‘binding’, ‘modifying resource tables’, and ‘throwing an error message’. The initiating phase collects a number of profiles needed for the binding. Next, the binding phase applies the binding function (6.39) to interpret and make use of context information. Later, the phase of modifying research tables updates the records of selected resources. Finally, the phase of throwing an error message handles an error that occurs during the binding.
Figure 6-23: Sequence of interface and modality binding

Figure 6-24: Phase to initiate the binding mechanism

Figure 6-24 illustrates more details of the initiating phase. During the initiating phase, MSB passes an application ID to Binding. A context broker profile and a presentation adaptation profile
Chapter 6. Interface and Modality Binding

are retrieved from ContextBroker and Adaptor, respectively. An application profile, including the information about application confidentiality, such as visibility, is obtained from AppPrivacy. Next, Binding consults Recommendation to analyse whether public devices are able to be utilised for this interaction session. The analysis is based on the confidential information of an application. The binding function in (6.39) is applied. User location, for example, is defined as a hard-value attribute. According to function (6.39), a binding function is derived as:

\[ B(UE_I, -) = ((\text{user location}), (AV_1 \cup AV_2), ((\text{private area}) \cup -), -, s) \]  
(6.43)

A dash '-' means an unspecified element. At this state, UE_Is are examined in general. Hence, identification of a particular UE_I is not required. Providing visibility of an application is private, UE_Is are allowed if and only if a user is currently in a private area. To find a value of AV_1, Recommendation asks BindingHelper for an identification of a context provider that hosts the information about a user location. Assuming, 's' is false (user is not in a private area), the DeviceHandler sets all UE_I to a hibernation state.

Afterwards, Binding requests Recommendation to inspect a conditions giving impacts on a performance of UE_C. UE_C’s battery remaining is used as an example. The binding function is then derived as:

\[ B(UE_C, UE_C \text{ address}) = ((\text{battery}), (AV_1 \cup AV_2), ((\text{battery remaining}) \cup -), -, s) \]  
(6.44)

To examine the status 's' in (6.44), the Recommendation component uses the private function 'checkBattery()'. If a battery charge is less than a minimum level, one of the following cases will occur.

- Case 1: The device table is empty. PopupAgent throws a warning message to a user. The user is forced to use the UE_C (UE_C will be then physically charged). And, 's' in the binding function (6.44) is set to true.
- Case 2: An availability tag of every entry in the device table is false. The initiating phase is continued as in case 1.
- Case 3: In (6.43), s is set to false. The initiating phase is continued as in case 1.
- Case 4: There are UE_Is present in a system, but none of them encompasses a CSM relation. Inside the Recommendation component, ‘automaticallyAdapting’ suggests whether MID-B can select a device without querying a user. If it is set as true, 's' in a binding function (6.44) is automatically set to false. UE_C now only provides data content, but not an interaction service. In opposite, automaticallyAdapting is false. MID-B consults the user by delivering a message to ask the user to select between:
Continuing to use the UE_C (UE_C must be physically charged and ‘s’ is true),
or
employing other services from UE_I(s) (‘s’ is false).

- Case 5: There is at least one UE_I with CSM relation and ‘s’ in (6.43) is true. This case extends the concept of the four previous cases by exploiting the idea of a ‘fake UE_C’. Yet, automaticallyAdapting in Recommendation is taken into account. If it is true, a fake UE_C is operated. The UECSwitch object in a recommendation component, see Figure 6-21, maintains an address of this fake UE_C. But if automaticallyAdapting is false, a user has three options:

- Using the UE_C. The UE_C must be physically charged. In (6.44), ‘s’ is true. UECSwitch.switched = false and UECSwitch.address = null.
- Using available UE_I(s). In (6.44), ‘s’ is false. UECSwitch.switched = false and UECSwitch.address = null.
- Using a UE_I that has CSM relation as a fake UE_C. In (6.44), ‘s’ is false. UECSwitch.switched = true and UECSwitch.address = an address of this UE_I.

Figure 6-25: Phase to operate a binding mechanism

After the initiating phase, the sequence enters the binding phase where a binding mechanism takes place. Figure 6-25 depicts the sequence of this phase and outlines the messaging between the
components. 'Binding' passes an application ID to 'Mechanism'. A modality requested for the application is examined. 'Mechanism' then requests information regarding a fake UE_C from 'Recommendation'. This information is used to examine an address of a core device employed for this particular interaction session. If a fake UE_C is incorporated within a system and 's' in (6.44) is false, an address of such a fake UE_C becomes a core address. Otherwise, an address of the actual UE_C is used.

Next, Mechanism obtains the tables of input and output modalities from ModalityHandler. Mechanism then discovers a set of device(s) that hosts a requested modality (Bottom-up search mechanism in Section 6.2.3 is used). Depending on a number of devices rendering this requested modality, there are five different algorithms to continue the binding mechanism. BindingUtil is responsible to compute a code indicating which algorithm will be used. The five algorithms include:

- **Code 1**: A requested modality 'is not' available within an environment. Hence, relation information between the modality (S₄) and other components on MST can not be investigated.

- **Code 2**: A requested modality has 'physically' existed in one of the modality tables, but it is 'logically' considered as a non-available modality. For instance, this may be triggered by the availabilityTag (set to false). Alternatively, such a modality may appear in a list of negative modalities maintained in Recommendation. In this case, MID-B can learn information related to this original modality from a MST. Finding an alternative modality will be carried out based on this information and the MST.

- **Code 3**: A request modality is provided on 'only' a UE_C (or a fake UE_C).

- **Code 4**: A request modality is 'not' provided on a UE_C (or a fake UE_C), but it is on UE_I(s).

- **Code 5**: A requested modality is provided on both UE_C (or fake UE_C) 'and' UE_I(s).

According to Figure 6-23, after the binding has been completed, ModalityHandler modifies inUseTag of a selected modality. And, InterfaceEquipment modifies inUseTag of the interface equipment(s) exploited in this interaction session. For any error case, PopupAgent throws an error message.

