A Structured Approach to the Development of Telematic Services Using Distributed Object-Oriented Platforms

by

Dionisis Adamopoulos

Thesis submitted to the University of Surrey for the degree of Doctor of Philosophy

Centre for Communication Systems Research
School of Electronic Engineering, IT and Mathematics
University of Surrey, UK

Copyright © Dionisis Adamopoulos
November 2000
Life is short,
the art long
timing is exact,
experience treacherous
judgement difficult

- Hippocrates -

... the occasion urgent.
ALL MISSING PAGES ARE BLANK

IN

ORIGINAL
Abstract

Rapid technological developments are taking place in computing and telecommunications. The evolving synergy between them, known as Telematics, provides a wide range of opportunities for the delivery of new advanced multimedia services. In this context, Information Networking plays a central role and Telecommunications Service Engineering emerges as an important new scientific discipline.

This thesis examines important issues that underpin the service creation process for advanced telematic services. It proposes a telecommunications service engineering framework consisting of a service development methodology, a service creation environment and a service support environment that encompasses a service execution environment, the latter based on distributed object technology, and populates that framework.

The starting point is the proposal of a complete methodology for the development of telematic services, that “covers” in a systematic and structured manner the entire service creation process. The methodology assumes a generic service environment based on the Telecommunications Information Networking Architecture (TINA), adopts an iterative and incremental use case driven approach, and uses the Unified Modelling Language (UML) and other results from object-oriented software engineering methodologies which it extends and applies to telematic services. Furthermore, the proposed methodology is validated and evaluated by applying it to the design and development of a complex representative telematic service i.e. a Multi-Media Conferencing Service for Education and Training (MMCS-ET).

The thesis also examines the role of various Distributed Processing Environments (DPEs) - also known as middleware - in service execution and argues that Microsoft's Distributed Component Object Model (DCOM) is a valid choice as a service engineering DPE. Furthermore, it enhances DCOM with the capability of handling continuous media streams such as voice, audio and video, by designing and implementing a number of suitable multimedia support services, together with a related Application Programming Interface (API). The use of the enhanced DCOM as a TINA compliant DPE is validated by using it as a service execution environment for the MMCS-ET service.
To all those who have offered support, inspiration and, most cherished of all, encouragement

Acknowledgements

I am truly indebted to a number of people, each of whom has contributed to making this thesis possible.

First of all, I would like to sincerely thank my supervisor Prof. George Pavlou for his expert guidance, technical advice, insightful discussions, and friendly encouragement during the process of working on this thesis. I was very fortunate to find someone that really understood my work. He was a very important source of inspiration and his help was indispensable in the detailed elaboration of this thesis.

I would also like to express my deep gratitude to Dr. Constantine A. Papandreou of the Hellenic Telecommunications Organisation (OTE) for his unconditional and inestimable support of my research work, his guidance and his patience, confidence, and encouragement in periods of stress. He was a constant source of useful advice and I am very grateful to him for his contributions on my academic development, and for his huge overall influence on my academic choices and on my personality.

Special thanks are also due to Prof. Barry Evans and Mr. Peter Jones who believed in my abilities and gave me the opportunity to conduct research in the area of telecommunications service engineering.

My gratitude goes also to Dr. Zhili Sun and Prof. George Haramis for the interesting discussions that we had and for their willingness to help me with my research work.

Further, I would like to thank the members of the networks research group and the researchers in the Centre for Communication Systems Research (CCSR), and especially Ilias Andrikopoulos, Costas Apostolas, Lloyd Wood, Paris Flegas, George Sfikas, Spyros Velentzas, Ioannis Mertzanis and Andreas Agius, for their help and advice. I also thank all my friends for their moral support and encouragement in this difficult and time-consuming endeavour.

Last, but not least, my thanks are due to my parents and my family in general. Their unfading interest in my efforts and faith in my ability have proved a continual source of inspiration to me. Their help, support and love was, and still is, invaluable.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>V</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>VII</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>IX</td>
</tr>
<tr>
<td>List of Figures</td>
<td>XV</td>
</tr>
<tr>
<td>List of Tables</td>
<td>XIX</td>
</tr>
<tr>
<td><strong>Chapter 1: Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1. The Telecommunications Environment</td>
<td>1</td>
</tr>
<tr>
<td>1.2. The Emergence of Telecommunications Service Engineering</td>
<td>6</td>
</tr>
<tr>
<td>1.3. Thesis Motivation</td>
<td>7</td>
</tr>
<tr>
<td>1.4. Thesis Objectives</td>
<td>9</td>
</tr>
<tr>
<td>1.5. Thesis Structure and Overview</td>
<td>13</td>
</tr>
<tr>
<td><strong>Chapter 2: Advanced Approaches in Telecommunications Service Engineering</strong></td>
<td>15</td>
</tr>
<tr>
<td>2.1. Introduction</td>
<td>15</td>
</tr>
<tr>
<td>2.2. Defining Telematic Services</td>
<td>17</td>
</tr>
<tr>
<td>2.3. Object-Oriented Development of Telematic Services</td>
<td>18</td>
</tr>
<tr>
<td>2.3.1. The Concept of Object Orientation</td>
<td>19</td>
</tr>
<tr>
<td>2.3.2. Object-Oriented Design of Telematic Services</td>
<td>20</td>
</tr>
<tr>
<td>2.3.3. Towards an Integrated Approach</td>
<td>23</td>
</tr>
<tr>
<td>2.4. Key Technologies in Emerging Multi-Service Networks</td>
<td>29</td>
</tr>
<tr>
<td>2.4.1. Intelligent Network</td>
<td>30</td>
</tr>
<tr>
<td>2.4.2. Telecommunication Management Network</td>
<td>31</td>
</tr>
<tr>
<td>2.4.3. Integration Issues</td>
<td>33</td>
</tr>
<tr>
<td>2.5. Architectural Frameworks for Telematic Services</td>
<td>34</td>
</tr>
<tr>
<td>2.5.1. The Open Services Architectural Framework</td>
<td>35</td>
</tr>
<tr>
<td>2.5.2. The TINA-C Initiative</td>
<td>36</td>
</tr>
<tr>
<td>2.5.3. Comparing OSA and TINA-C</td>
<td>41</td>
</tr>
<tr>
<td>2.5.4. Evaluation and Concluding Remarks</td>
<td>44</td>
</tr>
<tr>
<td>2.6. Service Creation Environments</td>
<td>48</td>
</tr>
<tr>
<td>2.7. Summary and Conclusions</td>
<td>54</td>
</tr>
</tbody>
</table>
# Table of Contents

**Chapter 3: Distributed Computing Platforms in Telematics** ............................. 57

3.1. Introduction .................................................................................................... 57

3.2. Important Distributed Processing Environments .......................................................... 59

3.2.1. ISO's / ITU-T's RM-ODP ............................................................................ 60

3.2.2. Open Group's DCE .................................................................................... 61

3.2.3. OMG's CORBA ....................................................................................... 62

3.2.4. Microsoft's COM/DCOM ........................................................................... 63

3.2.5. Sun's Java RMI ....................................................................................... 65

3.2.6. Evaluation.............................................................................................. 66

3.3. The Emergence of Distributed Object Platforms .......................................................... 68

3.4. A Comparison Between DCOM and CORBA ............................................................. 69

3.4.1. Core Properties ....................................................................................... 72

3.4.2. Service Engineering Related Properties ........................................................... 75

3.4.3. Examination of Performance ........................................................................ 78

3.4.3.1. The Experiment ................................................................................ 79

3.4.3.2. The Results .................................................................................... 80

3.4.3.3. DCOM Remoting Architecture ............................................................... 82

3.4.4. Interoperation with the World Wide Web .......................................................... 84

3.4.5. Comparison Remarks ................................................................................. 87

3.5. Interworking Between DCOM and CORBA............................................................... 88

3.6. The Distributed Processing Environment of Choice ...................................................... 89

3.7. Summary and Conclusions .................................................................... I .............. 90

**Chapter 4: A Methodology for the Development of Telematic Services**........... I. 93

4.1. Introduction .................................................................................................... 93

4.2. The Need for an Integrated and Systematic Approach .................................................... 95

4.3. Modelling Issues ............................................................................................... 100

4.4. Overview of the Proposed Methodology .................................................................... 101

4.5. Important Methodological Considerations .................................................................. 104

4.6. Requirements Capture and Analysis Phase ................................................................. 106

4.6.1. Definition of a Service Development Plan ......................................................... 107

4.6.2. Creation of a Preliminary Investigation Report .................................................. 109

4.6.3. Definition of Service Requirements .................................................................. 113

4.6.4. Definition of Use Cases ............................................................................. 115
# Table of Contents

4.6.5. *Definition of a Draft Service Conceptual Model* .................................................. 118
4.6.6. *Recording Terms in the Glossary* ................................................................... 118
4.6.7. *Considering the Application of Rapid Prototyping* ................................................ 118
4.6.8. *Definition of Draft Service Architecture Layers* .................................................. 118
4.6.9. *Refining the Service Development Plan* ............................................................ 120
4.6.10. *Complementary and Alternative Approaches* .................................................... 122

4.7. *Starting a Service Development Cycle* ...................................................................... 122

4.8. *Service Analysis Phase* ....................................................................................... 123
   4.8.1. *Definition of Essential Use Cases* .................................................................. 123
   4.8.2. *Refining the Use Case Diagram(s)* ................................................................. 123
   4.8.3. *Definition of Service Conceptual Models* ........................................................ 124
      4.8.3.1. *Specification of the Main Service Conceptual Model* ................................. 125
      4.8.3.2. *Specification of the Ancillary Service Conceptual Models* .......................... 128
   4.8.4. *Definition of Service Sequence Diagrams* ......................................................... 132
   4.8.5. *Definition of Service Operation Contracts* ........................................................ 133
   4.8.6. *Refining the Glossary* ................................................................................ 134
   4.8.7. *Definition of Service State Diagrams* .............................................................. 134
   4.8.8. *Concluding Remarks* .................................................................................. 135
   4.8.9. *Complementary and Alternative Approaches* ..................................................... 136

4.9. *Service Design Phase* ......................................................................................... 137
   4.9.1. *Definition of Real Use Cases* ......................................................................... 137
   4.9.2. *Definition of User Interface Aspects* ............................................................... 138
   4.9.3. *Definition of Service Interaction Diagrams* ....................................................... 139
   4.9.4. *Definition of Service Design Class Diagrams* ..................................................... 144
   4.9.5. *Definition of Service Architecture Layers* ......................................................... 145
   4.9.6. *Definition of a Database Schema* ................................................................... 147
   4.9.7. *Concluding Remarks* .................................................................................. 147
   4.9.8. *Complementary and Alternative Approaches* ..................................................... 147

4.10. *Service Implementation Phase* ............................................................................. 150
   4.10.1. *Structure of the Service Implementation Phase* .................................................. 152
   4.10.2. *Important Reusability Matters* .................................................................. 153

4.11. *Service Validation and Testing Phase* ................................................................. 154

4.12. *Support from Software Tools* ............................................................................ 155

4.13. *Summary and Conclusions* ................................................................................ 156
## Table of Contents

**Chapter 5: Validation and Evaluation of the Proposed Methodology**

5.1. Introduction ..................................................................................................... 159

5.2. Validation of the Proposed Methodology .......................................................... 160

5.2.1. The Development of a MMCS-ET ................................................................. 162

5.2.1.1. Telematic Services for Education and Training .............................................. 162

5.2.1.2. Requirements Capture and Analysis Phase .................................................. 164

5.2.1.3. Service Analysis Phase ........................................................................ 168

5.2.1.4. Service Design Phase .......................................................................... 175

5.2.1.5. Service Implementation Phase ................................................................ 178

5.2.2. Additional Simple Validation Attempts ............................................................ 182

5.2.3. Concluding Remarks .................................................................................. 187

5.3. Exploitation of Design Patterns ............................................................................. 187

5.4. Evaluation of the Proposed Methodology .......................................................... 194

5.5. Summary and Conclusions ................................................................................... 200

**Chapter 6: Continuous Media Support in the Distributed Component Object Model**

6.1. Introduction .................................................................................................... 205

6.2. Modelling Multimedia Telecommunications Services ............................................. 207

6.3. Enhancing DCOM for the Support of Continuous Media ...................................... 209

6.3.1. The Proposed Approach ............................................................................ 210

6.3.2. The Stream Communication Algorithm ........................................................ 216

6.3.3. Important Implementation Considerations .................................................... 218

6.4. Validation and Experimentation ............................................................................. 221

6.5. Summary and Conclusions ................................................................................... 224

**Chapter 7: Summary - Conclusions**

7.1. Introduction .................................................................................................... 227

7.2. Thesis Overview .............................................................................................. 227

7.3. Research Contributions and Main Findings of the Thesis .................................... 231

7.4. Possible Future Work .......................................................................................... 236

7.5. Concluding Remarks .......................................................................................... 238

Bibliography ............................................................................................................. 239

XII
# Table of Contents

**Appendix A: Detailed Presentation of the Development of a MMCS-ET using the Proposed Methodology** ......................................................... A-1

A.1. Introduction .................................................................................................... A-1

A.2. Requirements Capture and Analysis Phase .................................................. A-1

A.3. Service Analysis Phase .................................................................................. A-6

A.4. Service Design Phase ................................................................................... A-28

A.5. Service Implementation Phase ...................................................................... A-35

**Appendix B: Overview of the Proposed Service Development Methodology** .... B-1

B.1. Introduction .................................................................................................. B-1

B.2. Overview of the Requirements Capture and Analysis Phase ....................... B-1

B.3. Overview of the Service Analysis Phase ....................................................... B-3

B.4. Overview of the Service Design Phase ........................................................ B-4

B.5. Overview of the Service Implementation Phase ............................................ B-5

**Appendix C: Acronyms and Abbreviations** .................................................. C-1

**Appendix D: Publications** ............................................................................. D-1
List of Figures

Figure 1.1.: The proposed telecommunications service engineering framework ........................................... 10
Figure 2.1.: The relationship between the criteria and the principles for modularity in telematic services .......... 22
Figure 2.2.: The OMG reference model for OOAD modified for telematic services ........................................ 24
Figure 2.3.: Simplified architecture of the intelligent network ........................................................................ 30
Figure 2.4.: A simplified view of the functional TMN hierarchy ..................................................................... 32
Figure 2.5.: The scope of the session concept in TINA-C .................................................................................. 38
Figure 3.1.: The RM-ODP viewpoints ............................................................................................................... 60
Figure 3.2.: The DCE architecture .................................................................................................................. 61
Figure 3.3.: The architecture of an object request broker ................................................................................... 62
Figure 3.4.: Traditional and Java run-time systems .......................................................................................... 65
Figure 3.5.: Using mixed DCE and Microsoft RPC: (a) Microsoft RPC client - DCE server, (b) DCE client - NT RPC server ................................................................. 67
Figure 3.6.: Examination of local method calls in DCOM and CORBA ............................................................. 81
Figure 3.7.: Examination of remote method calls in DCOM and CORBA .......................................................... 81
Figure 3.8.: The DCOM remoting architecture and extensibility options .......................................................... 83
Figure 3.9.: Interoperability using applets and (remote or local) server objects ................................................ 85
Figure 3.10.: Interoperability using applets as notification objects ................................................................. 85
Figure 4.1.: The service life-cycle .................................................................................................................... 96
Figure 4.2.: The overall telematic service model .............................................................................................. 101
Figure 4.3.: Iterative service development cycles in the proposed methodology ........................................... 102
Figure 4.4.: Requirements capture and analysis phase activities ....................................................................... 107
Figure 4.5.: Requirements capture and analysis phase artifact dependencies ................................................. 108
Figure 4.6.: The collaboration space-time matrix ............................................................................................. 111
Figure 4.7.: The TINA-C Business Model (roles and relationships) ................................................................. 112
Figure 4.8.: Service architecture layers of a typical telematic service ............................................................ 120
Figure 4.9.: Service analysis phase activities .................................................................................................... 122
Figure 4.10.: Service analysis phase artifact dependencies ............................................................................... 123
Figure 4.11.: Important session related information models (TINA-C): (a) The session information model, (b) The session role information model ................................................. 128
Figure 4.12.: Important access session related information models (TINA-C): (a) Classification of the access session, (b) The access session information model ...................................... 129
List of Figures

Figure 4.13.: Important service session related information models (TINA-C):
(a) Classification of the service session (TINA-C), ......................................................... 130
(b) The service session information model (TINA-C)........................................................ 130
Figure 4.14.: Relationship between the service session information model and the SSG (TINA-C) ..... 130
Figure 4.15.: The Service Session Graph information model (TINA-C) ..................................... 131
Figure 4.16.: An example of a use case service state diagram.............................................. 135
Figure 4.17.: The service analysis model........................................................................ 136
Figure 4.18: Service design phase activities..................................................................... 137
Figure 4.19.: Service design phase artifact dependencies...................................................... 138
Figure 4.20.: A generic TINA-C service scenario involving user-provider roles....................... 142
Figure 4.21.: A generic TINA-C service scenario involving peer-to-peer access and usage roles...... 142
Figure 4.22.: A generic TINA-C service scenario involving a compound service session.......... 143
Figure 4.23.: Service architecture layers expressed in terms of UML packages...................... 146
Figure 4.24.: The service design model........................................................................... 148
Figure 4.25.: Service implementation phase activities.......................................................... 150
Figure 4.26.: Service implementation phase artifact dependencies....................................... 151
Figure 4.27.: The influence of service implementation work in the proposed iterative and incremental service development process................................................................. 155
Figure 5.1.: The business model of the MMCS-ET ........................................................... 165
Figure 5.2.: The high-level form of the target use cases....................................................... 168
Figure 5.3.: The expanded form of use case “Invite a student to join a MMCS-ET session”........ 169
Figure 5.4.: The expanded form of use case “Join a MMCS-ET session after being invited”.... 170
Figure 5.5.: The expanded form of use case “Engage in text communication with an active user”... 171
Figure 5.6.: The use case diagram of the MMCS-ET (simplified)........................................ 172
Figure 5.7.: The main service conceptual model of the MMCS-ET........................................ 172
Figure 5.8.: The ancillary service conceptual models of the MMCS-ET................................... 173
Figure 5.9.: Service sequence diagrams of the MMCS-ET for the target use cases.................... 174
Figure 5.10.: Service operation contract of service operation “InviteStudent”.......................... 174
Figure 5.11.: Service operation contract of service operation “JoinSession”............................ 175
Figure 5.12.: Service operation contract of service operation “EngageTextCom”....................... 175
Figure 5.13.: Service interaction diagram for inviting a student to join a MMCS-ET session........ 176
Figure 5.14.: Service interaction diagram for joining a MMCS-ET session after being invited....... 176
Figure 5.15.: Service interaction diagram for engaging in text communication with an active user... 177
Figure 5.16.: The service design class diagram of the MMCS-ET (simplified).......................... 178
Figure 5.17.: The login window of the MMCS-ET............................................................. 181
Figure 5.18.: The participant choice window of the MMCS-ET ............................................. 181
Figure 5.19.: The main user window of the MMCS-ET ...................................................... 181
Figure 5.20.: The common activities window of the MMCS-ET ............................................. 182
Figure 5.21.: Distributed collaborative design.................................................................. 183
Figure 5.22.: Distributed case handling.......................................................................... 183
Figure 5.23.: Remote monitoring................................................................................ 184
Figure 5.24.: Remote database access........................................................................ 184
Figure 5.25.: Remote database utilisation......................................................................... 185
Figure 5.26.: Remote access to expertise........................................................................ 185
Figure 5.27.: Remote application running........................................................................ 185
Figure 5.28.: Entertainment on demand (pay per view).................................................. 186
Figure 5.29.: Remote consultation................................................................................... 186
Figure 5.30.: Social conversation..................................................................................... 186
Figure 5.31.: Creating an instance of the ActiveUserInfo list............................................. 190
Figure 5.32.: Facilitating low coupling when sending a message from a teacher to a student.. 191
Figure 5.33.: Service interaction diagram for inviting a user to join a service session............. 193
Figure 5.34.: The essential elements of the NIMSAD framework...................................... 195
Figure 6.1.: Inheritance structure for (characteristic) continuous media devices............. 210
Figure 6.2.: The IChain Interface................................................................................ 211
Figure 6.3.: The IEndpoint interface............................................................................... 212
Figure 6.4.: The state transition diagram of a device........................................................... 213
Figure 6.5.: The IStreamBinder interface........................................................................ 213
Figure 6.6.: The IObjectGroup interface.......................................................................... 215
Figure 6.7.: An example scenario for the proposed multimedia support services.............. 216
Figure 6.8.: Using an interface of a remote server object in DCOM................................. 220
Figure 6.9.: The main COs of the MMCS-ET and the position of the interfaces of the proposed API.. 222
Figure 6.10.: Experimenting with the use (or not) of object groups in the stream communication algorithm ........................................................ 223
Figure 7.1.: A detailed version of the proposed telecommunications service engineering framework .... 230
Figure A.1.: The main service concepts of the MMCS-ET and their attributes.................. A-18
Figure A.2.: Service sequence diagrams of the MMCS-ET............................................... A-19
# List of Tables

Table 2.1.: Structure, function, and behaviour abstractions in the selected OOAD methodologies 26
Table 2.2.: Evaluating the selected OOAD methodologies for service development 27
Table 2.3.: The main service components that are proposed by the TINA-C Service Architecture 40
Table 3.1.: Summary of RM-ODP viewpoints and languages 61
Table 3.2.: Comparing DCOM and CORBA: Basic characteristics 70
Table 3.3.: Comparing DCOM and CORBA: Core properties 71
Table 3.4.: Comparing DCOM and CORBA: Service engineering related properties 71
Table 4.1.: Mapping between service concepts and TINA-C access session related service COs 140
Table 4.2.: Mapping between service concepts and TINA-C service session related service COs 141
Table 5.1.: Session related MMCS-ET functions 166
Table 5.2.: Interaction related MMCS-ET functions 167
Table 5.3.: Collaboration support MMCS-ET functions 167
Table 5.4.: Mapping of the service IOs of the main service conceptual model to appropriate service COs 177
Table 5.5.: Implementation characteristics of the MMCS-ET code 180
Table 5.6.: Evaluating the proposed methodology according to the NIMSAD framework: The “problem situation” element 197
Table 5.7.: Evaluating the proposed methodology according to the NIMSAD framework: The problem solver element 197
Table 5.8.: Evaluating the proposed methodology according to the NIMSAD framework: The problem-solving process element (problem formulation) 198
Table 5.9.: Evaluating the proposed methodology according to the NIMSAD framework: The problem-solving process element (solution design and design implementation) 199
Table 5.10.: Essential metrics related to the development of the MMCS-ET (during the formation of the proposed methodology) 201
Table 5.11.: Essential metrics related to the development of the MMCS-ET (after the formation of the proposed methodology) 202
Table 6.1.: Comparison of COM threading models using the proposed API 224
Table 6.2.: Comparison of modelling approaches for handling continuous media in CORBA and DCOM 224
Chapter 1:

Introduction

1.1. The Telecommunications Environment.

The telecommunications industry is currently undergoing a fundamental restructuring as the evolving synergy between computing and telecommunications technology, termed Telematics, is gradually gaining momentum. This evolution and diversification is being driven mainly by liberalisation, increasing competition in the marketplace, technological advancements, and demands from all customer segments for an increasingly sophisticated portfolio of telecommunications services, tailored to specific needs [ARMB97][HELD93]. It is expected that the forthcoming integrated (fixed and mobile) broadband networks will be openly available to existing and new service providers, constituting a world-wide common shared communications platform for a multitude of new advanced telecommunications services (telematic services - see also Section 2.2.) [ADA99d][AKIM97][MESS99]. The proliferation of these services will lead eventually to the transformation of the global information infrastructure to an open services market fuelled by deregulation, strategic partnerships, and joint interoperability activities [AUGU99][EGAN96].

Until recently, however, the traditional telecommunications environment consisted almost exclusively of telecommunications monopolies, in which public telecommunications organisations acted as both network operators and service providers. The choice of the telecommunications services offered by these Public Network Operators (PNOs) was driven mostly by available technologies rather than by customer demand [CARP92]. This was due to the fact that the telecommunications infrastructure was mostly based on vendor-specific network equipment with proprietary interfaces and limited capabilities. As each network platform provided a specific, functionally limited, set of services and because there was no common methodology for the introduction of new services across multiple network platforms, the development of new telecommunications services was a time consuming and very expensive activity.

Despite these conditions, the telecommunications world is currently experiencing dramatic changes. More specifically, the main directions of evolution that characterise the world-wide telecommunications
environment and are more likely to have a strong impact on it, can be summarised in the following (interrelated) trends:

1) **Rapid technological development.**

Progress in computing and telecommunications technology is proceeding at an increasingly fast pace [CIPR98]. As these technology advances are rapid and sustained, they have a disruptive impact on the markets that they create and serve. Continuing exponentially increasing advances in microelectronics [ISAA00], photonics [CHAT99][TODD93], mobile / wireless technology [TRIG98], and satellite communications [EVAN99][GOLD98] provide a cost-effective foundation and support the market drive towards the next generation of converged broadband telecommunications networks [DOWD98]. However, it will be the progress on telecommunications software [SARA96][VENI00], new (service and network) architectures and related practices / algorithms [BERNO0][FLOO98], and Quality of Service (QoS) designs [RUTK99], that will transform these networks to the appropriate infrastructure of a variety of telecommunications services, making them useful and thus desirable.

2) **Convergence of technologies and industries.**

Because of the expanding penetration of microelectronics and digitisation, the diversification and growing of information and communication needs, and the progress of the related technologies, telecommunications and computer science (or informatics or information technology) are no longer considered as two separate and distinct scientific fields or technologies. The term *Telematics* refers to their convergence, their continuously increasing integration, and to the new opportunities that emerge in that way.

Telematics creates also the necessary incentives and forces for more and more industries (e.g. broadcast, publishing, and video game industries) to try to join the same technological environment and create new industries, as well as new markets and services / products that will contribute to better life quality [RALP96]. It is evident that no enterprise / organisation will be able to exploit by itself the potential presented by the technological convergence. Therefore, achieving synergies among the different involved domains will become a necessity [GRAA96][HILL98]. For example, PNOs will have to define strategic alliances (by forming consortia, by acquisition, etc.) with other “partners”, such as software providers, publishers, equipment vendors, entertainment and information providers [GREE94]. Information will be the essence of this convergence trend [HELD94].

3) **Globalisation of economy and markets.**

A main trend affecting telecommunications policies is the increased integration of the world economy with the globalisation of major companies, without reference to national boundaries, in order to minimise costs

---

1 The term Telematics was created and first presented in 1978 by Simon Nora and Allan Mine as a synthesis of the words Telecommunications and Informatics [NORA78]. Many scientists consider the term Teleinformatics as a synonym of the term Telematics. Instead of these terms, in USA is in frequent use the term Computer Communications [HELD94].
and risks caused by economic, financial, and political modifications [CLEE96]. The merging of information technology and telecommunications provides a strategic infrastructure with tremendous capabilities for managing such global businesses.

In this environment, information becomes a commodity and the telecommunications business acquires gradually a global character. This is mainly due to the nature of its services (international traffic and services to multinational customers) and the wider internationalisation processes (acquisition of privatised public operators, procurement of new licenses in other countries, and strategic alliances) [HILL98].

4) **Radical changes in the regulatory environment.**

A shift is taking place from the traditional regulation approach, that was characterised by the monopolistic provision of telephony, to a new one concerning supply and demand relationships of telecommunications services [VENI00]. The main trend in the regulatory environment will be the high degree of liberalisation and privatisation of telecommunications markets, which is driven by the concepts of open network architecture in North America and Open Network Provision (ONP) in Europe. These two tendencies will be key factors for a nation's progress and integration into the emerging global economy, as the existing monopolies will disappear and new “players” will enter the scene, pushing for an open market of telecommunications services.

Regulation is expected to evolve, according to market conditions, network development, social and economic situations, incorporating more and more concerns of public interest, becoming as technology independent as possible, and moving away from mandatory harmonisation [EURE95].

5) **Increased competition.**

A main effect of liberalisation and privatisation is that telecommunications services and networks are opened to full international competition. In this way, services become less dependent on networks, which are made seamless by transparent agreements among competitors. The telecommunications market now involves multiple competing and cooperating providers of telecommunications services that can substitute or supplement each other, as services from one provider can be used by other providers as components of their own services, forming in that way value chains of services [ADA98c].

Competition is expected to become the best market mechanism to maximise user satisfaction by providing an increased choice of services, with better quality, lower prices, and continuous innovation [EGAN96]. However, it has to be noted that (more) competition does not mean necessarily less regulation [NIHO98]. New kinds of regulatory areas will appear regarding the management of competition conflicts with other policy objectives, the privacy and security of information, intellectual property rights, etc.

Furthermore, questions as to how more effective competition can be introduced, how quickly, and through what mechanisms, so as to stimulate innovation without jeopardising the financial viability of incumbent
telecommunications operators are being tackled increasingly frequently by policy makers. Although it is generally acknowledged that competition in the supply of the services and networks can provide a stimulus to innovation and efficiency, the timing of the relaxation of entry restrictions is the subject of a vigorous debate [MANS94].

6) Expanding standardisation activities.

To effectively answer the technological challenges of the emerging telecommunications environment and to provide both fair competition and regulated liberalisation, standardisation is necessary. The essential objectives of standardisation relate to interconnectivity and interoperability. Joint activities have to address suitable reference architectures, functional models, well-conceived building blocks, open interfaces and protocols, and appropriate migration scenarios, without restricting the needs and creativity of the different parties / countries involved [ARMB97]. Balance is also required between innovation and coordination taking into consideration the dynamic developments in regulation, markets and technologies, and realising the economic significance of standards [DAVI96]. Finally, the standards-making process is in need of a strong alignment between the technical aspects and the market and commercial considerations [WRIG95].

7) Proliferation and growing importance of the Internet.

The dramatic expansion of the Internet and its corporate incarnations (Intranets, Extranets), together with the popularity of the World Wide Web, are important developments that should not be underestimated. Graphical user interfaces provided by Web browsers give the user a convenient, visual, easily manipulated interface to the vast repository of (any type of) information available at Internet sites throughout the world. The Web has therefore become the first world-wide multimedia communication system. Albeit (relatively) slow and sometimes unreliable, the Internet is able to support voice telephony, electronic commerce activities, and multimedia telematic services, such as videoconferencing. For fast responses and a much higher reliability level several solutions are emerging, like the IPng (IP next generation), the Integrated Services Internet (ISI), TCP/IP over ATM, and cable-based Internet systems [CLAR96][RUTK99].

As the Internet paradigm prevails, telecommunications operators face an urgent need to re-assess their strategy with regard to it [BOUC98]. More specifically, they should make balanced, creative, and concerted efforts to devise methods and technologies in order to exploit the Internet and generate additional revenues through the provision of new services, thereby alleviating the (possible) revenue fall through service substitution [EURE97]. In any case, the state of the Internet and its evolution is a crucial matter for the future, whether it is going to be considered as the universal paradigm for the truly global information infrastructure, or as something destined to evolve naturally from its present state to be one of the elements of the overall telecommunications scenario [AKIM97][HUBA99].
A direct result of the above mentioned trends is the dramatic increase in the number, variety, and sophistication of telecommunications services that telecommunication companies offer in an attempt to satisfy the high and continuously expanding customer demand \(^2\) for powerful communication and information capabilities \([ADA99d][CIPR98][HELD93][MESS93][RUDG94][RUSS98]\). The timely availability of these new telecommunications services corresponding to the needs of the market is an important condition for gaining a competitive advantage and for the exploitation of the huge telecommunication potential that will be offered by the broadband network infrastructure that is currently under development \([MCK198]\). Therefore, telecommunications services have a characteristic and dominant role in the emerging telecommunications environment; a role that fully justifies the fact that they constitute the semantic "core" of the research work presented in this thesis.

In this context information networking is gradually gaining momentum, formulating an open market of new telecommunications services where the vision is "information any time, at any place, in any form" \([GREE96][MAGE95][MARS95]\). Within this electronic market, which is an important precondition for the emerging information society, information resources are available to everybody, without any practical restrictions, because in an all-digital environment, "there is no difference between bits which represent text, audio or video, since bits are bits" \([NEGR95]\).

The innovative developments that are taking place in today's diversified telecommunications world are simultaneously driven by progress in network / system architectures and software (technology push), and by the need for more universal and personal communication and information capabilities independent of location and time (market pull) \([ARMB97][CHAP95]\). It is thus evident, that in order to pave the way to the information society, it is necessary to define and develop a telecommunications service infrastructure (an information network), above the bearer network infrastructure, which will control and manage the distribution of information, in its various media manifestations, between geographically distributed user entities satisfying their needs in the best possible way \([ADA98d][MARS95]\).

In such an attempt software has a central role. The traditional method of telecommunications service development and provision is not possible to meet the increasing demands of a competitive, highly dynamic telecommunications services market. Such a market requires an increased "intelligence" of telecommunication networks, provided by software whose importance is constantly rising \([SARA96][TENN96]\). Based on the principles of distributed computing systems, an information network is realised as a distributed processing system in which the telecommunications services can be regarded as distributed software applications \([ADA00a]\). In this way the network becomes a programmable entity, made out of

\(^2\) The customer has an increasingly more demanding role in telecommunications. Customers are no longer driven by available service offerings but have an active role, asking for new, sophisticated, and personalised telecommunications services to solve their specific needs \([AUGU99]\).
service components and open generic Application Programming Interfaces (APIs), which is able to support network-transparent service provision [BERN00].

It is conceivable that the variety of existing and evolving networks will partially converge to establish full-service and application-ready networks. A common ATM core network could interlink the diverse access networks (including cordless and cellular mobile radio), ISDN, Broadband ISDN, Internet / Intranet and other data / corporate networks, as well as cable TV systems [ARMB97][POLE98][RAYC94]. This would be an important step towards the information infrastructures of tomorrow (e.g. information superhighway, the European and Global Information Infrastructure, EII-GII) that will enable people and systems to communicate securely with each other, any time, and anywhere with acceptable quality and at reasonable costs [EURE96][RALP96].

However, for the transfer towards the information society, many political and legal problems have still to be solved both on a national and international basis, in addition to technical and economical aspects [CLAR96][HELD94]. These problems relate to setting up an appropriate and stable regulation for competition, providing faster progress, greater choice, higher performance, and lower costs and to institute a suitable legislation for the content. The process of facing these issues requires common vision, harmonisation, joint action, and above all a careful balance between cooperation and competition.

1.2. The Emergence of Telecommunications Service Engineering.

Existing telecommunications systems are gradually converging into a ubiquitous information infrastructure inside an open deregulated multi-provider telecommunications market place. Additionally, the demand for (and the importance of) telecommunications services is increasing and will increase rapidly in the years to come. Based on these assumptions, which are fully justified by the analysis taken place in Section 1.1., it is evident that in order to derive a viable paradigm for the broadband information highway of the future, telecommunications service engineering (or simply service engineering) should emerge as a separate scientific discipline of strategic importance.

Service engineering can be defined as the discipline addressing the technologies and engineering processes required to define, design, implement, test, verify, validate, deploy, maintain, and manage telematic services that meet user needs in the current or future networks [BAIL95][TRIG95][ZNAT97]. Sometimes, the term integrated service engineering is used to emphasise the importance of considering different levels of functionality when examining telematic services. The attribute "integrated" makes reference to the need for information, management, service, component, and interface integration (integration aspects), required to support effectively the development of new telecommunications services [RACE95].

The concepts, principles and rules of service engineering have been initially borrowed from software engineering, although there is a clear distinction between the two disciplines [AIRC94]. However, in order to
handle the large size of possible services, their open-endedness and the variety of systems that will have to interwork, service engineering gradually incorporated a vast number of concepts already defined and developed in the fields of both information technology and telecommunications. Its main objective is to ensure the introduction of new and enhanced services and their management, in a fast and efficient manner. Therefore, service engineering covers the following domains:

- **Service creation**, where the service is considered as a distributed application running on the multiple nodes of a telecommunications network.

- **Service management**, which is concerned (mainly) with ensuring that services are provided to customers efficiently, and the service requirements agreed between the customer and the provider are satisfied.

- **Network management**, which concerns the management of network resources used to provide telecommunications services.

Telecommunications service engineering is an important research area at the boundaries of software engineering and telecommunications, relying heavily on open distributed processing and the object-oriented paradigm [ADA98b][CAMP94][CUSA93]. It ambitiously promises to significantly facilitate the offering of a wide variety of highly sophisticated and personalised services over the widest possible coverage area. In the light of these challenges, the research work presented in this thesis can be considered as a contribution to the scientific area of telecommunications service engineering.

1.3. Thesis Motivation.

Because of the recent diversification of the telecommunications environment, which was described and examined in Section I.1., the importance of telecommunications service engineering, as a discipline with profound practical implications, continuously increases. Therefore, service engineering activities, in order to retain their usefulness and fulfil the emerging increased expectations regarding their value and impact, have to transform and adapt to the new conditions that shape the telecommunications world of the 21st century.

The corresponding evolution process is characterised by a number of *requirements* that service engineering activities should satisfy in the new era. The incentive for considering most of these requirements has its origin in the experience gained by the application attempts of recent developments of the telecommunications and computing technology in real situations [PAPA96][PAPA97][PAP98a][PAP98b][PAP98c][PAPA99] and in the influence of recent research in service engineering [BERN00][CHAP95][MAGE97][MARS95][RACE95][TRIG95]. More specifically, service engineering activities should support:

- **The efficient and effective development of telecommunications services** by guiding successfully service developers during the entire service creation process, i.e. from the elicitation and analysis of the service...
requirements up to the actual implementation of the telecommunication service under examination.

(Requirement 1)

• The successful application in a telecommunications context (through the appropriate adaptation when necessary) of carefully selected concepts, models, techniques and practices that are proposed, developed, tested and (extensively) applied by a variety of computer science disciplines, ranging from software engineering to information systems development.

(Requirement 2)

• The reduction of complexity and the increase of efficiency during the design and implementation of telecommunications services by hiding from the service developers commonly encountered implementation details regarding (mainly) the heterogeneity and the distributed nature of the necessary underlying computing and network infrastructure.

(Requirement 3)

• The efficient automation of the service creation process, without semantic loss, with the use of appropriate, carefully designed and tested, customisable, and user-friendly software tools.

(Requirement 4)

• The development of a rich variety of telecommunications services, including conventional telephony and data transmission services, as well as advanced multimedia services with enhanced content, which can efficiently support a variety of communication, information, education, entertainment, and cooperation needs. The introduction of new types of services should also be possible.

(Requirement 5)

• The examination and exploitation of new and / or emerging communication concepts, such as Internet-based and agent-based services, terminal mobility, personal mobility, service mobility, application transfer, ubiquitous connectivity, single session abstraction, and application adaptation [ADA99d]. (Requirement 6)

• The representation, processing, management, and transmission (possibly in an integrated manner) of all the basic information types, which include data, text, vector, graphics, still images (bitmap), video signals (moving pictures or motion video), and audio signals (voice and sound).

(Requirement 7)

• Precise service semantics, because in a telecommunications market where several service providers offer a great variety of services, it is important for reasons of service interoperability and for maximising customer satisfaction to specify services in a clear and unambiguous way by using concepts that their semantic content can be accurately defined.

(Requirement 8)

• Reusability at different abstraction levels (e.g. reusable service requirements, service specifications, service components), with the intention to promote rapid service design and deployment.

(Requirement 9)

• The management of services in a flexible manner by facilitating the integration of control and management aspects of services.

(Requirement 10)

• The interoperability of services in a multi-provider (open) telecommunications environment (with multiple domains of management and ownership of services) by facilitating and promoting service composition and service federation.

(Requirement 11)
• **The accommodation of relevant standards** in the cases that the adherence to them is considered to be important and necessary.  
  \((\text{Requirement 12})\)

• **The efficient and accurate handling of service interactions** in multi-service networks, so that the (intended) functionality of a service is preserved by protecting it against accidental and intentional interference.  
  \((\text{Requirement 13})\)

• **Openness to all types of potential end-users of a service** considering all the possibly interested people (e.g. mobile users, residential users, professional users, etc.). Therefore, user-friendly services tailored to individual tastes, preferences and needs, and designed according to sound usability practices should always be the norm.  
  \((\text{Requirement 14})\)

• **Openness to change of service software and hardware** (computer and network infrastructure), because as technology advances, or as prices change, or as purchasing policies and needs dictate, different hardware should be able to be used without requiring new investment in the accompanying software, and vice versa.  
  \((\text{Requirement 15})\)

• **The accommodation of legacy telecommunications services and systems** as they represent significant investments that should be protected. In most cases, interworking with them (even if it is only for a transition period) is essential.  
  \((\text{Requirement 16})\)

The necessity for the fulfilment of the above mentioned requirements, and therefore the requirements themselves which should be fulfilled by the variety of service engineering activities that take place in the reshaped telecommunications market at a constantly increasing rate constitute the *motivation* of the work presented in this thesis.

The importance of this motivation is increased when considering that some of the above mentioned requirements are used to shape the “vision” and the strategic business orientation of telecommunications and Information Technology (IT) related organisations and corporations world-wide [BELL94][EURE95]. Furthermore, part of these requirements are mentioned (directly or indirectly) in the Bangemann report [BANG94] as important factors affecting the realisation of the future forthcoming information society and their scientific in-depth examination is clearly promoted by related European Union (EU) research activities [TRIG95][VENI00].

1.4. **Thesis Objectives.**

In an attempt to revitalise telecommunications service engineering and prepare it for the crucial role that is anticipated to have in the new emerging telecommunications environment, the telecommunications service engineering framework of *Figure 1.1.* is proposed. In this thesis, it is argued that the adoption of the proposed framework of *Figure 1.1.* will enable (the various) service engineering activities to satisfy all the
requirements identified and presented in Section 1.3. (not just a subset of them), fulfilling at the same time the corresponding expectations that were the motivation of this research work.

![Figure 1.1. The proposed telecommunications service engineering framework.](image)

As can be seen from Figure 1.1, the proposed framework is placed inside an "organisational context" in order to signify that service engineering activities are normally performed by service developers or service designers (called sometimes collectively service engineers) working for various organisations or enterprises (e.g. service providers, PTTs, outsourcing companies, etc., or specialised departments of large corporations). Therefore, the proposed framework is influenced (see also Section 5.4) by their knowledge, their problem solving attitude and their experience, and also by the more general telecommunications and strategic orientation of the organisation / enterprise that they work for.

The main constituent parts of the proposed telecommunications service engineering framework, which are depicted in Figure 1.1., are:

- **A service development methodology**: It is a methodology that guides service developers in a systematic and structured way during the entire process of service creation.

- **A Service Creation Environment (SCE)**: It is actually a collection of software tools (together with a reuse infrastructure) that are used according to the service development methodology with the aim to assist the service developer(s) when applying the service development methodology by automating and simplifying as much as possible the service creation process, and by facilitating consistency and verification checks.

3 In some cases, limited service engineering activities (usually related to the customisation of services) are performed by the end-users of a service. Then, the proposed framework is influenced by the knowledge, the problem solving attitude, and the experience of the specific end-users.

4 The SCE is in close cooperation with the service development methodology and therefore its software tools are considered to be used in an "integrated" way; hence the use of the word "environment" in the term SCE.
• A Service Support Environment: It is an environment aiming to facilitate, both the development of
telematic services (in cooperation with the service development methodology and the SCE), and their
execution under real conditions. It consists mainly of:

- Service engineering principles: These are concepts, guidelines, practices, and (in general) mental
constructs that are applicable to service engineering activities.

- A Service execution environment: It encompasses the necessary computing and network infrastructure
and the appropriate ancillary software (e.g. operating systems, database management systems, etc.),
which is needed for and during the execution of a telematic service. However, its most important part is
the Distributed Processing Environment (DPE) which abstracts over all the other parts and reduces
greatly the effort needed for the implementation of a telematic service.

From this brief presentation it is evident that the proposed framework is still an abstract entity that is in
need of further examination and specification. The aim of this thesis is to consider the proposed framework
as a conceptual “umbrella” for service engineering activities and gradually transform it to a more precise and
concrete construct (see also Chapter 7) by examining its constituent parts and reasoning about the way that
they support the requirements identified in Section 1.3. It has to be noted, that without affecting
completeness, two of the constituent parts of the proposed framework receive special attention and are
examined in detail as they present increased research interest and practical value. Therefore, these
constituent parts and their in depth critical study form the main objectives (which reflect the main research
contribution and correspond to the main deliverables) of the research work that is presented in this thesis.

More specifically, the main objectives of the thesis, expressed in a concise way, are the following:

• To propose and present a methodology, that was conceived and constructed for the development of new
telecommunications services, and “covers” in a systematic and structured manner the entire service
creation process through a requirements capture and analysis phase, a service analysis phase, a service
design phase, a service implementation phase, and a service validation and testing phase. This
methodology recognises the inefficiency of current general-purpose software engineering methodologies to
address successfully service engineering matters and proposes a novel service creation process based on
fundamental object-oriented analysis and design concepts and on important results of the
Telecommunications Information Networking Architecture Consortium (TINA-C). The novel character of
the proposed methodology is reinforced by the adoption of an incremental and iterative use case driven
approach, by the careful incorporation of the Unified Modelling Language (UML) notation throughout the
service creation process, by the exploitation of design patterns, and by the promotion of reusability (see
also Chapters 4 and 5).

• To validate and evaluate the proposed methodology, and provide a comprehensive example of its use by
applying it to the development (from requirements elicitation up to actual implementation) of a complex
representative telematic service (a MultiMedia Conferencing Service for Education and Training, MMCS-ET). This validation attempt, together with a number of smaller validation attempts (in terms of complexity, not in terms of scope) focusing on a number of simpler telematic services aim to provide tangible evidence about the correctness, the efficiency, and the true practical value of the proposed methodology (see also Chapter 5).

- To enhance Microsoft's Distributed Component Object Model (DCOM), which is the distributed object platform proposed and promoted by Microsoft, with the capability of handling continuous media interactions like those commonly required by new multimedia telecommunications services (as the MMCS-ET). In this way, DCOM becomes a strong candidate for a TINA-C compliant DPE and suitable for use in cooperation with the proposed service development methodology. Therefore, after designing, implementing, and evaluating a number of multimedia support services, which offer to DCOM the desired functionality, DCOM is used for the implementation of the MMCS-ET, during the validation of the proposed methodology, in full agreement with its intended role by the proposed telecommunications service engineering framework of Figure 1.1. (see also Chapter 6).

Beside the above main objectives, which have a central role in the thesis, other (secondary) objectives of the thesis are the following:

- To define important terms / concepts in the area of service engineering.
- To justify the use of object orientation in the design of telematic services.
- To consider important general purpose object-oriented software development methodologies for the development of telematic services.
- To compare important architectural frameworks for telematic services.
- To examine the future perspectives of TINA-C.
- To examine the structure, the role, and the purpose of a SCE.
- To evaluate the most important DPEs from the perspective of telecommunications service engineering and comment on their future evolution.
- To identify the main benefits of using distributed object platforms to design and implement telematic services.
- To propose a decision framework that facilitates service developers to select between Microsoft's DCOM and Object Management Group's (OMG's) Common Object Request Broker Architecture (CORBA).
- To examine the performance of DCOM and CORBA.
- To examine the ability of DCOM and CORBA to interwork with the WWW and the Internet.
- To compare DCOM and CORBA in the area of telecommunications service engineering.
• To specify the functionality of a multimedia conferencing service suitable for education and training purposes.
• To describe several simple service scenarios that can be used for validation purposes.
• To exploit the concept of design patterns in the service engineering area.

The approach that is followed for the realisation of the objectives of this research work is briefly outlined in Section 1.5. It is based on the recognition that service engineering activities should be examined and performed at different abstraction levels, the latter being the result of a critical filtering process that ignores characteristics that are not relevant from some point of view, with the minimum possible semantic loss.

Therefore, the overall research contribution of this thesis has an architectural and systems engineering nature, and focuses mostly on methodological, specification and modelling matters, with the validation done through software implementation. It is thus evident that this is not a “performance examination” type of thesis, although a performance analysis is taking place in certain parts of the thesis (see Section 3.4.3. and Section 6.4.) for comparison and evaluation reasons. Finally, it has to be noted that all the research work included in this thesis was undertaken by the author alone (any influences are stated clearly and suitable references are used) and not in the context of collaborative research projects.

1.5. Thesis Structure and Overview.

This chapter was the introduction to the thesis and revealed the scope, the motivation, the objectives, and the importance of the research work that was undertaken. The remainder of this thesis is organised as follows.

Chapter 2 considers advanced approaches in telecommunications service engineering. After discussing important issues associated with the object-oriented development of telematic services, it presents key technologies in emerging multi-service networks, and describes the two most important architectural frameworks for telematic services, reasoning about their role. It concludes by examining the structure and purpose of SCEs.

Chapter 3 provides a thorough analysis of the role of distributed computing platforms in Telematics. It examines the most important DPEs from the perspective of telecommunications service engineering, it discusses the benefits that distributed object technology offers to telematic services, and it compares DCOM and CORBA with the intention to assist service developers in a possible selection process between them.

---

5 This necessity becomes (especially) evident during the presentation of the proposed service development methodology in Chapter 4.
6 System engineering is the effective application of scientific and engineering efforts (in an integrated manner) in order to transform an operational need into a defined system configuration through the top-down iterative process of requirements analysis, functional analysis and allocation, synthesis, design optimisation, test, evaluation, and validation [BLAN98].
Chapter 1: Introduction

Chapter 4 presents and examines in detail all the phases of the proposed service development methodology, focusing on their desired functionality and on the role and purpose of essential service models and artifacts. It also explains the necessity for such a methodology and highlights its novel characteristics and its benefits.

Chapter 5 validates the proposed methodology by applying it to the development of a complex representative telematic service and to several simple service scenarios. It also extends the methodology by examining the exploitation of design patterns in the service engineering area, and evaluates it according to the Normative Information Model-based Systems Analysis and Design (NIMSAD) framework.

Chapter 6 enhances DCOM with the capability to handle continuous media interactions by designing and implementing a set of multimedia support services together with a related Application Programming Interface (API). It also presents a stream communication algorithm, considers important related implementation issues, validates the proposed modelling approach, discusses the experimentation with it, and compares it with the OMG's Audio / Video streams specification.

Chapter 7, summarises the results of the research work described in this thesis, identifies the original research contributions, outlines possible directions for further research, and contains a number of important concluding remarks.

It is evident that Chapters 4, 5, and 6 are related to the main objectives of the thesis and therefore constitute the most important part of it and the main research contributions, while Chapters 2 and 3 are related to the secondary objectives of the thesis. For reasons of semantic completeness and in order to preserve a clear and cohesive structure, state of the art work is presented throughout this thesis, in easily identifiable places, before the presentation of the research work regarding a particular matter. Finally, it has to be noted that the last section of each chapter includes references to the publications of the author that contain related material.

7 Throughout this thesis, research work undertaken by the author is indicated by using third person and / or passive voice (e.g. it is proposed, it was found, ..., etc., meaning, I propose, I found, ..., etc.).
Chapter 2:

Advanced Approaches in
Telecommunications Service Engineering

2.1. Introduction.

The telecommunications industry is characterised by rapid technological development, market growth, and deregulation and is heading towards a fully liberalised global market, where regulatory changes are breaking down the traditional barriers between public and private domains (see also Section 1.1.). The result of this liberalisation is a fragmented market with a multiplicity of competing and / or cooperating providers of telecommunications services, as services from one provider are used by other providers as components in their services forming “value chains” of services. Increasing customer needs press for the creation, operation, and management of many types of services, ranging from simple telephony services to new advanced multimedia telecommunications services, motivating in that way the further evolution of telecommunications service engineering.

This evolution path is shaped and directed by several concepts and technologies, which promise to revolutionise service engineering activities, increasing their efficiency and value. For this reason, and because the research work presented in this thesis aims also at the enhancement of service engineering activities by enabling them to fulfil a number of requirements (see Section 1.3.), this chapter examines advanced approaches in telecommunications service engineering in an environment open to competition and open to changes in market and technology. In this way, the rationale for proposing the telecommunications service engineering framework of Figure 1.1. is revealed and explained, and the necessary conceptual background and terminology for reasoning about its structure is introduced.

As was mentioned in Section 1.2., the discipline of telecommunications service engineering embraces the development of telematic services. Therefore, this chapter, after this introductory section (Section 2.1.), presents an attempt to define the term / concept “telematic service” and a set of related expressions in Section 2.2., with the intention to resolve the corresponding terminology confusion and enable the accurate
and unambiguous interpretation and comprehension of the research contribution of this thesis, which is inextricably intertwined with the notion of service in telecommunications.

After these necessary clarifications, Section 2.3. focuses on the development of telematic services in the light of the expanding penetration of the object-oriented paradigm in telecommunications. More specifically, after a brief introduction to important object-oriented concepts in Section 2.3.1., the main requirements associated with service design are identified and ways that object orientation can support them are examined in Section 2.3.2. Finally, Section 2.3.3. proposes the application of object-orientation on the complete life cycle of telematic services and considers for this purpose a set of carefully selected general purpose object-oriented software development methodologies. However, Section 2.3. is not only a place for state of the art descriptions, as it justifies the use of object orientation for the design of telematic services and highlights the inefficiency and inadequacy of current general purpose object-oriented methodologies for service development purposes, supporting in that manner the decision to propose a specialised object-oriented service creation methodology (see also Section 4.2.), which is one of the main objectives of this thesis.

Section 2.4. extends the discussion about the development of telematic services and comments on the evolution of telecommunications software, as a result of the application of object-oriented technology and the emergence of Multi-Service Networks (MSNs). In this context, Sections 2.4.1. and 2.4.2. present state of the art descriptions of the technologies of Intelligent Network (IN) and Telecommunication Management Network (TMN) respectively, and Section 2.4.3. considers their possible integration. In general, Section 2.4. reveals the main initial semantic underpinning of the research work presented in this thesis and points out the necessity for a new architecture beyond IN and TMN.

More specifically, Section 2.5. considers the Open Services Architectural (OSA) framework developed within the RACE project CASSIOPEIA and the framework defined by the Telecommunications Information Networking Architecture Consortium (TINA-C), as the two most important architectural frameworks for the unified provision of (new) communications and management services, using object-oriented software modelling techniques, within a common DPE. Sections 2.5.1. and 2.5.2. provide a state of the art overview of the basic OSA and TINA-C concepts respectively, with particular emphasis on their service architecture, while Section 2.5.3. compares and contrasts the two frameworks with the intention to offer a better understanding regarding their capabilities and their limitations. Finally, in Section 2.5.4. one of these architectural frameworks is selected to be the service support environment of the proposed telecommunications service engineering framework of Figure 1.1. The reasoning behind this decision is presented, together with an explanation of how Requirements 2, 3, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, and 16 (see Section 1.3.) are satisfied and a discussion about the practical value and the future perspectives of the chosen architectural framework.

Recognising the difficulty of applying architectural frameworks like OSA and TINA-C for the development of telecommunications services, Section 2.6., focuses on Service Creation Environments
Chapter 2: Advanced Approaches in Telecommunications Service Engineering

(SCEs), attempting to define them and identify their main characteristics. In order to reason about the role and purpose of the SCE in the proposed telecommunications service engineering framework of Figure 1.1., this section examines important related approaches in a critical manner, highlights the relation of the SCE with the service development methodology, and explains under which circumstances the SCE is able to satisfy Requirement 4 (see Section 1.3.).

Finally, this chapter ends with Section 2.7., which summarises the main findings of this chapter, highlights the research contributions in it, and paves the smooth transition to Chapter 3.

2.2. Defining Telematic Services.

With the advent of new technologies and recent changes in regulations, telecommunications services have grown chaotically. Because of these conditions, the terms “service” and “telecommunications service” have received many definitions in the service engineering literature and are commonly used in multiple senses.

Therefore, in the area of communication protocols, a service usually designates the set of primitives that a given protocol layer provides to the upper protocol layer. In [RACE92] (RACE project ROSA, see Section 2.5.3.), a telecommunications service is defined from an ODP enterprise viewpoint as “... a meaningful set of capabilities provided by an existing or intended network to all who utilise it, like customers, end users, network providers and service providers; each one sees a different perspective of the service”. [RAC92b] (RACE project CASSIOPEIA, see Section 2.5.1.) defines a service recursively as “... a set of service capabilities or a set of services”, considering a service capability as “... the ability of a system to cause, in order to satisfy a need of a using context, a predefined effect on substance”. Finally, a TINA-C service (see also Section 2.5.2.) “... is a meaningful set of capabilities provided by an existing or intended system to all business roles that utilise it; each business role sees a different perspective of the service” [TIN97a].

Taking into account these definition attempts, which are among the most important, in this thesis a service is considered to be a (packaged) set of capabilities, for which billing can be arranged, offered to a customer/user by a service provider with the use of telecommunication networks. More specifically, a telematic service is a geographically distributed entity (actually an encapsulation of a number of cooperating entities distributed over a geographical environment) providing a number of people (users, subscribers) a predefined, carefully selected, set of capabilities/facilities regarding the integrated coverage of a (possibly) wide range of information and communication needs, utilising the resources of (existing and future) telecommunication networks.

It has to be noted that the term “telecommunications service” emphasises the network requirements of a service and the network resources that are necessary for its provision, and therefore it is usually used by network providers and those with a similar perspective. On the contrary, the term “telematic service” emphasises the software structure of a service and its functional characteristics as perceived by its users. Thus, this term is usually preferred by service providers, service developers, end users, and those with a
similar perspective. In this thesis, in order to avoid terminology confusion, the two terms are used interchangeably, with the term “telematic service” or simply “service” being preferred.

2.3. Object-Oriented Development of Telematic Services.

As telecommunication networks become more advanced in the emerging telecommunications environment (see Section 1.1.), customers are motivated to request more varied and diverse telematic services, which in turn leads to larger, more geographically dispersed, and more complex telecommunication systems. Therefore, complexity tends to be a characteristic feature of telecommunication systems, which is manifested by the large number and the sophistication of the provided services, that have an increasingly complex nature [FITS96][VENI00]. A typical telematic service interacts frequently with other services and with its end users and is possible to consist of thousands of software objects that are running on hundreds of hardware objects utilising a great variety of network resources. In order to provide the necessary functionality, all these objects are interacting in a complex and almost unpredictable fashion [FITS95].

This high complexity makes the development of a telematic service, starting from requirements capture and proceeding towards the implementation of the service code a very difficult task [ADA00f]. To master this complexity, create new efficient telematic services as quickly as possible, without making quality compromises, and be able to manage them successfully through time ensuring their usability and reliability, telecommunications software must be built with ways that are easy to understand, apply, and extend [MOSS95].

Initially, structured techniques were used for this purpose. These techniques offer a top down decomposition of the subject matter (the service under development), focusing on the functional parts and the relations between these parts [AVIS95]. However, as the telecommunications market was becoming more dynamic, time-to-market started to be a crucial factor in service development and the ability to leverage on already existing service components gradually gained importance, as it can offer a competitive advantage. Under these conditions the demand for reusability was increased and the value of structured techniques started to be questioned, as they fail to offer the levels of reusability, modifiability and extensibility that would be necessary. Correspondingly, telematic services that are implemented using these techniques do not scale easily and lack the flexibility and the sophistication that is often required to ensure the viability of newly introduced services. Furthermore, service development projects that use structured techniques are increasingly facing long development cycles and high costs [MORE95][VENI00].

The application of object-oriented technology in the development of telematic services promises to provide a solution to these problems as object orientation presents some key features necessary for tackling such challenging problem domains [DECL97][YAMA93][YAU92]. Among them (see also Section 2.3.1.) the most important are the improved interaction between the service developer and the problem domain expert, the inherent support for reuse, the increased internal consistency of the service analysis and the service
design results, the explicit representation of commonality, and the construction of service specifications that are resilient to change [FITS95]. However, despite the great potential of object-oriented service development, it has to be stressed that there is no "silver bullet" [BROO87] that can unfailingly lead a service developer, without any kind of mental effort from his / her side, down the path from requirements specifications to the implementation of a telematic service.

In general, this section examines important issues regarding the application of object orientation to the development of telematic services motivated by the increased penetration of the object-oriented paradigm in the telecommunications world [ACT96a][EUR96b][ITU92a][RAC92b][TINA95a][VENI00]. Initially, basic concepts of the object-oriented technology are analysed. Then, the main requirements associated with the design of telematic services are identified and ways that object orientation can support them are examined. Finally, an integrated approach is considered.

### 2.3.1. The Concept of Object Orientation.

Object orientation reflects a deep and natural human perception of the world, as there is philosophic evidence for the existence of a human cognitive filter based on abstraction and classification. The fundamental construct is an object which is the "building block" of any object-oriented system. Objects are instances organised into classes with common features. These features comprise attributes and procedures, called operations or methods. Objects should be as far as possible based on the real-world entities and concepts of the application domain. Each object behaves or interacts with its environment in a prescribed and understood way. More specifically, objects (classes and their instances) communicate by message passing. Messages may alter the state of an object, but leave the identity of the object unchanged (persistent identity) [GRA95a].

The packaging together of state (data / attributes) and behaviour (process / methods) in an encapsulated description is a key characteristic of object orientation. In this manner, the data structures and implementation details of an object are hidden from other objects in a system. Furthermore, in combination with message passing, data duplication is eliminated and changes to data structures encapsulated within objects do not propagate their effects to other parts of a system. Thus, encapsulation or information hiding refers to the practice of including within an object everything it needs, and doing this in such a way that no other object need ever be aware of this internal structure. Closely related to encapsulation is the concept of abstraction, which is the process of identifying relevant objects in an application, ignoring the irrelevant detail [RADI96].

In an object-oriented system the structural and semantic relationships between instances and classes are dealt with by considering two main kinds of hierarchy: the inheritance, classification, generalisation or kind-of hierarchy and the composition, aggregation or part-of hierarchy. Another key concept of object orientation is that of self-recursion or self-reference by which objects can send messages to their own methods.
Chapter 2: Advanced Approaches in Telecommunications Service Engineering

recursively (i.e. send messages to themselves). Thus, an object must know its own unique identity and be able to store itself as the value of one of its own attributes [PARS97][SOMM93].

Related to the term object-oriented are the terms object-based and class-based, which differentiate in the following way [ECKE94]:

- object-based = encapsulation + object identity
- class-based = object-based + set abstraction
- object-oriented = class-based + inheritance + self-recursion

Object orientation can provide several benefits [BOOC94][COCK93][CUSA93][GRA95a][GRA95b][RADI96][RUMB91], which can be summarised in the formula ¹:

encapsulation + inheritance + identity ⇒ reusability + extensibility + semantic richness

Object-oriented concepts and ideas initially appeared in programming languages (e.g. Smalltalk, Eiffel, Objective C, C++, etc.), resulting in a new technique (or paradigm) for programming (object-oriented programming), which can be considered as the evolution of procedural programming and modular programming. However, object orientation soon started to affect the process by which a developer proceeds from a statement of the problem to a detailed design that is then materialised in software. The result of this influence was the appearance of numerous object-oriented decomposition methodologies [GRA95a], which vary considerably, and in most cases use divergent terminologies, distinct graphical notations and emphasise different aspects of the software development process (see also Section 2.3.3.).

2.3.2. Object-Oriented Design of Telematic Services.

The application of object orientation to the development of telematic services is more likely to start from the service design phase, as service developers can recognise its benefits more easily during the design of services [FIT96][GRAH95][YAMA93]. Therefore, they are more willing to apply the object-oriented principles during this phase. For this reason, this section emphasises on design matters. More specifically, the design of telematic services has to be done in a way able to support a number of requirements. The most important of them are quality, modularity, management of complexity, and distributed processing. All these requirements can be efficiently supported by the object-oriented paradigm as is explained in the following paragraphs [ADA98c].

**Quality**

Service engineering is concerned inter alia with the construction of high-quality telematic services for a reasonable outlay of effort and thus cost. The quality of a telematic service is to be evaluated (mainly)

---

¹ It must be noted that inheritance can compromise reusability if not carefully managed [GRA95a].
according to the following criteria [GRA95a][KELL97][RICH88][SOMM93]: correctness, verifiability, robustness, reliability, maintainability, reusability, interoperability, efficiency, security, and friendliness.

Object orientation contributes to most of these objectives as object-oriented systems are characterised by inherent reusability and extensibility, and because of the fact that formal methods of specification may be made to blend with the object-oriented approach. Additionally, object-oriented design contributes significantly to the issues of reusability, maintainability, and interoperability of telematic services by employing mainly the techniques of bottom-up design, information hiding, and inheritance [PARS97].

More specifically, a bottom-up approach to design improves significantly the potential for service reuse permitting the specification of new services by modifying existing specifications and the creation of new services by relying on facilities made available by existing services. This happens because service partitioning is accomplished by packaging objects and not by breaking service tasks down into smaller and smaller components (top-down functional decomposition), which is an activity that is nearly always highly application specific and therefore leads to non-reusable modules. Due also to information hiding service modules are accessed purely via their interface, which is equivalent to a specification of the function of the module. Thus, information hiding can also contribute to security. Additionally, a simple interface contributes to interoperability and friendliness. Finally, inheritance makes exception handling easier, improves the extensibility of telematic services, and can facilitate reuse.

Correctness and robustness are addressed by formal specification and prototyping, which can be harmonised with object-oriented design [GRA95a][RADI96]. Furthermore, reusable service modules are likely to be more robust as every time they are used they are tested as well. Maintainability, reusability, verifiability, and reliability of telematic services are all supported by the simultaneously “open” and “closed” nature of object-oriented systems 2 [CUSA93].

Modularity

The importance of modularity in the creation of telematic services has been greatly emphasised [FELD95][SEIi93]. The main criteria for modularity in telematic services are decomposability, composability, understandability, continuity, and protection [MEYE88].

More specifically, decomposability refers to the requirement that services be decomposable into manageable parts so that they can be modified more easily. Decomposability is strongly related to decentralisation, which helps to simplify the design of a service by eliminating hidden dependencies, so that design changes are isolated to specific service modules and do not automatically affect the rest of the service. Composability refers to the property of service modules to be freely combined even in services for which

---

2 Object-oriented systems have this nature mainly due to inheritance. Meyer’s “open-closed” principle states that reusable systems should be both open, in the sense that they are easy to extend, and closed, in the sense that they are ready to use [MEYE88].
they were not developed. Understandability assists people to comprehend a service by looking at its parts prior to gaining an understanding of the whole. Continuity in a service implies both that small changes made to it will result in small changes in its behaviour, and that small changes in the specification will require changes in only a few service modules. Finally, the criterion of module protection insists that exception and error conditions, either remain confined to the service module in which they occur, or propagate to only a few other closely related service modules.

The principles required to ensure these criteria for service modularity are linguistic modular units, few interfaces, small interfaces, explicit interfaces, and information hiding. The phrase "linguistic modular units" refers to the need for a correspondence between the modules of a service and the syntactic units provided by the method used to design the service. It is evident that object orientation supports this principle by default. Few interfaces implies that each service module should communicate with as few others as possible, because of the difficulty of tracing complex message passing. The principle of small interfaces states that the interfaces should pass as few data as possible. The need for explicit interfaces arises from the need to use service modules via their specification rather than their implementation, so that in composition and decomposition the intermodule connections are known clearly. The last three principles may be combined as the principle of simple interfaces. Finally, the principle of information hiding states that service modules should hide the design decisions used in their construction.

The relationship between the criteria and the principles for modularity in telematic services (which criteria are ensured by each principle) can be seen in Figure 2.1. Object orientation supports all five principles and thus guarantees the modularity of telematic services.

Management of complexity

The design of telematic services is an inherently complex task as it involves interaction among several processes. Every telematic service results from a thorough examination of a problem domain. An abstraction of the key elements of this problem domain and a decomposition into service objects and inheritance, aggregation, and usage structures based on real-world objects and concepts assists with the management of
complexity in several ways. First, the telematic service and the problem are in close correspondence, so that
general knowledge in service engineering can be mapped onto the solution. Second, encapsulation divides the
service up into coherent parts, which are small and simple enough to be understood as a whole. Third, object
orientation provides several ways to model structure and meaning. Lastly, the reusability and extensibility of
object-oriented systems mean that complex services may be assembled from simple ones (service compo-
nents) and that they can evolve incrementally in simple steps [GRA95a][PARS97].

**Distributed processing**

The vast majority of new telecommunications services can only be provided if a high level of cooperative
computing (not just signalling) based on distributed processing is made possible between network nodes and
terminal nodes with advanced processing capabilities. Additionally, telematic services must communicate
(service nesting) and share resources in a distributed manner [FELD95].

It is natural to think of a telematic service involving distributed processing as an entity composed of
objects communicating by message passing, which again is the most natural metaphor to use to describe
networked, cooperating service components. Thus, object orientation has the modelling expressiveness and
semantic richness to express distributed processing in a telematic service [EMME00]. In fact the application
of object orientation in the field of distributed processing has been so successful that resulted in the
emergence of the “distributed objects” paradigm (see also Section 3.3.) [GUER99].

### 2.3.3. Towards an Integrated Approach.

Currently, the application of object orientation to the telecommunications industry is mainly focused on the
modelling and specification of communications entities, especially network and service elements, and on the
programming of telecommunications software [VENI00]. However, the full potential of object orientation
can only be realised when applied to the complete life-cycle of a telematic service, which includes all the
activities that are required to support the development, the operation and the maintenance of a service (see
Section 4.2., Figure 4.1.), because then it can deliver, in the most efficient and effective manner, all its
benefits [ADA99d]. This application of object orientation can take the form of the reference model of Figure
2.2., which is the reference model of the Object Management Group (OMG) for Object-Oriented Analysis
and Design (OOAD) [OMG93] modified for telematic services.

In this reference model, object modelling provides a set of terms and concepts for representing everything
within the scope of a telematic service life-cycle as an object. Service requirements modelling refers to the
elicitation of the service requirements and to their appropriate structuring and representation. Service
analysis modelling covers the process of obtaining a description and gaining an understanding of the (main
concepts related to the) service under development. Service design modelling refers to the logical design of
the service and consists of adding information to service class specifications and producing a solution to
some particular problem using service objects. Service implementation modelling is physical design and involves designing service modules, the distribution strategy, and taking account of software (tools) and hardware to be used. Finally, service construction refers to the production of the service code, service testing to the examination of the implemented service under various conditions, and service deployment to the actual provision of the service by placing it in its real operational environment. In general, the object-oriented metaphor forms a unified conceptual model covering all the phases of the reference model of Figure 2.2. (paradigm consistency), in an integrated manner.

In order to expand the application of object orientation, according to Figure 2.2., a service creation methodology is necessary to guide the service developer(s) and structure the service development project [ADA00][DECL97]. One possibility to satisfy this need would be to use one of the numerous general purpose OOAD methodologies that have been appearing during the last decade [HEND99]. However, many of these methodologies are quite immature both theoretically and empirically, and they are based on very limited practical experience. Moreover, some of them are weakly documented or they are still undergoing evolution [IIVA95]. Therefore, the comparison of OOAD methodologies, in order to decide about their suitability for service creation purposes, seems to be unavoidable.

### Figure 2.2.: The OMG reference model for OOAD modified for telematic services.

Most of the existing comparative analysis of OOAD methodologies are non-empirical, focusing on the similarities and differences between the methodologies. [ARN091] compares systematically five object-oriented development methods with regard to important concepts, models and notations, process aspects, and pragmatic aspects. [SUTC91] includes a brief feature analysis of five OOAD methods, while [CHAM92] discusses typical features shared by the majority of Object-Oriented Analysis (OOA) methods as well as their differences, and also compares very briefly a number of methods with regard to their concepts

3 The terms method and methodology are used interchangeably in this section. See also Section 4.4.
and notations. [FICH92] compares conventional and OOAD methodologies, [MONA92] compares 24 OOAD methods with regard to features of the OOAD process, representations and complexity management, and [CIVE93] examines the concept of composite objects in six OOAD methodologies. [HONG93] uses a metamodelling approach to compare the concepts / notations and the development process in six OOAD methodologies, while [LOSA94] compares the graphical notations of three OOAD methodologies. [ECKE94] includes a comparison of four OOA methods focusing mainly on their structural aspects, [IIVA94] examines the suitability of six OOA methods for information systems analysis, and [YUAN96] evaluates six OOAD methodologies based on a number of modelling criteria and on model integrity. Finally, [HEND99] contrasts and compares two object-oriented development approaches focusing on process and life cycle support, metamodelling, and notation.

All the above comparisons, except for the fact that they refer to different OOAD methodologies, have a very broad scope and they consider matters from a purely software engineering perspective. Therefore, due to their lack of focus and their orientation, they are not really useful (in fact someone could argue that they are even confusing) for deciding about using a OOAD methodology for the development of telematic services. For this reason, in the following paragraphs a simple, but focused and well-thought, OOAD methodology comparison is attempted.

The OOAD methodologies that are considered are well documented, successfully applied in many (and various) cases, non-proprietary, reasonably specific, and enjoy a wider acceptance. These are:

- "Object-Oriented Analysis and Object-Oriented Design (OOA/OOD)" of Coad / Yourdon [COA91a] [COA91b].
- "Object Modelling Technique (OMT)" of Rumbaugh et al. [RUMB91].
- "Object-Oriented Analysis and Design (OOAD)" of Martin and Odell [MART92].
- "Object-Oriented Software Engineering (OOSE)" of Jacobson et al. [JACO92].
- "Object-Oriented Systems Analysis (OOSA)" of Shlaer and Mellor [SHLA88][SHLA92].
- "Object-Oriented Analysis and Design (OOAD)" of Booch [BOOC94].
- "Design Object-Oriented Software (DOOS)" of Wirfs-Brock et al. [WIRF90].
- "The Unified Software Development Process" of Jacobson, Booch, and Rumbaugh [JACO99].

Initially, the modelling ability of the selected OOAD methodologies is examined regarding structure, function and behaviour aspects, because all the corresponding modelling capabilities are considered to be absolutely necessary when developing telematic services [ACT96a][FITS95]. More specifically, structure aspects describe static service objects, their relationships, attributes and their possible states, function aspects describe the input-output transformations, and behaviour aspects the instantiation and dynamics of
the transformations with time [IIVA95]. It has to be noted that the classification into structure, function, and behaviour is a generalisation of the distinction between data, function and control, which is widely applied in the information systems literature [GRA95a], and corresponds to the distinctions between information model, process model and state model [SHLA92] and between object modelling, functional modelling and dynamic modelling used in the object-oriented community [RUMB91].

<table>
<thead>
<tr>
<th>Structure Abstractions</th>
<th>OOA/OOD, Coad/Yourdon</th>
<th>OMT, Rumbaugh et al.</th>
<th>OOAD, Martin and Odell</th>
<th>OOSE, Jacobson et al.</th>
<th>OOSA, Shlaer and Mellor</th>
<th>OOAD, Booch</th>
<th>DOOS, Wijs-Brock et al.</th>
<th>The Unified Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object &amp; class, Attribute, Gen.-spec. structure¹, Whole-part structure², Subject, Instance connection</td>
<td>Object &amp; class, Attribute, Generalisation¹, Aggregation¹ (fixed, variable, recursive), Association and link, Module, Sheet</td>
<td>Object type, Relationship, Generalisation², Composition³</td>
<td>Object and instance, Attribute, Domain, Control object, Attribute, Association (inheritance¹, acquaintance, consists-of¹), Subsystem</td>
<td>Object and instance, Attribute, Domain, Subsystem, Relationship, Subtype¹, Supertype¹</td>
<td>Object &amp; class, Attribute, Inheritance relationship¹, &quot;Is-part-of&quot; relationship, Subsystem</td>
<td>Class, Attribute, Subclass, superclass¹, Package, Template class, Composite object, Component</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

|-----------------------|---------|-------------------------------------|-----------|------------------------------------|---------------------------------|--------------------------|----------------------------------------|---------------------------------|

<table>
<thead>
<tr>
<th>Behaviour Abstractions</th>
<th>State transition, Message connection</th>
<th>State transition, Scenario, Event, Event trace, Event flow, Input/output values</th>
<th>State transition, Event, Trigger, Triggering rule, Control condition</th>
<th>Use case, Communication association</th>
<th>State transition, Bridge, Event, Accessor</th>
<th>State transition</th>
<th>Collaboration, Contract, Message</th>
<th>State transition, Event, Action, Activity</th>
</tr>
</thead>
</table>

¹ Refers to the modelling of inheritance.
² Refers to the modelling of composite or complex objects.

Table 2.1.: Structure, function, and behaviour abstractions in the selected OOAD methodologies.