### 6.4.2.2 Basic Algorithms for the Binding Mechanism

This section presents the basic algorithms to show the proof-of-concept of the theories defined in MID-B. In this thesis, these algorithms are developed at a top-level. Figure 6-26 presents the top-level flowchart explaining the binding mechanism. It starts by checking the algorithm code. Five
Possible codes have been defined above. In codes 1 and 2, an original modality cannot be manipulated. Hence, alternative modalities must be computed. On the other hand, code 3, 4 and 5 use the original modality. The mechanism to select new modality is therefore omitted.

Figure 6-26: Top-level flowchart: binding mechanism
Algorithm for codes 1 and 2

For codes 1 and 2, seven steps are invoked. First, an algorithm investigates a presentation-adaptation profile to explore a set of alternative modalities (AlternativeModality). Second, me AlternativeModality is evaluated by using the modality tables. Any m that has not been registered in the tables is removed from the AlternativeModality set. Third, a set of positive modalities (PositiveRecom) is retrieved from a recommendation component. Further, this third step proceeds as following:

- Compute the intersection between the sets of alternative modalities and positive modalities, i.e. PositiveRecom = PositiveRecom ∩ AlternativeModality
- Investigate the availability tag of every modality in PositiveRecom, and remove any modality that is not currently in reach from the set.
- Examine the algorithm code.
  - Code 1, a requested modality is not available in the system. The third step is terminated
  - Code 2, a requested modality has physically existed in a modality table, but it is logically considered as a non-available modality. According to the MST theory, the algorithm is able to learn about interface services (S3) that provide the original modality (S4). The algorithm uses ModalityOf relation to separate positiveRecom set into two smaller sets as:
    - positiveRecom_1 containing modalities (S4) provided by the same interface (S3) as an original modality, and
    - positiveRecom_2 containing modalities (S4) provided by the different interface (S3) as an original modality,

Next, the forth step considers a fact that an agent of presentation adaptor may give a new modality of which a format is not compatible with available devices. To avoid this, the algorithm compares the abilities of the available agents with the devices in a system. A modality with an incompatible format is then removed from PositiveRecom (for code 1), or PositiveRecom_1 and PositiveRecom_2 (for code 2). Fifth, preferences of modalities in these sets are sorted by using the principle of NegativeDegree attribute. Sixth, select a modality based on the set(s) that is derived at the fifth step. Figure 6-27 depicts a sequence of the sixth step.
In the sixth step, Binding consults Recommendation to select an appropriate modality. Recommendation obtains a modality table from ModalityHandling. Only a modality with availabilityTag is true and isInUse is false is taken into account. Any irrelevant modality is then removed from PositiveRecom (for code 1), and PositiveRecom_1 and PositiveRecom_2 (for code 2).

Then, the binding function (6.45) is applied to every modality in these sets.

\[
B(\text{modality name, hashcode}) = (A, A_v, A_R, E, s)
\]  

(6.45)

Generally, A and \(A_R\) in (6.45) are assigned differently, depending on, for example, type of modality.

Since MID-B is a virtual device, a modality could be hosted on different physical user devices. Yet, in some occasion, Recommendation may select a modality that is currently functioning in one device, but not another. Commonly, such a modality should be rejected. An ‘automaticallyAdapting’ in Recommendation suggests whether the algorithm should automatically return to the beginning of the sixth step to find a new modality. Or, a PopupAgent throws a warning message to the user. The user then can decide to continue using the selected modality or finding a new modality.

Once a modality has been selected, the sixth step is also responsible to re-compute an algorithm code. A number of devices that provide the selected modality is discovered. Based on the five
code explained above, a new code can be allocated depending on where the new selected modality is contained.

For the final step of the original algorithm codes 1 – 2, interface equipment used to render the selected modality is investigated. This will be thoroughly explained later in the following section.

**Algorithm codes 3 to 5**

Considering the original algorithm codes 3 – 5, a requested modality is currently available and accessible on UE_C and/or UE_I. And, the modality must not be included in the negative lists in the Recommendation component. Such a modality is assumed as the most appropriate. A mechanism to replace the original is not required. According to Figure 6-26, for codes 3 – 5, the binding algorithm firstly examines a modality table. All table entries that are associated with the original modality are obtained. Any entry with availabilityTag is false is ignored. In MID-B, one modality is allowed to be located in several interface equipments. Hence, similar to codes 1 and 2, the binding algorithm finally enters a step to select appropriate interface equipment.

**6.4.2.2.1 Function ‘checkInterfaceEq()’**

Selecting interface equipment (S2) invokes the function ‘checkInterfaceEq()’ in BindingUtil. As shown in a class diagram in Figure 6-21, this function takes in three parameters as follows.

- A set of modality entries. Each entry details the capabilities of the selected modality (S4) for a particular user device (S1) and specific interface equipment (S2).
- An algorithm code. If an original algorithm code is 1 or 2, the new re-computed code, described in the previous section, is passed into this function. Otherwise, an original code 3, 4 or 5 is passed.
- The details of media content of an application.
Figure 6-28: The function ‘checkInterfaceEq()’

Figure 6-28 illustrates the algorithm taking place in checkInterfaceEq() function. This function firstly declares an empty set ‘A’. For every modality entry (the first parameter), the function calls the getIEofModality() function, defined in MSTSearching, to find a list of interface equipments that provide the selected modality. This list is added into the set A. After all entries are examined, the size of set A is checked.

- If the size of set A is greater than one, the binding consults a Recommendation component by using an function consultToSelectAmbiguousIEs(). The algorithm code and the details of media content (the second and the third parameters) are passed to this function.

- Providing A’s size is equal to one indicates that only one interface equipment on a particular user device provides the selected modality. The function checkInterfaceEq() then examines the algorithm code (the second parameter).
If a device is an UE_C (code 3), a final result of binding is now derived. And, the binding is then terminated. The final result is gaining its knowledge as follow:

- For the original algorithm code 1 and 2, a modality name corresponds to a new selected modality. If the original code is 3, a modality name corresponds to an original modality.
- Selected Interface equipment. The function checkInterfaceEq() ask for an interface-equipment table from an interface equipment handler. A record of this interface equipment is retrieved from this table.
- Because an interaction session is carried out on the UE_C, information about content routing is null.

On the other hand, an algorithm code 4 suggests that the selected interface equipment is accommodated on an UE_I. The function ‘Connectivity.getConnectivity()’ is called to compute how to reroute the media content between the UE_C and the UE_I. The algorithm code and the details of media content (the second and the third parameters) are passed to the called function.

### 6.4.2.2.2 Function ‘consultToSelectAmbiguousIEs()’

The function ‘consultToSelectAmbiguousIEs()’ is called when ‘checkInterfaceEq()’ encounters the situations that there are ‘several’ interface equipments \((S_2)\) in a system rendering a particular modality service \((S_4)\). In MID-B, ‘several interface equipments \((S_2)\)’ is twofold. It can refer to ‘different’ types of interface equipments \((S_2)\) hosted on the ‘same’ or ‘different’ device \((S_1)\). Alternatively, it can imply the ‘same’ interface equipments \((S_2)\) hosted on different’ devices \((S_1)\).

The binding function (6.46) is applied to all interface equipments.

\[
B(\text{interface equipment, hashcode}) = (A, A_v, A_R, E, s) \quad \text{(6.46)}
\]

Generally, \(A\) and \(A_R\) in (6.46) are assigned differently, depending on type and hashcode of the interface equipment. Yet, this thesis considers a method to transmit data content between a UE_C and UE_I as one of the vital factors. Therefore, attribute \(A\) in (6.46) is derived as:

\[
A = ((a_I \in \text{hard-value attributes}) \cup \text{connectivity}) \cup (b_I \in \text{soft-value attributes}) \quad \text{(6.47)}
\]

To select a connectivity method, a function ‘getConnectivity()’ implemented in ‘Connectivity’ class is invoked. This method will be thoroughly explained in the following section.
However, there could be more than one interface equipments with status ‘s’ set to true. This implies the selection of interface equipment is still ambiguous. Then, a final selection will be manually made by the user.

![Diagram](image)

**Figure 6-29: The function consultToSelectAmbigiousIEs()**

### 6.4.2.2.3 Function 'getConnectivity()' 

Generally, a connectivity selection is needed if an interaction session can take place on UE_I. This invokes the 'getConnectivity()' function.

![Connectivity Information](image)

**Figure 6-30: Connectivity information**
getConnectivity() gains knowledge about an underlying protocol selected for a particular interaction session. As shown in Figure 6-30, this knowledge includes:

- An address of the device deployed for this interaction session.
- A hashcode of the interface equipment on that device.
- Routing information. Routing information describes the details how to retransmit a data content stream to an UE_I. From Figure 6-30, this information is composed of four attributes.
  
  o ‘isReroute’. Although discovery in MID-B is based on an ah-hoc manner (using Bluetooth), MID-B allows the freedom to transmit data content between the two devices. Mainly, isReroute is a Boolean used to indicate whether data transmission is operated in ad-hoc or infrastructure style. Value ‘true’ means the transmission is taking place in an infrastructure network. Data content is transmitted to the UE_I with assistance from a third-party, such as a router. In contrast, the attribute is set as false. The data content is transmitted by an underlying Bluetooth protocol.
  
  o ‘connectivityMedia’. This specifies a certain protocol to deliver the data content. A value of this attribute must match with isReroute. For instance, WLAN or LAN is relevant to isReroute equal to true.
  
  o ‘reRouteType’. This attribute is null when isReroute is false. Otherwise, this attribute is characterised by an enumeration set T. T = {local resource, remote resource}. Mainly, reRouteType specifies where data is located. For instance, data may be locally stored on the UE_C. Alternatively, it could be remotely maintained somewhere in a server connected to a MID-B environment.
  
  o ‘routerURL’. The meaning of this attribute is related to reRouteType. Assuming reRouteType is ‘local resource’, routerURL is an ‘identifier’ of a router. For a ‘remote resource’, routerURL is an ‘URL’ where the data is stored.

With the explanation above, three possible outcomes could be gained from the routing information. They are summarised in Table 6-1.
Table 6-1: Information how to reroute a data content stream to an UE_I

<table>
<thead>
<tr>
<th>isReroute</th>
<th>connectivityMedia</th>
<th>reRouteType</th>
<th>routerURL</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>Null</td>
<td>Null</td>
<td>Bluetooth</td>
</tr>
<tr>
<td>True</td>
<td>WLAN, LAN, etc.</td>
<td>Remote resource</td>
<td>URL where the data is stored</td>
</tr>
<tr>
<td>True</td>
<td>WLAN, LAN, etc.</td>
<td>Local resource</td>
<td>Identifier of a router</td>
</tr>
</tbody>
</table>

To execute getConnectivity() function, two parameters are passed. The first parameter is an algorithm code. In general, connectivity selection is needed when UE_I is involved; this is associated with the algorithm codes 4 and 5. According to Figure 6-27, if an original code is code 1 or 2, the code will be converted to 3, 4, or 5, once a new modality has been selected. The new code is then passed into this function. Next, the second parameter contains information about content of an application. This information is gained from an application profile.

An algorithms inside getConnectivity() function are visualised in Figures 6-31 to 6-34. The algorithm firstly examines the location of data content. Figure 6-31 illustrates a procedure on condition that data is maintained in a remote server. In Figure 6-31, for every UE_I hosting a selected modality, the algorithm investigates communication characteristics maintained in the UE_I profiles. Next the algorithm analyses whether there is any UE_I in an environment can be directly connected to the remote resource or not.

- If not, the algorithm code is examined.
  - If the code is 5, the UE_C can be directly connected to the remote resource and provides a selected modality. The binding mechanism is then terminated.
  - For code 4, none of UE_I is networked with the remote resource and the UE_C does not support the selected modality. Hence, data is sent to the UE_C. Then, it is task of the UE_C to forward the data to other devices. In such a case, the algorithm will continue as shown in Figure 6-32.

- Or else, the algorithm continues to select a device to which the remote content will be sent. Often, a device has multiple communication methods. UE_C and UE_I is able to be connected to the remote resource by several network interfaces. However, the network should be activated accordingly to the needs of a content stream. The function selectDeviceProvidingMaxTransferRate() denotes the criteria for selecting the precise network interface and a particular device used for this interaction session. A maximum
data transfer rate is used as an example. A device and precise network giving the maximum transfer rate to be selected.

- If the selected device is an UE_C, further computing for routing information is not required.
- In contrast, a UE_I is nominated to carry out the user interface service. A private function `getCommunicationFeature()` is then called. This function explores the profile of this selected UE_I and discovers the information about the network interface between this UE_I and the remote resource.