The structure, function, and behaviour abstractions used in the selected OOAD methodologies are summarised in Table 2.1. Regarding the structure abstractions, from this table is evident that the methodologies are quite similar in the way in which they structure individual objects and classes, even though the terms differ slightly from one to the other. Furthermore, inheritance is a common feature of all the methodologies, while only some of them (Coad/Yourdon, OMT, Martin and Odell, OOSE, Booch, the Unified process) include the idea of composite or complex objects. Important is also the fact that all the methodologies, except for that of Martin and Odell, introduce concepts to describe collections of objects and classes. Regarding the function abstractions, Table 2.1. reveals that there is considerable variety in the modelling of the functionality of individual objects and classes, and that the specification of the functionality of set of objects is only addressed by some of the methodologies (OOSE, OOSA, DOOS, the Unified process). Finally, Table 2.1. indicates that the modelling of the internal behaviour of individual objects is quite uniform in the selected methodologies, as all of them, except OOSE and DOOS, adhere to the use of state transition graphs / diagrams. However, there is considerable variety between the methodologies with regard to their ways of modelling the behaviour of set of objects.
In general, Table 2.1 reveals that all the selected OOAD methodologies are more or less capable to satisfy the modelling needs that appear during the development of telematic services. However, when considering the detailed descriptions of the methodologies using Table 2.1 as a guide, it becomes evident that most of them are biased towards either structural, functional or behavioural modelling. More specifically, Coad / Yourdon, OMT, OOSA and Booch emphasise structural modelling, OOSE favours functional modelling, and Martin and Odell and DOOS focus on behavioural modelling. Only the Unified process is characterised by a balanced attention to the three modelling perspectives, which are also considered in detail. Because of this fact, and because the Unified process adopts the use of the Unified Modelling Language (UML) [BOOC98] (see also Section 4.3.), which is rapidly emerging as a standard modelling notation, this OOAD methodology appears to be more capable to support the modelling of telematic services.

<table>
<thead>
<tr>
<th>Methodology</th>
<th>Direct support for service concepts / constructs</th>
<th>Support for reuse</th>
<th>Tool support</th>
<th>Customisability</th>
<th>Integration</th>
<th>Precision</th>
<th>Organisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>OOA/DD, Coad / Yourdon</td>
<td>No</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
<td>High</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>OMT, Rumbaugh et al.</td>
<td>No</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>OOAD, Martin and Odell</td>
<td>No</td>
<td>Low</td>
<td>Medium</td>
<td>Medium</td>
<td>Low</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>OOSE, Jacobson et al.</td>
<td>No</td>
<td>No</td>
<td>Medium</td>
<td>Medium</td>
<td>Low</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>OOSA, Shiue and Mellor</td>
<td>No</td>
<td>High</td>
<td>High</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>OOAD, Booch</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>DOOS, Wirfs-Brock et al.</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.2: Evaluating the selected OOAD methodologies for service development.

The OOAD methodology comparison is completed by considering, for each of the selected OOAD methodologies, a set of properties (direct support for service concepts / constructs, support for reuse, tool support, customisability, integration, precision, organisation) that are essential for applying successfully, in an effective and efficient manner, a methodology to the development of telematic services. The results of this examination are summarised in Table 2.2., which reveals that most of the methodologies are characterised by most of the necessary properties (at varying degrees). The problem is clearly created by the first property (direct support for service concepts / constructs) which does not characterise any of the selected OOAD methodologies and therefore makes them unsuitable to be used for service development, although they possess the necessary modelling abilities (as was explained earlier).

To be more precise, the unsuitability of the selected OOAD methodologies stems mainly from their general purpose character. These methodologies are designed with the objective to support the development of every possible software system. Therefore, they are rather abstract and sometimes complex as they try to take into account every possible situation, but without losing their generic character. These methodologies

---

4 This property indirectly characterises the understandability and ease of use of the methodology.
have thus to be customised in order to be used for the development of telematic services [EUR96b]. Otherwise, if such a methodology is applied as it is, the service developer will spend considerable time and effort to "reinvent the wheel", for example, by devising again basic service engineering concepts (as that of a session), and try to incorporate them appropriately in the original methodology. It is obvious that such an approach is impractical, inefficient, and very open to errors. At this point, it has to be stressed that the necessity to customise general purpose methodologies for use in a specific domain or for specific purposes is widely acknowledged by the information systems research community [CHAT97] and has led to the appearance of the "methodology engineering" paradigm [BRIN96][HIDD97][KUMA92].

The customisation of a general purpose OOAD methodology for service engineering activities is generally a difficult and time consuming task. More specifically, best practices regarding service development have to be identified, expressed in an appropriate manner and incorporated successfully in the methodology, together with various important service concepts. Additionally, proper validation of the resultant methodology is necessary in order to ensure its applicability and its practical value. Such a customisation attempt is more difficult, more demanding, and more risky than creating a service development methodology from scratch, because the service developer who will do the customisation must have a very good knowledge of the original methodology (and even significant practical experience on its use), in order to (among others) select and remove the parts of the methodology that are not necessary (because they increase complexity without a reason), introduce the desired constructs / concepts without violating any of the basic methodology principles, amend the proposed sequence of activities (usually in the form of steps) to incorporate the changes, and ensure the overall consistency. Finally, it has to be noted that the customisation of a methodology increases the possibility of inheriting its deficiencies (because the emphasis is on the introduction of new concepts / constructs), while it does not guarantee the preservation of its advantages.

Another reason for the inadequacy of the selected OOAD methodologies for service development purposes is that most of them, either lack completely a requirements capture and analysis phase, or they do not support sufficiently the related activities, because they consider the system requirements (their elicitation and their structuring) as given (by means external to the methodologies) [GRA95a]. This is a serious inefficiency in the context of service creation where the success of a telematic service (its acceptance by the end users) depends significantly on the accurate identification and the clear and efficient representation and structuring of the service requirements [CLAR94]. This inefficiency should be addressed when customising a general purpose OOAD methodology. Another matter that should be addressed then refers to the modelling notation used by the methodology, which should be UML (wherever possible) due to its important benefits and its excellent perspective (see also Section 4.3.).

5 It is even recognised by most of the selected OOAD methodologies. For example, the creators of the Unified process state explicitly that it is actually a "customisable framework" [JACO99].
Finally, from the above discussion the need for a specialise service development methodology that adopts fundamental OOAD concepts, has a requirements capture and analysis phase, and uses UML should be evident. This need is also highlighted in Section 2.6. and is finally satisfied in Chapter 4 by proposing and presenting such a methodology. In this way the full potential of object orientation in service engineering will be realised, facilitating the transition to an open integrated broadband telematic service infrastructure, populated by a great variety of services.

2.4. Key Technologies in Emerging Multi-Service Networks.

The rapidly expanding penetration of the object-oriented paradigm in almost every software development activity combined with the increased demand for new telecommunications services that started to characterise the transforming telecommunications environment (see Section 1.1.), resulted in the strengthening of the awareness regarding the potential benefits of the application of object orientation in telecommunications (see Section 2.3.) and in a corresponding growing interest for the introduction of object-oriented technology in the development of telecommunications services [EUR96b][FIT95][VENI00].

Furthermore, telecommunication networks began to evolve towards integrated services networks or Multi-Service Networks (MSNs), which are networks capable of supporting a wide range of services [SMIT94]. Traditional telecommunication networks are monolithic constructs developed with specific services in mind and their structuring, their protocols, and the procedures and concepts they define reflect this. On the contrary, MSNs constitute and promote a generic network infrastructure with a large set of simple facilities which are employed by actual services; the more complex a service is, the greater the set of elementary facilities it uses and the more intricate the interactions between them that it prescribes. Therefore, in MSNs, services are viewed as value adding applications, composed using more elementary facilities available underneath and operating on top of a general purpose communications subsystem. For this reason, the emphasis is placed on finding a set of elementary facilities that would lend themselves to systematic arrangements and would form a coherent framework for the composition of (more complex) services [GEIH96][VENI00].

Under these conditions, the telecommunication software has also significantly evolved. This evolution is reflected in miscellaneous (yet interrelated) areas, such as the signalling (control) software, the (service and network) management software, and the application of the object-oriented and distributed processing paradigms. It is this evolution that is mostly related with the research work presented in this thesis, as it affects mostly the development of telematic services. Therefore, the following subsections present in a concise manner the main concepts pertaining to the technologies of Intelligent Network (IN) and Telecommunication Management Network (TMN), and also comment on their possible future integration. These two technologies constitute the initial semantic underpinning of this thesis as they were the first that incorporated successfully and efficiently state of the art information technology into the telecommunications world.
2.4.1. Intelligent Network.

The Intelligent Network (IN) is an evolving open service-oriented network architecture which represents a basis for the uniform creation, introduction, control, and management of services beyond the basic telephone services in the telecommunication environment.

IN represents a first step towards programmable switched telecommunication networks [CHAP95]. Instead of replicating the service logic and data in all switching nodes, the basic idea of the IN is to move the network intelligence to centralised, high performance and reliable computing nodes in the network, and remotely control the establishment of call connections performed by the switches. For this reason, as can be seen in Figure 2.3., within each switch a Service Switching Point (SSP) is introduced. On receipt of certain phone numbers, the SSP triggers intelligent processing by passing a request to a Service Control Point (SCP) [FAYN96]. The fundamental prerequisite for this approach is the real-time connection between the SSPs and the SCPs, which is achieved by a Common Channel Signalling no. 7 (CCS7) network, supporting also the IN Application Protocol (INAP) between SSP and SCP [ITU95a].

Figure 2.3.: Simplified architecture of the intelligent network.

In order to support the provision of an open set of services, the IN defines an extensible set of generic and reusable service components, known as Service Independent Building blocks (SIBs). These can be combined into higher level service elements (service features) in order to enable the construction of new services. This process is facilitated by the IN Conceptual Model (INCM), which defines a top down approach for the structuring of IN architectures based on IN service capabilities through four planes addressing service design aspects, global and distributed service provisioning functionality, and physical aspects of an IN-structured network [ITU95b].

The main strength of INs originate from the fact that they are an overlay on an existing network, and thus require little change to the existing systems. However, this is also their weakness, since the capability of the underlying network is limited to telephony. Future broadband services will require the ability to handle large multi-party calls (e.g. videoconferencing) and modify the connections during the progress of a call (e.g. to
increase and decrease bandwidth on demand). It is doubtful whether the current underlying telephony network and the IN concept as it is can meet these future service needs without extensive modification [CHAP95][HUBA99].

The standardisation of IN takes place within ITU-T and ETSI in a coordinated manner [ITU93]. Both bodies aim for the definition of an IN Long-Term Architecture (LTA) in stages and focus their work on the development of recommendations for a series of upwardly compatible IN Capability Sets (CSs) 6. Each CS is defined in terms of the services to be supported and the functional architecture supporting these services. More specifically, CS-1 covers single-ended services and is followed by two other phases (CS-2 and CS-3), which extend it with more sophisticated capabilities [MAGE96]. This phased approach of standardisation and thus evolution facilitates the extension of the IN service capabilities and architecture 7 as required by the market demands and the incorporation of important IT technological developments (e.g. the emerging intelligent broadband network) [TRIG98][VENI00].

2.4.2. Telecommunication Management Network.

The Telecommunication Management Network (TMN) framework defines a well-agreed open management 8 architecture that provides a minimum but necessary platform for providing interoperability between different management systems and network components in the short term and a basis of an integrated system for managing all kinds of telecommunication networks and services in the long term, supporting their planning, provisioning, installing, maintaining, and administering [ITU92a][MAGE96]. The basic concept behind the TMN is the provision of an organised network structure for achieving the interconnection of various types of management systems (which carry out specific management tasks) with the telecommunications equipment in the underlying managed network [HALL93]. A TMN is conceptually a separate network that interfaces with the telecommunication network that it manages 9 at several different points to send/receive information to/from it and to control its operations [SHRE95].

As can be seen in Figure 2.4., a TMN has a distributed hierarchical nature and is based on a hierarchy of five management layers. Each layer restricts management activities within the boundary of the layer to a clearly defined rank (business, service, network, and element management). On the bottom layer is the physical equipment or network elements. Above this is the element management layer which contains functions for the management of each network element on an individual basis, whereas the network management layer has the responsibility for the management of all network elements, both individually and

---

6 It has to be noted that the objectives of the Advanced Intelligent Network (AIN) releases defined by Bellcore in North America are similar [BELL94].

7 Although this process in practice proved to be difficult and time-consuming [MAGE96].

8 Management refers to a set of capabilities related to the FCAPS management areas (Fault, Configuration, Accounting, Performance and Security) [ITU92b].

9 A TMN may use parts of the telecommunication network to provide its own communication.
Chapter 2: Advanced Approaches in Telecommunications Service Engineering

as a set. The network management layer provides the functionality to manage a network by coordinating activity across the network and supports the "network demands" made by the service management layer, which contains functions for the provisioning of services and the management of customer subscriptions. The business management layer contains functions related to managing the business processes of a network operator or service provider [AIDA98].

![Figure 2.4.: A simplified view of the functional TMN hierarchy.](image)

As indicated in Figure 2.4, each layer is represented by a corresponding Operations System Function (OSF), which processes information related to specific telecommunications management activities [MAGE93], and all vertical and horizontal interactions within a domain take place at generic "q" reference points, while interactions between the OSFs of different TMNs take place at "x" reference points. Furthermore, the TMN defines a manager-agent model, according to which manageable resources are modelled by managed objects at different levels of abstraction [SHRE95].

The definition of the TMN information model is based on object orientation. Therefore, the TMN adopted the OSI Systems Management (OSI-SM) [ITU92b] as the basis for the TMN information architecture and thus the Common Management Information Service and Protocol (CMIS/P) [ITU91a][ITU91b] for distributed object access across standardised interfaces. However, it has to be noted that the TMN is not tied to the OSI-SM framework and other technologies have also been considered [AIDA98][PAVL96][PAVL98].

Finally, for the design of TMN interfaces, a top-down interface specification methodology is described in [ITU95c]. According to this, the starting point is the identification of the desirable management services [ITU95d] and their decomposition to management service components, which state the requirements for actions to be performed on the managed network. These service components are then further decomposed into management functions [ITU95e], which are the smallest part of a TMN management service as perceived by the user of the service. This process ends with an appropriate object modelling phase which maps the man-
management functions onto managed object classes and a consolidation phase which produces the final management information schema. This methodology has been extended and refined in the area of constituents of management services and hierarchical decomposition by the RACE projects ICM and PREPARE [GRIF96].

### 2.4.3. Integration Issues.

The IN and TMN architectures have many similarities and complement each other, since the IN facilitates the uniform service creation and provision, whereas the TMN enables the uniform management of services and networks. However, there are also some important differences, as both architectures are defining their own sets of protocols, and use different terminology, notations and techniques to structure software. Therefore, the co-existence and integration of IN and TMN is urgently required and thus it is the subject of several international research and standardisation activities [ETSI92][MAGE96][PAVL98].

More specifically, effort is being spent in defining how IN systems and services can be managed using the TMN architecture, as TMN is expected to be deployed for nearly all telecommunication systems [KOCK95]. This will result in the reusability of management functions and associated logic (management service components) for all network technologies that share the same TMN management philosophy and to the unification of management processes [PAVL98]. The starting point for a TMN-based IN management solution is a comprehensive management requirements analysis, covering for each role present in the IN environment (service subscriber, service provider, network operator) all management activities within the entire lifetime of an IN service [ETSI94]. Based on the identified IN management requirements, a TMN management solution can be developed, comprising by a set of appropriate IN management services, a corresponding IN management information model, and an IN management architecture [MAGE93][MAGE96].

Another, more controversial activity regarding the IN and TMN integration is the attempt to achieve IN service capabilities by using TMN concepts [MAGE95]. In that way, IN functional entities are replaced by equivalent object-oriented TMN functional blocks, communicating with each other over the signalling plane (e.g. using CMIS/P, instead of INAP) [PAVL98]. There is evidence that this approach preserves the basic IN principles and achieves the desired functionality by considering the IN as a short-term implementation of a service management network [ECKA95].

Finally, the IN and TMN integration efforts revealed the importance and necessity of globally manageable telecommunications networks with a seamless service delivery far beyond the possibilities of INs. Therefore, the challenge for the long term is to integrate IN and TMN within a common platform, allowing the integrated creation, provision, and management of telecommunications services (comprising both telecommunications and management capabilities), by adopting object-oriented techniques and the ODP modelling approach, and by making use of a DPE as a ubiquitous supporting infrastructure which encapsulates the transport network [AIDA98][PAVL98].

33
2.5. Architectural Frameworks for Telematic Services.

In order to advance beyond IN and TMN, and realise the vision of information networking, the prerequisite is (as was explained in Section 1.1.) the provision of a global service infrastructure, based on a network independent open service platform, which should hide both the complexity of the heterogeneous network infrastructure and the diversity of the existing services, facilitating at the same time the fast and efficient creation, provision, and management of new telecommunications services [MCK198]. The correct design, the portability, and the reuse of interoperable software components realising telematic services will be of prime importance in this context [BER95b][VENI00].

Furthermore, as is evident from the analysis of Section 2A, IN and TMN are closely related [MAGE93], since they cover complementary aspects, but they are neither harmonised nor aligned with object-oriented service design and modelling principles as promoted by emerging Open Distributed Processing (ODP) standards (see also Section 3.2.1.) [ITU95a]. Therefore, various activities to “merge” IN and TMN have been initiated. However, these activities will fail unless the, often implicit, architectural incompatibilities are addressed and resolved. There is strong evidence, that an integration of IN and TMN should be done on a conceptual level and be based on an explicit architectural framework that unifies control and management of telecommunications services 10, while promoting object-oriented modelling techniques and distributed information processing [PAVL98].

Therefore, there is a need for an innovative architectural framework supporting in an efficient manner the rapid and flexible introduction of new telecommunications services, and the ability to manage them and the network infrastructure in an integrated way [PROZ97]. This framework is necessary to make use of recent advances in the field of distributed computing, incorporate object-oriented design principles for service modelling, consider both IN and TMN, and define a software architecture that offers reusable software components, supports network-wide software interoperability, cases and improves service design, construction, testing, deployment and operation, and hides from the service developer the heterogeneity of the underlying technologies and the complexities introduced by distribution [ROWB96].

The most important architectural frameworks, which have been developed taking into account the above mentioned needs / requirements, are the Open Services Architectural (OSA) framework within the RACE project CASSIOPEIA [RAC95a][TRIG95], and the framework defined by the Telecommunications Information Networking Architecture Consortium (TINA-C) [BARR93][CHAP95][INOU98][MAGE97]. In the following sections these two frameworks are presented focusing on their essential characteristics that are

10 The term “control” generally indicates network and terminal functionality allowing users to access and use services by the set-up, exploitation, and release of necessary resources. The term “management”, instead, indicates network and terminal functionality able to create, monitor, measure, and modify the status of resources and services. “Control” has traditionally been the realm of IN, whereas “management” has been in the scope of TMN [INOU99][TRIG98][VENI00].
most relevant with the research work presented in this thesis. Then they are compared and contrasted with emphasis given on important approaches followed pertaining their service architecture and component models. Finally, as a result of this evaluation process, one of these architectural frameworks is selected to be the service support environment of the proposed telecommunications service engineering framework of Figure 1.1.

Before proceeding, it has to be noted that there are also other approaches in the service engineering area that focus on (almost) the same direction with OSA and TINA-C. The most important of them are the future Intelligent Network (IN) as developed by the Eurescom Project P103 “Evolution of the Intelligent Network” [EUR94a][MUDH961 and the Information Networking Architecture (INA) initiative of Bellcore [NATA92]. However, these approaches are not further considered in this thesis, because the P103 approach has been very strongly influenced by IN technology, and the INA approach has been undertaken by a single company (Bellcore) without being specified at the appropriate level of detail. Furthermore, all of them represent research work that took place earlier or in parallel with OSA and TINA-C, and therefore their main findings are incorporated (directly or indirectly) into both OSA and TINA-C.

2.5.1. The Open Services Architectural Framework.

Recognising the need for a thorough re-evaluation of the way telecommunications services should be conceived, deployed, provided and managed in the emerging Integrated Broadband Communications (IBC) environment, the RACE (Research into Advanced Communication in Europe) program, focused through a series of projects, on the definition and validation of an architectural framework for open service design and deployment [TRIG95]. The most important of these projects was CASSIOPEIA [RACE95a] which resulted in the development of the Open Services Architectural framework (OSA) 11.

OSA is comprised of two architectures: the Open Services Design Architecture (OSA\_d) for the development of telecommunications services, and the Open Services Provisioning Architecture (OSA\_p) for the development of the environment in which these services will be provided. These two peer architectures complement each other. Each architecture includes means for the conceptual modelling during analysis and design of either services or systems in a suitable and problem oriented way.

In OSA, services are specified at different levels of abstraction in terms of OSA-components. These are mapped onto deployable components, defining the necessary computational objects to provide the service in a selected system environment. The decomposition and refinement of the OSA-components is supported and guided by OSA\_p, with the aim to achieve reuse and reusability of the designed services. OSA doesn't specify accurately the functionality that should be provided in any particular system environment, but allows the environment to be populated with any suitable set of deployable components, which can be used and combined to realise other services, and enables this set to grow with the deployment of new services.

11 RACE projects ROSA, SCORE, BOOST, DRAGON, EURSAF were also related to the development of OSA.
Chapter 2: Advanced Approaches in Telecommunications Service Engineering

The computational abstraction of OSA is called a service machine, and is populated by executable service specifications [RAC92b]. It represents the functionality enabling the deployment and provision of services, abstracting from its actual distribution over the network nodes. Any new deployed service becomes part of the service machine and can be reused by other services. A minimal service machine represents a minimal set of “functions” that can be taken for granted by any service designer. The functionality of the minimal service machine is organised around the following groupings: nucleus, resource access, distributed processing, and deployment. The service machine is engineered as a service network, which is defined as a network of service nodes, each hosting a “local projection” of the service machine and ensuring the availability of the minimal service machine’s functionality [TRIG95].

2.5.2. The TINA-C Initiative.

The emerging open service platform, envisaged by information networking, will result in the “virtualisation” of the (computing and communications) infrastructure, increasing significantly the proliferation of telematic services. Such a virtual infrastructure, which is the outcome of the adoption of a software perspective in telecommunications, embraces a much more dynamic and heterogeneous approach to the representation of information, the configuration of hardware and software systems that process it, and the binding of resources to its distribution, with a profound impact on almost every telecommunications related activity [TENN96].

Because of this perspective some of the world’s leading telecommunications and information technology companies established in 1992 the Telecommunications Information Networking Architecture Consortium (TINA-C). The main objective of TINA-C is to define and validate an open, innovative, and coherent architectural framework (a long term architecture for telematic services) that would address in an integrated manner service control and service management [TIN95b]. This framework encompasses the long term objectives of both IN and TMN [APPE93], applies ODP standards and object-oriented design principles, facilitates the design and provision of services in a heterogeneous system and network environment with different domains of ownership, and ensures the introduction of new and enhanced services and their management, much faster and more efficiently than with current approaches [BARR93][CHAP95][INO98][INO99][MAGE97][PROZ97][ROWB96].

TINA-C services are considered as software-based applications that operate on a distributed computing platform (a DPE). This platform hides from services the underlying technologies and distribution concerns, and supports in this way the portability and interoperability of the service code. Therefore, a service is realised by a set of interacting service components (i.e. computational objects interacting via their computational interfaces), which are distributed across different network elements [MAGE97][POLY98].

The encapsulated service and network resource components reside on top of a DPE, which consists of DPE kernels, implemented in different heterogeneous computing environments (DPE nodes) and
interconnected by a logical signalling network, known as the *kernel Transport Network (kTN)*, which is designed to transfer information between different DPE nodes [NILS95]. The functionality of the DPE kernels is enhanced by generically defined *DPE services* (trading, notification, transaction, repository, configuration, security and performance monitoring service), which provide useful additional features to the basic DPE communication environment [HAMA97][KELL95]. The DPE services are described independently of the *Native Computing and Communication Environment (NCCE)*, which supports the DPE by providing an abstraction of the computing and network hardware, the operating system and the communication protocol stack of the service nodes. Finally, the *Transport Network (TN)* models the underlying network and guarantees the handling of continuous audio and video streams for multimedia services [BERN00][INOU98].

The TINA-C overall architecture encompasses basic design and modelling rules, which are applicable to a wide range of services, and decomposes the complexity of the problem space into the following (sub)architectures [TIN95a]:

- **Computing Architecture**: It defines modelling concepts for designing and building distributed service software and the DPE. It describes the service component properties, their interaction mechanisms, and how such components can be executed, administered, and supported in a DPE. This DPE resides in heterogeneous pieces of equipment and, by hiding the distribution of the components, makes them function as a single supportive system for telematic services [TIN94a][TIN94b][TIN95e][TIN96a][TIN96b][TIN98a].

- **Service Architecture**: It provides a set of concepts and principles for service design, specification, provision, management, reuse, access, and usage. It uses the notion of session to offer a coherent view of the various events and relationships taking place during the provision of services [TIN94c][TIN95d] [TIN97a][TIN97e].

- **Network Architecture**: It describes a generic, technology independent model for setting up connections and managing telecommunications networks. It inherits concepts used in ITU-T and other standards bodies. It extends these concepts to integrate network control and management for different network technologies [TIN95c][TIN97c][TIN97d].

In TINA-C, the session concept has a central role [BER95b]. A *session* is defined as the temporary relationship among a group of objects that are assigned to collectively fulfil a task for a period of time [TRIG98]. A session has a state that may change during its lifetime and represents an abstract and simplified view of the management and usage of the involved objects and their shared information. Objects in a session are subject to common policies (specified in the management context) that govern the session [TIN97a]. Sessions can span multiple business administrative domains [GARC98]. However, since policies are only valid within a business administrative domain, it is useful to define a portion of the session that covers a single domain (the domain session). The scope of the session concept in TINA-C is depicted in

---

12 These concepts are based on the five viewpoints identified in the Reference Model for ODP (see Section 3.2.1.).
Chapter 2: Advanced Approaches in Telecommunications Service Engineering

Figure 2.5. In this figure a consumer is interacting with another consumer within a service session supplied by a retailer (e.g. during a videoconference service). It has to be noted that the business administrative domains participating in a service are able to take different session roles (user/party, provider, and peer when interacting with each other\textsuperscript{13} (see also Section 4.6.2.) [BER95c] [GARC98].

![Diagram of TINA-C session concept](image)

**Figure 2.5.:** The scope of the session concept in TINA-C.

As can be seen in Figure 2.5., a TINA-C session can be:

- **An Access Session:** It represents information and functionality related to user / provider interactions for choosing, starting, and maintaining a service. It is established when two Domain Access Sessions (D\_AS) are bound together in a secure relationship (i.e. in a domain session binding). The D\_AS is an abstract concept, which is further specialised, according to the different access session roles, into User Domain Access Session (UD\_AS), Provider Domain Access Session (PD\_AS), and Peer Domain Access Session (PeerD\_AS).

- **A Service Session:** It represents information and functionality related to the ability of executing, controlling, and managing services. A service session is the single activation of a service\textsuperscript{14}. It relates the users of a service together so that they can interact with each other and share various entities. The service session comprises a provider service session and usage service session(s):
  - **Provider Service Session (PSS):** It represents the core service logic and any additional provider information / logic necessary to execute service requests and maintain the session for the one or more domains participating in the service in one role or another. It contains the service capabilities common to multiple members of the service session.

\textsuperscript{13} In TINA-C, service provisioning between two actors is based on the user-provider paradigm. According to this paradigm, an actor in the user role will use a service provided by another actor in the provider role. To support the paradigm, two basic concepts are defined in TINA-C; the user domain and the provider domain [INOU99].

\textsuperscript{14} A service session cannot exist without the access session of the party holding the ownership of the service session.
Chapter 2: Advanced Approaches in Telecommunications Service Engineering

- **Usage Service Session (USS)**: It represents the participation of other domains in the PSS (it is their view of the service), and therefore reflects an across domain relationship between two Domain USSs (D_USS). The D_USSs are specialised according to the role of the participating domains within the USS. Specialisations are the User Domain USS (UD_USS), the Provider Domain USS (PD_USS), the Peer Domain USS (PeerD_USS), and the Composer Domain USS (CompD_USS).

- **A Communication Session**: It represents an abstract, service view of stream connections and network technology. In general, it handles connectivity provider interactions for a service session by managing the communication resources required to establish end to end connections.

The purpose of the session concept is to separate different concerns of service usage and to promote the distribution of service functionality [VENI00]. The separation of the access session and the service session allows the access methods and technology to vary for different users and the location of users accessing a service to change [MAGE97]. The separation of the PSS and the USS within the service session enables the distribution of the service logic. Finally, the separation of the access session and the service session from the communication session corresponds to the separation between call and connection (and hence between call control and connection control) [PAVO96].

In accordance with the supported session types, the TINA-C service architecture introduces a set of **generic software components** for the realisation of telecommunications services (service components), which can provide access session, service session, and communication session related functionality constituting in that way a framework for segmenting the overall functionality of TINA-C services [MAGE97][YAG95]. The **TINA-C service components** are high level abstractions 15, structured in a uniform manner according to the Universal Service Component Model (USCM) 16 [BER95b][YAG95]. They can be decomposed into (interacting) **Computational Objects (COs)** and / or **Computational Object Groups (COGs)** [TIN96b]. COs are defined as a unit of distribution over DPE nodes and thus all the functionality of a CO has to be supported by a single DPE node. COGs are not defined as a unit of distribution, but only in terms of their internal COs and COGs. It has to be stressed that it is up to the designer of a TINA-C service to decide precisely how each service component is decomposed into COs and COGs. The TINA-C service architecture does not place any restrictions on how the service components are decomposed and deployed, and it does not define their internal structure [INO99][PROZ97]. However, in practice COs are usually defined at different levels of abstraction, and are decomposed and distributed over multiple DPE nodes (if necessary) [DME99][POLY98]. Thus, for reasons of simplicity and ambiguity avoidance, in this thesis it is assumed that a service component corresponds to a CO and the two terms are used interchangeably.

---

15 Service components are defined in the computational viewpoint and therefore their definition is a computational specification [MERC97].

16 The USCM defines four parts in every component: core, usage, management and substance. Therefore, it extends the computational modelling concepts by allowing the grouping of objects that have a common purpose, and also provides guidelines on how to define (semantically) complete and manageable objects [TIN97e].
Chapter 2: Advanced Approaches in Telecommunications Service Engineering

The main generic service components \(^{17}\) defined by the TINA-C service architecture, with the objective to provide a framework offering a set of generic functionality (a kind of a generic “construction kit”) for the realisation of TINA-based telecommunications services in a stakeholder independent, service independent and interoperable way [ECKE97], are listed in Table 2.3. It has to be noted that when developing telecommunications services using the generic TINA-C service components, service developers (almost) always want to include additional functionality or customise the already existing functionality, in order to provide some competitive added value for their service. This can be achieved by deriving new service components from existing ones using specialisation and/or composition (see also Section 6.3.1.) [TIN97a][YAG195].

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>TINA-C Service Components</th>
<th>Main Functionality</th>
<th>Domain Role</th>
<th>Session Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>as-UAP</td>
<td>access session User Application</td>
<td>It models/represents a variety of applications &amp; programs in the user domain.</td>
<td>user</td>
<td>access</td>
</tr>
<tr>
<td>PA</td>
<td>Provider Agent</td>
<td>It is the user’s end-point of an access session.</td>
<td>user</td>
<td>access</td>
</tr>
<tr>
<td>IA</td>
<td>Initial Agent</td>
<td>It is the initial access point to a domain.</td>
<td>provider, peer</td>
<td>access</td>
</tr>
<tr>
<td>UA</td>
<td>User Agent</td>
<td>It represents a user in the provider domain.</td>
<td>provider</td>
<td>access</td>
</tr>
<tr>
<td>namedUA</td>
<td>named User Agent</td>
<td>It is a specialisation of the UA for a user that is an end-user or a subscriber of the provider.</td>
<td>provider</td>
<td>access</td>
</tr>
<tr>
<td>anonUA</td>
<td>anonymous User Agent</td>
<td>It is a specialisation of the UA for users that do not wish to disclose their identity to the provider.</td>
<td>provider</td>
<td>access</td>
</tr>
<tr>
<td>PeerA</td>
<td>Peer Agent</td>
<td>It represents a peer in another domain.</td>
<td>peer</td>
<td>access</td>
</tr>
<tr>
<td>ss-UAP</td>
<td>service session User Application</td>
<td>Enables a user to make use of the capabilities of a service session, through an appropriate user interface.</td>
<td>party</td>
<td>service</td>
</tr>
<tr>
<td>SF</td>
<td>Service Factory</td>
<td>It creates the service session components for a service type and controls their life-cycle according to requests from UAs.</td>
<td>provider, peer</td>
<td>service</td>
</tr>
<tr>
<td>USM</td>
<td>User service Session Manager</td>
<td>It represents and holds the context of a party or resource in a service session.</td>
<td>provider</td>
<td>service</td>
</tr>
<tr>
<td>SSM</td>
<td>Service Session Manager</td>
<td>It supports service capabilities that are shared among users in a service session.</td>
<td>provider</td>
<td>service</td>
</tr>
<tr>
<td>CompUSM</td>
<td>Composer Usage Session Manager</td>
<td>It supports composition of service sessions.</td>
<td>party</td>
<td>service</td>
</tr>
<tr>
<td>PeerUSM</td>
<td>Peer Usage Session Manager</td>
<td>It supports peer to peer relationships between service sessions in different domains.</td>
<td>peer</td>
<td>service</td>
</tr>
<tr>
<td>CSM</td>
<td>Communication Session Manager</td>
<td>It provides the appropriate connectivity functionality to the SSM and manages application-level, end-to-end bindings between stream interfaces (stream flow connections).</td>
<td>provider</td>
<td>communication</td>
</tr>
<tr>
<td>TCSM</td>
<td>Terminal Communication Session Manager</td>
<td>It interacts with the CSM to realise application’s requests for network connections.</td>
<td>user</td>
<td>communication</td>
</tr>
</tbody>
</table>

Table 2.3.: The main service components that are proposed by the TINA-C Service Architecture.

Due to the nature of the research work presented in this thesis, this section emphasises the TINA-C service architecture, which provides the basis for the realisation of future multimedia multi-party telecommunications services. A more complete “picture” of TINA-C can be formed and a better understanding of it can be achieved by considering the related service engineering literature, which (beside the official TINA-C documents) addresses all the important aspects of TINA-C in considerable detail: the computing architecture [MERC97][WAKA94], the proposed structure of the DPE [DANG96][KELL95], the service architecture [BERN94][BER95b][GARC98][YAG195], the management approach followed [BER95c][DELA94][HAMA97][PAVO96], and the network architecture [BLOE95][LENG96] \(^{18}\).

\(^{17}\) It is stressed that only the most important COs that are relevant to this research work are mentioned. TINA-C defines many more COs addressing a variety of aspects (e.g. accounting management, subscription, etc.) [TIN98a].

\(^{18}\) The references that are given are indicative and not exhaustive.
Furthermore, [BARR93] [BERN00] [CHAP95] [INOU98] [INOU99] [MAGE97] [NILS95] [PROZ97] provide an overview of TINA-C at different time periods and thus they reflect its progress through time. From their examination is evident that the important architectural parts of TINA-C were delivered by the end of 1997, when its first phase was completed. During this first phase, the European Union (EU) supported several TINA-related research projects (TINA auxiliary projects) through the ACTS program (e.g. DOLMEN, REFORM, ReTINA, SCREEN, TOSCA, VITAL), which resulted in research output that has been widely acknowledged as a valuable contribution to TINA-C, as it influenced significantly its refinement in certain areas (through conceptual extensions and specification enhancements) and its solidness (through validation experiments) [BERN00]. The feasibility of key architectural concepts and relevant component specifications were proven by international trials (The TINA Trial - TTT - project) in the 1996 - 1998 period. The second phase of TINA-C started in 1998 with the objective to facilitate the market-driven adoption of TINA-C by focusing on the provision of detailed component specifications and their implementation, aiming at selected business cases. In this context, nine TINA-C technical workgroups and one special interest group have been formed, creating specifications on a variety of matters, by following a Request for Proposal (RfP) process that favours proposals of practical value, characterised by the availability of prototypes, pre-products or products [BERN00][INOU99].

Following this evolution path, TINA-C has reached a mature state and TINA-C products gradually appear in the market, enforcing the value, the importance, and the influence of TINA-C, which will continue to progress at a steady pace [MOTA99].

In the future, more sophisticated services breaking away from the simple call model (e.g. multimedia, multi-party conferencing) will need to be rapidly and efficiently introduced, deployed, operated, and managed. TINA-C makes this possible, by capitalising on the latest advances in computer and telecommunications technologies to rationalise the organisation of complex software for services and network management. For this reason, it prepares the emerging global, multi-supplier, multi-operator telecommunications sector and promises to make the future open information market a reality.

2.5.3. Comparing OSA and TINA-C.

Both OSA and TINA-C have been influenced by the previous RACE project ROSA (RACE Open Services Architecture). ROSA aimed to define an architecture for the specification, design, construction and implementation of a broad range of service types, including conventional telephony and advanced information services, based on object orientation and advanced distributed processing techniques [HALL93][OSHI92]. In addition to a similar ancestry, OSA and TINA-C share also a common service concept. In both of them, services are considered as software-based applications that operate on a DPE that enables interactions between distributed objects, which collectively realise a specific telecommunications service [MAGE97].

41
Driven by similar objectives, both architectural frameworks share some fundamental modelling and separation principles, which assist in service development and ensure the identification of correct problem domain structures increasing the possibility of reuse. Therefore, these principles are followed as guides and views are provided on how they are reflected in OSA and TINA-C [ADA98d][BER95a][CHAP95].

**Mission oriented and system independent modelling of services**

Services can be modelled from a user/customer perspective (mission oriented) and be independent of any system in which they will be provided. In OSA this principle is reflected in the OSA-component which is structured into mission and ancillary facets. Additionally, to achieve system independent modelling, OSA decouples service and system concerns by the division into OSA\textsubscript{APP} and OSA\textsubscript{SYS}, and contains in OSA\textsubscript{APP} system independent abstractions of resources. In TINA-C this principle is interpreted in two ways. Firstly, TINA-C specifies the control of the execution of services but does not restrict the functionality of the service logic. Secondly, abstractions of the resource and computing infrastructures are defined, which provide technology independent interfaces on which services can be built.

**Coherent design of management and control**

Control and management should not be taken as separate tasks in the service design, but have to be integrated to ensure the design of manageable services. In OSA there is no difference in how control and management services are modelled. OSA contains no specific support for designing management applications, but facilitates (e.g. through ancillary facets in OSA-components) services to be designed such that they are manageable. In TINA-C control and management are treated in a consistent way as there isn't a syntactic distinction between management and control operations. The interfaces offered by objects in a TINA-C system are defined in the same way, irrespective of whether the interface will be used for management or control. Furthermore, management services should be designed according to the TINA-C service architecture and offered by encapsulating them into service sessions [DELA94].

**Service nesting for definition and provision of new services**

Specifications of services are made in a general context, with the view that its users may be other services, not necessarily humans, and its resources may be other services as well. In OSA nesting is decomposed into two separated (although linked) sets of problems: the definition of services based on the definition of other services, and the provision of services through reuse and combination of other services. Concepts and rules in OSA\textsubscript{APP} intend to promote reusability of designed services. To support nesting of services during provisioning, any new deployed service becomes part of the service machine and can be reused by other services. In TINA-C the execution of a service is accomplished by the creation and control of a service session via user agents. In principle, it is possible for two service sessions to interact with one another, where one service session acts like a user agent.
Customisation and personalisation of services

Users and customers must be able to set their preferences for the use of a specific service. OSA has no predefined set of services and assumes that customisation is the rule and not an exception. It contains the concept of service configurations, which through use of guidelines, can be easily combined and configured according to the needs of individual users or particular groups of users. Like OSA, TINA-C has no predefined set of services either. It provides a consistent deployment and subscription architecture, but leaves several opportunities for customisation and personalisation (e.g. in user agents and in usage service sessions).