![Diagram](image)

**Figure 6-31: Function getConnectivity (1)**

An algorithm shown in Figure 6-32 is executed under two conditions. Firstly, data content is locally maintained on a UE_C. Secondly, none of UE_Is can be directly connected to a remote storage and the algorithm code is 4. In Figure 6-32, an empty set BT is declared. The algorithm...
considers the fact that to operate a specific task; UE_C and UE_I must implement the same basic Bluetooth profiles. A UE_I that satisfies this fact is added to the BT set. For every element in this set, the algorithm calculates the duration of time consumed to transfer data, via Bluetooth, from UE_C to UE_I. If this is over a defined limit, the UE_I is removed from the BT set. Next, the algorithm code is examined. For code 4, an algorithm continues as shown in Figure 6-33. Figure 6-34 is carried out if the code is 5.

![Diagram of Function getConnectivity](2)

In Code 4, a selected modality is not accommodated on a UE_C. Importantly, data must be sent to the UE_I. Figure 6-33 explains the algorithm at a top level. The algorithm firstly investigates size of the BT set. A non-empty set implies that Bluetooth is used as a connection media and transmission is performed as an ah-hoc manner. A UE_I selected for this interaction session is included in the set. Yet, there could be more than one UE_I in the set. If there is equal preference, the user will manually make a final selection.
On the other hand, BT is empty. According to a computation in Figure 6-32, Bluetooth connectivity is not preferable. Conveying a data stream from a UE_C to a UE_I needs a third-party. Under this condition, the content-router profile, maintaining a mapping between a device and a supporting router, is investigated. In an error case, none of router supports the UE_Is in an environment. An error message is then thrown. Or else, the algorithm calls a private function selectDeviceProvidingMaxTransferRate() to calculate a rate to transmit data stream from UE_C to UE_I via a supporting router. This function returns an address of UE_I that provides maximum transfer rate. Such an UE_I is considered as a selected device. Next, getCommunicationFeature() returns information about supporting underlying protocol exploited in the is transmission.

However, assorted UE_Is could provide the same transfer rate. A final device selection will be made by the user. Again, getCommunicationFeature() delivers the information of connection media.
The algorithm shown in Figure 6-34 expands the mechanism of Figure 6-33. In the extension, a UE_C is taken into account (an algorithm code is signified as 5). As shown in Figure 6-34, the information from the content-router profile is obtained. The algorithm declares set R as an empty set. Mainly, Set R maintains the information of UE_I and its supporting router. For every element in the set, duration of time used for transmitting data stream from UE_C to UE_I, via a supporting router, is calculated. If it is over a defined limit, that UE_I is removed from R. Subsequently, the algorithm considers the sets R and BT (BT was calculated in Figure 6-32). Providing neither of them contains any element, the UE_C-UE_I communication will fail. Hence, a user interface on the UE_C will be deployed.
6.5 Summary

In this chapter a theoretical approach to unite a range of physical devices into one single virtual device is presented. To describe the diversity of available interfaces and the constraints imposed by these interfaces, this chapter takes advantages of mathematical methods to form a Modality Service Tree (MST). The MST is formed and visualised by exploiting graph theory. It introduces a graphical syntax to express interaction means of a device at different levels of abstraction. In the MST, mappings between these levels are specified. Because MST is based on mathematical expressions, it is generic and can be used for any device type. MST facilitates MST search mechanism. Such a mechanism can explore information hosted on MST at any level.

In a complex environment with rich physical interface devices, this chapter uses the MST to define the rules addressing relational dependency between modalities offered by a number of devices. In MSB, these devices include a ‘control device’ (UE_C) and ‘public devices’ (UE_J). Assuming, a device ‘A’ is a public device and a device ‘B’ is a control device. Fundamentally, this chapter identifies the dependencies to express the semantic relation between modalities as five following cases.

- A set of modalities hosted on the two devices are identical.
- The two devices provide the same set of modalities, regardless of their capabilities.
- Some of modalities provided by device ‘A’ are also provided by device ‘B’. Meanwhile, all modalities of device ‘B’ are provided by device ‘A’.
- Some of modalities provided by device ‘A’ are not provided by device ‘B’.
- All modalities provided by device ‘A’ are not provided by device ‘B’.

Further, this chapter presents the mechanism for resource handling. The mechanism extends the principle of ‘MST’ to an ‘MST forest’. Similar to a single MST, the MST forest is formed as the hierarchical structure, of which each layer presents different information. MID-B considers the MST forest as one single virtual device. The MST forest includes the patterns to describe the combination of multiple input/output modalities that are provided by different devices. In MID-B, resource information is associated with the syntax of the MST forest.

Finally, this chapter formalises the theoretical approach for interface and modality binding. Due to the binding, MID-B facilitates new virtual capabilities for more than one physical device. The binding employs a context space model to consolidate information of contexts. Different categories of context are taken into account. Two main categories are internal and external context. The internal context is further grouped as ‘hard-value’ and ‘soft-value’. At the end of this
chapter, a ‘basic’ algorithm of the binding mechanism is presented. In the following chapter, the thesis conclusions are provided. Also, suggestions for future work are presented.
Chapter 7

7 Conclusions and Further Research

7.1 Thesis Conclusions

This thesis addresses and documents a fundamental theoretical solution to form a ‘virtual’ device environment. Throughout the thesis, three facts are considered:

- The design of mobile user interfaces (UIs) is becoming increasingly diverse and advanced and any design process should consider and exploit the wide range of modalities that are available in modern mobile terminals.
- However, still UIs are restricted by the mobile terminal’s physical interface features and capabilities.
- The environment surrounding a mobile user is equipped with a wide range of additional user interfaces; they are hosted on ubiquitously available devices, which provide enhanced features and capabilities. This includes screens, audio systems, as well as printers and so forth.

Taking advantage of these interfaces, this thesis extends the traditional limited UI into a multimodal multi-device environment. The thesis describes the resulting construct as a ‘virtual device’. A theoretical concept to describe this virtual device is introduced. For the proof-of-concept, an architectural framework called ‘Multi Interface-Device Binding (MID-B)’ is introduced. The thesis describes MID-B. Its event-driven mechanism is visualised by means of a deterministic Finite State Machine (FSM). In the context of this state machine, the sequential logic-based implementation of the MID-B system is described.