Separation of media services from their control and management

Services for information generation, transport, transformation, and presentation (media services) differentiate from services controlling, coordinating, and managing media handling resources (media control services). Efficiency is the focus in the former case, while flexibility is the focus in the latter case. This separation is directly reflected in the telecommunications service domain of OSAApp, where media services and media control services are seen as separate domains. OSA does not aim to give support for engineering of the media services. In TINA-C end-user applications exchange multimedia information through the use of streams. During a service session, when streams are needed, a CSM provides a service-oriented abstraction of connections in the transport network (communication session).

Separation of local from global aspects of services

The users of a service have, as part of their contract with the service provider, a local view of the service. On the other hand, the service provider is necessarily has a global view of the service in order to mediate and coordinate the users effectively. In OSA this separation is reflected in the use of local and global sessions. In this way, a user necessarily has only a local view of the services which participates. In TINA-C local aspects are separated from global aspects in two cases. Firstly, in subscription applications, it is possible to define nestings of groups of people where capabilities are defined by the containing group, and subgroups can extend or refine the capabilities. Secondly, an executing service will comprise a provider service session and at least one usage service session (see also Section 2.5.2.).

Separation of local resource handling from global resource coordination

The provision of services to users requires the allocation, coordination, and use of resources. A separation between handling of local resources (e.g. links and switching devices), and coordination of global network resources (e.g. connections), need to be made. Network resources in OSA are considered as part of the network infrastructure. The services of these resources are provided within an OSA system via resource components. Consequently, local resource handling is regarded as a task of the resource infrastructure, while global resource coordination is handled by the media service control services (e.g. connection management services). In TINA-C streams between terminal COs are established by communication sessions, via service
sessions upon the request from the SSM. Therefore, terminals are responsible for their own internal resource management of ports (or sockets) and bindings, and connection management is responsible for the management of network resources.

**Separation of service access from the service core**

Users should be able to use different access arrangements, at different locations, to access a service, and access many services using the same access arrangements. In OSA access services and guidelines for their support belong to the enterprise support domain of OSA_{APP}. An access service is used during an access session, to ensure that a user (user agent) has access to a set of OSA services in a providing context. For this reason, access services use user profiles, which are sets of user specific information (e.g. user authentication information). In TINA-C user access is distinguished from terminal access. A user agent represents and acts on behalf of a user. It receives requests from users to establish service sessions, or to join existing service sessions. A terminal agent is responsible for representing a terminal. To access a service, users must associate their user agents with terminal agents.

**Separation between application / session oriented and resource / communication oriented problems**

The application oriented problems are related to the support of users and applications in their use of communication services. The resource oriented problems are related to allocation, monitoring, and management of communication resources. In OSA (specifically OSA_{APP}) application support services (supporting the modelling of session control and mobility) are separated from the connectivity and communication problems supported by the domains of media services and media services control services. In TINA-C this separation is incorporated in the differentiation between service sessions and communication sessions.

### 2.5.4. Evaluation and Concluding Remarks.

From Section 2.5.3, it is evident that OSA and TINA-C have many similarities and present a surprising commonality in the technical approach that they follow. They both have in common a service driven approach, as opposed to being driven by network and resource technology, the separation of service design and system / DPE design, integrating management and management systems with services and resource infrastructures, and applying object orientation and ODP principles. However, OSA and TINA-C have some important differences in scope, which lead to different ways to express their architectures.

More specifically, OSA is emphasising requirements engineering, while TINA-C has the explicit goal of leading to a software architecture, and thus addresses software engineering issues. As a consequence, TINA-C is much more focused on computational issues, while OSA is more oriented towards problem decomposition on a more conceptual level. There are no major inconsistencies between the architectures. A service modelled using the OSA framework can probably be implemented using the TINA-C architecture, and similarly a TINA-C service would not violate the main rules of OSA [RAC95a][TIN97a].
However, it must be noted that OSA was developed and examined in a number of RACE projects (e.g. ROSA, CASSIOPEIA, DRAGON) which have ended a long time ago. Current EU research projects do not address OSA directly. On the other hand, as was explained in Section 2.5.2., TINA-C is now a mature architectural framework with significant momentum, which incorporates most of the important findings of OSA. Therefore, taking into account the ongoing TINA-C related activity and the similarities and compatibility between OSA and TINA-C, the architectural framework of TINA-C is selected as the most suitable service support environment for the proposed telecommunications service engineering framework of Figure 1.1. Regarding the two main constituent parts of the service support environment (see also Section 1.4.), the service engineering principles are specified by the TINA-C service architecture and the service execution environment (which is examined in Chapter 3) is influenced by the TINA-C approach pertaining the DPE.

TINA-C enables the service support environment (depicted in Figure 1.1.) to satisfy Requirements 2, 3, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, and 16 (see Section 1.3.). More specifically, as can be deduced by the careful examination of the official TINA-C documents, the related literature and Section 2.5.2., TINA-C provides an architectural framework that exhibits the following distinctive features:

- Combination of a collection of new computing technologies and application of them coherently to a complete set of architectural principles for telematic services. Thus, it satisfies Requirements 2 and 12.
- Separation of service delivery from network transport in order to facilitate the rapid introduction of services. Thus, it satisfies Requirements 3 and 15.
- Support of new services with a special focus on services with multi-party and multimedia features. Thus, it satisfies Requirement 5.
- Inherent mobility support and capability to quickly incorporate advanced communication concepts. Thus, it satisfies Requirement 6.
- Flexibility against regulations by defining reference points according to market driven business roles. Thus, it satisfies Requirements 5 and 6.
- Openness, with the consequence of supporting interfaces that have clearly defined and accurate semantics. Thus, it satisfies Requirements 8, 11, 12, 13, 14, and 15.
- Reusability of service specifications and service software. Thus, it satisfies Requirement 9.
- Uniform support of any kind of management functionality, which is viewed simply as another kind of service. Thus, it satisfies Requirement 10.
- Support for the distribution of the service logic among several service provider domains. Thus, it satisfies Requirements 11 and 13.
- Network technology independence (evolvability). Thus, it satisfies Requirement 15.
- Support for gradual evolution and co-existence of the new (service, system) with the present. Thus, it satisfies Requirement 16.
Chapter 2: Advanced Approaches in Telecommunications Service Engineering

The choice of TINA-C as the service support environment of Figure 1.1. was based on its maturity, the commitment for its ongoing development and improvement, the gradual appearance of TINA-C products, and the undeniable advantages of its architectural framework. However, in order to fully justify such a choice the perspectives of TINA-C to significantly penetrate the telecommunications environment should be considered in a realistic manner, taking into account the (technical and economic) implications that are created by the proliferation and strong presence of other (widely accepted) related technologies (IN, TMN, the Internet, mobile communications).

Therefore, smooth migration and interworking scenarios have to be devised and carefully examined. Whereas migration is related to the gradual introduction of TINA-C concepts in current telecommunications networks, interworking is related to the cooperation between TINA-C conformant and conventional systems by means of particular adaptation or interworking units. Migration has to be considered because there are a lot of existing legacy systems in the telecommunications industry for which considerable investments have been made by the telecommunications network operators. Consequently, the success of TINA-C depends largely on well investigated migration scenarios which are based on slowly expanding islands of TINA-C conformity. The result of migration is a full TINA-C compliant network. However, as long as this goal is not met, interworking between legacy and TINA-C systems (at different levels of abstraction) will be necessary. Since TINA-C is a solution for the middle and long term, interworking is expected to be required for a long time [ELTR97a][REYN98][VENI98][VENI00].

TINA-C has much in common with IN, in both its principles and its operational concepts, with regard to call control and service creation [MAGE96][VENI00]. Furthermore, TINA-C shares with TMN a distributed style of management and has inherited and adopted TMN concepts in its network resource view and connection management [GRIF97][TIN97d]. As CORBA-based TMN systems [AIDA98][PAVL97] are expected to be dominant in the future, TINA-C and TMN will share a common design more and more. Hence, in both contexts, TINA-C can be considered as an adequate evolution.

Scenarios detailing the migration from current networks to TINA-C, as well as the interworking of those technologies, have been fairly extensively studied [BROW94][EUR96a][EUR98a][EUR99a][HAUW98][HERZ97][REYN98]. The overall conclusion of these considerations is that the migration towards TINA-C from IN and TMN is indeed possible and feasible, even if it may not be an easy task. The basic strategy they suggest to follow is to utilise existing components as long as they can sufficiently substitute for the generic TINA-C components. Therefore, the optimum evolution path follows a step by step approach while capitalising as much as possible (and for as long as possible) on already deployed equipment and software [MAGE96][VENI98]. This will avoid network instability problems, revenue loss, and customer complaints.

As was mentioned in Section 1.1., the exponential growth of the Internet and its proven scalability, positions the Internet Protocol (IP) in the centre of any future (global) communications network. Due to this...
fact, much of the transport technologies envisioned by TINA-C, namely the TN and some portion of the kTN, is expected to be predominantly IP-based, in the not so distant future. Therefore, if TINA-C is to be widely adopted in the near to medium term it must be able to accommodate current and evolving protocol based communication control mechanisms, and in particular (QoS-aware) IP networks [LEW198]. This necessity is reinforced when considering that the interworking between the PSTN and the Internet is already being attempted through the WebIN architecture [LOW97] that fits into the PINT reference model [KRIS97] and that the Internet is emerging as a competing medium transport infrastructure to the legacy PSTN through the introduction of IP telephony [HASS00].

The Internet is quite distant from the TINA-C architecture, because in TINA-C the network model is fundamentally connection-oriented, as opposed to the best-effort, connectionless datagram model of IP protocols. On the one hand, Internet is a threat to TINA-C, because broadband networks are not spreading as fast as was forecast, and interactive multimedia services are provided by the Internet in an easy and cost effective way, regardless of the unpredictable performance degradation (in terms of QoS and service availability). This fact will become less important considering the new IPv6 protocol and the Integrated Services architecture with the RSVP signalling protocol [RUTK99] which will guarantee a certain QoS. On the other hand, TINA-C could learn from the Internet experience because many Internet technologies could be used to support TINA concepts (e.g. the use of Java for building distributed applications, or the concept of the network computer) [VENI98].

In general, there is evidence that a synergy between TINA-C and the Internet is feasible [CANA99] [DEZE97][GEIH98][LEW198][LICC97][SMITT97]. More specifically, it seems possible that despite their fundamental architectural differences, certain parts of TINA-C can be applied successfully in the Internet in the near term and add control and management value on top of the Internet. In particular, TINA-C access and service session concepts, together with other features of the TINA-C service architecture, can be the enabling factor for integrating IP-based services with traditional telecommunications services [HUBA99].

Finally, because of the increasing demand for universal connectivity, mobile communications are becoming fundamental attributes of future telecommunication systems. With the promising business opportunities of universal personal communications services, it has been recognised that mobility functions should be fully supported by programmable intelligent service infrastructures like TINA-C [DAOU98][TRIG98].

This would require a modification of the TINA-C architecture [TIRO98]. Such an effort has already been undertaken by several EU research projects. Among them, DOLMEN provided a number of TINA-C extensions in the area of mobility, validated these extensions through an international field trial, and highlighted a number of important open issues that will be the target of further research in this area [TRIG98]. For this reason, it is envisaged that TINA-C will provide universal programmable connectivity between mobile users supporting a variety of mobile services (cordless, cellular, paging, mobile satellite) in
the framework of the Universal Broadband Mobile Telecommunication System (UBMTS) that will emerge as the result of the activities pertaining the fourth generation of mobile systems [DAOU98]. In this way fixed and mobile networks will eventually converge enabling the realisation of a Virtual Home Environment (VHE) [TORA98]. In this environment, the clear distinction between fixed and mobile networks, as well as services, is fading out through increased similarities of network functions in both network types, representing an opportunity to provide significant user benefits in the performance and availability of services without regard to how the service is accessed [HARR99].

Because of the flexibility of TINA-C in adapting and incorporating new and existing approaches and technologies, and because of its wide scope, the choice of TINA-C as the service support environment of Figure 1.1. is fully justified. Furthermore, TINA-C can be considered as a valuable complement to GII activities. It provides an architectural underpinning for the GII, as it is directed to the provisioning of any kind of services, running on a global scale, on different network technologies, allowing the combination of any kind of media and any kind of connectivity, and facilitating third-party connection set-up and broadcasting as well as multiparty involvement.

2.6. Service Creation Environments.

The architectural framework of TINA-C, that was selected as the service support environment of the proposed telecommunications service engineering framework of Figure 1.1., provides a rich set of concepts, guidelines, and practices for the creation of telecommunications services, especially through its service architecture (see Section 2.5.2.). However, the application of these service engineering principles for the development of specific telematic services is not an easy task, as TINA-C does not provide any additional support for this purpose in the form of a suitable methodology or a Service Creation Environment (SCE).

For this reason, the proposed telecommunications service engineering framework, contains both a service development methodology and a SCE, as two of its main constituent parts (see also Section 1.4.) recognising their importance when performing service engineering activities. The rest of this section will focus on the SCE, attempting to identify its main characteristics, examine important related approaches, highlight its relation with the service development methodology (which is examined in Chapter 4), and reason about its role and purpose in the proposed telecommunications service engineering framework.

The importance of SCEs, combined with an appropriate service development methodology, is constantly increasing because, in the deregulated and highly competitive telecommunications market, the success of service providers is determined by the efficiency with which services are developed [DEME99][MUDH96][POLY98]. Therefore, there is a need to capture accurately the customer demands, and create and introduce rapidly the pertinent services, in a way that they retain certain desired quality characteristics despite their sophistication [CONC97]. A SCE should support the service developer in addressing these requirements.
Despite their significance, SCEs are not perceived in the same manner by all service developers. The term SCE is rather abstract and is defined / interpreted in various ways by different people [PONT94]. However, what is undeniable (by simply considering its name) is that service creation activities fall into the scope of a SCE [KOSM97]. Service creation is the first stage of the service life-cycle (see Figure 4.1.) [BERN94], refers to the transformation of the often vague and imprecise requirements for a new service into code that implements the required service, and includes all the activities that are necessary in order to create a new service, either from other existing services / components or starting anew [CAPE95][MUDH94].

In this context, and taking into account that the service creation process should be supported by a suitable methodology, a SCE is considered as a logical framework incorporating a collection of appropriate, carefully designed and tested, customisable, and user-friendly software tools (together with a reuse infrastructure) that are used according to the service development methodology with the aim to assist the service developer(s) when applying the methodology by automating and simplifying as much as possible the service creation process, and by facilitating consistency and verification checks. This definition of a SCE is used throughout this thesis and emphasises that a SCE is actually a development environment aiming to support (in the most general case) teams of service developers working towards a common goal (the development of a specific telematic service) using shared information and a number of software tools.

According to this definition, a SCE is in close cooperation with the service development methodology. Additionally, the SCE definition reveals that the two most important constituent parts of a SCE are:

- A set of software tools ranging from GUI-driven rapid prototyping tools and high level (specification) languages for the design of the service logic to powerful code generators for translating the high level design into the corresponding service code. For this purpose, various existing software technologies and tools can be used, depending on their commercial availability, their compatibility, and their ability to interact in an integrated manner [DEME98][DEME99][RAC95c].

- A reuse infrastructure supporting reuse at different levels of abstraction using appropriate notation and modelling constructs, and possibly code written in a certain programming language. This infrastructure includes mainly a service component library that provides reusable service components, which can be used for the construction of telematic services (see Section 4.10.2.) [POLY98][RAC95b]. For structuring a service component library and starting to developing it, the information and computational modelling concepts and guidelines of an architectural framework for services (like OSA and TINA-C) should be considered.

The value of a SCE is increased as it is used for supporting the development of a variety of services. During this process, needs regarding software tools are identified and possibly satisfied, the service component library is enriched with new service components, and service developers gain important related experience [PONT94]. The result is a more advanced SCE with improved effectiveness. However, the success of a SCE depends also on more abstract factors. More specifically, since a SCE is used by service
developers working in an organisation / enterprise (e.g. a service provider), it is influenced by the culture and general philosophy of the organisation regarding service development and the related technologies. It is also subject to the limitations of the organisation in terms of tool and personnel support [KOSM97].

Although the definition of a SCE is relatively simple and clear, constructing an effective SCE is a challenging task. For this reason, considerable research effort is being devoted to the development of SCEs and the examination of their structure [ACT97a][ACT99a][DEME98][DEME99][EUR94a][POLY98][RAC95d][RACE96]. The main related attempts will be briefly presented in a critical manner in the following paragraphs, in order to facilitate the reasoning about the purpose and the structure of the SCE included in the proposed telecommunications service engineering framework of Figure 1.1.

SCEs initially appeared in conjunction with the technology of IN. Legacy IN systems usually encompassed powerful SCEs, where the service designer had only to select icons (representing predefined primitive components that rarely corresponded to standardised SIBs), interconnect them to each other, and fill-in some dialogue-based forms to parameterise them. This procedure was an easy one and the work of the service designer was simplified. However, these systems were closed, not able to seamlessly embed user-written code in order to enhance the functionality of the services. Hence, the creation of more complex services was tricky and demanding [VENI00]. With the progress of IN standardisation this problem was largely overcome and SCEs were introduced enabling the quick customisation or creation of IN services out of predefined open components (SIBs) [CONC97]. Thus, the existing commercially available SCEs speed up the design of the IN service logic (even in the case of complex services), but the whole process is still relatively long. This is mainly due to the very limited support for specification, validation, and testing (especially early in the development process) provided by the current SCEs. Therefore, there is a need for adequate support by appropriate software tools [KIM99b].

The functionality and role of SCEs regarding the technology of IN was significantly extended by the Eurescom project P103 “Evolution of the Intelligent Network” [EUR94a][MUDH96]. The SCE that was developed by this project [EUR94c] was accompanied by a service creation process [EUR94b], which combined the Object Oriented Role Analysis Method (OORAM) [REEN92], Message Sequence Charts (MSCs) [ITU92c], and the Specification and Description Language (SDL) [ITU95f]. It contained several role models 19 (which are units of modularity in OORAM) for the various service creation activities, without restricting the way they may be combined, and a “service constituent storage”, storing service constituents at various abstraction levels for reuse purposes and allowing their easy retrieval and maintenance [CAPE95][MUDH94]. Rather than defining a new storage model, it has been decided to adapt an existing model developed within the RACE project SCORE [RAC95b] (see below).

---

19 A role model is made up of a set of interacting objects [MUDH96].
In parallel with project P103, research regarding SCEs was also taken place in the context of the OSA architectural framework by the RACE project SCORE (Service Creation in an Object-oriented Reuse Environment) \(^{20}\). SCORE developed a service creation process model, which is generic, as it specifies roles, activities and associated tools, but it does not specify a particular process or temporal interconnection [RAC94a]. This is left to be done by the enterprise developing and deploying services according to its specific needs [TRIG95]. SCORE adopts a very broad view of what an SCE is [RAC95d], and after identifying the most important requirements that should be met by SCEs [HALL94], considers the service creation process model as a generic SCE [PONT94]. Thus, a particular SCE (dedicated SCE) is an instantiation of the service creation process model, in which the particular tools and methods appropriate to the type of service being created, and to the enterprise doing the creation, are brought together \(^{21}\) [PONT94][RACE96]. Furthermore, SCORE developed a component model, which can be used as a standard template for a repository of reusable service components designed according to possibly different architectural approaches and targeted to possibly different service execution environments [RAC95b].

Finally, most recently, SCEs were examined in alignment with the architectural framework of TINA-C by the ACTS projects SCREEN (Service CREation Engineering ENvironment) [ACT99a] and TOSCA (TINA Open Service Creation Architecture) [ACT97a]. More specifically, the main objective of SCREEN was to consolidate and extend existing and emerging technologies in order to produce a seamless tool-supported approach to component based service creation [POLY98]. Therefore, it developed an open, multi-vendor SCE, which consists of service logic development tools that provide the necessary means (software technologies) for the design, specification and implementation of the service logic, a service component library that promotes and facilitates the reuse of service components, and QoS related facilities that enable cost-effective service provision at the appropriate quality levels. The operation of this SCE is supported by a service creation practice structured according to a number of phases (requirements gathering and analysis, service analysis, service design, validation and simulation, and DPE targeting) [ACT98a][DEME99].

TOSCA was primarily focused on the development of a SCE capable of supporting the rapid (automatic) creation and validation of telecommunications services in an effective manner. The TOSCA approach assumes that for a certain class of service, a flexible and reliable object-oriented software framework based on TINA-C principles is developed [LODG99]. A framework is a set of service components that can be used to build a large number of standard and customised services (see also Section 5.3.). It may be specialised through inheritance and / or delegation to acquire service-specific behaviour and represent the logic of a particular service. Framework specialisation is carried out through the use of a paradigm tool. This is a graphical CASE (Computer-Aided Software Engineering) tool that allows the service designers to specify

\(^{20}\) SCORE had a strong relation with the RACE project BOOST (Broadband Object-Oriented Service Technology) [RACE96], and thus part of the SCE research took place in both these projects.

\(^{21}\) The mechanism chosen for the integration of particular sets of tools was Tcl/Tk [RAC94b].
Chapter 2: Advanced Approaches in Telecommunications Service Engineering

the behaviour of a service at an abstract level, and then automatically creates the new service by
appropriately customising a framework with the specific service behaviour [KIMB99]. Thus, the TOSCA
SCE consists of a framework and a corresponding paradigm tool supplemented with service validation tools
[ACT97a][SINN98]. Recognising the fact that the development of frameworks is a considerably more
complex task than the development of services, TOSCA has also developed a process architecture for the
development and use of frameworks in service creation [ACT98b][LODG99].

The SCREEN and TOSCA approaches to SCEs are complementary [ACT98c][LODG99]. Taking into
account that the resources required to develop a framework are far greater than the resources required to
create one of the services in the class covered by the framework, the SCREEN approach should be used for
the initial development of new types of services. Then, if there is a high confidence level that there will be a
need for similar services at some point in the future, an object-oriented framework should be developed 22
(using components from the original service) to facilitate the creation of a class of similar services using the
TOSCA approach.

A detailed comparison between different existing SCEs that are applicable to TINA-C is not attempted,
because it is argued that every SCE can be sufficient and effective in a certain service development context.
Thus, the importance of supporting efficiently the entire service creation process is highlighted, because the
characteristics of a specific service development methodology determine the kind of support that the service
developer(s) expect from an SCE and thus (should) affect significantly its structure and its constituent parts.
In this context, the success of a SCE is measured against its ability to facilitate the service developer(s)
when applying each of the phases of a particular service development methodology, and is determined by the
quality and the effectiveness of the corresponding methodology.

The construction and use of an SCE, without having a specific service development methodology in mind,
is not suggested, because it cannot offer any kind of guarantee regarding the result of the service creation
process. Only an experienced service developer will manage to design and implement a successful telematic
service (i.e. according to user requirements), without any methodological support and only the use of a SCE
and some abstract guidelines. Furthermore, his / her attempt will require an undefined and impossible to
predict amount of time and will involve a significant amount of ad hoc decisions on a variety of matters,
preventing in that way any kind of project management activity (including the possible cooperation with
other developers) and seriously jeopardising the maintainability of the service. Finally, there is always the
danger of a service developer who, mislead by the capabilities of the (software tools of the) SCE, proceeds to
activities which either destroys his / her focus on the real service development problems, or directs his / her
interest away from considering the real user requirements, with obvious negative effects on both the service
creation process and on the resultant telematic service.

22 Possibly using a variant of the SCREEN service creation practice [KIMB99].
An intermediate approach is followed by the SCREEN project, which recognises the importance of specifying a service creation methodology when developing and using an SCE. Thus, SCREEN provides (a kind of) a methodology addressing service creation matters. However, this methodology is abstract. It has a rather obvious structure, which is not elaborated, and acts more like a loosely-coupled framework for fitting in several concepts, practices and technologies. Thus, SCREEN fails to provide sufficient methodological support regarding the service creation process (see also Section 4.2.) and thus the use of the SCE that it prescribes does not deliver the anticipated benefits for reasons already explained previously.

The same remark applies also to the TOSCA approach, which although it has a much wider scope than SCREEN, it does not specify a service development methodology. Furthermore, TOSCA underestimates the need for such a methodology by adopting a paradigm based approach to service creation and a navigation and parameterisation approach to service customisation [ACT97a]. However, composing telecommunications services from reusable service components without the support of an appropriate service creation methodology has important drawbacks (see also Section 4.2.). Additionally, in the case of the TOSCA paradigm based approach, each TOSCA SCE only allows the service developer to create services within some specific set of possible services (the class of services covered by the framework). This set may be large but it is inevitably restricted by both the functionality of the framework and the nature of the paradigm chosen. Similar limitations exist also in the case of navigation and parameterisation, as the service type required must be already available if the service developer is to be able to select it. Therefore, for the creation of any arbitrary service a process of software development in the form of a service creation methodology must be followed.

In order to avoid the possible negative implications of using an SCE without adequate methodological support, one of the main objectives of this thesis is to propose and present a service development methodology (see Section 1.4. and Chapter 4). This methodology has a central position and role in the proposed telecommunications engineering framework of Figure 1.1., and specifies in a systematic and structured manner the entire service creation process. Therefore, the purpose of the SCE in this framework is to increase the applicability of the service development methodology, by efficiently facilitating the service developer(s), and not to substitute it. For this reason, the SCE is proposed to be constructed (or customised) by the service developer(s), prior to the start of the service development project, according to:

- The exact characteristics and nature of the service development methodology that they plan to use.
- Their experience regarding the use of appropriate software tools.
- The requirements (the "design culture") of the organisation / enterprise that they work for.
- The requirements of the organisation / enterprise that will use (and possibly maintain) the telematic service that will be developed.
- Any special requirements that the telematic service under development may impose.
In this manner, the maximum flexibility is achieved and the service creation process is facilitated in the best possible way by the SCE, satisfying Requirement 4 (see Section 1.3.). It is evident that this approach can incorporate (after appropriate customisation) the main findings regarding SCEs of both SCREEN and TOSCA, as long as they do not contradict the service development methodology that is used. This remark applies especially to the reuse infrastructure which is less affected by the service development methodology.

2.7. Summary and Conclusions.

In the traditional Public Switched Telephone Network (PSTN) the only service offered initially was the classical, two-way Plain Old Telephone Service (POTS). To offer this service, the telecommunication network has to allocate and reserve resources across voice trunks passing through a number of switches. First implementations were entirely hardware based, where electric pulses steered electromechanical switches to select the appropriate outgoing circuit. It was not until these systems were replaced by programmable computers and electrical pulses by "proper" signalling protocols that it became feasible to design services in a more flexible manner.

Although it did not fundamentally change the way services were provided, nor did it enable the provision of radically new services, the IN concept (see also Section 2.4.1.) represented a significant evolutionary step in that respect since it made it possible to introduce more intelligence in selected network nodes without affecting the majority of switching systems. The intelligence required for the provision of a service was now placed in dedicated IN servers (SCPs) instead of every switch of the network. The switch functionality was restricted to basic call processing, and additionally, to the identification of IN service calls and to the routing of these calls to the IN servers. In this way, variations of the basic phone service can be easily implemented.

However, even IN-augmented PSTN, fails to meet the demand for a great diversity of broadband, multimedia, and multi-party telecommunications services with enriched functionality. Such services require a more flexible service provisioning approach, in which flexibility should be pursued even if it is not advantageous from a performance point of view, and in which service - transport network dependencies are eradicated. As will be explained in Chapter 3, DPEs can satisfy this need by permitting very complex signalling interactions, abstracting over the use of low level signalling protocols without semantic loss. Under these conditions telematic services can be considered as complex distributed software applications designed and implemented as a set of interacting service components operating upon a DPE. This complexity together with the generic character of modern networks, which are not bound to a specific "service paradigm", highlight the necessity for a service creation methodology that will bridge the gap between the initial conception of a service and its actual implementation. Recognising this need, this chapter examines a number of advanced approaches in the area of service engineering that are related to the research work presented in this thesis.

In order to fully understand the objectives, the driving forces and the scope of telecommunications service engineering, and to improve the semantic coherence of the entire thesis, the starting point of this chapter is the definition of some important terms / concepts (Section 2.2.). Important issues associated with the object-
oriented development of telematic services are considered next by this chapter (Section 2.3.). The emphasis of this chapter on telecommunications software continues with Section 2.4. In this section the main characteristics of IN and the TMN, in the framework of the constantly expanding MSNs, are presented and their co-existence and integration is examined. Then, the need for an open universal service platform capable of supporting the control and management of telecommunications services in an integrated manner is highlighted.

The two most important architectural frameworks that promise to satisfy this need and make the future open information market a reality are OSA and TINA-C, which are presented in Section 2.5. In the same section, OSA and TINA-C are also compared and contrasted considering mainly their service architectures. Between them, TINA-C appears to be more robust, more complete and more promising for the future, as various problems, weaknesses, and pitfalls will be tackled by the ongoing efforts of the TINA consortium. Therefore, TINA-C is selected as the most suitable service support environment for the proposed telecommunications service engineering framework of Figure 1.1. Section 2.5., after reasoning about this choice, identifies a number of distinctive features in TINA-C that satisfy Requirements 2, 3, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, and 16 (see Section 1.3.), and examines the future perspectives of TINA-C by considering its relation with INs, the TMN, the Internet and mobile communications.

In general, the discussion regarding TINA-C in Section 2.5. reveals that TINA-C is already a mature architectural framework which is gaining acceptance, through validation activities and the examination of interworking concepts and migration strategies from existing technologies. Furthermore, TINA-C is gradually acquiring industrial strength and the related technology moves from the research field into the area of industrial application. The appearance of related products exposes TINA-C specifications to a wide and public process of industry interaction and the feedback that is generated in that way, will either back up the consistency of the specifications or will determine the need for alternative specifications. Up to now, the feasibility of the TINA-C approach has been proved in many different contexts and its main concepts have been challenged in many different ways. Therefore, there is evidence that TINA-C is a very interesting and promising solution for the middle and long term, and is not considered any more as the ultimate vision, but as an essential evolutionary step towards the optimum (re)structuring of the emerging MSNs [VEN198].

However, there is still limited practical experience regarding the performance stability and scalability of TINA-C in the time-critical network control area. Additionally, although TINA-C is flexible in adapting new and existing approaches and technologies, the tumultuous development of the Internet is likely to allow only certain parts of TINA-C to be implemented and deployed widely. In any case, this possibility does not limit the importance and value of TINA-C, as TINA-C is not an "all-or-nothing" approach and the principles and concepts of TINA-C will continue to be very valuable as a conceptual guideline in any context, irrespective of the penetration of the complete TINA-C architectural framework [GEIH98]. This belief is also reflected in the proposed methodology which considers TINA-C in a critical manner (see Section 4.4.).

In Section 2.6., the difficulty of applying TINA-C for the development of telematic services is identified and a SCE together with a service development methodology are considered as a possible solution,

It is "a cooperative solution for a competitive world" as is officially characterised [INOU99].
explaining thus the rationale for proposing the telecommunications service engineering framework of Figure 1.1. The rest of this section focuses on the SCE and it is argued that in the proposed framework the SCE is in close cooperation with the service creation methodology and aims to increase its applicability and not to substitute it. Therefore, in order to satisfy Requirement 4 (see Section 1.3.), the SCE is proposed to be constructed / customised by the service developer(s), prior to the start of a specific service development project.

Although this chapter is mainly an introductory chapter, regarding the main objectives of this thesis (see Section 1.4.), that contains state of the art descriptions, some parts of it can be considered as a research contribution, because the related matters were addressed in a critical manner, trying to clarify and structure a number of issues, and always explaining the reasons behind the various decisions and proposals. Therefore, the research contributions in this chapter, which are among the secondary objectives of this thesis (see Section 1.4.), are the following [ADA98c][ADA98d][ADA99d][ADA00c][ADA00f]:

- The definition of the term / concept “telematic service” and the discussion about similar definition attempts found in the service engineering literature (Section 2.2.).
- The justification of the object-oriented design of telematic services by identifying the main requirements associated with the design of telematic services and examining ways that object orientation can support them (Section 2.3.2.).
- The comparison of a number of mature and well documented general purpose object-oriented software development methodologies and the consideration of their suitability for the development of telematic services (Section 2.3.3.).
- The comparison and contradistinction of the architectural frameworks of OSA and TINA-C according (mainly) to their service architectures (Section 2.5.3.).
- The discussion on the (future) perspectives of TINA-C to significantly penetrate the telecommunications environment by considering its flexibility in adapting and embracing new and existing approaches and technologies (IN, TMN, the Internet, mobile communications) (Section 2.5.4.).
- The identification of the main constituent parts of a SCE and the discussion about its role and purpose in relation with a service development methodology by previously considering other important related approaches (Section 2.6.).

This chapter examines the SCE and the service support environment of the proposed telecommunications service engineering framework of Figure 1.1. However, as was mentioned in Section 1.4., the service support environment has two main constituent parts. First, the service engineering principles are specified by the TINA-C service architecture. And second, the service execution environment, which is the other constituent part of the service support environment, is only influenced by the TINA-C approach regarding the DPE, as TINA-C does not suggest a particular DPE. Such a choice, which is essential for the distributed realisation of telecommunications services, is made in the next chapter (Chapter 3), which examines in detail important issues and technologies regarding DPEs and therefore regarding the service execution environment of the proposed telecommunications service engineering framework.
Chapter 3:

Distributed Computing Platforms
in Telematics

3.1. Introduction.

There are many driving forces which have compelled telecommunications operators and vendors to seek new solutions in telecommunications service engineering. Among them, growing competition and the progressive convergence of information and telecommunications technologies led to an increased focus on how a great variety of advanced multimedia telematic services with enhanced functionality can be efficiently and effectively developed and deployed in shorter time frames taking advantage of different network technologies, end-systems, communications protocols, operating systems, and programming language environments.

Distributed object platforms have been recognised as a key technology solution to this problem, because, due to recent developments in object orientation and distributed computing, they promise the benefits of more flexible service design and service deployment, increased software reuse, and increased interconnection capabilities with external resources such as the Internet and private database systems [ADA99c]. Furthermore, as services evolved, it became necessary for the user to communicate more complex information to the network. Accordingly, adjacent nodes in a network needed to exchange more information. Signalling is used for both these purposes. Signalling was initially in charge of the bearer connection control. In its early steps it had rudimentary capabilities, which were adequate for the services for which it was at that time used. Increasing service complexity and the increased needs imposed by more sophisticated service control has since then sustained a continuous signalling evolution; an evolution which was fostered by advances in software engineering practices. As signalling protocols become more complex, they can be implemented by distributed objects in order to facilitate easier implementation and maintenance, and software reuse.

Therefore, distributed object platforms, which are actually object-oriented Distributed Processing Environments (DPEs), promise to revolutionise the service control area by providing a new approach to advanced signalling (see also Section 7.2.). They are used for both the transmission of service control information (in the form of remote method invocations) and the multimedia streams that may flow between
user applications, and function as a unifying abstraction layer, presenting a homogeneous view of the transport capacity to telematic services, which are realised by a set of interacting service objects that are physically distributed across different network elements [ADA00f].

Taking into account the growing significance of distributed computing platforms in the development, construction, and management of telecommunications services, the proposed telecommunications service engineering framework of Figure 1.1. considers the DPE as the most important part of the service execution environment (see Section 1.4.). The other ancillary software that is included in this environment is usually used through the DPE and has been extensively studied, especially regarding distributed operating systems [SINH97] and database management systems [GOEB95][GRA95a]. For this reason, this chapter focuses on DPEs and examines in considerable detail and in a critical manner important issues and technologies regarding DPEs, and therefore regarding the service execution environment of the proposed telecommunications service engineering framework. The overall objective of the chapter is to justify the central role of DPEs in the service execution environment, explain the way that they satisfy Requirement 3 (see Section 1.3.), and select a specific DPE technology that will be used in combination with the service development methodology that will be proposed and presented in Chapter 4.

After this initial section, Section 3.2. examines the most important de jure and de facto DPEs from the perspective of telecommunications service engineering. More specifically, after a brief introduction to open distributed processing environments, that emphasise their goals and main characteristics, Sections 3.2.1., 3.2.2., 3.2.3., 3.2.4., and 3.2.5. present respectively a concise state of the art description of ISO’s / ITU-T’s Reference Model for Open Distributed Processing (RM-ODP), Open Group’s Distributed Computing Environment (DCE), OMG’s Common Object Request Broker Architecture (CORBA), Microsoft’s (Distributed) Component Object Model (COM / DCOM), and Sun’s Java Remote Method Invocation (RMI), highlighting the discerning characteristics of each of them. Finally, Section 3.2.6. considers the relationship between these initiatives and evaluates the corresponding DPEs by identifying their role, commenting on their perspective, and proposing and examining a potential path to convergence, which reveals that DCOM and CORBA are currently the most significant DPE technologies for service engineering activities.

This remark is further justified in Section 3.3., by identifying and explaining the most important benefits that distributed object platforms like DCOM and CORBA can offer to telecommunications services. Recognising their importance, Section 3.4. compares DCOM and CORBA with the intention to inform service developers about the expectations they should have when using these platforms, and thus assist them in a possible selection process between them. For this reason, this section proposes a decision framework, which is constructed by identifying a set of basic, core and service engineering related properties, and examining the way that DCOM and CORBA support these properties (in Sections 3.4.1. and 3.4.2. respectively). The proposed decision framework is complemented by an examination of the performance of DCOM and CORBA, and by considering their interworking with the World Wide Web (in Section 3.4.4.). The main results of this comparison attempt are summarised and briefly discussed in Section 3.4.5.
Based on the comparison results, and taking into account Section 3.5. that examines the interworking between DCOM and CORBA, Section 3.6. discusses the suitability of each one of these platforms for different service development requirements. Additionally, this section, after considering the main TINA-C requirements from a DPE, selects DCOM as the DPE that is going to be used and examined in the rest of this thesis, explaining the rationale behind this choice. Finally, Section 3.7. offers a synopsis of the main findings of this chapter, highlights the research contributions included in it, and explains its relationship with the subsequent chapters.

3.2. Important Distributed Processing Environments.

Telecommunications providers are increasingly looking towards distributed system technologies related to DPEs, in order to address the problems of the construction and subsequent management of telecommunications services. The main goal of a DPE is to enable open distributed processing, and thus the interaction between services in a geographically distributed telecommunications system without concern for the underlying environment. This level of openness requires the conformance to well defined and well documented interfaces and has the advantage of achieving interoperability \(^1\) and portability \(^2\) [BLA196] [BLA198]. Additionally, open distributed processing environments aim to enable the development of telecommunications services which are themselves open, in the sense that the individual components of the services are open. Then, the benefits of interoperability and portability characterise also all the components of the service, which can be extended by introducing new components at a later date and can be specialised or evolve by changing the implementation of individual components.

The main problem in achieving these benefits is the heterogeneity of the underlying environment (in terms of heterogeneous hardware, systems platforms, programming languages, and management policies). This problem can be solved if a level of independence is provided from the heterogeneous infrastructure. More specifically, hardware independence is achieved by providing abstractions over the characteristics of the underlying physical network and computing environment. For platform independence abstractions over the underlying systems platforms (e.g. operating systems and network protocols) are necessary and language independence implies providing abstractions over the interfaces offered by different programming languages. Finally, to overcome heterogeneous management policies, it is necessary to provide a meta-architecture for management which can accommodate specific approaches and policies. DPEs which offer all the above levels of abstractions are often referred to as middleware [CAMP99].

Another important goal of DPEs is to provide distribution transparency, either full or partial / selective, by masking out problems occurring in the distributed environment. Therefore, a high level of distribution transparency implies that the service developer need not be aware of the distributed nature of the telematic

\(^1\) Then, an implementation of an open distributed processing platform developed by one manufacturer should be able to interwork with a different platform implemented by a (potentially) different manufacturer.

\(^2\) Then, a service running on an open distributed processing platform developed by one manufacturer should be able to be ported directly to a different platform implemented by a (potentially) different manufacturer.
service, and thus the complexity of the required distributed programming is significantly reduced. However, it must be noted that the service developer may sometimes have to be aware of the distribution of the service (e.g. in mobile environments or when the real-time behaviour of a service is crucial), and thus DPEs are preferable to support selective transparency. The most important distribution transparencies are access, location, failure, migration, persistence, relocation, replication, concurrency, and transaction transparency [BLAI98][GAY97][ISO95].

The following sections examine the most significant existing de jure and de facto DPEs, focusing on their essential characteristics, with the intention to evaluate them afterwards from the perspective of telecommunications service engineering [ADA98b].

3.2.1. ISO's / ITU-T's RM-ODP.

The Reference Model for Open Distributed Processing (RM-ODP) is a joint standardisation activity by both ISO and ITU-T. The aim of the RM-ODP is “the development of standards that allow the benefits of distribution of information processing services to be realised in an environment of heterogeneous resources and multiple organisational domains” [ISO95]. However, it must be noted that the RM-ODP does not itself prescribe particular standards for open distributed processing. Rather, it provides a framework (in terms of architectural concepts and terminology) to enable specific standards to emerge. RM-ODP should thus be considered a meta-standard for open distributed processing.

Figure 3.1.: The RM-ODP viewpoints.

Because of the large amount of information required to produce a complete specification of a distributed system, the RM-ODP introduces the concept of viewpoints, which are abstractions used to partition the specification of a distributed system into a number of different parts. Each viewpoint is a complete and self-contained description of the required distributed system targeted towards a particular audience [GEIH93]. As can be seen in Figure 3.1., the RM-ODP defines five viewpoints; namely the enterprise, information, computational, engineering, and technology viewpoints. Each viewpoint has a corresponding language. However, these languages do not prescribe a full syntax and semantics for specifying the relevant concerns. Rather, they provide the basic terminology required to model the concerns of a given viewpoint. In this sense, the viewpoint languages can be considered as meta-languages which can be instantiated with particular notations in a given usage [GAY97].
Chapter 3: Distributed Computing Platforms in Telematics

Table 3.1.: Summary of RM-ODP viewpoints and languages.

<table>
<thead>
<tr>
<th>Viewpoint</th>
<th>Areas of Concern</th>
<th>Concerned Audience</th>
<th>Language Concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise</td>
<td>System objectives, business concerns, policy constraints</td>
<td>Corporate managers, policy makers</td>
<td>Contracts, agents, artifacts, roles, etc.</td>
</tr>
<tr>
<td>Information</td>
<td>Information models, information flows and associated processes</td>
<td>Information managers, information engineers, business &amp; system analysts</td>
<td>Objects, composite objects, schemata (invariant, static, dynamic), etc.</td>
</tr>
<tr>
<td>Computational</td>
<td>Logical partitioning of distributed applications</td>
<td>Application developers, system designers</td>
<td>Objects, interfaces, binding (operational, stream, signal), environmental contract, etc.</td>
</tr>
<tr>
<td>Engineering</td>
<td>Distributed infrastructure to support applications</td>
<td>Operating system or communications designers, system programmers</td>
<td>Stubs, binders, protocol objects, nodes, capsules, clusters, etc.</td>
</tr>
<tr>
<td>Technology</td>
<td>Technology procurement and installation</td>
<td>System procurers, system maintainers / administrators, system managers</td>
<td>Implementation eXtra Information for Testing (IXIT), configuration diagrams, etc.</td>
</tr>
</tbody>
</table>

The RM-ODP viewpoints should not be seen as architecture layers or levels of refinement. The order of these viewpoints does not imply any particular precedence relationship. It reflects to some degree the stages of a typical design process, where the analysis of the enterprise is at the beginning, and an implementation is finally produced. A summary of the various viewpoints and languages can be seen in Table 3.1.

3.2.2. Open Group's DCE.

The Distributed Computing Environment (DCE) is a standardisation activity sponsored by the Open Group (former Open Software Foundation, OSF) [OSF92]. DCE is an operating system independent and network independent software platform aiming to achieve distributed processing. It is actually an integrated set of services (in the sense of facilities) offering collectively a DPE. Unlike RM-ODP, DCE does not follow an object-oriented approach. Instead, it has a classic client-server architecture. Thus, it does not support object-oriented features such as class / type, subclass / subtype, object instantiation, and inheritance. Implementations of DCE are available on several platforms including PCs and many UNIX platforms.

The overall architecture of DCE is illustrated in Figure 3.2. DCE services can be divided into the fundamental services (or secure core) and the data sharing services. The fundamental services are the threads service, the Remote Procedure Call (RPC) service, the time service, the naming service, and the security service. The data sharing services are the distributed file service and the associated diskless support service. It has to be noted that DCE provides a limited number of management tools for some of its services.

![Figure 3.2.: The DCE architecture.](image-url)
Distributed systems management can also make use of the Open Group's Distributed Management Environment (DME) [OSF94].

Finally, DCE uses an Interface Definition Language (IDL) for the description of interfaces, which are collections of functions logically grouped together that can be called through the RPC mechanism. The DCE IDL consists of the IDL header, which contains a Universal Unique IDentifier (UUID) and a version number, and the IDL body, which includes definitions of every operation on an interface by stating the operation name, and the name and type of the parameters and results.

3.2.3. OMG's CORBA.

The Common Object Request Broker Architecture (CORBA) [OMG98a] is supported by the Object Management Group (OMG) as part of an initiative to develop a comprehensive Object Management Architecture (OMA) for object-oriented computing [OMG93]. CORBA adopts an object-oriented approach. Its object model defines an object as an identifiable encapsulated entity which provides one or more services that can be requested by a client. CORBA objects support interfaces, which are described in terms of an IDL.

CORBA has a special component called the Object Request Broker (ORB) that is responsible for making object distribution transparent and providing a mechanism for trading, enabling object requests to be carried out in a heterogeneous distributed environment. Clients can issue requests on object implementations (object servers) and the ORB will undertake finding the objects, sending each request to the appropriate object, preparing this object to receive and process the request, and return the results back to the client. Therefore, the ORB implements a level of distribution transparency (in fact access and location transparency). It also provides persistent storage of information concerning objects using the interface repository and the implementation repository [ORFA97]. The ORB is an object in CORBA and is thus defined in terms of IDL.

![Figure 3.3.: The architecture of an object request broker.](image_url)

As can be seen in Figure 3.3., an ORB offers a number of different interfaces to the object developer. The ORB core is the lowest level in the ORB architecture and supports the basic representation of objects and the means of communicating between objects. A client usually accesses an object implementation using the interfaces offered by the IDL stubs and the IDL skeletons (Static Interface Invocation, SII). However, object-oriented applications sometimes require the flexibility offered by the Dynamic Interface Invocation

---

3 OMG is an international non-profit consortium, founded in 1989, with over 800 members. It aims to promote the theory and practice of object technology for the development of distributed computing systems [VINO97].
A client can use this interface when an application object dynamically locates an object and hasn’t got the appropriate stub accessible through the stub interface (e.g. when browsing an object-oriented database).

An object implementation normally accesses services provided by an ORB through the object adapter interface. This interface provides access to services including the generation and interpretation of object references, method invocation, security of interactions, object activation and deactivation, mapping object references to implementations, and registration of implementations. Different styles of objects can be supported by different object adapters. However, every implementation is necessary to provide a minimum environment for the management of an object, supporting a Basic Object Adapter (BOA), which offers a core set of services to the object developer [VIVO97].

CORBA does not constrain the protocol that each vendor's ORB uses to pass messages across a network. However, compliance with the CORBA 2.0 specification can only be achieved if there is a gateway between whatever protocol an ORB uses and the Internet Inter-ORB Protocol (IIOP) [OMG96].

Besides CORBA, OMA defines also certain key object interfaces. These can be divided into the lower-level CORBA services and the higher-level CORBA facilities. CORBA services define such object interfaces as naming, event, life cycle, persistence, transaction, concurrency control, relationship, externalisation, query, licensing, property, time, security, object trader, and object collection [OMG98b][TALL98]. CORBA facilities provide horizontal and vertical application frameworks, by defining collections of facilities that processes may use through CORBA objects, such as compound documents, user interfaces and system management [ORFA97]. Major ORB vendors have CORBA ORB implementations for Windows, Unix and mainframe platforms, like Inprise through their VisiBroker product and Iona through their Orbix product.

3.2.4. Microsoft's COM/DCOM.

The Component Object Model (COM) constitutes the foundation of Microsoft’s object services and has been put forward to the Open Group for standardisation. COM is a binary standard for the creation of software components, which allows components written according to the standard to be shared between applications [MICR95]. Being the core of object creation and management, COM also acts as a foundation for further infrastructure, such as persistent intelligent names, and structured and persistent storage. On top of these are built Microsoft's Object Linking and Embedding (OLE) technology and the facilities specified by the ActiveX 4 Application Programming Interfaces (APIs) [GRIM97]. According to the COM architecture a client uses COM objects through interfaces. An object interface is defined using Microsoft's Interface Definition Language (MIDL), which is an extension of DCE's IDL.

Distributed COM (DCOM) is the distributed extension to COM that builds an Object Remote Procedure Call (ORPC) layer on top of the DCE RPC to support remote objects [BROW98]. DCOM makes COM

4 ActiveX controls are lightweight COM objects that can be “hosted” on Web sites.
objects location-independent, and adds security and multithreading to COM. It specifies facilities that enable interoperability between objects implemented in different languages, for various operating systems, running in different address spaces. Objects may interoperate within a single process, between processes on a single machine, or between processes distributed across a network. Thus, DCOM provides an efficient and effective solution for the integration of heterogeneous components in a distributed environment.

A DCOM server can create object instances of multiple object classes. Every DCOM server object can support multiple interfaces, each representing a different view or behaviour of the object. An interface consists of a set of functionally related methods. A DCOM client interacts with a DCOM object by acquiring a pointer to one of the server object’s interfaces. This interface may be the generic interface that all DCOM objects support (IUnknown) or it may be another interface that the client requested. The client invokes the server object’s exposed methods through the acquired interface pointer (using appropriate proxy and stub code) as if the server object resided in the client’s address space [RUBI99]. Beside the static method of invoking methods on objects, DCOM supports a dynamic one, using the standard IDispatch interface and object type libraries. To enable clients to find the objects that they require, all objects are registered in the local system registry. Furthermore, object classes are uniquely identified by the Microsoft GUID (Globally Unique IDentifier), which can be either a CLSID (class identifier) or an IID (interface identifier).

DCOM is primarily a specification for how objects and their clients interact and addresses basic issues, such as interface negotiation, reference counting, rules for memory allocation (between independently developed components), and error reporting. A particular design goal, which is also a strength of DCOM, is the ability to manage different versions of components. Since the COM specification is at the binary level, DCOM server components can be written in diverse programming languages, such as C++, Java, Object Pascal (Delphi), Visual Basic, and even COBOL.

As long as a platform supports COM services, DCOM can be used on that platform. DCOM is extensively supported on the Windows platform. Other companies provide DCOM implementations on other platforms. Examples are Software AG (through its EntireX product) for Unix, i.e. Linux and other flavours, Digital for the Open VMS platform, and Microsoft for the Solaris platform [THOM97].

Microsoft has recently produced a new version of COM, called COM+, which is an advanced run-time environment that unifies COM, DCOM and the infrastructure offered for component based distributed applications by the Microsoft Transaction Server (MTS), into a coherent and homogeneous component technology [PLAT99]. COM+ merges the COM and MTS programming models with few extensions and new features such as queued components, event service, in-memory database and load balancing. Furthermore, COM+ is an integrated part of Windows 2000 and the heart of Microsoft’s Windows Distributed interNet Applications (DNA) architecture, which attempts to create a seamless environment for developing multitier distributed computing applications [VOTH98].
3.2.5. Sun's Java RMI.

The term Java, except from the object-oriented programming language, characterises also the specification of the bytecode to which the source code is compiled and the Virtual Machine (VM) that interprets this bytecode. Taken together, these define an object model [CAMP99]. All the different facets of Java have been defined by JavaSoft, a business unit of Sun Microsystems Inc. It has to be noted that, although the standardisation of Java was under way through ISO's Java Study Group (SC22) and in ECMA, Sun recently (February 2000) stopped this process.

![Figure 3.4.: Traditional and Java run-time systems.](image)

The Java run-time system is shown in Figure 3.4. alongside a traditional run-time system (using the object code). The bytecode specification is designed to have safety features for network download. The VM prevents any direct access to the real machine or the operating system running on it. It hides the differences between heterogeneous operating systems (a different VM is needed for each platform) so that one version of the bytecode will run on any platform. Moreover, it is envisaged that in the future the VM will be tailored for particular hardware taking over the role of the operating system. The VM links Java programs together at run time, eliminating the need for this at compile time, as Java's object model supports dynamic binding.

Java has been written especially for use in the Internet, and in this case Java code is copied between machines in a form that allows it to be run on the machine that it is copied to. However, in a DPE the code running on a server machine is accessed by a client. In Java, this can be done by using the Remote Method Invocation (RMI) API [SRID97]. More specifically, RMI enables a client object that resides in one Java VM to invoke a method from a remote server object that resides in a different Java VM. As these VMs may actually execute on different hosts, Java RMI can be regarded as an object-oriented DPE and remote method invocations in Java can be considered as object requests between Java objects that are distributed across different Java VMs [EMME00].

To use RMI, the Java Remote interface has to be extended to include the methods that will be accessible from the remote machine. The client object locates the server object through a registry running on the server machine that holds information about the available server objects. It has to be noted that both the client and

---

5 Java is a hybrid language because it is both compiled and interpreted.

6 This is particularly likely in less modular equipment than general purpose computers, e.g. communications equipment, household appliances, etc.
the server object have to be written in Java, because RMI depends on Java serialisation, which allows objects and primitive data types to be marshalled (transmitted as a stream).

Sun has provided a series of Java classes that are CORBA-compliant, along with an IDL compiler called id1gen that creates CORBA stubs and skeletons in Java, and a light-weight ORB called Door ORB that is implemented over TCP/IP [GRIM97]. As RMI was meant to be a rather lean provision of distribution capabilities on top of the Java VM, Sun is also providing higher-level services on top of RMI. One such set of services is Jini which supports application construction for portable hosts [ARN099]. Another example is the Enterprise Javabeans which aim to provide component based development support for server components by combining Javabeans with RMI and adding facilities, such as transactions, persistence and security [MONS99].

3.2.6. Evaluation.

Comparing the DPEs that were presented in the previous sections and taking into account their use for service engineering activities, it is evident that:

- **RM-ODP** is a meta-standard defining a framework to enable the emergence of more specific object-oriented technologies for open distributed processing. Thus, it has a very valuable conceptual influence to service engineering (e.g. in the case of TINA-C, see Section 2.5.2).

- **DCE** is a client-server architecture achieving integration between a range of key facilities. The importance of DCE in the area of service engineering is currently limited because services are engineered as distributed object-oriented applications and DCE does not follow an object-oriented approach.

- **CORBA** is a specific object-oriented technology providing interoperability of objects in a heterogeneous environment by offering an architecture with associated interfaces expressed in an IDL language. Thus, CORBA can support the construction of telematic services.

- **COM / DCOM** is a specific object-oriented technology (as CORBA) facilitating object creation, interaction, and management. It is also capable to support the construction of telematic services.

- **Java RMI** is an object-oriented mechanism for the communication of distributed Java objects, which is yet rather immature as a lot of the transparencies built on the core object models of OMA and COM / DCOM are yet to be defined for Java-RMI 7. However, the significance of Java RMI in the area of service engineering is expected to raise rapidly if its standardisation is resumed and completed, and as more related experience is gained [KEIJ00].

From the above remarks it can easily be deduced that DCOM and CORBA are the only two DPE technologies that have currently the capabilities to support service engineering activities in an efficient and

---

7 It has to be noted that Java-written applications can be built within the object model provided by the OMA or COM / DCOM, as these both offer multiple language support. This adds abstractions to Java's own object model and covers some of its deficiencies. The advantage over RMI is language independence for each of the components making up a telecommunications service.
effective manner. Irrespective of this fact, the various initiatives in the field of open distributed processing can be considered as complementary with a potential path to convergence. RM-ODP is the most comprehensive of the initiatives, tackling issues such as object orientation, persistency, migration, security, transaction processing, and multimedia. Therefore, it can be used as a reference point for the development of current and future open distributed processing solutions. In contrast, DCE, CORBA, COM / DCOM, and Java RMI can be viewed as more immediate realisations of open distributed processing with DCE identifying and integrating key facilities, and CORBA and COM / DCOM offering mature object-oriented abstractions. One proposed path to convergence is to offer a CORBA or COM / DCOM interface on top of DCE services and then to evolve gradually this architecture towards a full RM-ODP architecture through the provision of additional facilities and through the definition of further viewpoint languages.

According to this proposal, irrespective of the use of DCOM or CORBA, DCE will be the low-level infrastructure upon which new object-oriented telecommunications services will be developed and provided. However, the service engineer should be aware of the fact that there are two versions of DCE, as Microsoft has taken DCE RPC and implemented it in its own operating systems [BROW98][GRIM97]. Microsoft RPC is compliant with DCE RPC, as Microsoft’s IDL compiler supports all the functionality of the DCE IDL compiler. However, Microsoft RPC has its own way of handling naming and security services.

Despite their differences, DCE and Microsoft RPC can interoperate when using mixed DCE and Microsoft client and servers. To overcome the naming problem, in order to enable Microsoft RPC client and servers to use the DCE Cell Directory naming Service (CDS), the name service interface daemon (nsid) is proposed to be used as in Figure 3.5. As DCE servers are unable to use Microsoft named pipes, the only possible transport mechanism is TCP/IP. This removes any possibility of mixed DCE and Microsoft RPC systems using authenticated RPCs.

Figure 3.5.: Using mixed DCE and Microsoft RPC:

(a) Microsoft RPC client - DCE server,  (b) DCE client - NT RPC server.

Upon the basic infrastructure provided by DCE, DCOM or CORBA can be used to complete the development of a telecommunications service in an object-oriented manner. It must be noted that COM / DCOM is based on Microsoft RPC. On the contrary, an ORB can be implemented upon either DCE or
Microsoft RPC, because CORBA is only a specification defining how clients can use objects via the ORB, leaving vendors to decide upon implementation matters.

Although the proposed convergence path is realistic and useful for guiding the future evolution of existing DPEs, many problems remain to be solved before full convergence can be achieved. Significant examples of these problems are technical differences, functional deficiencies, different level of maturity, and policy conflicts between different competing organisations and enterprises. In the meantime, DCOM and CORBA are undoubtfully the two most important object-oriented DPEs for service engineering activities. For this reason, the rest of this chapter, after highlighting their importance once more in the following section, focuses on their detailed examination always from a telecommunications service engineering perspective.

3.3. The Emergence of Distributed Object Platforms.

As the telecommunications environment is gradually changing its face towards an open market of information services, it is becoming apparent that major private and public networks are actually large distributed object systems. These systems are populated by a dispersed set of objects that can request services from one another through a communications mechanism, using interfaces defined in a consistent IDL [REDL98].

Whereas regular objects “reside” in a single program and do not even exist as separate entities once the program is compiled, distributed objects are extended objects that can reside anywhere on a network and continue to exist as physical stand alone entities while remaining accessible remotely by other objects. Robust distributed object platforms, such as DCOM and CORBA, allow objects implemented in different programming languages to communicate seamlessly via standardised messaging protocols. Such object frameworks enable higher levels of transparency and interoperability between distributed objects [EMME00].

The most important benefits offered by distributed object technology, and more specifically by distributed object platforms like DCOM and CORBA, to telecommunications services engineered in this way are the following [KRIE98][LEWA98][REDL98][SALE99]:

- **Ease of development and maintenance**: Distributed object platforms increase the self-management capability of objects by regulating the necessary inter-object communications and by providing object lifecycle services. Self-managing objects are used easily by other objects since no management burdens are imposed on the client objects. Furthermore, they rely on a robust event model that allows objects to broadcast specific messages and generate certain events. These capabilities simplify the programming of telecommunications services and also assist to provide an accurate representation of the real-world system / situation modelled. It has to be noted that distributed objects incorporating advanced self-management features are often called components. Such components follow a plug-and-play philosophy, operate across networks, take responsibility for their own resources, interact with other objects/components, and maintain
the transparency of their location and implementation. Components may contain multiple distributed or local objects, and they are often used to centralise and secure a specific operation.

- **Abstraction**: Because of the strict encapsulation that distributed objects provide, they are ideally suited for the construction of telecommunications services, especially when the separation of data is important. Furthermore, a client does not have to know the details of a service providing entity.

- **Modularity**: An object is a self-contained entity whose services can be invoked by several different clients designed separately from it. Cooperating distributed objects can form / model accurately the logic of telecommunications services because of the rich interaction semantics they can realise.

- **Reusability**: Libraries of object services and facilities are provided with or for distributed object platforms. They offer functions, such as security, object life-cycle management, naming service, notification service, and various special-purpose facilities common to a wide variety of telematic services.

- **Granularity flexibility**: Objects can be as small as required to provide the correct functionality combinations or as large and complex as required to encapsulate completely the logic of a particular telematic service segment without unwarranted reliance on other objects. Additionally, since distributed objects allow telematic services to be split up into lightweight pieces that can be executed on separate machines, less powerful machines can be used for running demanding services.

Distributed object technology is already in use in the telecommunications world [AIDA98][REDL98]. A characteristic example is the information model of the TMN [PAVL97], and an example where this influence is maximised is TINA-C (see Section 2.5.2.).

Taking into account the fact that object-oriented architectures for distributed computing have become a foundation technology for both telecommunications services and networks, it is inevitable that soon the use of DCOM and CORBA by service developers will be a common activity. Under these conditions, it is necessary that service developers have an in-depth knowledge of both DCOM and CORBA in order to be able, not only to apply each of them in an efficient manner, but also to select the most appropriate of them for a specific service engineering activity. For this reason, the following section compares and contrasts DCOM and CORBA with the objective to clarify important issues regarding their capabilities and use in the area of service engineering, improve in this way the related comprehension of the service developers, and also guide them during a possible selection process [ADA99c].

### 3.4. A Comparison Between DCOM and CORBA.

Considering the basic characteristics of DCOM and CORBA, some of which were mentioned in Sections 3.2.4. and 3.2.3. respectively, Table 3.2. represents an initial comparison attempt. This table reveals the wide scope and the richness of both platforms, and it is more concise and informative than other comparison attempts found in the literature, which are based extensively on code examples [CHUN98][GOPA98].
Chapter 3: Distributed Computing Platforms in Telematics

Basic Characteristics

<table>
<thead>
<tr>
<th></th>
<th>COM / DCOM</th>
<th>CORBA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inheritance of a base interface</td>
<td>Every object implements <strong>InUnknown</strong> Through its interface pointer(s)</td>
<td>Every interface inherits from <strong>CORBA::object</strong> Through an object reference (objref)</td>
</tr>
<tr>
<td>Unique identification of a remote server object</td>
<td>Using the concept of Interface IDs (IIDs) Using the concept of Class IDs (CLSIDs), the mapping of which is found in the registry</td>
<td>Using the interface reference By mapping a name to a reference through the naming service</td>
</tr>
<tr>
<td>Unique identification of an interface</td>
<td>Performed on the wire protocol by the Object Exporter</td>
<td>Performed on the wire protocol by the Object Adapter</td>
</tr>
<tr>
<td>Unique identification of a named implementation of a server object</td>
<td>Either explicitly performed by the server program or handled dynamically by the DCOM run-time system</td>
<td>Performed implicitly by the constructor</td>
</tr>
<tr>
<td>Reference generation of the remote server object</td>
<td>Object Remote Procedure Call (ORPC)</td>
<td>Internet Inter-ORB Protocol (IIOP)</td>
</tr>
<tr>
<td>Handling of common tasks like object registration, skeleton instantiation, etc.</td>
<td>Mainly by using <code>CreateInstance()</code> Handled by the registry Interface repository Called a proxy or stub Called a skeleton</td>
<td></td>
</tr>
<tr>
<td>Underlying remoting protocol</td>
<td>Type library Called a proxy Called a stub</td>
<td>All interface types are passed by reference. All other objects are passed by value, including highly complex data types</td>
</tr>
<tr>
<td>Activation of a server object</td>
<td>Defined in the interface at the IDL file. Depending on what the IDL specifies, parameters are passed either by value or by reference Complex types that will cross interface boundaries must be declared in the IDL</td>
<td>Complex types that will cross interface boundaries must be declared in the IDL No</td>
</tr>
<tr>
<td>Mapping of object name to its implementation</td>
<td>On the wire by a pinging mechanism which garbage collects remote object references and encapsulates them in the <strong>IDCOM::Object</strong> interface</td>
<td></td>
</tr>
<tr>
<td>Storage of type information</td>
<td>Any platform as long as there is a COM service implementation for that platform</td>
<td>Any platform as long as there is a CORBA ORB implementation for that platform</td>
</tr>
<tr>
<td>Definition of parameters passed between the client and server objects</td>
<td>Since the specification is at the binary level, diverse programming languages can be used</td>
<td>Since it is just a specification, diverse programming languages can be used, as long as there are ORB libraries suitable for coding in a specific language</td>
</tr>
<tr>
<td>Definition of complex types</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support of distributed garbage collection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platform support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programming language support</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.2.: Comparing DCOM and CORBA: Basic characteristics.

These code examples, while reasonable and correct, are extremely limited and involve choices among a variety of possible approaches. Furthermore, it is very difficult to keep the two implementations exactly equivalent. Thus, the DCOM code appears to be much more verbose than the CORBA code. However, in common practice one would use the ActiveX Template Library (ATL), a C++ class library, and perhaps the ATL wizard, to reduce the manual coding to roughly the same amount as in CORBA. Therefore, comparisons based heavily on code examples can only be used as a means to become familiar with DCOM and CORBA, and not as a basis for general conclusions about either technology.

In order to derive such desired conclusions, a set of core properties that should characterise every distributed object platform are identified. Table 3.3. summarises the way that DCOM and CORBA support each of these properties, and offers an insight on the capabilities of the platforms pertaining their use in practical situations. This is explained in more detail in Section 3.4.1.

Furthermore, in order to compare the suitability and applicability of DCOM and CORBA in the telecommunications field, a set of (telecommunications) service engineering related properties are also identified, and their support by DCOM and CORBA is summarised in Table 3.4. This table focuses on how DCOM and CORBA provide a solution for developing effective (possibly large-scale) telematic services and how they assist in the deployment of these telematic services across the Internet, within an Intranet, over an Extranet, or simply with a Web front end.
Chapter 3: Distributed Computing Platforms in Telematics

<table>
<thead>
<tr>
<th>Core Properties</th>
<th>COM / DCOM</th>
<th>CORBA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Locator</td>
<td>Locally maintained</td>
<td>Centralised in the domain</td>
</tr>
<tr>
<td>Server Activation</td>
<td>Yes (Service Control Manager)</td>
<td>Yes (Basic Object Adapter)</td>
</tr>
<tr>
<td>Data Typing</td>
<td>Strong and predetermined (vtable method)</td>
<td>Strong and predetermined (Static Interface Invocation, SII)</td>
</tr>
<tr>
<td>Dynamic Invocation</td>
<td>Dispatch interface</td>
<td>Dynamic Interface Invocation (DII)</td>
</tr>
<tr>
<td>Communication Type</td>
<td>Synchronous, Asynchronous (callback support)</td>
<td>Synchronous, Asynchronous, Deferred</td>
</tr>
<tr>
<td>Inheritance</td>
<td>Interface, Implementation (containment, aggregation)</td>
<td>Implementation</td>
</tr>
<tr>
<td>State Persistence</td>
<td>Yes (2 models)</td>
<td>Yes (Persistence Service)</td>
</tr>
<tr>
<td>Load Balancing</td>
<td>Yes (load balancing router)</td>
<td>No</td>
</tr>
<tr>
<td>Exceptions Handling</td>
<td>Not directly (error reporting)</td>
<td>Yes (CORBA IDL)</td>
</tr>
<tr>
<td>Multithreading</td>
<td>Yes (3 models)</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 3.3.: Comparing DCOM and CORBA: Core properties.

During this comparison attempt, the various value-added services provided by DCOM (taking also into account the extensions and enhancements introduced by COM+) and CORBA are considered. These include, for DCOM, the Microsoft Transaction Server (MTS), the MicroSoft Message Queue server (MSMQ), the Microsoft Cluster Server (MCS), and the Microsoft Management Console (MMC), and for CORBA 2.3, the naming, event, life cycle, persistence, transaction, concurrency control, relationship, externalisation, query, licensing, property, time, security, object trader, and object collection services (CORBA services) [OMG98b][ORFA97][PINN98][RUB199]. Table 3.4. is explained in more detail in Section 3.4.2.

<table>
<thead>
<tr>
<th>Service Engineering Related Properties</th>
<th>COM / DCOM</th>
<th>CORBA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalability</td>
<td>MTS, Active Directory Service Interface (Win 2000)</td>
<td>Naming service, Trader service</td>
</tr>
<tr>
<td></td>
<td>MTS, MCS, MSMQ</td>
<td>Transaction service</td>
</tr>
<tr>
<td>Reliability</td>
<td>Built-in:</td>
<td>Platform dependent: 3 security levels (0, 1, 2)</td>
</tr>
<tr>
<td></td>
<td>NT LAN Manager, MTS, MS Crypto API, Authenticode SDK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MMC</td>
<td>Vendor specific tools, Transaction service</td>
</tr>
<tr>
<td></td>
<td>ActiveX, MS Active Server Page Technology</td>
<td>JavaScript/Java</td>
</tr>
<tr>
<td>Manageability</td>
<td>Two-factor authentication, Remote Data Service (RDS)</td>
<td>Two-factor authentication, Secure Socket Layer (SSL)</td>
</tr>
<tr>
<td>Support for Web-based Telematic Services</td>
<td>Desktop tools, ActiveX, Active Data Objects, MSMQ</td>
<td>Desktop tools (via a bridge), Event service, Persistence service</td>
</tr>
<tr>
<td>Support for Internet/Extranets</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support for Intranets</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.4.: Comparing DCOM and CORBA: Service engineering related properties.

Before reaching some conclusions regarding the comparison between DCOM and CORBA, their performance is also examined (in Section 3.4.3.), together with their capability to interoperate with the World Wide Web (in Section 3.4.4.). The outcome of these examinations complement Table 3.2., Table 3.3. and Table 3.4., which collectively constitute a concise and well structured decision framework that supports the selection between DCOM and CORBA [ADA99c], and facilitates further the service developer(s) in performing such a selection.
3.4.1. Core Properties.

Object Locator

A mechanism by which objects can be located and subsequently activated is necessary. If a server machine agrees that it can activate an object, it must be able to select the right executable, and the right object implemented in that executable. Both DCOM and CORBA avoid using machine addresses and port numbers for this purpose, because such schemes are inflexible and can lead to collisions.

DCOM allows for a locally maintained object locator on the server machine using object names, while CORBA and MTS centralise the locator on a single (or perhaps a few) domain machines that can identify object servers in the domain. More specifically, the most well-known object locator in DCOM is the registry. It maps a CLSID (or a readable name called ProgID) to the path name of the server executable that supports the CLSID. However, the registry is consulted only after the Service Control Manager (SCM) has failed to locate any running object instance. In CORBA, an object can locate another object in a system, by using either the naming service or the trader service. When using the naming service, an object looks up another object by the name under which the object registered itself with the ORB on initialisation. Alternatively, objects can query the trader service about objects with certain service characteristics that have registered.

Server Activation

A DCOM object server is not necessary to be running when a client request is made to instantiate an object. DCOM will locate the server code through the registry, and will start the server using SCM. DCOM also allows access to servers that are already running when the client request is made, as running objects are registered with the Running Object Table (ROT). In CORBA, server activation is handled by the BOA. If a client makes a request for an object that isn’t running, then the BOA will find the server and launch it to create the object. In this process it uses the implementation repository, which holds information about the location of every server object.

Data Typing

Once an object has been located and activated, the client will need to be able to communicate with it. Strong data typing is supported by both DCOM and CORBA through the use of interfaces. In the static method of invoking operations on DCOM objects, the MIDL compiler, based on the IDL definition of the object and its interfaces, creates the corresponding proxy and stub code. Due to the way that the static invocation is implemented, this is often referred to as the vtable method for invoking objects [EMME00]. In CORBA, in the case of the Static Interface Invocation (SII), all methods are specified in advance and are known to the client and server through the stubs and skeletons that are produced by the IDL compiler.
**Dynamic Invocation**

Although strong and predetermined data typing is extremely important for the creation of complex and robust code, sometimes the flexibility of slightly looser typing, similar to the kind which is important to interpreted scripting languages, is necessary. This can be allowed by enabling the dynamic querying of objects for the functionality that they support. DCOM provides this facility through its \texttt{IDispatch} interface and CORBA through its DII mechanism. In essence, predetermined typed interfaces are used that allow a dynamic interface to be queried.

More specifically, in DCOM dynamic invocation is based on the use of the \texttt{IDispatch} interface. Type libraries are special files that describe the methods implemented by an object and the parameters that these methods require, and DCOM provides interfaces, obtained through the \texttt{IDispatch} interface, to query an object's type library. In DCOM, an object whose methods are dynamically invoked must be written to specifically support the \texttt{IDispatch} interface. Furthermore, if the object has just the \texttt{IDispatch} interface then it can only be invoked in this way. However, DCOM also allows for an object to implement dual interfaces, where an interface is implemented both through \texttt{IDispatch} and through the vtable method [RUBI99].

In CORBA, the IDL compiler generates type information for each method in an interface and stores it in the interface repository. A client can thus query this repository to get run-time information about a particular interface and then use that information to invoke a method on the remote CORBA server object dynamically through the DII. Thus, the DII acts as a generic interface that does not require stubs, and allows an object to learn about the methods of other objects at run time. Similarly, on the server side, the Dynamic Skeleton Interface (DSI) allows a client to invoke an operation of a remote CORBA server object that has no compile time knowledge of the type of object it is implementing [VINO97]. It has to be noted, that unlike DCOM, any object can be invoked with DII, as long as the object information is in the implementation repository.

**Communication Type**

The communication between objects can be either synchronous or asynchronous. In synchronous communication, when a client makes a request for an object, the client process is blocked until the object replies. Usually, this is not a problem, since the requests from the client are made because the continuing action of the client depends on the data returned from the server. However, there are situations when the object request may take a large amount of time or when the data returned is not required immediately (e.g. during initialisation). While one way round this problem is to use multi-threaded execution, asynchronous communication provides a better solution enabling a message-passing paradigm. This may though complicate the client code if it requires some callback mechanism to indicate when the request has been serviced.

DCOM is mainly synchronous. However, it allows for flexible callback mechanisms, such as connection points, to be implemented [GRIM97]. In CORBA, a client can invoke a method, either synchronously or post it asynchronously. Asynchronous invocations have always been possible through the DII \textit{deferred} mode.
but they became possible for static invocations only in CORBA 3.0 [VINO98]. Posting has the meaning that the calling object is not blocked waiting for the reply. Instead, it can specify which of its methods the response should invoke. It has to be noted that the receiver cannot tell the difference between a synchronous or an asynchronous call.

Inheritance

DCOM allows interface inheritance, whereas CORBA allows implementation inheritance. In interface inheritance, when one interface is derived from another, the derived interface must supply an implementation for the methods of the base interface; all it inherits is the responsibility to supply the interface. In implementation inheritance a derived interface inherits the interface and an implementation. DCOM provides a similar mechanism using containment or aggregation (see also Section 6.3.1.) [RUBI99].

State Persistence

Objects represent both functionality and data. A client wishing to access an object would typically create the object, access its services, and then destroy it. The object server will need to be able to associate a client connection with a particular object, since each client will have some assumption about the state of the object when it last accessed it.

Both DCOM and CORBA use the notion of saving object state for later reactivation. DCOM has two persistence models. The original model requires that objects implement an interface that supports persistence using one of several known storage media (file, stream or storage). To reactivate an object, the client indicates explicitly where an object’s state is stored and requests its activation, usually through a file moniker. A more recent persistence model in MTS provide server-managed storage. Instances implemented using MTS retain no identity between transactions. MTS can destroy an instance and recreate it on subsequent method invocations without the client’s knowledge. The client must therefore identify the target of each method in its parameters [PINN98][PLATT99].

In CORBA, the persistence mechanism is completely transparent (persistence service). The client has no legitimate means of determining where or how an object is stored (unless some object with knowledge of the storage details provides an interface with a method that will divulge the information). The implementation is exclusively responsible for managing persistence [EMME00][ORFA97].

Load Balancing

A server machine may provide several object servers corresponding to several object types. Thus, the server machine may become a bottleneck in the distribution of objects and this leads to the need for load balancing. In essence, the server should employ a mechanism to allow the objects to be distributed across the network to other servers, so that the load is spread. A load balancing facility determines the server’s load and when an object request is made, it chooses the server with the least loading.
DCOM implements load balancing at the object level. Thus, when a client requests a specific object it first contacts a load balancing router. This router contains information about a cluster of machines belonging to the distributed application and balances the workload among these servers. Once the desired object has been instantiated on one of the servers in the application cluster, the client receives a reference directly to the object on the particular server machine. Thus, any future requests by the client go directly to the object. A load balancing facility is not offered currently in CORBA, but development is underway [VINO97].