MID-B provides the functionality to discover and logically combine coexisting devices and services into an overall interaction session. Rendering of the mobile content can then take place on any device which has the capabilities that are closest to meet the user’s preferences while considering the operational context where an interaction takes place.

In this thesis, factors having an impact of MID-B operation are investigated and MID-B requirements are analysed. The MID-B framework functions are multidisciplinary. A number of different functionalities are required; they include device & service discovery, detection of user’s
contexts, user-interface adaptation and presentation-adaptation & connectivity-management. The contributions of this thesis overall, as well as to these functionalities are summarised in Figure 7-1 and summarised thereafter.

<table>
<thead>
<tr>
<th>Device and service discovery</th>
<th>User-interface adaptation</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Extension to Bluetooth SDP</td>
<td>• Device and modality dependency</td>
</tr>
<tr>
<td>• Bluetooth-based and IP-based discoveries</td>
<td>• MST search mechanism</td>
</tr>
<tr>
<td>• Three-party model</td>
<td>• MST forest</td>
</tr>
<tr>
<td>• Registry table</td>
<td>• Resource handler</td>
</tr>
<tr>
<td>• MST search mechanism</td>
<td>• Binding function</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Detection and handling of context</th>
<th>Presentation-adaptation &amp; connectivity-management</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Set of profiles</td>
<td>• Algorithm</td>
</tr>
<tr>
<td>• Cooperation between the profiles of user and affected-interfaces</td>
<td></td>
</tr>
<tr>
<td>• Modality Service Tree (MST)</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 7-1: The contributions of the thesis over four functionalities**

**Device and service discovery**

- **Extension to Bluetooth Service Discovery Protocol (SDP).** To discover fundamental service information, MID-B exploits the mechanism introduced in the Bluetooth SDP transaction. This thesis proposes two extensions to the SDP. First, User Equipment Discovery (UED) which is responsible for finding physical interfaces available in the MID-B environment. Second, the Multimodality (MM) transaction which collects information about the available interaction means. With the extension, MID-B facilitates a mechanism for discovering in-depth information about modalities that are available on devices within the vicinity, including their capabilities and limitations, available in the system.

- **Bluetooth- and IP-based discovery mechanisms.** The extension of Bluetooth SDP is emulated in two different network protocols. Firstly, Bluetooth-based discovery uses the Bluetooth stack as a reference. In the Bluetooth stack, UED and SDP have already been defined in the lower and SDP layers. MID-B adds an MST layer into the stack at the same level as SDP. The added MST layer facilitates the MM transaction. This type of discovery is emulated by using J2ME and Bluetooth Java APIs. Secondly, IP-based discovery makes MID-B more flexible for general IP networks. The OSI-7 model is used as a reference. IP networks do not include or implement SDP functionality; comparable mechanisms are available in IP-based discovery mechanisms. MID-B can be used in both
cases, it has been defined to discover and use Bluetooth-enabled devices as well as common user interfaces connected to an IP network.

- **Three-party model.** In general, SDP is based on a two-party model, i.e. client/server. In MID-B, the discovery mechanism is developed, based on a three-party model. The three parties include a core private device (UE_C), public interface devices (UE_Is), and interaction management (MSB). UE_C is a client searching for supporting services from UE_I(s). MSB maintains the information of service descriptions and facilitates service mappings. The developed three-party model gives an advantage over the SDP, it adds management functions for distributed devices and modalities that may join or leave the virtual device vicinity at any time.

- **Registry table.** In MID-B, a registry table is implemented atop the SDP principle. It is maintained in the MSB. Indeed, the added registry table maintains the records of discovered UE_Is. It also provides a pointer to the location that stores information about resources available on a particular UE_I. In addition, the registry implements an availability tag to indicate whether a UE_I is currently present within an environment. All the UE_I's context information is stored even if it is not currently present. Hence, UE_C does not need to recover service and modality information of a rediscovered UE_I.

Detection and handling of context

- **Set of profiles.** A set of profiles is a collector that acts as database of context information needed for the device and service binding. A number of profiles defined in MID-B are documented in XML-based format, employing the XML tags proposed in the User Agent Profile (UAProf) specification. This ensures that the structures of these profiles are extendable and flexible.

- **Cooperation between end-user and affected-interface profiles.** Generally, the information about a user's physical abilities is captured within a user profile. The thesis introduces an affected-interface profile that contains the relation between a 'human's physical (sensing) organ' and 'affected interfaces'. The thesis describes a mechanism how the two profiles cooperate. Such a mechanism gains knowledge about which underlying 'modalities', provided by these 'interfaces', will be affected due to disability of a particular 'physical (sensing) organ'. This helps the binding mechanism to learn about negative modalities (i.e. modalities that should be excluded from use).

- **Modality Service Tree (MST).** MST provides a generalised approach to express a device and its context. This thesis introduces a mathematical approach to define and describe the information of this context. Device characteristics are described at different levels of
Chapter 7. Conclusions and Further Research

detail. The MST is inspired by the idea of a multi-branching tree, allowing a mixed structure with immediate-offspring. In MST, relations between two levels are identified and described. The mathematical expressions allow the description of devices independent on device type or model.

User-interface adaptation

- **Theoretical description of device dependencies.** A dependency describes the relation between UE_C and UE_I. Two concepts are introduced; firstly, a logical relation defines the semantic dependencies between a number of modality services provided by the UE_C and UE_I. Secondly, the cluster relation describes the physical dependencies within a closed group of UE_Is (and one UE_C). Implementing the two concepts, a binding mechanism is able to understand how different devices in a system are related. This helps the binding mechanism to unite these devices to form a virtual device.

- **MST search mechanisms.** Two types of MST search mechanisms are defined. Firstly, the top-down search mechanism discovers interface and/or modality service(s) provided by a particular physical equipment. When a modality/interface service is known, the bottom-up search mechanism discovers interface equipment(s) and device(s) supporting this service. With the two search mechanisms, context information provided by the MST can be explored and discovered at any level of the tree.

- **MST forest.** In a virtual device environment, the principle of MST is extended into an 'MST forest'. The MST forest combines different MSTs to envision a set of interface and modality services provided in such an environment.