**Exceptions Handling**

DCOM has a standard way of handling error data through the return of a 32 bit error code, called an HRESULT, by all methods (see also Section 5.2.1.5). At the language / tool level, a set of conventions and system provided services (the IErrorInfo interface) allows failure HRESULTs to be converted into exceptions in a way natural to the programming language [GRIM97]. On the other hand, CORBA specifies an extensible exception capability that maps naturally to languages that have native exceptions, like C++ and Java, and that maps into exception data in languages that do not. It is based on user-defined exception types declared in CORBA IDL [EMME00].

**Multithreading**

DCOM supports multithreaded server objects, but it requires that the DCOM libraries be initialised in the threads that use them. There are three main models; namely the simple single threaded model, the Single Threaded Apartment (STA) model, and the MultiThreaded Apartment (MTA) model (see also Section 6.3.3) [RUBI99]. CORBA object servers can also be multithreaded. Issues such as, for example, whether an object is created in a new process or in a new thread are handled by the ORB through the object adapter [VINO98].

### 3.4.2. Service Engineering Related Properties.

**Scalability**

A critical factor for a telematic service is its ability to accommodate easily, quickly, and efficiently an increase of the number of users, the amount of data, and / or the required functionality in terms of the number of service objects. The telematic service should be small and fast when the demands are minimal, but it should be able to handle additional demands without sacrificing performance or reliability.

MTS provides a set of DCOM interfaces and libraries that promises to allow telematic services to easily scale as the size of the problem increases. It also provides automated thread management to facilitate the handling of increased demand, and offers resource pooling and database connection multiplexing in order to distribute more efficiently user and data load across multiple replicated service components. In Windows 2000, DCOM is also equipped with the Active Directory Service Interface (ADSI), which allows components to seamlessly use a variety of existing naming services, such as the Netware Directory Service.
Chapter 3: Distributed Computing Platforms in Telematics

(NDS), the Lightweight Directory Access Protocol (LDAP) or even the Windows registry [GRIM97]. In this way, a telematic service is able to handle an increasing number of geographically dispersed users.

In CORBA, the Object Activation Daemon (OAD) and the implementation repository allow efficient use of resources by only instantiating objects when required. The centralised naming service provides location independence for telematic services and their users, while the trader service allows more sophisticated component searches [EMME00].

Reliability

Distributed objects should offer transparency to a client and part of this transparency is the guarantee that the object connection will be reliable throughout the client’s use of an object. In the optimum case, a remote object must be as reliable as a local object. Reliability can be supported by using a transaction monitor, like MTS or the OMG transaction service, to efficiently exploit the replication of service objects around a network. MTS allows telematic services to use distributed transactions to reliably update data across disparate data stores, while the OMG transaction service supports an Open Group Distributed Transaction Processing (DTP)-compliant model for distributed transactions. As far as DCOM is concerned, MCS and MSMQ also increase reliability. MCS can automatically restart telematic services and provides automated failover between server machines, and MSMQ, through the use of reliable message queues for communication, provides features such as guaranteed delivery of object requests and disconnected operation.

Security

DCOM has been designed with security built in, while CORBA objects can implement their own security mechanisms for the platform on which they are implemented. More specifically, the NT LAN Manager (NTLM) and the MTS authenticate users and authorise checking via Access Control Lists (ACLs). Additionally, the MS Crypto API provides data encryption and integrity to prevent eavesdropping and tampering, while the Authenticode SDK uses digital signatures to provide non-repudiation. The challenge for the future is to integrate all these value-added services in a single solution [PLAT99].

On the other hand, CORBA defines two security levels; Level 1 and Level 2. Level 1 allows a telematic service that is unaware of security to participate in a secure domain. It provides user authentication, authorisation via ACLs, data encryption and integrity, and optionally non-repudiation. Level 2 requires telematic services to be security-aware, and thus it is enforcing stronger versions of the security policies [ORFA97]. Some CORBA vendors, such as Iona and Inprise, have provided a Secure Socket Layer (SSL) implementation of IIOP, called Level 0, that allows user authentication and data encryption. Level 1 implementations are currently available from Iona and IBM, and are based on DCE security. Level 2 implementations are not currently available.
Manageability

Microsoft Management Console (MMC) provides a unified GUI for managing MTS and MSMQ based service components. Features include centralised configuration and administration, as well as remote deployment of components. MTS gathers various statistics about service components under its control, including transactional outcome, mean time between object requests, and overall activity level. This information can then be viewed in the MMC.

As far as CORBA is concerned, vendors such as Inprise and Iona have sophisticated tools for centrally configuring and administering CORBA-based telematic services. Iona also allows these services to be centrally managed from any Simple Network Management Protocol (SNMP)-compliant system management console (e.g. OpenView). The transaction service collects various usage statistics (such as transactional outcome, mean time between object requests, and maximum load per service component) that can be used for performance tuning, and provides extensive logging and auditing features.

Support for Web-based Telematic Services

A telematic service is Web-based when its front end is a Web browser and it does not necessarily mean that the telematic service is deployed over the Internet. The front end is typically developed using regular HyperText Markup Language (HTML), or either JavaScript / Java or ActiveX (when it needs to be more sophisticated) [DOLG97].

DCOM front ends in the form of ActiveX controls can execute within Internet Explorer and, via a plug-in, within Netscape Navigator. ActiveX controls can be installed into the browser without user intervention. In addition, DCOM is available natively on all Windows platforms, so that regardless of which browser is used, the application does not have to download the runtime service. Furthermore, Microsoft’s Active Server Page (ASP) technology allows the seamless integration of both HTML and ActiveX clients with DCOM servers. It also allows DCOM services such as MTS and MSMQ to be used with Web-based telematic services [VOTH98].

On the other hand, Java-based front ends to CORBA-based telematic services can execute on all major browsers and platforms. Because Netscape Navigator is supplied with the CORBA runtime service, front ends deployed using Netscape can be downloaded faster since they do not have to download the runtime service. Furthermore, the Netscape Enterprise Server provides the Web Application Interface, which allows HTTP-based clients to communicate with CORBA servers.

Support for Internet / Extranets

Telematic services that need to operate across the public Internet or a semi-public Extranet are typically deployed across great distances and often through several firewalls. Such telematic services, when they

---

8 Enhancements of HTML, like Dynamic HTML (DHTML) and eXtensible Markup Language (XML), can also be used for such purposes.
Chapter 3: Distributed Computing Platforms in Telematics

involve transactions executed over the Internet, they require additional security measures to ensure accuracy, confidentiality, and credibility. These measures include support for a public-key security infrastructure (e.g. SSL), which is able to deal with the unpredictable and anonymous nature of Internet users better than a traditional private-key system (e.g. Kerberos), two-step verification of the user's identity based on both a password and a physical token such as a smart card (two-factor authentication), digital validation of important transactions via a third party to ensure accuracy and timeliness, and integration of Internet-specific security policies with the existing enterprise security infrastructure to ensure seamless end-to-end security.

**DCOM** provides two-factor authentication (through public certificate and smart cards) and Remote Data Service (RDS) support. Currently, there is limited support for SSL and its integration with NTLM security. However, this will be expanded to include SSL-to-Kerberos integration in a future version of COM+. On the other hand, several CORBA vendors support both SSL and two-factor authentication in their implementation, although these features are still immature. Furthermore, CORBA 3.0 recently defined a set of interfaces for allowing IIOP requests and replies to pass through a firewall [OMG98c].

**Support for Intranets**

Telematic services which are limited inside an Intranet, are usually optimised for use within an organisation, have higher network bandwidth, and little or no firewall restrictions. Thus, they can be built with more sophisticated front ends, both in terms of user interface and functionality. Sophisticated functional requirements might include local object caching and persistence, as well as a publish / subscribe capability.

DCOM-based telematic services can be built with sophisticated user interface and functional features, as every major development environment on Windows supports the rapid development of graphical DCOM applications and there are a great variety of ActiveX controls available, either for free or for a nominal charge, that provide additional charting, graphing or industry-specific features. Furthermore, Active Data Objects allows the support of persistence, while MSMQ provides publish / subscribe capabilities. On the other hand, CORBA-based telematic services can be integrated with desktop tools using a DCOM / CORBA bridge (see also Section 3.5) available from several CORBA vendors. CORBA services such as event and persistence can also be used to add publish / subscribe and persistence features.

### 3.4.3. Examination of Performance.

The performance of distributed software, usually expressed in terms of execution time for a variety of operations / actions, is a critical factor for the development and provision of successful (user accepted and efficient) telematic services, especially when real-time functionality and / or multimedia characteristics are required. Therefore, a comparison between DCOM and CORBA has to address performance matters in order to be complete and offer a full insight of the practical value of the two most prominent distributed object platforms [ADA00e].
Performance comparisons between DCOM and CORBA that are found in the literature use a simple example of distributed code (two objects residing in two different computers that communicate via a network) and measure the time needed for the successful completion of a single remote method invocation (when one of the objects calls a method of the other object and waits for the result) [LEWA98][MOUN97]. Although such an approach is valid, a performance comparison based not just on a single remote method invocation, but on a series of logically related remote method invocations (forming a usage scenario / pattern) can lead to more accurate, reliable and illustrative results, and can also be the stimulus for the deduction of a number of more general, albeit useful and of practical value, performance related conclusions.

For this reason, considering the latter remark, an experiment involving multiple remote method invocations under both DCOM and CORBA was conducted. This experiment and the results obtained from it will be examined in the following subsections in an attempt to evaluate the performance of DCOM and CORBA under conditions which are common in telecommunications services engineered as distributed object applications.

Before proceeding, it has to be stressed that the performance of object-oriented DPEs, like DCOM and CORBA, that provide high-level network programming interfaces is comparable (under certain circumstances) with the performance experienced when using low-level, procedure-oriented, non-typesafe programming interfaces, such as BSD sockets [FAT097]. Furthermore, the performance of both DCOM and CORBA keeps improving with the application of compiler optimisation techniques and the utilisation of light-weight communication protocols [CHEN99][GOKH96]. Nevertheless, service developers seem to be willing to accept a certain performance penalty given all the benefits (and especially extensibility, maintainability, and reusability) they are gaining from using distributed object platforms [FISC97][SALE99].

3.4.3.1. The Experiment.

A simple distributed object application implemented under both DCOM and CORBA constitutes the basis of the experiment that was conducted. More specifically, a server object returns (after an appropriate request) fixed length strings (each 80 characters long) to a client object in two different ways: one string after the other as a result of separate consecutive method calls, or by gathering a number of strings and returning them all together as a result of a single method call. Equivalently, the client object, when interacting with the server object, can either make multiple method calls for small amounts of data (one string) or a single method call for a larger amount of data (several strings). The IDL description of the server object interface in DCOM, which is similar to that in CORBA, is (using a simplified variation of MIDL):

```c
interface ITestServer : IUnknown
{
    HRESULT GetSingleString([in] LONG index, [out] BSTR* item);
    HRESULT GetMultipleString([in] LONG index, [in] LONG count,
                               [out] LONG* got, [out] BSTR** item);
};
```
The first method (\texttt{GetSingleString()}) of this interface returns a single string, based on the ID of that string that is included in \texttt{index}, as all strings are kept in an array until the data is requested. The second method (\texttt{GetMultipleString()}) returns a number of strings (\texttt{count}) starting at \texttt{index}. It has to be noted that in order to obtain comparable results, care was taken to have both the DCOM and CORBA versions of the testing code execute on the same operating system platform (MS Windows NT 4.0) and on exactly the same hardware and network infrastructure. To achieve this uniformity, the CORBA implementation of the client and server objects used Iona's Orbix ORB under MS Windows NT 4.0, and all the testing activity took place using two 350 MHz Pentium II computers with 64 MB of memory interconnected by a 10 Mbit/s Ethernet LAN. In the following paragraph, important parts of the code of the client and server objects are briefly discussed.

The server object has an embedded object of class \texttt{Elements}. When this object is created, it loads the strings in the specified file (\texttt{elements.dat}) and it makes each string 80 characters long by adding the appropriate number of dashes at the end of the string. Then, it keeps these strings in an array until they are requested. The client object determines whether the single or multiple case is used, specifies the start value and the number of items (strings) to get, and calculates the average time required to make the call. If it is a single test, the \texttt{GetSingleString()} method is called. Otherwise the \texttt{GetMultipleString()} method is called. \texttt{GetSingleString()} simply sends a single value and thus it is called for each of the required values (the number of strings and the first index are specified by the client). In order to increase the accuracy of the tests, the tests are repeated the number of times specified by the client. For the multiple case, the \texttt{GetMultipleString()} method is called on the server object just once. In both cases, the calls to the server object are timed and the average of the time needed to get the requested data is calculated.

\subsection*{3.4.3.2. The Results.}

During the experiment, two types of measurements were carried out, using both the DCOM and CORBA versions of the testing code. Initially, the client and the server objects were placed on the same machine, and the time (in ms) needed to transfer a number of strings from the server object to the client object, as a result of calling (on the server object), either \texttt{GetSingleString()} many times or \texttt{GetMultipleString()} once, was calculated for several different numbers of strings (Figure 3.6.). In this way, the performance of (the usually neglected) \textit{local method calls} is examined under DCOM (in fact COM) and CORBA, recognising the fact that local object interactions are common even in large scale telematic services and thus they shouldn't be ignored or underestimated. As can be seen in Figure 3.6., local method calls are fast in both DCOM and CORBA, with DCOM being slightly faster than CORBA. Additionally, for both DCOM and CORBA, the single method call (\texttt{GetMultipleString()}) is about 10 times faster than making multiple method calls (\texttt{GetSingleString()}) for a specific number of strings.
Chapter 3: Distributed Computing Platforms in Telematics

Figure 3.6.: Examination of local method calls in DCOM and CORBA.

The second type of measurements focused on remote method calls, which are the ones that affect mostly the performance of a telematic service. In this case, the time (in ms) needed to transfer a number of strings from the server object to the client object was calculated as in the first type of measurements, with the exception that the client and server objects were placed on two different machines connected via a network (Figure 3.7.). As would be expected, remote method calls take much longer than local method calls in both DCOM and CORBA, although the measurements have been taken on a very quiet network of just two machines running only the test software. From Figure 3.7. it is evident that CORBA is slightly faster than DCOM regarding remote object interactions, and that in both DCOM and CORBA, the multiple calls of GetSingleString() take about 5 times longer than the GetMultipleString() call for a specific number of strings.

Figure 3.7.: Examination of remote method calls in DCOM and CORBA.

Taking into account all the measurements and evaluating the whole experiment it is apparent that in both DCOM and CORBA remote method calls are slower than local method calls, and many single method calls are five to ten times slower than a single multiple method call. Finally, Figures 3.6. and 3.7. clearly illustrate that DCOM and CORBA have a comparable performance under the MS Windows operating system platform. Therefore, for this operating environment, a choice between DCOM and CORBA should
not be based exclusively on performance considerations, but it should also take into account other more
general and abstract/qualitative issues [ADA99c].

This conclusion is reinforced even more by the fact that CORBA performance depends significantly on
the implementation of the ORB by a specific ORB vendor and thus differs between different products
[VINO97]. Furthermore, there exist special purpose ORBs, usually developed during research projects,
which are characterised by increased performance. A characteristic example is the TAO ORB, that executes
on real-time operating system platforms, and is primarily designed for real-time applications [SCHM98].

A similar situation is also true for DCOM, as DCOM's performance can be improved in certain
circumstances by extending its remoting architecture which has built-in extensibility [GRIM97]. The way
that this can be achieved is examined separately in the following section, because of the significant
potential benefits it can offer to DCOM-based telecommunications services.

3.4.3.3. DCOM Remoting Architecture.

Distributed object systems, such as DCOM and CORBA, provide the necessary infrastructure for supporting
remote object activation and remote method invocation in a client-transparent way. The term *remoting
architecture* refers to the entire infrastructure that connects clients to server objects [BROW98][GRIM97].

A distributed object system does not necessarily have to specify how the entire remoting architecture
should be structured. It can treat it as a black box as far as user applications are concerned. This approach
has the advantage of allowing vendors to use their best performance optimisation techniques. However, a
disadvantage is that such architectures are usually difficult to extend [WANG98]. Therefore, when low-
level system properties, such as load balancing and fault tolerance, are required by a telematic service
under development, they need to be either tightly integrated with the infrastructure [MAFF98] or provided
through interception mechanisms outside the infrastructure [NARA97].

The DCOM remoting architecture can be seen in *Figure 3.8*. Its main constituent parts are the following
[BROW98][GRIM97]:

- **Object proxies**: They act as the client-side representatives of server objects and connect directly to the
  client.

- **Interface proxies**: They perform client-side data marshaling and are aggregated into object proxies 9.

- **Client-side channel objects**: They use RPCs to forward the marshaled calls.

- **Server-side endpoints**: They receive RPC requests from clients.

- **Stub managers**: They are located in the server and they dispatch calls to the appropriate interface stubs.

9 It has to be noted that the interface proxies and stubs are application-specific and are generated by executing the
MIDL compiler on application-specific IDL files, while all the other objects/parts are application-independent and
are provided by DCOM.
Chapter 3: Distributed Computing Platforms in Telematics

- **Interface stubs**: They perform server-side data marshaling and make actual calls on the appropriate server objects.

- **Standard marshaler**: It marshals interface pointers into object references on the server side and unmarshals the object references on the client side.

![Diagram of DCOM remoting architecture and extensibility options](image)

**Figure 3.8.**: The DCOM remoting architecture and extensibility options.

Usually, DCOM-based telecommunications services use *standard marshaling*. However, some telecommunications services may need to customise the client-server connection in order to express the correct semantics and improve performance. In these cases, the DCOM remoting architecture has to be extended.

The extensibility provided by DCOM can be divided into three categories; namely, *below, above, and within* [BROW98][GRIM97][WANG98]. The first category extends DCOM at the RPC layer and below, as shown in *Figure 3.8.*, in a way totally transparent to the standard remoting architecture. To achieve the other two types of extensibility, DCOM supports a *custom marshaling* mechanism which allows a server object to bypass the standard remoting architecture and construct a custom one, optimised for a particular situation, without requiring source code modifications to the former. A server object declares that it wants to implement custom marshaling by supporting the IMarshal interface. By implementing the method calls of this interface the server object can construct a custom object reference and specify the CLSID of the client-side custom unmarshaler, which will be automatically instantiated by DCOM to receive and interpret the custom object reference and to make the custom connection using an application-specific communication mechanism.
According to the second type of extensibility in DCOM, a handler layer can be inserted above the standard remoting architecture and below the user application (service components). This activity is often called semi-custom marshaling (or handler marshaling) because most of the tasks are eventually delegated to the standard remoting architecture, as shown in Figure 3.8. As part of the marshaling/unmarshaling process, a custom proxy and a custom stub are inserted to allow additional processing of each method invocation.

The third type of extensibility in DCOM is the most general and the most promising one. According to it, DCOM's remoting architecture, should not only provide the basis for building distributed component-based applications (e.g. telematic services), but it should also be a distributed component-based application by itself. More specifically, as the remoting architecture is constructed at runtime by instantiating and connecting various components, it should be possible for a custom architecture to reuse some of the binary components from the standard one and supply only the necessary custom objects. The construction of such a custom architecture is hard in the current DCOM architecture, but it can be facilitated significantly by specialised architectures developed for this purpose. Such architectures can also efficiently support the incorporation and flexible customisation of low-level system properties (e.g. load balancing, fault tolerance, etc.) into the DCOM infrastructure [WANG98].

CORBA does not specify a standard remoting architecture. Therefore, incorporating stronger system properties into CORBA-based telecommunications services and improving their semantics and performance is usually not done by exploiting the extensibility of the remoting architecture. Furthermore, while some CORBA-based systems allow the replacement of the marshaling code for a given interface (sometimes called smart proxies), DCOM is unique in that the remoting behaviour is polymorphically bound at runtime on an object-by-object basis, as two references of identical type may be using custom or standard marshaling independently. This allows object implementers to safely evolve their remoting implementation based on performance needs without rebuilding client applications.

3.4.4. Interoperation with the World Wide Web.

The Internet, mainly because of the unparalleled success and proliferation of the World Wide Web (WWW), is currently the second largest distributed system in the world, behind only the telephone network. For this reason, telematic services engineering as a collection of DCOM/CORBA objects should be able to interface with Internet applications and use efficiently Internet technology and resources. Therefore, both DCOM and CORBA, despite their deterministic nature, is necessary to interoperate with the WWW and the Internet, in which many requirements for total reliability are willingly relaxed in order to achieve scaling to global proportions [BERN92]. Two approaches for such an interoperation are proposed and briefly examined in the following paragraphs [ADA98b].
Adding Web-like characteristics into DCOM and CORBA will initially result into relatively disconnected islands of interacting objects, offering high performance, reliability, manageability, etc. (objects on the Web). These islands of distributed objects will inevitably start to interconnect, as new capabilities provided by other islands are required, resulting in the long term in globally interconnected interacting objects (a web of objects) [BRIS97].

To increase the functionality of the Web browser the arrangement of Figure 3.9. is proposed. According to this, any user interface that is not provided by standard browser elements is downloaded as mobile code (an “applet”), which can be written in Java or in a scripting language like Visual Basic Script. In this way, any infrastructure elements that are not available at the user end, such as a protocol stack for direct communication with a CORBA or a DCOM server, can also be downloaded. However, it is obviously preferable that such commonly used software is not continuously downloaded and this has been recognised by the main browser vendors. It must also be noted that, as can be seen in Figure 3.9., the server object can be installed at the user end.

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{fig39.png}
\caption{Interoperability using applets and (remote or local) server objects.}
\end{figure}

In the previous approaches, the user end has been assumed to be initiating a call. However, in many cases retroaction takes place utilising a call-back model, where an interest in a certain type of reply is registered by the user entity, leaving a call-back address, and asynchronously, replies or event notifications appear any time later. Such a situation can be seen in Figure 3.10., where the necessary code and protocol stack is downloaded as mobile code. In this arrangement a solution has to be provided to the problem of opening up a firewall to admit asynchronous messages on connections not directly initiated from inside.

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{fig310.png}
\caption{Interoperability using applets as notification objects.}
\end{figure}
From the discussion of the proposed approaches, it is evident that DCOM and CORBA are equally capable to interoperate with the WWW. Therefore, islands of distributed object systems can be deployed over the ubiquitous WWW infrastructure resulting in very promising new telecommunications services. However, there is no guarantee that the interconnection of these islands into a global object-oriented telecommunications service will work reliably. Further research is necessary, together with considerable standardisation efforts, before achieving full integration in a global scale [ADA98b][BRIS97].

3.4.5. Comparison Remarks.

The liberalisation of telecommunications markets has exposed service providers to a high level of competition. This competition is forcing them to reduce costs, improve customer satisfaction, and rapidly introduce new services. One key way in which these pressures can be addressed is through the increased exploitation of distributed object platforms, such as DCOM and CORBA.

From the proposed decision framework (Table 3.2., Table 3.3., and Table 3.4.) it is evident that DCOM and CORBA have similar architectures as both provide the infrastructure for supporting remote object activation and remote method invocation in a client-transparent way. Both adopt a client / server based programming style and agree on the most fundamental aspects of their object models. More specifically, both approaches use abstract interfaces (specified by an IDL) to define how objects represent their functionality to clients. These interfaces allow only method invocations, so that implementation details are hidden from the client. This encapsulation reduces the linkage between the client and the server object implementation to the minimum possible extent and enables location, platform, and programming language independence. It also allows programs to select the object implementations that will use during execution, rather than at compilation or linking. This dynamic binding of executable code makes upgrading and reusing implementations easier.

As far as the support of service engineering related properties is concerned, DCOM and CORBA differ in many respects. Most significantly, while DCOM provides a rich set of tools and technologies, it is essentially a Windows-only solution. Even though DCOM is available on other operating systems, key pieces such as MTS, MSMQ, and MCS are not currently offered. Additionally, many of DCOM's value-added services are very new and are still maturing. On the other hand, CORBA's main strength, which is its availability by different vendors, is also its biggest weakness. It is very difficult (if not impossible), for example, to get one vendor's security implementation to work with another vendor's transaction service. Since no vendor has a complete solution, integration issues are usually introduced when CORBA is used to build telematic services. For this reason, neither technology provides a complete solution for service engineering activities. However, both provide a solid infrastructure and there are specific scenarios in which either excels over the other.

Finally, in Section 3.4.3. the performance of DCOM and CORBA was examined focusing on their ability to support object interactions commonly used in new telecommunications services. The experiment that was
conducted revealed that DCOM and CORBA have a comparable performance, although DCOM appears to be more flexible and with a significant potential for improved performance due to its extensible and customisable remoting architecture. Taking also into account that both DCOM and CORBA can interoperate with the WWW, it can easily be deduced that selecting between DCOM and CORBA for a specific service engineering activity can be a complicated task that requires experience, an accurate appreciation of the main parameters of the problem domain, and an ability to consider the merits of these technologies from a less technical and more strategic perspective (see also Section 3.6.).

In general, the proposed decision framework and the activities that complement it offer a deeper understanding of the capabilities of DCOM and CORBA in the service engineering area and therefore facilitate significantly service developers in a possible selection process. Furthermore, they also make evident that DCOM and CORBA have much in common and continue to converge in several aspects, although each architecture has different origins, with consequent strengths and weaknesses.

3.5. Interworking Between DCOM and CORBA.

Because both DCOM and CORBA are being used in practice with considerable success, and because of the economic implications that result from this fact, it is unlikely that one platform will soon overwhelm the other. Therefore, interworking between DCOM and CORBA is crucial, not only for system architects, who often have to integrate software components of different origin, but also for developers who want to maximise their productivity by using the tools they are most familiar with, and thus minimise the learning curve associated with working with a new object model [ADA99c][DOLG97].

Since CORBA 2.2, the interworking between DCOM and CORBA is part of the CORBA specification and it is achieved by bridges which receive object invocations from a CORBA application, translate them into equivalent data structures for DCOM, and have the function call executed in the DCOM application. Since from DCOM’s point of view the bridge was the client for this call, the bridge will receive the results, which it will send back to the actual client inside the CORBA application. In a similar manner, DCOM clients can access CORBA objects through bridges. The difficult part of this process is the translation of CORBA data structures into DCOM data structures and vice versa, because a bridge is not part of an application, and thus, it is not given information about application-specific data structures. However, without this knowledge, a bridge can’t convert the received byte sequences back into in-memory representation of the data [EMME00].

This problem can be overcome by using either static or dynamic bridges. Static bridges are provided with a rich set of data structure definitions that often occur in communications between clients and servers. However, this set of data structures is statically compiled into the bridge (i.e. it is fixed). Thus, if at runtime a data type is used that is not contained in this set, the bridge will not understand it and will report an error.
The other alternative is the use of a dynamic bridge. Dynamic bridges take advantage of the fact that in both DCOM and CORBA information about the interfaces and the data structures is stored for efficiency reasons in suitable infrastructure components (type libraries, interface repository). With the knowledge obtainable from these IDL repositories, a dynamic bridge can learn about new data types as they occur (i.e. at runtime). However, due to efficient caching mechanisms, access to IDL repositories is only needed when new data types are discovered. Hence, the performance of dynamic and static bridges is almost the same, but the implementation of dynamic bridges is much more complicated than that of static bridges [REDL98].

In both cases, the mapping between the DCOM and CORBA object models is implemented in a compiler. The developer of a DCOM component will use this compiler to obtain the interfaces from a DCOM point of view. Likewise, the CORBA developer will use the compiler to translate MIDL into CORBA IDL. From then on the developer will be unaware that the objects are implemented in a different distributed object technology.

### 3.6. The Distributed Processing Environment of Choice.

Advances in distributed object platforms have been rapid in the past few years. These advances have been largely driven by increasing demand for efficient object creation, interaction, management, and distribution. Both DCOM and CORBA address these issues, and are increasingly being used to develop new telecommunications services as distributed object applications. However, further progress is expected, and as both technologies are still evolving, it is likely that in the near future they will converge in more areas.

DCOM is built on a proven desktop component architecture. DCOM-based applications are robust and perform well, while DCOM’s integration into development languages and tools simplifies greatly application development. Furthermore, as Windows-based desktop systems exist in nearly all organisations today, these organisations will probably choose to use DCOM. Additionally, Microsoft services (MTS, MSMQ, and other mainframe integration tools) make DCOM an attractive infrastructure even for large organisations in enterprise-wide applications. However, DCOM is not a well partitioned architecture and relies on a key optimisation for a single platform.

In contrast, CORBA has a more complete and well-defined architecture and provides a better solution for heterogeneous environments. It offers advantages in (value-added) services, platform and tool support, maturity, and overall architectural integrity. Furthermore, OMG IDL ensures an extensible architecture and support for both new and legacy applications. The disadvantages of CORBA are its complexity and variation in vendor implementations.

Therefore, DCOM is an effective solution for the development of telematic services in Windows-based environments, particularly by medium and small organisations and departments. On the other side, a requirement for multi-platform support or for a choice with the least technological risk will drive an organisation towards a CORBA solution. However, such a decision will (should) be highly influenced by
more general factors, such as the available Information Technology (IT) resources and skills / experience, the IT structural characteristics and its relation to business units of the organisation, the desired level of standardisation, and the capability to adopt new technologies [TALL98].

From this discussion and from Section 3.4., it is evident that both DCOM and CORBA can be used as a DPE technology in the service execution environment of the proposed telecommunications service engineering framework of Figure 1.1., as they both satisfy Requirement 3 (see Section 1.3.) and have comparable characteristics and equivalent capabilities. However, as the service support environment of the proposed framework (which includes the service execution environment) is influenced by TINA-C (see Section 2.5.2.), the main requirements that TINA-C has from a DPE should be satisfied. According to these requirements, a TINA-C DPE should at least provide object life-cycle and trading functionality, support multiple interfaces per object, and model continuous media interactions using appropriate stream interfaces [CARR96][TIN94a].

CORBA provides object life-cycle and trading functionality through the respective CORBA services (see Section 3.2.3.), while DCOM offers object life cycle facilities natively (e.g. through the use of the IUnknown interface) and supports a mechanism for locating objects through the use of the registry and the MTS (see Section 3.4.1.). It has to be noted that although DCOM does not provide itself true trading functionality, a trading service for DCOM (based on the RM-ODP trader model) has been designed, implemented, and used successfully [OUTH97]. Furthermore, DCOM (and MIDL) allows an object to have multiple interfaces, while this is not possible in any current implementation of CORBA 10. However, this shortcoming of CORBA implementations is expected to soon be addressed, as CORBA 3.0 specifies and supports components, which allow the composition of multiple interfaces [OMG99]. Finally, CORBA prescribes the use of stream interfaces for the support of continuous media communications (see also Section 6.2.) [OMG98d], while DCOM does not address this issue at all. Considering altogether the TINA-C requirements from a DPE, it is evident that the most appropriate DPE technologies for the service execution environment of the proposed framework are DCOM enhanced with the modelling of continuous media interactions and an ORB implementing CORBA 3.0 specification.

For the rest of this thesis, DCOM is selected to be used because it is comparable with CORBA, it is a relatively new technology that presents increased research interest, its significance (and its use) is expected to raise very quickly as Windows penetrate more and more the market of the operating systems for enterprise servers [PLAT99], and because its serious support and rapid evolution is guaranteed by the commitment of Microsoft to this technology 11. Therefore, DCOM will be enhanced with the support of continuous media in

---

10 A common technique to enable this, is to use multiple inheritance and inherit state and behaviour from multiple CORBA interface classes.

11 The commitment of Microsoft to DCOM increases also the competition with CORBA and motivate in this way OMG to speed up the (traditionally) slow standards making process with obvious benefits for distributed object technology in general.
Chapter 3: Distributed Computing Platforms in Telematics

order to qualify for a TINA-C DPE and it will be used for the implementation of service code in the framework of specific service creation activities.

3.7. Summary and Conclusions.

In response to major trends in the telecommunications market today and under influence of the emerging distributed computing technology the telecommunications industry is embracing distributed object platforms as a means enabling the successful participation in the open global services market of the foreseen era with new and advanced service offerings under increasing competition in a multi-vendor environment.

In this realm, this chapter starts by considering the most significant existing de jure and de facto DPEs (Section 3.2.). More specifically, after explaining the importance of open distributed processing, it presents ISO's / ITU-T's RM-ODP, Open Group's DCE, OMG's CORBA, Microsoft's COM / DCOM and Sun's Java RMI focusing on their essential characteristics, and then it evaluates them from the perspective of telecommunications service engineering. During this evaluation attempt it is argued that the various initiatives in the field of open distributed processing are complementary and a path to convergence is proposed which clearly reveals that DCOM and CORBA are the only two DPE technologies that have currently the capabilities to support service engineering activities in an efficient and effective manner.

After justifying the importance of DCOM and CORBA in Section 3.3., by discussing the main benefits that distributed object technology can offer to telecommunications services, the rest of the chapter focuses exclusively on the detailed examination of the role of DCOM and CORBA in service engineering. Therefore, in Section 3.4. a comparison between DCOM and CORBA is attempted, by constructing an appropriate decision framework (based on a number of basic characteristics and on a carefully selected set of core and service engineering related properties), by examining the performance of DCOM and CORBA with a suitably structured experiment, and by considering their interworking with the WWW. At the end of this comparison attempt it is highlighted that selecting between DCOM and CORBA for a specific service engineering activity can be a complicated task, as both these technologies provide a solid infrastructure, have a comparable performance, and are equally capable to interwork with the WWW. In general, it is argued that both DCOM and CORBA can assist service developers to cope efficiently with the complexity inherent in the process of realising telecommunications services as open distributed software applications comprised by service components which may be scattered across multiple organisations and distant locations, because both support communication and cooperation between service components, allow their identification and location, enable software portability, and provide all the other basic functions needed to overcome the difficulties of heterogeneity and distribution, which are common in distributed systems.

Finally, in Section 3.5. the interworking between DCOM and CORBA is examined and in Section 3.6. the problem of selecting between DCOM and CORBA is revisited once again by discussing the suitability of
each one of these platforms in different service provisioning situations. In this section, the main TINA-C requirements from a DPE are also taken into account and **DCOM is selected** as the distributed object platform that will be used in the rest of this thesis.

This chapter is mainly an introductory chapter regarding the main objectives of this thesis (see Section 1.4.), and it reveals that both DCOM and CORBA can satisfy **Requirement 3** (see Section 1.3.) and can be used as a DPE technology in the service execution environment of the proposed telecommunications service engineering framework of **Figure 1.1.** The research contributions of this chapter, which are among the secondary objectives of this thesis (see Section 1.4.), are the following [ADA98b][ADA99c][ADA00e]:

- The evaluation of the most important existing de jure and de facto DPEs from the perspective of telecommunications service engineering and the proposal of a potential path for their convergence (Section 3.2.6.).
- The discussion about the benefits from engineering telecommunications services as distributed object-oriented applications upon distributed object platforms (Section 3.3.).
- The construction of a decision framework that facilitates the selection between DCOM and CORBA by identifying a set of basic, core and service engineering related properties, and examining the way that DCOM and CORBA supports these properties (Section 3.4., and especially Sections 3.4.1. and 3.4.2.).
- The examination of the performance of DCOM and CORBA by conducting a suitably structured experiment that focuses on their ability to support object interactions commonly used in telematic services (Section 3.4.3.).
- The examination of the ability of DCOM and CORBA to interwork with the WWW and the Internet, and the proposal of a number of possible approaches for such an interworking (Section 3.4.4.).
- The comparison and contradistinction of DCOM and CORBA in the area of telecommunications service engineering (Section 3.4. and 3.6.).

With this chapter, all the constituent parts of the proposed telecommunications service engineering framework of **Figure 1.1.** have been examined, except from the service development methodology, which has a central role in this framework and is related to one of the main objectives of this thesis. This examination will take place in the next chapter (Chapter 4) utilising the main findings of both Chapter 2 and this chapter. DCOM will be used during the validation of the proposed service development methodology in Chapter 5 and for this reason it will be enhanced with the ability to model continuous media interactions in Chapter 6 (see also Section 5.5.).
4.1. Introduction.

The advent of deregulation combined with new opportunities opened by advances in telecommunications technologies has dramatically changed the paradigm of telecommunications services. Due to the evolving synergy between information and telecommunications technology, termed telematics, the number and capabilities of services is rapidly growing, and services and network infrastructure is expected to be more and more unbundled. Services have become increasingly software based, taking the form of distributed (multimedia) applications operating on DPEs and reusing and combining other already deployed services as well as capabilities of telecommunication and computing resources [ADA00a]. Furthermore, in the emerging liberalised telecommunications market, multiple competing and cooperating players handle the process of creating, deploying, and providing new telecommunications services (telematic services) [ADA98a][DELA97].

There is much incentive to stay ahead of this global market and offer new and/or improved services before the competition. Pressure on service providers is increased as they need to be able to react quickly and flexibly to ever changing customer needs by developing and offering services of enhanced functionality and significant diversity in shorter time-frames. Therefore, the rapid deployment of new or improved services is critical, and the service life-cycle has to accelerate so that new services can be provided fast enough to meet the changing customer demands in a competitive manner. However, the fast and cost-effective provision of new efficient services requires not only an open service architectural framework, like the one specified by TINA-C (see Section 2.5.2.), but also appropriate support for the service development process [ADA00f][DEME99].

The creation of telecommunications services within an open environment is a highly complex activity. This complexity stems not only from the technical nature of the tasks involved, but also from the number of the participating actors and the variety in their roles, concerns, and skills. Therefore, there is a need to support the complex service creation process in order to ensure that resulting services actually perform as
planned and as required by customers and service providers [DIDR99]. A methodology is an important part of such an attempt, as it provides a systematic and structured base for the flexible and efficient management of the development of telecommunications services, ensuring also that the roles of the participating actors are clearly identified and that their behaviours are consistent throughout the whole process of service creation.

In this chapter, in order to structure and control the service development process from requirements capture and analysis to service implementation and testing, to reduce the inherent complexity, and to ensure the thorough compatibility among the many involved tasks, a novel TINA-C conformant object-oriented service creation methodology is proposed. Its intention is to provide valuable answers to several important service engineering matters and thus facilitate the transition to a telecommunications environment where many different (enhanced) services are offered by a multiplicity of service providers to several categories of customers within an open market.

As was mentioned in Section 1.4., proposing and presenting such a service development methodology is one of the main objectives of this thesis and constitutes an important and central research contribution of it. Furthermore, as it was presented in Figure 1.1., a service development methodology is one of the main constituent parts of the proposed telecommunications service engineering framework and, as the rest of this chapter will illustrate and explain, it enables service engineering activities to satisfy Requirements 1, 2, 5, 8, 9, 11, and 14 \(^1\) (see Section 1.3.). Due to the nature of a methodology for service creation processes and also for the reader to fully understand the rational behind it, the important decisions that underpin its structure and the way that it can be applied, this chapter is extensive, as it comprises several sections (4.6., 4.8., and 4.9.) that are organised as “super sections”, remaining however strongly semantically related with each other.

Section 4.2. complements the introduction of this chapter (Section 4.1.) and explains further the necessity for a service development methodology by introducing the concept of the service life-cycle. Then, it summarises relevant work found in the service engineering literature and discusses the inefficiencies and the shortcomings of the related approaches in order to point out the significance and the novel character of the proposed methodology. The importance of this research contribution is further reinforced by reasoning about the weakness of general purpose object-oriented methodologies and RM-ODP to address efficiently the service creation process. It is thus evident that this section serves not only as a place for state of the art descriptions, but justifies also the decision to propose a service development methodology.

Section 4.3. considers briefly a number of service modelling issues and reasons about the decision to use the UML notation in the proposed methodology. An overview of this methodology is given in Section 4.4. by describing its main characteristics and by presenting its overall structure in terms of a number of phases. A deeper understanding of the proposed methodology is promoted by Section 4.5., which considers important issues that underpin the service creation process in the highly competitive environment of service provisioning.

\(^1\) In fact more requirements are satisfied, but those mentioned are satisfied in a more direct manner.
Chapter 4: A Methodology for the Development of Telematic Services

All the phases of the proposed methodology (requirements capture and analysis phase, service analysis phase, service design phase, service implementation phase, and service validation and testing phase) are presented and examined in Sections 4.6., 4.7., 4.8., 4.9., 4.10., and 4.11., focusing on their desired functionality and on the role and purpose of essential service models and artifacts, examining their dependencies and illustrating the significance of UML. It has to be noted that each methodology phase is considered as a set of interrelated activities and each of these activities is realised by a series of steps. For the most important phases of the proposed methodology significant complementary and alternative approaches that can be considered by the service developer(s) during a (possible) customisation of the methodology are identified. Finally, Section 4.12. highlights the importance of selecting appropriate software tools for supporting the proposed methodology, and Section 4.13. presents a summary of the main findings of this chapter together with some conclusions and highlights the research contributions in it.

4.2. The Need for an Integrated and Systematic Approach.

An architectural framework is by its definition an abstract entity, which consists of a set of concepts/principles and a set of guidelines and rules. For this reason, TINA-C is more descriptive rather than prescriptive, and its application can be a complex task [RANA99]. Furthermore, there seems to be no end to the emergence of new services, each requiring a new set of communications capabilities. In a world already replete with a multitude of services, the addition of new intricate services can be a daunting challenge.

Closely related to this challenge is the concept of the service life-cycle which has a central role in telecommunications service engineering. All services go through a service life-cycle, which contains descriptions of activities, in the form of an ordered collection of steps or processes, that are required to support the development, the operation, and the maintenance of a service [DELA97][MUDH94]. The logical grouping of these activities gives rise to a number of distinct sets, which are known as stages (service creation, service deployment, service operation/utilisation, and service withdrawal). Further grouping of the activities within a stage gives rise to the concept of phases (or actions). The description of the phases includes all the essential details of the activities that take place in a given stage. The service life-cycle establishes a common terminology to be used when discussing a service, and thus facilitates a common understanding of service matters.

Figure 4.1. depicts a graphical representation of a service life-cycle, which is an enriched variation of the TINA-C life-cycle model [TIN97a]. The rectangles are the phases/actions, while the ellipses are the main states that a service goes through. In this figure, the following states are identified:

- **Conceived but not planned**: The service has been conceived, but no details about its implementation are known.
- **Planned but not installed**: The service has been planned, but it does not exist in a (TINA-C) service execution environment (although it might have existed in the past).
- **Installed but not activated:** The components of the service exist in a (TINA-C) service execution environment, but the service cannot be instantiated.

- **Activated but not instantiated:** The service has the potential for being instantiated.

- **Instantiated (executing):** An instance of the service is available.

![Service Life-Cycle Diagram](image)

Figure 4.1.: The service life-cycle.

The service life-cycle of Figure 4.1. is a combination of traditional software engineering methodologies (focusing on development issues) with the activities required to operate, use, and maintain a service (focusing on post-development issues) [BERN94][DECL97]. It should be noted that it is not a strict waterfall model (it is not a strict top-down approach) of system development [JAC096]. It is possible, in each phase of the life-cycle (and especially in the service creation phases) to return to a previous phase if refinements are needed and / or requirements are added during the service development.

Among all the stages of the service life-cycle of Figure 4.1. (in TINA-C and in service engineering in general) **service creation** is one of the most abstract and general, since there are not many detailed guidelines (advisory statements) available on how to structure each of its phases. Furthermore, it is also one of the most important as it determines the efficiency with which the services will be developed and thus the success of service providers in a highly competitive market, where the main strengths of a service provider is the
diversity of its service portfolio and the ability to respond to new market demands quickly (see also Sections 1.1. and 2.6.). In such a competitive environment efficient service creation enables the fast launching of new services without compromising the overall service quality.

For this reason, considerable effort is being devoted in Europe and world-wide to the definition of advanced service creation practices. The necessity for appropriate methodological support during service development is highlighted by both OSA, which clearly states that "... only the combination of architecture and methodology will meet the required expectations on long-term efficiency of service creation, provision and management" [RACE95], and TINA-C, which emphasises its importance [BERN94]. Furthermore, both these two architectural frameworks recognise the need to (gradually) leave behind service creation practices that are market and technology dependent, as it is clear that they can't address the requirements imposed on service engineering activities by the emerging telecommunications environment (see Sections 1.1. and 1.3.).

Such practices, which are not complete methodologies, were the result of the proliferation and widespread adoption of specific technologies, and in many cases are still under use, although their problems and inefficiencies are more and more visible. More specifically, the three stage method used for defining services in ISDN implies a traditional telecommunications model where the underlying signalling protocols need to be updated for each new service [ITU88]. Furthermore, in the IN Conceptual Model (INCM), the existence of a Basic Call Process (BCP) on the Global Functional Plane (GFP) is mandatory. Short term IN is therefore constrained to address only services that can be expressed as extensions to the BCP, i.e. telephony based services [ITU93a].

The first attempt to facilitate the service creation process in a technology independent manner considering the fundamental restructuring that the telecommunications industry is undergoing, took place in the RACE project ROSA (see Section 2.5.3.), which accompanied the service architectural framework that it proposed with a number of guidelines for the development of services (the ROSA methodology) [OSHI92][RACE92]. These guidelines were expressed in a more detailed and structured manner ² in a related RACE project, forming the Eurobridge methodology [MAHO93]. In the same direction, service creation matters where then examined in the framework of IN [EUR94b][EUR97c] and by the RACE projects SCORE [RAC94a] and BOOST [RACE96], which however focused mainly on SCEs (see Section 2.6.).

Furthermore, there are several other attempts to define methodologies in the area of service engineering, although most of them are not directly related to service creation and they are not expressed at the appropriate level of detail ³. [ACT96b] proposes the Performance Evaluation Methodology (PEM) for evaluating the impact of mobility functions on a service architecture and [ACT98d] discusses the Prospect

---

² Despite their characterisation as a methodology, these guidelines were very far from being able to qualify as a service development methodology.
³ The references that are given in this paragraph are indicative and not exhaustive.
methodology for the design of multi-domain management services in an open services environment. [EUR97b] describes a user-centred method that improves the user-friendliness of telecommunications services by taking into account usability and human factors tools and techniques, and [BYER91], [BYER92] and [BYER93] present the enabling states method, which starts with an analysis of the usage context and the human end-user's goal tasks, and produces a specification of the service requirements promoting a user-centred design. [GASP96] and [OLDE98] propose an approach for the development of distributed services based on the ODP viewpoints. [EUR98b] and [DELA97] examine a methodology for the design of distributed management systems for multimedia services, and [LEW199] propose a methodology for the development of open service management systems. Finally, [STRI98] discusses a methodology for the specification and design of multi-domain management systems in a distributed telecommunication environment, and [DEPU98] presents a methodology for interoperable network and service management system design.

The most important from all these approaches is the one proposed by the ACTS project SCREEN (see also Section 2.6.), which focuses on component-based service creation [ACT97a][ACT99a]. More specifically, according to SCREEN, the construction of a service should ideally consist of the simple activity of assembling together already built "pieces" of services commonly referred to as components, which are the units of construction and reuse. For this reason, SCREEN has established a component library that enables the storage, management, and easy browsing and retrieval of reusable component representations. Thus, in an ideal setting and for relatively simple services, a service developer should be able to define the logic of an entire service, by simply selecting the appropriate components from the component library and connecting them together. For building new components and for customising the already existing ones, SCREEN provides a (kind of) service creation methodology (see also Section 2.6.). However, this methodology (which is actually a service creation practice), except from being abstract, focuses mainly on the service design phase, underestimating all the previous phases, and is heavily based on SDL (see Section 4.9.8. for the implications of using SDL), ignoring UML. Therefore, the SCREEN methodology can be considered as a loosely-coupled framework, incorporating mostly service design related guidelines and technologies, that fails to efficiently support the entire service creation process.

Additionally, important disadvantages of the SCREEN approach, that characterise in general component-based service creation practices [VERSO0][ZAVE98], are the following:

- Service developers focus on the service design underestimating or even completely ignoring activities related to the elicitation of requirements and service analysis. In this way, the possibility of having a service design that doesn't reflect (correctly or even at all) the real service requirements, and thus a telematic service that is not acceptable by its users, is increased.

- Maintaining a (not trivial) telematic service that has been constructed by assembling reusable components is a difficult task, because the semantic origin of these components is not known. Due to the lack of
Chapter 4: A Methodology for the Development of Telematic Services

(proper) requirements elicitation and service analysis results, when a feature of the service has to be changed, the components that should be modified can be found only in ad-hoc ways.

- Criteria for determining the usefulness and suitability of a reusable component when creating a telematic service are not specified. Therefore, the possibility for poor choices that can lead to unnecessary complex service designs (especially when the service developer is not experienced) is increased.

- There is significant dependence on the quality and completeness of the available library of reusable components. In general, it is very hard to foresee in advance all the different needs of services to be constructed and have the right (properly designed) reusable components available. Furthermore, there is a lack of commercially available service component libraries and the efficient management of these libraries is still a significant unsolved problem.

- Creating carefully designed reusable service components is a difficult task and can increase significantly the cost for creating a telematic service [GRA95a]. Thus, there is always a temptation to develop a service as quickly as possible even at the expense of future reusability.

From these remarks, and taking into account the inadequacy of general purpose OOAD methodologies for service creation purposes (see Section 2.3.3.), the necessity for a service development methodology is evident. Such a methodology can be combined with component-based service creation practices, offering an efficient, realistic, integrated, and systematic approach for the development of telematic services.

Finally, the need for a service development methodology is reinforced when considering ODP. As is mentioned in Section 3.2.1., ODP offers a conceptual background (a meta-framework) by providing a reference model and powerful concepts to support the specification and design of distributed systems. Moreover, the use of different viewpoints advocated by ODP, with each viewpoint specifying the original system by taking into account different concerns, allows the inherent complexity of distributed systems to be reduced and make their specification more understandable. However, ODP does not provide any methodological support to facilitate the modelling inside each viewpoint or to enable the specifications in different viewpoints to be linked. It does not offer a methodology that starts from enterprise views and proceeds through all the viewpoints establishing the appropriate models in an integrated manner.

In order to meet the challenges identified in this section, a service creation methodology is proposed with the intention to accelerate the service life-cycle so that new and enhanced services can be developed and deployed at a faster rate, in a cost effective manner, without making quality compromises in an open deregulated multi-provider telecommunications market place. This methodology, given a set of requirements that a service should meet, a set of the available service independent features (normally in the form of service components), and a target TINA-C compliant DPE wherein the service will be deployed, facilitates the design and implementation of a TINA-C compliant service, which meets the desired requirements by promoting the use of the service independent features [POLY98].

A telematic service, due to its (potentially) enhanced functionality and its inherent distributed nature, is usually overwhelmingly complex and thus it is necessary to decompose it into understandable parts / segments in order to fully comprehend its semantics and manage the complexity. These parts / segments may be represented as models which describe and abstract essential aspects of the telematic service.

Therefore, a useful activity during the development of a telematic service is to create models of the service, which organise in a concise way and communicate with accuracy the important details of the telematic service under examination. These service models should contain cohesive, strongly related elements and are usually composed of other (simpler) models or artifacts, comprising basically diagrams and documents which describe concepts and entities and reveal the relations between them.

Service models in the proposed methodology are presented using the Unified Modelling Language (UML), which is an emerging industry standard for specifying, constructing, visualising, and documenting software-intensive systems [BOOC98][EVIT00]. UML is an elegant, expressive, and flexible object-oriented modelling language, capable of supporting effective and consistent communication and of enhancing the ability to understand and act. Although UML must be applied in the context of a process, it does not define a standard development process, as it has been experienced that different organisations and problem domains require different processes. Therefore, UML consists only of a metamodel (which unifies semantics) and a notation (which offers a human rendering of these semantics). It has to be noted that, the recently proposed Unified process is actually a “customisable framework” incorporating the use of UML (see Section 2.3.3.) [JACO99].

UML was chosen as the main modelling notation for the proposed methodology because:

- It provides service designers and service developers a ready-to-use, expressive, visual modelling language so that they can develop easily and exchange efficiently meaningful and consistent service models.
- It provides extensibility and specialisation mechanisms to extend the core concepts and thus it can be tailored to the specific needs of telematic services.
- It is independent of particular programming languages and development processes.
- It provides a formal basis for understanding the modelling language.
- It encourages the growth of the object-oriented software tools market and thus the availability of appropriate tool support for the entire service creation life cycle is guaranteed.
- It supports useful higher-level development concepts, such as collaborations, frameworks, (design) patterns, and components.
- It integrates best practices from popular first-generation object-oriented analysis and design methods.

The overall telematic service model that is created when applying the proposed methodology to the development of a specific telematic service can be seen in Figure 4.2. and is composed of the service
Chapter 4: A Methodology for the Development of Telematic Services

In the proposed methodology, independent of how artifacts are organised into service models, there are influential dependencies between them. It is useful to understand these dependencies so that consistency checks and traceability can be achieved, and so that dependent artifacts are effectively used as input to creating later artifacts [LARM98]. Finally, it has to be stressed that the overall modelling approach followed by the proposed methodology is strongly influenced by the TINA-C service architecture and by the modelling guidelines and rules that it encompasses [LEVI98][TIN97a]. Therefore, telematic services are considered as software-based applications that operate on a distributed computing platform (a DPE). They are realised by a set of interacting service components (i.e. computational objects interacting via their computational interfaces), which are distributed across different network elements.

4.4. Overview of the Proposed Methodology.

Telecommunications operators need to master the complexity of service software, because of the highly diversified market demands, and consequently, because of the necessity to quickly and economically develop and introduce a broad range of new services [DECL97][MOSS95]. To achieve such an ambitious, yet strategic to the telecommunications operators goal, a service creation methodology based on the rich conceptual model of TINA-C is proposed.

A methodology is considered to be a coherent and integrated set of methods / procedures from which a coherent subset can be selected by developers for particular cases. A method / procedure is a systematic way to achieve a specific goal 4. It is implemented by techniques, and some techniques are supported, or automated, by tools [ADA98a][DELA97]. Therefore, the proposed methodology contains a conceptual model of constructs and a series of guidelines, essential to the development of telematic services, together with a set of ordered activities, realised by a number of specific steps, suggesting the direction to proceed.

---

4 Methodology is a Greek term meaning “the study of methods”. However, the term “methodology” is pragmatically well established within the field of information systems to mean the same as the term “method” [JAYA94]. Therefore, in this thesis the two terms are used interchangeably, although the term “methodology” is preferred.
Chapter 4: A Methodology for the Development of Telematic Services

A high-level or macro-level view of the proposed service creation methodology can be seen in Figure 4.3. The proposed service development process is based on an iterative and incremental, use case driven approach. An iterative service creation life cycle is adopted, which is based on successive enlargement and refinement of a telematic service through multiple service development cycles within each one the telematic service grows as it is enriched with new functions. More specifically, after the requirements capture and analysis phase, service development proceeds in a service formation phase, through a series of service development cycles. Each cycle tackles a relatively small set of service requirements, proceeding through the phases of service analysis, service design, service implementation, and service validation and testing. The telematic service grows incrementally as each cycle is completed. The main benefits of adopting such an iterative and incremental process are the following:

- The complexity is never overwhelming, because only manageable units of complexity are tackled in a service development cycle.
- Early feedback is generated, because implementation and testing occurs rapidly for a small subset of the telematic service. This feedback is an important source of information that can enhance the service analysis and the service design phases of subsequent service development cycles.
- The service developers of the service development team can gradually reapply their growing experience and maturity with the tools and the techniques / guidelines / practices proposed by the methodology in a more and more efficient and effective manner.
- Service requirements can be adjusted (if necessary), as the telematic service is under development.

These phases are actually subphases of the service formation phase.
The proposed methodology recognises the value and the importance of TINA-C in the service engineering area, and therefore it considers seriously the entire architectural framework that is presented in a set of TINA-C documents 6 (see also Section 2.5.2.). However, it does not adopt everything that is prescribed by TINA-C. On the contrary, it critically examines and evaluates the approach that is proposed by TINA-C, and selects only certain parts of it (see also Section 4.9.8.). In this way, the proposed methodology is especially influenced by the TINA-C service architecture [TIN97a]. More specifically, as can be seen in more detail in the following sections of this chapter, it adopts a set of important concepts (e.g. the concept of a session), the TINA-C business model, a number of information models, some information and computational guidelines, and the generic service components (their purpose and part of their functionality) that are proposed by TINA-C.

According to Figure 4.3. the main phases of the proposed methodology are the following:

- **Requirements capture and analysis phase**: It identifies the telematic service requirements (together with a number of roles), examines them in a critical manner, process them appropriately, and presents them in a structured way. Use cases are defined to facilitate the further refinement of the requirements.

- **Service analysis phase**: It describes the semantics of the problem domain that the telematic service is designed for. Thus, it identifies the objects that compose a service (information service objects), their types, and their relationships.

- **Service design phase**: It produces the design specifications of the telematic service under examination. Computational modelling is taking place in this phase and thus the service is described in terms of (TINA-C) computational objects interacting with each other.

- **Service implementation phase**: In this phase the pieces of the service software (computational objects) are defined and implemented in an object-oriented programming language (e.g. C++, Java), inside a TINA-C compliant DPE.

- **Service validation and testing phase**: It subjects the implemented telematic service in a variety of tests in order to ensure its correct and reliable operation.

- **Service optimisation phase**: It examines thoroughly the service code in order to improve its performance in the target DPE and thus prepares the telematic service for a successful deployment.

As can be seen from Figure 4.3., the proposed methodology is conceptually consistent with the viewpoint separation as advocated by TINA-C in accordance with the RM-ODP and uses the service life-cycle of Figure 4.1. as a roadmap. Therefore, it does not imply a waterfall model in which each activity is done once for the entire set of service requirements, because the traditional waterfall model for software development with its sequential phases is inadequate to support the development of telecommunications services. Its limi-

---

6 These documents are publicly available and can be found at the official TINA-C web site (http://www.tina-c.com).
tations lie in the possibility for causing complexity overload and delayed feedback, and in the lack of support for specific telecommunication features and such advanced concepts as reuse, the exploitation of rapid prototyping techniques for iterative requirements specification, and the evolutionary nature of service creation.

Graphical and textual notations are proposed in each phase in order to improve the readability of the related results and ensure a level of formalism sufficient to prevent any ambiguity. The proposed use of the Unified Modelling Language (UML) notation in almost all the phases of the methodology, has the advantage of making both the service description more coherent and the process of proceeding from one phase to another more natural and efficient [LARM98]. For this reason, TINA-C could also consider UML in a future version of its computing and service architectures.

In the following paragraphs, a number of important issues regarding the proposed methodology as a whole are considered and the most important phases of the methodology are examined focusing on their essential characteristics and artifacts [ADA00]. The service optimisation phase has been omitted, because it depends highly on the selected programming language and on the target DPE.

4.5. Important Methodological Considerations.

The proposed methodology specifies a service development process from the identification of service requirements through to the actual implementation of a telematic service. This service development process, except from the fact that it organises all the activities related to the creation of telematic services, it also provides a foundation for creating a manageable, repeatable, and successful service development project. It has to be noted that a service development process is also related to broader issues (service project issues), such as the complete life-cycle of a telematic service (and not only service creation, see Figure 4.1.), documentation, budget management, support and training, parallel team interaction, project management, and coordination between parties. These issues are out of the scope of the proposed methodology.

The most important phases of the proposed methodology are the requirements capture and analysis phase, and the phases of service analysis and service design which are performed inside repeated service development cycles. A useful approach is to bound each of these cycles within a rigidly fixed time frame (a time-box). All work related to a specific cycle must be accomplished in that time frame. Depending on the telematic service under examination, a time frame ranging between two weeks and two months is usually appropriate [JAC099]. Any less, and it is difficult to complete tasks; any more and the complexity becomes overwhelming and feedback is delayed [LARM98]. To succeed with a time-box schedule, it is necessary to choose the service requirements carefully, as iterative service development cycles are organised according to the identified use case requirements, i.e. the process is broken in steps as it will be discussed later (see Section 4.6.9.).

To successfully create a telematic service, a clear description of the problem and of the service requirements is necessary (i.e. what the problem is about and what the telematic service must do). It is also necessary
to have high level and detailed descriptions of a logical solution which fulfills the identified service requirements and satisfies any potential constraints. Service analysis emphasises an investigation of the problem rather than how a solution is defined, while service design emphasises the construction of a logical solution according to the service requirements. Ultimately, the service specifications produced by the service design phase will be implemented in software with the appropriate use of computing and networking infrastructure.

Service development is complex and therefore decomposition ("divide-and-conquer") is the primary strategy considered to deal with this complexity by breaking a telematic service up into manageable units. The proposed methodology applies object-oriented analysis and design, which emphasises the consideration of a problem domain and an associated logical solution from the perspective of objects (things, concepts, or entities) [ADA98c][BOOC98]. Therefore, during the (object-oriented) service analysis phase there is an emphasis on finding and describing the service concepts (service information objects) in the service domain, whose boundaries are determined, as accurately as possible, during the requirements capture and analysis phase. Furthermore, during the (object-oriented) service design phase, there is an emphasis on defining logical (software) service objects (service computational objects or service components) with attributes and methods that will ultimately be implemented in an object-oriented programming language, such as C++ or Java, inside a TINA-C compliant DPE during the service implementation phase.

It has to be stressed that the division between service analysis and service design is fuzzy. In service engineering (and sometimes in other disciplines also), analysis and design work exists on a continuum, and different practitioners of service analysis and service design classify an activity at varying points on the continuum [DECL97][POLY98]. Therefore, it is not helpful being rigid about what constitutes a service analysis versus a service design step. Nevertheless, some consistent distinction is useful in practice between investigation (service analysis) and solution (service design) because it is advantageous to have a well-defined step that emphasises an inquiry into what the problem is before examining in detail how to create a solution.

Irrespective of the exact scope of service analysis and service design, the most important ability in both these two phases is to successfully assign responsibilities to service components [DELA97][LARM98][VERSO00]. This is the most critical skill, because this activity must be performed and it has the most profound effect on the robustness, maintainability, and reusability of the resulting service components, which are the main building blocks of telematic services. Assigning responsibilities is inevitable even when a service developer hasn’t got the opportunity to perform any other service analysis or service design activities (a “rush to code” service development process). Next to assigning responsibilities, another important activity is finding suitable service objects or abstractions. Both activities are critical, but responsibility assignment tends to be the more challenging skill to master, as it is difficult to devise and apply guidelines for this activity.

Finally, the timing of the creation of some artifacts during the application of the proposed methodology needs to be discussed and examined. More specifically, certain artifacts created during the service analysis
phase, such as a service conceptual model and expanded use cases, may also be created during the requirements capture and analysis phase.

A *service conceptual model* is a graphical representation of service concepts in the service domain under examination (see also Section 4.8.3.). The amount of effort applied to the creation of a draft service conceptual model during the requirements capture and analysis phase needs to be tempered. The goal is to obtain a basic understanding of the vocabulary and concepts used in the service requirements. Therefore, a fine-grained investigation is not required, as it also increases the possibility of front-loading the investigation (complexity overload). The recommended strategy is to quickly create a rough service conceptual model where the emphasis is on finding obvious service concepts expressed in the service requirements while deferring a thorough investigation. Later, within each service development cycle, the service conceptual model will be incrementally refined and extended for the service requirements under consideration within that cycle. Another possible strategy is to completely defer creation of the service conceptual model until the start of the service development cycles. This has the advantage of deferring complexity, but the disadvantage of less up-front information, which may have been useful for achieving better comprehension.

Another artifact that the timing of its creation needs to be examined is *use cases* (see also Section 4.6.4.). During the requirements capture and analysis phase it is proposed to create all the high-level use cases, but to only rewrite the most critical and important use cases in an expanded (long) format, deferring the rest until the service development cycle in which they are examined. As with the service conceptual model, there are trade-offs in terms of the benefit of the early acquisition of information versus facing too much complexity. The advantage of investigating and writing all the detailed expanded use cases during the requirements capture and analysis phase is more information which can improve comprehension. However, the disadvantage is early complexity overload, as this investigation will generate many more detailed issues. Furthermore, the expanded use cases may not be very reliable because of incomplete or incorrect information, and because the service requirements may be under continual change. Therefore, the recommended strategy is to expend effort investigating only the most important use cases in detail during the requirements capture and analysis phase.

### 4.6. Requirements Capture and Analysis Phase.

During this phase the service developer assembles, documents, and structures the *requirements* on the service (service requirements, service needs) from the different stakeholders involved. The focus is on modelling the concepts that are visible at the service boundary and thus the service logic is viewed as a black box. The requirements capture and analysis phase is a critical phase because the correct and thorough specification of the service requirements is essential for a successful telematic service. The primary goal of this phase is to identify what functionality is really needed to include in the telematic service (i.e. which are the desired service processes) and document it in a form that is easily understandable and unambiguous.
Chapter 4: A Methodology for the Development of Telematic Services

Figure 4.4.: Requirements capture and analysis phase activities.

The activities that take place in this phase can be seen in Figure 4.4. Although this figure suggests a linear order of artifact creation, that is not strictly the case. Some artifacts may be created in parallel. This is especially true for the draft conceptual model, the glossary, the use cases, and the use case diagram(s). The dependencies between the artifacts produced during the requirements capture and analysis phase can be seen in Figure 4.5. The activities of this phase are examined in the following sections.

4.6.1. Definition of a Service Development Plan.

The requirements capture and analysis phase starts with an attempt to specify a service development plan. This plan is a structured document containing information about important matters that affect the entire service development process. The exact nature of the service development plan is the responsibility of the service developer (or the manager of the service development team) and depends on the nature of the telematic service under examination and his / her perception of the problem domain. However, there are some matters that is proposed to be addressed during this activity.

First of all, it should be stated that the service development methodology examined in this chapter will be applied. If a customised version of this methodology is intended to be used then the differences / variations from the original methodology should be clearly described and / or pointers to related documentation (if any) should be given.

The service development plan should also include information about the organisation / enterprise that requested the development of the telematic service under examination (the client). For example, it should
mention the main business objectives / targets of the client, any special requirements that the client has on the service (e.g. the use use of a particular technology), its general attitude regarding IT and telecommunications, etc. Furthermore, the plan should also specify the service developer(s) that will take part in the service development process, together with their knowledge and expertise.

![Diagram of Service Development Process](image)

**Figure 4.5.:** Requirements capture and analysis phase artifact dependencies.

The required infrastructure for the development of the service is another matter that should be addressed by the service development plan. More specifically, it should identify and propose the necessary computing and network infrastructure and the necessary software tools. Based on the availability of the proposed infrastructure, the client's requirements (if any), and the knowledge / expertise of the service developer(s), this activity should also initiate the process for selecting a programming language and a DPE.\(^7\) (see also Section 3.4.).

Finally, a schedule considering all the phases of the proposed methodology should be created during this activity. However, as a time-box approach based on use case requirements is proposed (see Section 4.5.) this task is postponed for later in this phase (see Section 4.6.9.).

It has to be noted that this activity can act as a link between the proposed service development process and broader service project issues. Therefore, the service developer, depending on his / her judgement, may include some service project issues (or references to the appropriate documents) that he / she expects to have a significant impact on the service development process (e.g. the available budget).

Taking into account the above mentioned guidelines and remarks, this activity consists mainly of the following steps:\(^8\):

**Step 1:** Specify the exact service development methodology that will be used.

**Step 2:** Provide important information about the client organisation / enterprise.

**Step 3:** Specify the service developer(s) that will participate in the service development process.

---

\(^7\) The selection process that is followed is outside of the scope of the proposed methodology. However, it is stressed that such a process is not always an easy task [ADA99c].

\(^8\) It has to be stressed that the main purpose of expressing an activity as a sequence of steps is to present the main points pertaining this activity in a concise and organised manner. Therefore, they are used to provide a quick overview of the methodology (see also Appendix B), but they can't substitute the detailed examination of the phases of the proposed methodology that takes place in the current chapter.
Step 4: Specify the required computing and network infrastructure, and the required software tools.

Step 5: Initiate a selection process for a programming language and a DPE that will be used for the implementation of the service.

Step 6: Mention and examine all other matters that affect the service development process.


The service developer after gathering enough material regarding the telematic service under examination with various means (e.g. interviews, group meetings, study of related documents, etc.) and in various forms / formats (e.g. notes, sketches / graphs, audio recordings, etc.), attempts to process this material and structure it appropriately in order to elicit from it a set of requirements.

This effort of the service developer is reflected in the preliminary investigation report. More specifically, this report contains the outcome of the process that resulted to the identification of the most important service requirements and an initial attempt to structure and process these service requirements.

The necessary steps for the creation of a preliminary investigation report are:

Step 1: Gather useful information regarding the telematic service under development.

The starting point of this step, which will also reveal the degree of commitment of the client, is to identify accurately key users and domain experts (from the client organisation) and assess their suitability, availability, and willingness for acting as sources of information. Depending on the result of this assessment and if the client commitment is substantial, several fact finding techniques can be applied. The most important and effective of them is interviewing selected people in a structured and focused manner, either individually or by conducting workshop sessions (Joint Application Development - JAD workshops) [AVIS95]. Interviews can be complemented by background reading ⁹, observation, document sampling / review, informal brainstorming group sessions, process flow analysis, the use of questionnaires, role-playing, and the application of selected knowledge acquisition methods, like Kelly grids (or repertory grids), object templates, and task / topic analysis [GRA95a].

It has to be stressed, that most of these techniques, in order to be truly effective, require an increased ability for interpersonal communication and experience in human relations. However, if these techniques are applied correctly can, not only elicit the required information, but also ensure the continual and effective user involvement throughout the development of the service, maximising in this way the chance of success. This is an important consequence, as user involvement is expected and promoted by the proposed methodology. More specifically, due to the iterative and incremental nature of the methodology, users have an active role and (can) express their opinion regarding the telematic service under development at the end of every service

⁹ Especially about the wider application domain that encompasses services like the one under development.
development cycle. This can also happen during a service development cycle with the application of rapid prototyping techniques (see Section 4.6.7).

Special attention is needed when the proposed methodology is applied for the development, not of a completely new telematic service, but for a service that is going to replace an already existing service. The new service can either be entirely different from the old one, or a variation/evolution of the old service. In these cases information should also be gathered about the existing service, emphasising the reasons (weaknesses, problems) that initiated the decision for its replacement. It is evident that any available information regarding the analysis and design of the old service will speed up considerably the application of the proposed methodology for the development of the new service and (consequently) will cause changes to the service development plan (see also Section 4.6.1.).

**Step 2:** State in a concise manner the main objective of the service.

After critically examining the material that was gathered regarding the telematic service, the main objective that the service should fulfil after its creation, is expressed in a concise manner using one or two sentences (5 to 8 lines). This statement answers the question “Why the specific telematic service is to be developed?”. It represents the minimum information that the service developer(s) should know before starting to apply the proposed methodology. Furthermore, it helps them to identify the problem domain and makes them focus on the solution that they have to provide.

**Step 3:** Analyse the constituent parts of the main service objective.

Each word or phrase (depending on the semantics) contained in the statement produced in step 2 is analysed briefly with the intention to interpret its real meaning and express it in a clear and unambiguous way. It is evident that the effectiveness of this analysis depends on the quality of the information that was gathered in step 1 and if done correctly will significantly assist the service developer(s) to understand the dimensions and the scope of the service development problem.

**Step 4:** Identify the initial service requirements.

Based on the analysis that took place in the previous step the most important service requirements are identified and presented concisely. These service requirements are termed “initial service requirements”, because they are expressed in an abstract and relatively vague way.

**Step 5:** Structure the identified initial service requirements.

The initial service requirements that were identified in step 4 are divided into categories according to the functionality that they imply. These categories were devised based on the observation that every telematic service involves at least two communicating entities (which can be considered as a group): either two humans (as in the case of videoconferencing) or a human and a “machine” (as in the case of video on
demand). These entities when communicating with each other form a session (see also Section 2.5.2.) which is necessary to be managed. They also have to interact in some way (e.g. by establishing audio / video links between them), as interaction is required in order to be able to justify the existence of a telematic service.

Furthermore, in some cases, the communicating humans need to collaborate in order to perform a common task. This collaboration can be either synchronous or asynchronous as can be seen in Figure 4.6. Synchronous distributed collaboration is more challenging because the participating humans must be present at the same time in order to perform their collaborative work, and thus support of real-time communication is necessary. However, in practice, work often switches rapidly between synchronous and asynchronous mode, and thus, usually, there is a need to support both types of collaboration.

![Collaboration Space-Time Matrix](image)

Figure 4.6.: The collaboration space-time matrix.

For these reasons, the initial service requirements are categorised in the following way:

- **Group communication support requirements:**
  - Session related requirements: Examples are the support of service creation, suspension, and shutdown.
  - Interaction requirements: Examples are the support of audio / video, text and file communication.

- **Collaboration support requirements:**
  - Synchronous requirements: Examples are the support of chat and voting.
  - Asynchronous requirements: Examples are the support of file transfer and electronic mail.

It has to be noted that the fulfilment of the interaction requirements can support implicitly, unintentionally, and (usually) inefficiently the collaboration between the users of a service. To emphasise the importance of collaboration and to support it efficiently, collaboration support requirements are identified separately from the group communication support requirements, although they have a relation with them. Collaboration support requirements are usually more complex from the other requirements as they presuppose the fulfilment of some (or all) of the group communication requirements.

Both steps 4 and 5 represent an attempt to extract the main service requirements from the information that was gathered in step 1 and structure them in a simple, but practical and intuitive manner that will facilitate their further exploitation from the service developer(s).
Chapter 4: A Methodology for the Development of Telematic Services

**Step 6:** Consider the TINA-C Business Model.

One of the most important tasks that the service developer has to perform in this phase is the identification of the independent actors/entities, which are involved (by their collaboration) in the operation of the service within and across business administrative domain boundaries (actors of the service). These actors correspond to roles modelling a well-defined grouping of functionality under control of a specific stakeholder [WIND98]. Due to its virtual nature, a role can be carried by one or more persons at a time. This initial task is important because the gathered service requirements are further structured in the service analysis and service design phases by considering the identified roles and the relationships between them.

![Figure 4.7: The TINA-C Business Model (roles and relationships).](image)

A role can be either *generic* or *specific*. The main generic (business) roles are specified by the TINA-C Business Model [TIN97b], which can be seen in Figure 4.7., and each of them (consumer, retailer, broker, third party service provider, connectivity provider) is considered to be performed by a separate business administrative domain. Each generic role corresponds to one or more specific roles. One or more persons (actors) can carry out a specific role (role-holders). There are two kinds of specific roles; namely, service independent roles, and service dependent roles [TIRO98]. It has to be noted that the service dependent specific roles, in combination with the information gathered in step 1, reveal in a relatively straightforward way the actors of the telematic service. For example, a "teacher" represents a service dependent specific role in an interactive teletraining service and "teacher Jones" is an actor of that service. Common service independent specific roles that can be associated with the consumer generic role are the roles of the end-user, the anonymous user, and the subscriber. The TINA-C Business Model specifies also the following (business) relationships (shown in Figure 4.7.): Retailer (Ret), Broker (Bkr), Third Party (3pty), Retailer-to-Retailer (RtR), Connectivity Service (ConS), and Terminal Connection (TCon) [PROZ97][TIN97a][TIN98b].

From Figure 4.7, it can easily be deduced that business administrative domains interact with each other in order to provide a specific telematic service. These interactions can be access interactions and usage interactions (service usage and communications usage interaction), and correspond to the different types of sessions that are established between them (see Section 2.5.2.). Therefore access interactions take place during an access session, service usage interactions take place during a service session, and communications
usage interactions take place during a communications session. Furthermore, the interactions between business administrative domains are also mapped one-to-one to inter-domain reference points, which are defined in relation to the (business) relationships that they support (e.g. the Retailer reference point corresponds to the Retailer relationship). Therefore, the functionality offered by a reference point can be divided into an access part and a (primary and ancillary) usage part [INOU98][TIN97a][TIN97b].

Finally, business administrative domains are able to take different session roles during access and usage interactions. More specifically, according to TINA-C a domain can take a user / party role (in order to request and use services), a provider role (in order to supply services), or a peer role (in order to alternate between party and provider). The session roles are dynamically performed by domains and so a domain can change roles as frequently as is required by the interactions that take place. Thus, a domain can have many session roles. In any case, session roles are always complementary and come in pairs [BERNOO][TIN97a].

Taking into account the previous discussion and the initial service requirements, the following actions take place when considering the TINA-C business model during this step:

- Identify the generic (business) roles that the actors of the service can take.
- Construct a business model for the service by adapting the model proposed by TINA-C.
- Identify the service independent specific role(s) that the actors of the service can take.
- Identify the service dependent specific role(s) that the actors of the service can take.
- Identify the actors of the service.
- Specify the type of interactions (access, service usage, communications usage) that should be considered between each pair of generic (business) roles.
- Identify the session roles that the actors of the service can take.

4.6.3. Definition of Service Requirements.

The definition of the service requirements involves the identification of (desirable) service functions and (desirable) service attributes based on the examination of the information included in the preliminary investigation report (see Section 4.6.2.) using the initial service requirements as a guide.

Service functions represent what the telematic service under examination is supposed to do. They are identified, expressed in a concise way, and listed in logical cohesive groupings according to the categories used for the structuring of the initial service requirements (i.e. group communication support functions, collaboration support functions, and the appropriate subcategories). To verify that some action <X> is indeed a service function, it should make sense in the following sentence:

The telematic service should do <X>

10 The term "user" is used for access, while the term "party" is used for usage as in TINA-C [TIN97a].
Service functions should be characterised by specifying their type in order to be able to prioritise them and to identify those that might otherwise be taken for granted (but which consume time and other resources).