- **Resource handler.** In MID-B environment, resources are captured at different levels of the MST forest. A resource handler is a component providing a management of resource information. In addition to the registry table described above, the resource handler implements three more tables maintaining information about interface equipments, input, and output modalities. Once a device is registered in the registry table, its resource information is extorted and recorded in these tables. Besides, the handler is responsible to monitor usability and availability of those resources. That makes a binding mechanism able to learn which resource can or cannot be bound in a particular interaction session.

- **Theoretical approach to the interface and modality binding.** This thesis introduces a theoretical approach to the interface and modality binding. A concept of context space model is exploited to formalise a binding function. The function compares a value of context with an accepted range. Depending on type of context, the value outside the range 'does' or 'may' render a focused service inactive. In the binding function, different
categories of binding are distinguished, such as interface equipment binding or modality binding. This helps MID-B to consider interaction means at different levels.

**Presentation-adaptation & connectivity-management**

- **Basic algorithm.** MID-B defines a ‘basic’ algorithm facilitating presentation adaptation and connectivity management. In fact, this algorithm is logically a part of the user-interface adaptation. This decides 1) whether presentation of content will be on the UE_C or another UE_I, and 2) which network connectivity will be used to reroute the content to the UE_I. Accordingly, a new selected pattern of user interaction is coordinated with available presentation adaptor and supporting communication protocol.

**Management of distributed modalities**

- **MID-B framework.** The last contribution includes the combination of all aforementioned mechanisms, techniques and enhancements. The definition and prototype implementation of the MID-B framework enables the discovery, analysis and binding of modalities that are hosted on different wirelessly connected devices, thus providing the basics for mobile multi-modal user interface operation.

### 7.2 Future Work

This thesis defines a framework for mobile multi-modality (MID-B) and it extends the functionality and definitions of many of the enablers that are needed to implement such framework. The thesis covers the framework and discusses many of the mechanisms and techniques necessary to implement the framework, at the same time however, it opens additional areas where additional research would be needed to further enhance the functionality of the framework. The following list discusses such areas for possible improvements and additional research that could be undertaken.

- **Group Application.** Currently, MID-B only provides a mechanism to form a virtual device environment for a single user. In Section 4.3.2.2.3, the user profile used in MID-B extends a profile defined in the MobiLife project. This profile includes information about social relations of the user. This will extend the current scope of MID-B; the extension will enable many users to jointly build one shared virtual user/group interface. Such group application/shared user interface would of course raise further issues, in particular with regard to security and information sharing.

- **Multiple core devices.** This thesis assumes that a user always carries a portable (private) device. This device is considered as the core device that controls possible interface
devices. There is only one core device allowed at any time. In a real scenario, users may have multiple private devices; mechanisms to manage such situations are required.

- **Extension to MST.** MST defines a unified representation of device contexts. It is formed by using graph theory. The principles of which provide a useful tool to express datatype of devices and services. However, additional detail can be added to the MST, depending on the requirements, the number of MST levels could be dynamic (currently four levels are defined). The current second level, representing physical interface equipment, could be broken down, depending on the device model. For example, contemporary smart phones, being a single interface equipment may host other smaller interface equipments. Such as, a touch screen hosts an on-screen keypad. The dynamic handling of such descriptions may further limit the discovery and analysis times.

- **Considering additional context information.** The work reported in this thesis considers device context and information about user abilities. To be able to cover additional operational scenarios, other types of context need to be taken into account. Profiles defined in this thesis are sufficiently flexible to be expanded and such extensions would facilitate an interaction service that satisfies even more fine grained conditions.

- **Generalising the criteria for device and modality selection.** In this thesis, the binding function is defined in (6.39). Nevertheless, the thesis does not yet address criteria to explain which attributes are to be considered for a particular device and modality. A formal description of these criteria is needed as part of the future work. Moreover, the condition (6.41) only gives a suggestion that a service is most likely to be used if soft-value attribute is in an agreed range. These criteria are yet to be defined.

- **Improving the algorithm performance.** In 6.4.2.2, the binding is using a rather 'basic' algorithm. The thesis introduces a top-level implementation and high-level planning mechanism. To improve the performance of a real system, work addressing a more detailed algorithm considering a wider range of input parameters should be defined.

- **Connectivity control.** In the architecture of MID-B, user interface control is an important feature. For future work, the architecture should be refined considering the actual connectivity between the nodes. One initial improvement should include the definition of concrete mechanisms and the implementation of features for connectivity control.
Chapter 8

8 Bibliography

8.1 References


[22] J. C. Martin, "On the use of the multimodal clues in observed human behavior for the
modeling of agent cooperative behavior," presented at 18th National Conference on

[23] J. Anil K, "Multimodal user interfaces: Who's the user?," in ICMI 5th International

adaptive classifier cascade based on modality reliability," IEEE Transaction on

programming system," in IEEE 2002 Symposia on Human Centric Computing Languages

(Last access on 21.07.06).

[27] M. Perakakis, A. Potamianos, and M. Toutoudakis, "Guidelines for multimodal user


Wnogard, J. Landay, J. Larson, and D. Ferro, "Designing for the user interface for
multimodal speech and pen-based gesture applications: State-of-the-art systems and
future research directions," in Human-Computer Interaction, vol. 15: Lawrence Erlbaum

on Audio and Video-based Biometric Person Authentication (AVBP' 97). Crans-Montana,
Switzerland, 1997.

[32] J. Yang, F. Huang, and W. Kunz, "User registration using your face and mouth," in International Conference on Speech and Language Processing (ICSLP). Beijing, China,
2000.


8. Bibliography


Chapter 8. Bibliography


[147] [http://www.ist-mobilife.org/](http://www.ist-mobilife.org/) (Last access 02.02.07).


8.2 Publications

Conference Papers


Journals/Magazines


# Appendix A.1

## Universal Attributes of Bluetooth SDP

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute ID (Hexadecimal number)</th>
<th>Attribute Value Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServiceRecordHandle</td>
<td>0x0000</td>
<td>32-bit unsigned integer</td>
</tr>
<tr>
<td>ServiceClassIDList</td>
<td>0x0001</td>
<td>Data element sequence</td>
</tr>
<tr>
<td>ServiceRecordState</td>
<td>0x0002</td>
<td>32-bit unsigned integer</td>
</tr>
<tr>
<td>ServiceID</td>
<td>0x0003</td>
<td>UUID</td>
</tr>
<tr>
<td>ProtocolDescriptorList</td>
<td>0x0004</td>
<td>Data element sequence</td>
</tr>
<tr>
<td>BrowseGroupList</td>
<td>0x0005</td>
<td>Data element sequence</td>
</tr>
<tr>
<td>LanguageBasedAttributeIDList</td>
<td>0x0006</td>
<td>Data element sequence</td>
</tr>
<tr>
<td>ServiceInfoTimeToLive</td>
<td>0x0007</td>
<td>32-bit unsigned integer</td>
</tr>
<tr>
<td>ServiceAvailability</td>
<td>0x0008</td>
<td>8-bit unsigned integer</td>
</tr>
<tr>
<td>BluetoothProfileDescriptorList</td>
<td>0x0009</td>
<td>Data element sequence</td>
</tr>
<tr>
<td>DocumentationURL</td>
<td>0x000A</td>
<td>URL</td>
</tr>
<tr>
<td>ClientExecutableURL</td>
<td>0x000B</td>
<td>URL</td>
</tr>
<tr>
<td>IconURL</td>
<td>0x000C</td>
<td>URL</td>
</tr>
<tr>
<td>ServiceName</td>
<td>0x0000 + Attribute ID based (contained in the LanguageBasedAttributeIDList)</td>
<td>String</td>
</tr>
<tr>
<td>ServiceDescriptor</td>
<td>0x0002 + Attribute ID based (contained in the)</td>
<td>String</td>
</tr>
<tr>
<td>ProviderName</td>
<td>LanguageBasedAttributeValueList)</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------</td>
<td></td>
</tr>
<tr>
<td>ProviderName</td>
<td>0x0003 + Attribute ID based</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(contained in the LanguageBasedAttributeValueList)</td>
<td>String</td>
</tr>
<tr>
<td>Reserved Fields</td>
<td>0x000D – 0x01FF</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Appendix A.2

Example for ServiceSearchAttribute Transaction

Assuming, a mobile client is searching for available Bluetooth headset service. The transaction initiates with a ServiceSearchAttribute request PDU transmitted from the client to the server, shown in Figure A2-1.

<table>
<thead>
<tr>
<th></th>
<th>PDU ID = 0x06</th>
<th>Transaction ID = 0x0003</th>
<th>Parameter length = 0x0012</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max attribute byte = 0x00100</td>
<td>T = 6</td>
<td>S = 3</td>
<td>T = 3</td>
</tr>
<tr>
<td></td>
<td>SI = 5</td>
<td></td>
<td>UUID</td>
</tr>
<tr>
<td></td>
<td>HeadsetClassID = 0x1108</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>S = 8</td>
<td>T = 1</td>
<td>SI = 2</td>
</tr>
<tr>
<td>ID range = 0x00000001</td>
<td>Con. S. = 0x00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

T = The Type Description  
SI = The Size Index  
S = An amount of size in byte  
Con. S. = Continuation State

Figure A2-1: SDP ServiceSearchAttribute request (1)

PDU ID          The PDU ID of the SDP ServiceSearchAttribute request is 0x06.
Transaction ID  Assuming, the transaction ID is 0x0003.
Parameter Length The length of parameter is 18 bytes.
ServiceSearchPattern This field begins with the data element description for the whole sequence of the ServiceSearchPattern. In this example, there is only one pattern, i.e. the UUID of the headset class. The pattern is prefixed with the data element description in which the type descriptor is 3 (UUID) and
the size index is 1 (2 bytes).

**MaximumAttributeByteCount**

Assuming, the maximum size of the response PDU is 256 bytes (0x0100). The format of this field is fixed, so the data element description is not necessary.

**AttributeIDLists**

This field begins with the data element description for the entire sequence of the attribute IDs. In this example, the set of the attribute IDs contains three elements, i.e. 0x000, 0x0001 and 0x000c. These define ServiceRecordHandle, ServiceClassIDList and IconURL, respectively. The list can be split into two ID ranges, these are 0x000 – 0x0001 and 0x000c. Every element in the range will share the data element description. At the first ID range, the type description is 1 (integer) and the size index is 2 (4 bytes). Following, the second range is prefixed by the type description = 1 (integer) and the size index = 1 (2 bytes).

**ContinuationState**

Since this PDU initialises the transaction, the ContinuationState is set to 0.

Once receives the request, a server forms the corresponding response PDU. The example assumes that the server contains two services records and each record contains the service attributes of which the IDs are 0x0000 and 0x0001. The information of the AttributeLists parameter must be retrieved from both records. The structure of the response is shown in Figure A2-2.
The PDU ID of the SDP ServiceSearchAttribute response is 0x07.

Matching with the request, the transaction ID is 0x0003.

The length of parameter is 44 bytes.

Number of bytes in the attribute list field = 39 (0x027).

The attribute lists are composed of service attributes in every service records matching the request’s ServiceSearchPattern. As usual, the field is prefixed by data element description in which
Appendix A.2

Type description is 6 (data sequence). Because the following size index is 6, the actual size of the whole field, 36 bytes (0x24), is contained in the next additional 2 bytes. The data sequence of this field is comprised of two elements. In fact, each element is formed as a sequence of service attributes in each individual service record. Every element has its own data element description. For instance, the first element is created by the attributes in the first service record. Its data element description is made up of type description = 6 (sequence of service attributes), size index = 6 and two extra bytes (length of all information from the first service record = 22 bytes, i.e. 0x16). The service attribute consists of an ID and value(s). The data element description is also required prior to the information of both of the ID and the value(s). Repeatedly, the rest of this field is established in the similar way until reaching the end of the second record. However, the example gives the uncompleted response. The part of second record’s information is not yet finished.

ContinuationState On account of the uncompleted AttributeLists field, the ContinuationState = 2, implying that continuation information is in the next 2 bytes.

When the client gets the response with non-zero ContinuationState, it resends the second request PDU with the identical ContinuationState, shown in Figure A2-3. On the other hand, the server receives the second request and interprets the continuation information for the stopping point of the first response PDU. Subsequently, it constructs the response PDU, shown in Figure A2-4, containing the rest of service attributes in the second service record. Since all the rest of information is embraced in the second response PDU, the ContinuationState is set to 0. This indicates the end of the transaction.
Appendix A.2

Figure A2-3: SDP ServiceSearchAttribute request (2)

Figure A2-4: SDP ServiceSearchAttribute response (2)
Appendix A.3

Example for MST

Using a real device model to present the MST, a multimedia-supporting smart phone represents an abstract prototype. The phone is equipped with a touch-screen but no physical keypad, Figure A3-1 shows the MST of such a phone.

MST of smart phone ($S$) = \{$S_1$, $S_2$, $S_3$, $S_4$\}

User equipment device ($S_1$) = Smart phone

Interface Equipment ($S_2$) = \{Hand-free, Microphone, Loud speaker, Screen, Camera, Stylus, Virtual dial-pad, On-screen keyboard\}

Interface Service ($S_3$) = \{(Audio, Un-Co), (Voice, Un-Co), (Visual, Un-Co), (Gesture, Sync), (Virtual dial-pad, Sync), (On-screen keyboard, Sync)\}

Modality Service ($S_4$) = \{(Pre-recorded sound, Output), (Non-verbal sound, Output), (Voice, Telephony), (ASR, Input), (DTMF, Input), (Graphic, Output), (Text, Output), (Visual, Input), (Gesture, Output)\}

Figure A.3-1: The MST representing smart phone
A laptop with built-in camera is another example of a collection of interface equipments (combined in one single device).

The MST helps a system developer to understand the fact of the differences and similarities between user devices. Figure A3-1 and Figure A3-2 show how the MST expresses the service hierarchy obtained from different devices. At the level of interface equipments (S2), the MSTs representing a smart phone and a laptop host a loud speaker, a screen and a camera. Yet, these interface equipments enclose dissimilar services. For instance, the loud speaker of the phone can render an audio interface or voice interface, but the laptop does not have a functionality of a voice service. In addition, a similar interface and modality service provided by both of the devices, in fact, may gain the access from different interface equipments. A conventional-keyboard service, for example, is offered in both MSTs. However, for the laptop, the service is accessed via an
actual physical keyboard, whereas the smart phone uses an on-screen keyboard and stylus as combined physical interface equipment implementing a conventional-keyboard service.
Appendix A.4

MST Representing Equipment with a single User Interface

A printer is used as the example for a device with single user interface equipment.

![Diagram of MST representing a printer]

Figure A4-1: MST representing a printer

MST of printer (S) = \{S_1, S_2, S_3, S_4\}
User equipment device (S_1) = Printer
Interface Equipment (S_2) = \{Printer\}
Interface Service (S_3) = \{(Visual, null)\}
Modality Service (S_4) = \{(Hard Copy, Output)\}
Appendix A.5

MST Representing a Composite Device with multiple User Interfaces

Figure A5-1 illustrates the MST of a simple desktop accommodating a keyboard, a mouse and a monitor. The main user equipment device, i.e. the desktop, embraces 1) a 'physical connectivity' to a number of the plugged-in components (i.e. a keyboard, a mouse and a monitor), and 2) a 'logical connectivity' between the desktop (level 1, a user equipment device) and the user interface equipments of all the components (level 2, these are a keyboard, a mouse and a screen).

MST of desktop (S) = \{S_1, S_2, S_3, S_4\}

User equipment device (S_1) = Desktop

Interface Equipment (S_2) = \{Keyboard, Mouse, Screen\}

Interface Service (S_3) = \{(Conventional-keyboard, null), (Conventional-mouse, null), (Visual, null)\}

Modality Service (S_4) = \{(Conventional-keyboard, Input), (Conventional-Mouse, Input), (Graphic, Output), (Text, Output)\}
Appendix A.6

Semantics of Interface Equipment ‘A’ exchanged between UE_C and UE_I

From the Appendix A.3, the MST representing a smart phone is used to describe the UE_C. On the other hand, the UE_I is characterised by the MST of a laptop. This appendix illustrates the semantic of the interface equipment ‘A’ between the phone and the laptop.

Assuming, 

A = \{a | a \in S_2 \text{ of } \text{MST}_{\text{phone}} \cap S_2 \text{ of } \text{MST}_{\text{laptop}} \} \text{ is a set of semantic interface equipment between the phone and the laptop,}

B is a set of child nodes of set ‘A’ at the MST_{phone}, and

C is a set of child node nodes of set ‘A’ at the MST_{laptop}.

Case 1:

A = \{a | a \in S_2 \text{ of } \text{MST}_{\text{phone}} \cap S_2 \text{ of } \text{MST}_{\text{laptop}} \} = \{\text{Loud speaker, Screen, Camera}\}

B = \{\text{Audio, Voice, Visual, Gesture, Virtual dial-pad, On-screen keyboard}\}

C = \{\text{Audio, Visual}\}

(B \cap C) = \{\text{Audio, Visual}\}

(B \cup C) = \{\text{Audio, Voice, Gesture, Visual, Virtual Dial-Pad, On-Screen Keyboard}\}

\therefore A \times (B \cap C) \neq A \times (B \cup C)

\therefore The MST_{phone} and the MST_{laptop} do not encompass a semantic of interface equipment on \{\text{Loud speaker, Screen, Camera}\}.

Case 2:

Let A_i \in A, B_i \in B and C_i \in C.

Case 2.1:

A_1 = \{\text{Loud Speaker}\}, B_1 = \{\text{Audio, Voice}\}, C_i = \{\text{Audio}\}

188
The MST\textsubscript{phone} and the MST\textsubscript{laptop} do not encompass a semantic of interface equipment on ‘Loud Speaker’.

Case 2.2:

\[ A_2 = \{\text{Screen}\}, \quad B_2 = \{\text{Visual, Gesture, Virtual Dial-Pad, On-Screen Keyboard}\}, \quad C_1 = \{\text{Visual}\} \]

\( \therefore \) The MST\textsubscript{phone} and the MST\textsubscript{laptop} do not encompass semantic interface equipment on ‘Screen’.

Case 2.3:

\[ A_3 = \{\text{Camera}\}, \quad B_3 = \{\text{Visual}\}, \quad C_1 = \{\text{Visual}\} \]

\( \therefore \) The MST\textsubscript{phone} and the MST\textsubscript{laptop} encompass semantic interface equipment on ‘Camera’. 