The proposed service function types are:

- **Evident**: It should take place and the actors of the service should be cognisant that it is performed.
- **Hidden**: The service function should take place without being visible to the actors of the service. This is usually desirable for underlying service functionality, which is not associated with a user interface. Hidden service functions are often (incorrectly) missed during the requirements capture and analysis phase.
- **Optional**: Performing the service function does not significantly affect cost or other service functions.

In contrast with service functions, **service attributes** are non-functional telematic service qualities and characteristics, that are often confused with service functions. Examples are ease of use, response time, interface metaphor, security level, error tolerance, etc. Service attributes may penetrate all the service functions or be specific to a particular service function or group of service functions. They can be associated with a set of attribute details, which tend to be discrete, fuzzy, symbolic values of the service attribute, such as:

\[
\text{response time} = \{\text{psychologically appropriate}\}
\]

\[
\text{interface metaphor} = \{\text{graphical, colourful, forms-based}\}
\]

On the other hand, some service attributes can be associated with attribute boundary constraints, which are mandatory boundary conditions, usually expressed with a numeric range of values for an attribute, such as:

\[
\text{response time} = \{\text{five seconds maximum}\}
\]

It is useful to describe clearly the way that the service functions are related to the service attributes by using a suitably structured table. In this table, service attributes may be characterised as either "required" or "optional".

Taking into account the above mentioned guidelines and remarks, this activity consists mainly of the following steps:

**Step 1**: Identify the (desirable) service functions.

**Step 2**: Structure the identified service functions by categorising them.

**Step 3**: Specify the type of the service functions.

**Step 4**: Identify the (desirable) service properties (attributes).

**Step 5**: Specify the attribute details or the boundary conditions for the identified service attributes.

**Step 6**: Specify the type of the service attributes.

**Step 7**: Relate service functions and service attributes.

**Step 8**: Include all necessary information in the appropriate document (specifications of service requirements).
4.6.4. Definition of Use Cases.

In order to improve the understanding of the service requirements, use cases are created. Use cases are textual narrative descriptions of service domain processes. They describe the sequence of events generated by an actor using a telematic service to complete a specific service process. They are stories or cases of using a telematic service. They are dependent on having at least a partial understanding of the service requirements, as were identified in the previous activity (Section 4.6.3.), because they usually involve more than one service function. Use cases are not exactly requirements or functional specifications by themselves, but they illustrate and imply service requirements in the stories they present [DANO97][EVIT00].

In the general case, a use case describes interaction with a “system”. In the proposed methodology the system is the telematic service under examination. Before starting to define the use cases, it is important to specify the boundaries of the telematic service, in order to better understand what the responsibilities of the telematic service are, and to identify what is external to the service and what is internal to it. The external environment is represented only by (external) actors, which stimulate the service with input events and / or receive something from it. The choice of the telematic service boundaries is influenced by the specific service needs, as were expressed in Sections 4.6.2. and 4.6.3.

For the identification of use cases from service requirements two alternative, but equivalent, methods are proposed. One of them is actor-based, and the other is event-based.

The actor-based method consists of the following actions:

1. Identify the (external) actors related to the telematic service.
2. For each actor, identify the service processes that he / she initiates or participates in and form use cases.

The event-based method consists of the following actions:

1. Identify the external events that a telematic service must respond to.
2. Relate these events to (external) actors and form use cases.

Both methods for identifying use cases involve critical thinking, brainstorming, and reviewing all the information gathered so far in this phase regarding the service. A common error in identifying use cases is to represent individual service functions, operations, or transactions as use cases. A use case is a relatively large end-to-end service process that typically includes many service functions; it is not normally an individual step or activity in a service process.

---

11 Use cases are not actually an object-oriented analysis artifact, but they are an important and widely practised early step in many general purpose OOAD methods [CONS97][JACO99], and they are also part of the UML [BOOC97].

12 These are processes that the telematic service should support.

13 The actors of the service that are involved with the provision and the management of the service (e.g. service providers) are usually considered to be internal. The internal actors and their actions are not further examined.
Use cases, after their identification, may be expressed with varying degrees of detail and commitment to design decisions. Therefore, the same use case may be written in different formats, with different levels of detail. There are two basic formats that a use case can take, leading respectively to high-level use cases and to expanded use cases.

A high-level use case describes a service process very briefly, usually in two or three sentences. It is useful to create this type of use case during the requirements capture and analysis phase in order to quickly understand the degree of complexity and functionality in a telematic service and obtain a complete view of the major service processes. High-level use cases are very terse and vague on design decisions. On the other hand, an expanded use case describes a service process in more detail than a high-level one as it has a section which describes the step-by-step events. During the requirements capture and analysis phase, it is useful to write the most important and influential use cases in the expanded format, and defer the less important ones until the service development cycle in which they are being tackled (see also Section 4.5.).

Furthermore, use cases can be either essential or real. More specifically, essential use cases are expanded use cases that are expressed in a form that remains relatively free of technology and implementation details, as design decisions (especially those related to the user interface) are deferred and abstracted [CONS97]. An essential use case describes a service process in terms of its essential activities and motivation. High-level use cases are always essential in nature, due to their brevity and abstraction. It is desirable to create essential use cases during the requirements capture and analysis phase in order to more fully understand the scope of the problem and the service functions required. They are advantageous because they reveal the essence and the fundamental motivation of the service process that they describe. They also tend to be correct for a long period of time, and thus easily customisable and reusable, since they exclude design decisions [DAN097].

In contrast, a real use case concretely describes a service process in terms of its real current design and committed to specific technologies (e.g. when a user interface is involved, real use cases often show screen samples or screen shots and discuss interaction with the widgets). Ideally, real use cases are created during the service design phase of a service development cycle, since they are a design artifact. If early design decisions regarding the user interface are expected, then real use cases must be created during the requirements capture and analysis phase. Otherwise, it is undesirable to create real use cases so early, because of the premature commitment to a specific service design and the overwhelming complexity involved 14.

The structure / format of a high-level use case is proposed to be the following (see also Figure 5.2.):

<table>
<thead>
<tr>
<th><strong>Use Case:</strong></th>
<th>Name of the use case under examination. It should start with a verb in order to emphasise that it refers to a service process.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actors:</strong></td>
<td>List of participating actors, indicating who initiates the use case (initiating actor who generates the starting stimulus).</td>
</tr>
<tr>
<td><strong>Type:</strong></td>
<td>The type of the use case. This can be primary (for use cases representing major common service processes), secondary (for use cases representing minor or rare service processes), or optional (for use cases representing service processes that is not necessary to be tackled).</td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td>The objective of the use case.</td>
</tr>
</tbody>
</table>

14 This increases the possibility of delay and decreases flexibility in the subsequent phases of the methodology.
Chapter 4: A Methodology for the Development of Telematic Services

It has to be noted that UML does not specify a rigid format for use cases. Therefore, the proposed format may be altered to meet the exact needs and scope of documentation and assure clarity of communication in a specific service development project.

As expanded use cases aim to offer a deeper understanding of the service requirements, they have to be presented in a "conversational" style between the actors and the telematic service. Therefore, the structure / format of an expanded use case is proposed to be the following (see also, for example, Figure 5.3.):

<table>
<thead>
<tr>
<th>Use Case:</th>
<th>Name of the use case under examination.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>List of participating actors, indicating the initiating actor.</td>
</tr>
<tr>
<td>Purpose:</td>
<td>Intention of the use case.</td>
</tr>
<tr>
<td>Overview:</td>
<td>Repetition of the high-level use case description, or a similar summary.</td>
</tr>
<tr>
<td>Type:</td>
<td>1. Primary, secondary or optional. 2. Essential or real.</td>
</tr>
<tr>
<td>Cross References:</td>
<td>Related service functions and use cases. This section provides an important &quot;link&quot; in terms of traceability between the artifacts, as it can be used to verify that all service functions have been allocated to use cases.</td>
</tr>
</tbody>
</table>

**Typical Course of Events:**

<table>
<thead>
<tr>
<th>Actor Action</th>
<th>Service Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numbered actions of the actors.</td>
<td>Numbered descriptions of service responses.</td>
</tr>
</tbody>
</table>

An expanded use case should start with the following schema: "1. This use case begins when <Actor> <initiates an event>". This encourages a clear identification of the initiating actor and the event.

**Alternative Courses:**

- Alternatives that may arise at a specific event. Description of the exception.

The middle section of an expanded use case (the "Typical Course of Events") is the most important as it describes in detail the required interaction between the actors and the telematic service. A critical aspect of this section is that it describes the most common sequence of activities needed for the successful completion of a service process. Alternative situations or exceptions that may arise with respect to the typical course are included in the final section of an expanded use case (the "Alternative Courses"). If complex, these alternatives may themselves be expanded into their own use cases.

After the identification of use cases and concurrently with their specification, a use case diagram is proposed to be created. It illustrates a set of use cases for a telematic service, the actors involved, and the relation between the actors and the use cases. The purpose of this diagram is to present a kind of context diagram by which one can quickly understand the external actors of a telematic service and the key ways in which they use it. In a typical use case diagram use cases are illustrated in ovals, while actors are represented by stick figures. There are lines of communication between the use cases and the actors. The arrows indicate flow of information or stimulus. High level and essential use cases and use case diagrams are members of the service analysis use case model (see Sections 4.8.1. and 4.8.2. and Figure 4.17.).

Taking into account the above mentioned guidelines and remarks, the definition of use cases consists mainly of the following steps:

**Step 1:** Specify the boundaries of the telematic service.

**Step 2:** Identify use cases using either the actor-based or the event-based method.
Step 3: Write all use cases in the high-level format.

Step 4: Draw a (draft) use case diagram.

Step 5: Write the most critical and influential use cases in the expanded essential format. Defer writing the expanded essential form of less critical use cases until the service development cycles in which they will be tackled.

Step 6: Create some real use cases only if concrete descriptions significantly aid comprehension or the client demands the specification of service processes in this fashion.

4.6.5. Definition of a Draft Service Conceptual Model.

The service conceptual model is an artifact and the timing of its creation should be carefully considered (see also Section 4.5.). During this activity a rough (preliminary) service conceptual model is proposed to be defined as an aid to understanding the vocabulary of the service domain, especially as it is used in the service specifications and in the use cases. More detailed versions of this service conceptual model will be created during the service analysis phase of the subsequent service development cycles (see Section 4.8.3.).

4.6.6. Recording Terms in the Glossary.

The glossary is a simple document that defines terms, related to the telematic service under examination, that are used in the service development process. At the very least, a glossary (or model dictionary) lists and defines all the terms that require clarification in order to improve communication and reduce the risk of misunderstanding. Consistent meaning and a shared understanding of terms is extremely important during service development, especially when many service developers are involved.

The glossary is originally created during the requirements capture and analysis phase as terms are generated, and is continually refined within each service development cycle as new terms are encountered (see Section 4.8.6.). It is usually made in parallel with the specification of the service requirements, the definition of use cases, and the definition of a draft service conceptual model. Maintaining the glossary is an ongoing activity throughout the proposed methodology. It has to be noted that the glossary is also a useful document within which to record service domain rules / assumptions, constraints, and other similar information. This kind of information can also be recorded in use cases and service contracts, and by attaching appropriate notes to elements (such as service concepts) to which a rule or a constraint applies.

4.6.7. Considering the Application of Rapid Prototyping.

A prototype is an approximation of a system that exhibits the essential features of the final version of that system. Prototyping is used in many areas, including engineering and information systems development. In

---

15 Ideally real use cases should be deferred until the service design phase of the service development cycles in which they will be tackled.
Chapter 4: A Methodology for the Development of Telematic Services

the latter area, prototyping (which is also known as rapid prototyping when suitable tools are used to support the rapid creation of design elements) addresses the complaint that users only saw their information system at implementation time when it was too late to make changes. In that case, the risk of failure because of user dissatisfaction, including outright user rejection, is significant [AVIS95]. On the other hand, by implementing a prototype first, the analyst can show the users something tangible, before finally committing the user to the new design. Furthermore, it may only be by using this approach that the users discover exactly what they want from the system, as well as what is feasible [ADA98a].

The proposed methodology is inherently compatible with the rapid prototyping approach due to its iterative and incremental nature (see Section 4.11.). Therefore, the consideration of rapid prototyping in this activity refers to its application regarding user interface matters in order to promote user involvement, not only between the service development cycles, but also inside them (see also Section 4.6.2.).

The fundamental purpose of constructing a prototype of the service user interface early in every service development cycle (e.g. during the refinement of the service requirements, see Figure 4.3.) is to show to the users of the service how the service is expected to behave in front of them, how it will look-like, and how they can interact with the service. From the feedback of the users, the service developer can understand better and specify more accurately the interactions between the users and the service, improving at the same time his / her comprehension of the use cases. In this way, uncertainties about the service requirements can be resolved early and the short-term additional costs for prototyping can result in long-term savings.

When building prototypes of the service user interface, the service developer has to find out the fundamental aspects that have to be represented and decide about how he / she is going to implement the prototype. Recommended technologies for Graphical User Interface (GUI) prototyping, that reduce the costs associated with prototyping and increase the ease and speed with which prototypes can be created and modified, are Visual Basic, Java, Tcl/Tk, VisualTk and HTML [ACT99a]. The prototypes produced in this manner can either be discarded after their evaluation ("throw-away" prototypes) or used as the basis for the actual implementation of the service user interface in activity 4 of the service implementation phase (see Section 4.10.1.) of the current and the subsequent service development cycles (exploratory prototyping).


As was stated in Section 4.2. and as it is evident throughout Chapter 4, the proposed methodology uses a number of concepts and ideas from the TINA-C service architecture. Additionally, it adopts the way that TINA-C considers telematic services. Based on these influences and taking also into account common architectural issues related to distributed information systems, such as three-tier and multi-tier client / server architectures [LEWA98], a typical telematic service is proposed to be architecturally designed in terms of the following (service architecture) layers (see Figure 4.8.):
• **Presentation Layer:** It is responsible for issues regarding the Graphical User Interface (GUI) of the service, report generating, etc.

• **Service Logic Layer:** Its functionality is represented by:
  - **Service Domain Objects:** These are objects representing service domain concepts (e.g. user agent, service session, etc.) that fulfil the service requirements. They are the primary focus of the proposed methodology.
  - **Ancillary / Supporting Objects:** These are non service domain objects that provide supporting functionality (such as naming capabilities, interfacing with a database, etc.) to the service domain objects.

• **Resource Layer:** It abstracts over the computing and network infrastructure, which is necessary for the realisation of the service. It is also responsible for the persistent storage mechanism (such as an object-oriented or relational database) that may be required by a telematic service.

![Service architecture layers of a typical telematic service.](image)

The proposed methodology primarily emphasises the service domain objects, allocating responsibilities to them to fulfil the service requirements. In this design approach, the presentation layer has very little responsibility and thus it is considered to be “thin”. Windows do not contain code that handles service logic or (any kind of) processing. Rather, task requests are forwarded on to the service logic layer.

This activity consists mainly of the following steps:

**Step 1:** Identify the service architecture layers.

**Step 2:** Comment on their importance for the telematic service under examination.

**Step 3:** Record the necessary information in the service development plan.

### 4.6.9. Refining the Service Development Plan.

During this activity the contents of the service development plan that was created at the beginning of the requirements capture and analysis phase (see Section 4.6.1.) are being reconsidered taking into account all
the artifacts produced so far. The most important new element of the refined service development plan is a schedule covering all the phases of the proposed methodology. The creation of this schedule is based on the allocation of use cases to service development cycles.

Assuming all desired artifacts have been generated (e.g. specifications of the service requirements, use cases) during the requirements capture and analysis phase, the next step is the transition to the first development cycle of the service formation phase (see Figure 4.3.) in order to start implementing the telematic service. The fulfilment of the use cases is distributed over multiple service development cycles, which are organised around the use case requirements. More specifically, a service development cycle is assigned to implement one or more use cases, or simplified versions of use cases, when the complete use case is too complex to tackle in one cycle.

Before deciding about such matters, use cases need to be ranked, and those with a high ranking need to be tackled in early service development cycles. The proposed strategy is to first select use cases that significantly influence the core functionality of the telematic service. The ranking scheme that is used consists of a number of criteria that are considered for each use case and it can apply either a simple fuzzy classification (such as high-medium-low) or arithmetic scores (possibly augmented with weighting). The specification of an exact ranking scheme is the responsibility of the service developer(s). However, as a guideline it is proposed to increase the ranking of a use case when it includes complex or time-critical service functions, has a significant impact on the overall service design (e.g. when adding many service classes), represents primary service processes, can offer significant information and insight regarding the overall service design with relatively small effort, involves significant research or the use of new technology, or is considered important by the client or the users of the service.

In the proposed methodology a time-box scheduling strategy is assumed, in which a service development cycle is set to a fixed time limit. Therefore, whenever a use case is assigned to a service development cycle, it is necessary to estimate if the entire use case can be tackled within the limited time-box of a cycle, or if the work of the use case needs to be distributed across multiple service development cycles. In this situation, the use case is redefined in terms of several simplified use case versions, each of which encompasses more and more of the requirements of the complete use case. Each use case version is limited to cover what is estimated to be a reasonable amount of work within the confines of the service development cycle time-box. These versions are then distributed over a series of service development cycles, along (possibly) with other use cases.

Finally, it has to be noted that virtually all telematic services have an initialising ("Start Up") use case. Although this use case may not acquire a high rank by the ranking scheme applied, it is necessary to tackle at least a simplified version of it in the first service development cycle so that the initialisation assumed by other use cases is provided. Within each service development cycle, the "Start Up" use case is incrementally developed to satisfy the start up needs of the appropriate use cases.
Taking into account the above mentioned guidelines and remarks, this activity consists mainly of the following steps:

**Step 1:** Reconsider the contents of the service development cycles.

**Step 2:** Devise an appropriate ranking scheme.

**Step 3:** Rank the use cases according to this ranking scheme.

**Step 4:** Allocate the use cases to service development cycles.

**Step 5:** Create a schedule for the proposed methodology.

### 4.6.10. Complementary and Alternative Approaches.

In the requirements capture and analysis phase, the application of usability engineering principles can complement use cases and describe the expectations of the users from the telematic service under development more accurately, improving also their interaction with it [BYER91][BYER92][BYER93]. Furthermore, scenarios can be used for the structuring of the service requirements [SUTC98], and the WinWin technique [HORO99] can aid in the capture, negotiation, and recording of the service requirements.

### 4.7. Starting a Service Development Cycle.

After the completion of the requirements capture and analysis phase, the *service formation phase* starts, within which iterative service development cycles occur (see Figure 4.3.).

![Service analysis phase activities.

Figure 4.9.: Service analysis phase activities.](image)

The initial activities within each service development cycle (requirements refinement, artefacts synchronisation) are mostly project management related. In the general case, the refinement of the service
requirements is followed by (or more likely occurring in parallel with) a synchronisation of documentation (e.g. various types of artifacts) from the last cycle with the actual state of the service code, because during the service implementation phase of the last service development cycle the design artifacts and the service code invariably diverge (see also Section 4.11). After these initial activities the service analysis phase starts, within which the use cases of the current service development cycle are examined in detail.

4.8. Service Analysis Phase.

The aim of this phase is to determine the functionality needed for satisfying the service requirements that were identified in the previous phase and to define the software architecture of the service implementation. For this reason, the focal point shifts from the service boundary to the internal service structure [ADA98a].

The activities that take place in this phase can be seen in Figure 4.9. As with the requirements capture and analysis phase artifacts, the linear order that may be inferred from this figure is not strictly the case, as some artifacts may be created in parallel (e.g. the service conceptual model and the glossary). The dependencies between the artifacts produced during the service analysis phase can be seen in Figure 4.10. The activities of this phase are examined in the following sections.

4.8.1. Definition of Essential Use Cases.

As was suggested in Section 4.6.4., essential use cases (which are expressed in an expanded format) is desirable (but not necessary) to be created during the requirements capture and analysis phase. However, if the service developer decides (for some reasons) not to follow this suggestion, then the essential use cases that are examined in the current service development cycle should be defined during this (first) activity of the service analysis phase.

4.8.2. Refining the Use Case Diagram(s).

The use case diagram(s) that were created in the requirements capture and analysis phase (Section 4.6.4.) depict only the absolutely necessary information regarding the identified use cases and therefore they have a draft nature. During this activity, more details are added to the use case diagram(s) [MAR98b].
specifically, use cases are related and their relationships are illustrated in the use case diagram. In an “extends” relationship the behaviour defined for a use case can follow the behaviour defined for another use case. If one use case initiates the behaviour of another use case, it is said to “use” the second use case and the two use cases are said to be in a “uses” relationship. The use case that uses the behaviour of another should indicate the connection with the “initiate” keyword in the use case text.

4.8.3. Definition of Service Conceptual Models.

The service analysis phase is the first phase of the service creation process where the telematic service is decomposed into its constituent parts (service information objects or service concepts), with the appropriate relationships among them, in an attempt to gain an overall understanding of the service. The resulting (main) service conceptual model, which is the most important artifact that is created during the service analysis phase, represents a restatement, in a graphical notation, of the problem statement, as it was expressed in the previous phase.

It involves identifying a rich set of service concepts regarding the telematic service under examination by investigating the service domain and by analysing the essential use cases (see Section 4.8.1.). Therefore, it describes what the service is in terms of interesting and meaningful (to the service developer) entities / concepts that constitute it and couplings / associations between them [FOW96a][MART95]. These couplings define relationships between two service Information Object (IO) classes. Each service IO participating in a relationship has a role in that relationship. Each role possesses a certain multiplicity that quantifies the number of instances of a service IO class having a role that may participate in a relationship with each instance of the service IO class having the other role [GASP96]. In UML, a service conceptual model can be illustrated with a set of static structure diagrams in which no operations are defined.

It has to be stressed that a service conceptual model is a representation of real-world concepts or actual things, and not a representation of software components (software entities). A good service conceptual model captures the essential abstractions and information required to understand the service domain in the context of the current service requirements, and aids service developers in understanding the service domain (its concepts, terminology, and relationships). However, there is no such thing as a single correct service conceptual model. All service conceptual models are approximations of the service domain under examination [BOOC98][LARM98].

The main service conceptual model is accompanied by a set of ancillary service conceptual models. These models are derived by (and correspond to) a number of generic information models deduced from the TINA-C service architecture and complement semantically the main service conceptual model with useful session related concepts and structures. More specifically, the ancillary models refer to the modelling of session roles, (TINA-C) sessions, access and service sessions, and to the classification of access and service sessions.
This activity of the service analysis phase consists mainly of the following steps:

**Step 1:** Identify the service concepts.

**Step 2:** Identify associations between the service concepts.

**Step 3:** Identify attributes of the service concepts.

**Step 4:** Draw the main service conceptual model.

**Step 5:** Specify the ancillary service conceptual models.

These steps are examined in detail in the following paragraphs grouped into two sections: one related to the main service conceptual model (steps 1-4), and one related to the ancillary service conceptual models (step 5).

### 4.8.3.1. Specification of the Main Service Conceptual Model.

The main service conceptual model is specified by the following steps:

**Step 1:** Identify the service concepts.

A central task when creating a service conceptual model is the identification of the service concepts. It has to be noted as a general guideline that it is better to overspecify a service conceptual model with many fine-grained concepts than to underspecify it. However, it is possible that some service concepts will be missed during the initial attempt to identify service concepts. These concepts will be discovered later during the consideration of associations or attributes, or during the service design phase. When found, the initially missed service concepts should be added to the service conceptual model.

Two techniques are proposed for the identification of service concepts. The first is based on the use of a service concept category list, which contains categories that are usually worth considering, though not in any particular order of importance. In this case, the creation of a service conceptual model begins by making a list of candidate service concepts from the service concept category list. An indicative service concept category list has the following categories: session related, interaction related and collaboration related service concepts (see Section 4.6.2., step 5), profiles of external actors, service (dependent / independent) specific roles (see Section 4.6.2., step 6), physical or tangible objects, specifications or descriptions, processes, rules and policies, abstract noun concepts, events, containers of other "items", catalogues, transactions, etc.

Another useful technique for the identification of service concepts is to consider the noun phrases in the text of the expanded use cases as candidate service concepts or attributes [GRA95a]. However, this noun phrase identification technique must be carefully applied, as a mechanical noun-to-concept mapping isn't possible, and words in natural languages are ambiguous. Therefore, a weakness of this approach originates from the inherent imprecision of natural languages (e.g. different noun phrases may represent the same concept or attribute). Nevertheless, this technique is recommended to be used in combination with the service concept category list technique.
Chapter 4: A Methodology for the Development of Telematic Services

Some of the identified service concepts are possible (and useful) if organised into a generalisation-specialisation type hierarchy (or simply type hierarchy) in which the supertype represents a more general concept and the subtypes more specialised ones. Identifying such commonality among service concepts is of value in a service conceptual model because the respective type hierarchies facilitate the understanding of service concepts in more general, refined, and abstract terms. Furthermore, they can lead to economy of expression, improved comprehension, and a reduction in repeated information.

Taking into account the previous discussion, the following actions take place when identifying the service concepts during this step:

- Apply the service concept category list technique.
- Apply the noun phrase identification technique.
- Draw an initial service conceptual model by representing graphically only the service concepts.
- Identify useful type hierarchies (optionally).

Step 2: Identify associations between the service concepts.

After identifying the service concepts, it is also necessary to identify those associations of the service concepts that are needed to satisfy the information requirements of the current use case(s) under development and which aid the comprehension of the service conceptual model. The associations that should be considered in order to be included in a service conceptual model are the associations for which the service requirements suggest or imply that knowledge of the relationship that they present needs to be preserved for some duration ("need-to-know" associations) or are otherwise strongly suggested in the service developer's perception of the problem domain.

Furthermore, it is usually worth considering, taking into account the identified service concepts and the current use cases, a common associations list which contains some common categories of associations. An indicative common associations list (between service concepts, for example, A and B) has the following associations: A supports B, A participates in B, A is a member of B, A contains B (either physically or logically), A is a physical or logical part of B, A is characterised by B, A is specified by B, A is a description of B, A is known / logged / recorded / reported / captured in B, A maintains B, A uses or manages B, A communicates with B, A is owned by B, etc. It has to be noted that it is generally undesirable to overwhelm the service conceptual model with associations that are derivable, not strongly required and which do not increase understanding. Too many associations tend to confuse a service conceptual model rather than clarify it. Their discovery can be time-consuming and with marginal benefit. However, associations should also ensure that an essential understanding of the important service concepts is offered by the service conceptual model.

---

16 Generally, an association is a relationship between concepts that indicates some meaningful and interesting connection. All associations are inherently bi-directional.
Chapter 4: A Methodology for the Development of Telematic Services

During the service analysis phase, an association is not a statement with software implications in the service code. It is just a statement that a relationship is meaningful in a purely analytical sense. Many of these relationships will be implemented in the service code as paths of navigation and visibility, but their presence in the service conceptual model does not require their implementation. Therefore, a service conceptual model may contain associations that are not necessary during programming. Conversely, if during the service implementation phase, associations are discovered that need to be implemented but were missed during the service analysis phase, then the service conceptual model should be updated to reflect these discoveries (see Section 4.10.).

Taking into account the previous discussion, the following actions take place when identifying associations between the service concepts during this step:

- Find the need-to-know associations.
- Consider the common associations list.
- Consider aggregation (composite or shared), derived, qualified, and recursive or reflexive associations, based on their value in improving understanding of the service domain (optional action).
- Select the desirable associations that will be included in the main service conceptual model.

Step 3: Identify attributes of the service concepts.

A service conceptual model should include all the attributes of the identified service concepts for which the service requirements suggest or imply a need to remember information. These attributes should preferably be simple attributes or pure data values. Caution is needed to avoid modelling a (complex) service concept as an attribute or relating two service concepts with an attribute instead of an association. Furthermore, it is stressed that some (not obvious) attributes may not be identified during service analysis. However, when they are discovered (either during the service design or during the service implementation) they should be added to the service conceptual model.

Step 4: Draw the main service conceptual model.

The main service conceptual model is formed by adding the identified type hierarchies, associations and attributes to the initial service conceptual model. It has to be noted that a verb phrase should be used for naming an association, in such a way that the association’s name together with the names of the service concepts that it relates create a sequence that is readable and meaningful 17. Finally, associative types, which are related to an association between two service concepts, should be added to the service conceptual model if considered to be necessary. Usually, associative types are useful when two service concepts are linked with a many-to-many association and there is important information related to it.

---

17 The proposed direction to read such a sequence is left to right or top to bottom. This is not imposed by UML, but it is a relatively common convention. Furthermore, the use of a direction arrow can clearly indicate the direction to read similar sequences (see, for example, Figure 5.7.).

As was stated in Section 4.4., the proposed methodology considers the TINA-C service architecture in a critical manner with the intention to extract from it useful concepts and guidelines / techniques. Taking into account this attitude, a number of generic TINA-C information models were formed, by considering parts of the documentation that is available by TINA-C (see also Section 2.5.2.). The customisation of these information models according to the service requirements results in the ancillary service conceptual models, which in combination with the main service conceptual model, describe the semantics of the service domain under examination in a clear, concise, and unambiguous way.

The generic TINA-C information models that are the basis of the ancillary service conceptual models are briefly described in the following paragraphs. It is stressed that these information models have a pure conceptual / analytical role and they are considered by the proposed methodology released from most of the details that the TINA-C service architecture associates with them (e.g. feature sets, see Section 4.9.8.).

The session related information models

The session information model is depicted in Figure 4.11.(a) and is valid for all session related telematic services. As a session related service may extend over multiple business administrative domains, it is intuitive and useful to model the service as an aggregation of one or more domain services, where each domain service represents a part of a service confined to a single domain. Just as a session represents an instance of a service, a domain session represents an instance of a domain service. Domain sessions may interact to establish services extending over multiple domains.

![Figure 4.11.: Important session related information models (TINA-C):](image)

(a) The session information model, (b) The session role information model.

Therefore, according to Figure 4.11.(a), a session related service is an aggregation of other session related services and domain services. Each domain service is instantiated by one domain session, which is established
in and managed by its associated domain. A domain session may be bound to another domain session via a
domain session binding, which represents the dynamic information used to link the two domain sessions.

As can be seen in the session role information model which is depicted in Figure 4.11.(b), business
administrative domains are able to take different session roles during the access and usage interactions that
occur between them. The same figure shows also the classification of the session role concept (see also
Section 4.6.2.). It has to be noted that only the “leaf” roles (at the bottom of the figure) can be taken by a
business administrative domain as the other roles are abstract roles.

The access session related information models

Taking into account the generic TINA-C session related information models of Figure 4.11. and the
different types of sessions that can be established between business administrative domains (see Section
2.5.2.), access sessions can be classified according to the specialisation hierarchy shown in Figure 4.12.(a).

![Diagram of access session related information models](image)

**Figure 4.12.** Important access session related information models (TINA-C):

(a) Classification of the access session,  
(b) The access session information model.

The access session related service IOs and their relationships are depicted in the information model of
Figure 4.12.(b). In this figure, the Domain Access Session (D_AS) service IO is associated with a particular
domain and represents the generic information required to establish and support access interactions between
two domains. Furthermore, it is specialised into UD_AS (managed by the user), PD_AS (managed by the
provider) and PeerD_AS service IOs (see also Section 2.5.2.), as each D_AS is associated with a particular
access role. All information that is used directly by the D_AS for authorisation decisions, constraints and
customisation of the D_ASs, Access Sessions and Service Sessions is contained in the User Profile.

The service session related information models

Service sessions can be classified according to the specialisation hierarchy shown in Figure 4.13.(a). The
service session related service IOs and their relationships are depicted in the information model of Figure
4.13.(b) (see also Section 2.5.2.). Every service session consists of usage and provider service sessions.
Each member of a session, i.e. an end-user, a resource or another session, is associated with a usage service session. Furthermore, each usage service session can extend over two domains and is composed of two complementary Domain Usage Service Sessions (D_USSs). The Domain Usage Service Session Binding (D_USS Binding) represents the dynamic information associated with the binding of two D_USSs.

Figure 4.13.: Important service session related information models (TINA-C):
(a) Classification of the service session, (b) The service session information model.

The Service Session Graph information model

The Service Session Graph (SSG) offers a generic framework to describe information in service sessions and is used to model and control the state of a service session. An instance, at a certain point of time, of the SSG models a "snapshot" of the resources, the parties, the peers, and the relationships established into the service session. Figure 4.14. illustrates how the service session information model relates to the SSG information model.

Figure 4.14.: Relationship between the service session information model and the SSG (TINA-C).

The capabilities modelled in the SSG, which can be seen in Figure 4.15., are party invitation and addition, stream binding and stream composition, and explicit control of the use of resources (e.g. various devices participating in continuous media communication). The SSG supports the definition of control relationships through the ControlSR IO type. More specifically, the OwnershipSR class expresses ownership relations, which determine the owners of a service IO that need to be involved in the negotiation of session
management operations. Furthermore, the PermissionSR class expresses the desired access control policy. Additionally, the SSG supports also the definition of composing relationships through the CompSR IO type, which facilitates the representation of composition and federation of services. SessionMember and SessionRelationship IOs can be aggregated into groups (SessionMemberGroup and SessionRelationshipGroup respectively) in order to ease the establishment of repeated relationships.

The invitation and addition of users to a service session is modelled by the Party IO type. A session participating in another session, where both sessions can initiate actions on the other, is modelled by the Peer IO type. A source of support for the execution of a service session (e.g. a file to be retrieved) is modelled by the Resource IO type. Finally, stream bindings are modelled by the association of StreamInterface IOs to StreamBindingSR IOs via the appropriate SessionMember IOs. The StreamFlowEndPoint IOs have associated Quality of Service (QoS) attributes and can be aggregated into stream interfaces at a party’s end system or at a Resource IO.

Taking into account the previous discussion, the following actions take place when specifying the ancillary service conceptual models:

- Customise the session related information models (Figure 4.11.) according to the service requirements.
- Customise the access session related information models (Figure 4.12.) according to the service requirements.
- Customise the service session related information models (Figures 4.13. and 4.14.) according to the service requirements.
- Customise the SSG information model (Figure 4.15.) according to the service requirements.

![Figure 4.15.: The Service Session Graph information model (TINA-C).]
4.8.4. Definition of Service Sequence Diagrams.

Before proceeding to a logical design of how a telematic service will work in terms of software components, its behaviour is necessary to be examined and defined as a black box. In this way, service behaviour is considered as a description of what the telematic service does, without explaining how it does it. One part of that description are service sequence diagrams.

Use cases suggest how actors interact with the telematic service under examination. During this interaction an actor generates events to the telematic service (service events), requesting some operation in response. It is desirable to isolate and illustrate the operations that an actor requests of a telematic service (service operations), because they are an important part of understanding service behaviour.

For this reason, a service sequence diagram, which is a UML sequence diagram, shows, for a particular scenario of a use case 18, the service events that external actors generate and their order. All telematic services are treated as a black box. The emphasis of the diagram is on events that cross the service boundary from actors to telematic services. A service event is an external input event generated by an actor to a telematic service. An event initiates a responding operation. A service operation is an operation of the telematic service that executes in response to a service event. The names of the service event and the service operation are identical. The distinction between them is that the service event is the named stimulus, while the service operation is the response 19. It is evident that the set of all required service operations is determined by identifying the service events.

A service sequence diagram should be done for the typical course of events of each use case and sometimes for the most important alternative courses. It depicts, for a particular course of events within a use case, the external actors that interact directly with the telematic service, the telematic service (as a black box), and the service events that the actors generate. Time proceeds downwards and the ordering of events should follow their order in the use case. Service events (and their associated service operations) should be expressed in an abstract way, emphasising their intention, and not in an implementation specific manner. In that way the name used captures the intent of the service event/service operation, while remaining abstract and noncommittal with respect to design choices. It also improves clarity to start the name of a service event/service operation with a verb (e.g. enter, add, end, make, etc.), since it emphasises their command orientation.

Taking into account the above mentioned guidelines and remarks, this activity consists mainly of the following steps:

**Step 1:** Draw a vertical line representing the telematic service as a black box.

**Step 2:** Identify each actor that directly operates on (or interacts with) the telematic service.

---

18 A scenario of a use case is a particular instance of the use case or a realised path through the use case.
19 A similar relation is the relation between messages and methods in the object-oriented paradigm [ECKE94].
Step 3: Draw a vertical line for each actor.

Step 4: Identify the (external) service events that each actor generates by examining the use case typical course of events text.

Step 5: Illustrate the identified service events in the correct order on the diagram.

Step 6: Include (fragments of) the use case text to the left of the diagram (optionally).

4.8.5. Definition of Service Operation Contracts.

The behaviour of a telematic service is further defined by service operation contracts (or service contracts)\(^{20}\), as they describe the effect of service operations upon the telematic service. A service sequence diagram depicts the external events that an actor generates, but it does not elaborate on the details of the functionality associated with the service operations invoked. All the details that are necessary to understand the service response (and thus the actual service behaviour) are missing. These details are included in service operation contracts, which describe changes in the state of the overall telematic service when a service operation is invoked.

The creation of service operation contracts is dependent on the prior development of use cases and service sequence diagrams, and on the identification of service operations (see Figure 4.10.) in the following order:

use cases → service sequence diagrams → service operations → service operation contracts

More specifically, the use cases suggest the service events and lead to the construction of the service sequence diagrams. The service operations can then be identified. The effect of the service operations is described in service operation contracts\(^{21}\). UML contains support for defining service contracts by allowing the definition of pre- and post-conditions of service operations [EVIT00].

A service operation contract is proposed to include the following sections\(^{22}\) (see also, e.g., Figure 5.10.):

<table>
<thead>
<tr>
<th>Name:</th>
<th>Name of the service operation and its parameters.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Responsibilities:</td>
<td>An informal description of the purpose of the service operation.</td>
</tr>
<tr>
<td>Type:</td>
<td>Name of type that the service operation belongs to.</td>
</tr>
<tr>
<td>Cross References:</td>
<td>Pointers to important related artifacts (e.g., service function reference numbers, names of use cases, etc.).</td>
</tr>
<tr>
<td>Notes:</td>
<td>Design statements regarding the service operation (design guidelines, algorithms, etc.).</td>
</tr>
<tr>
<td>Exceptions:</td>
<td>Reaction of the service operation to exceptional situations.</td>
</tr>
<tr>
<td>Output:</td>
<td>Non user interface outputs.</td>
</tr>
<tr>
<td>Pre-conditions:</td>
<td>Assumptions about the state of the telematic service before execution of the service operation.</td>
</tr>
<tr>
<td>Post-conditions:</td>
<td>The state of the telematic service after completion of the service operation.</td>
</tr>
</tbody>
</table>

The most important section is the “Post-conditions” section which declaratively describes the state changes that occur to service IOs in the service conceptual model, using a number of suitably selected

\(^{20}\) In general, a contract, which is a term borrowed from the software engineering community [MEYE88], is a document that describes what an operation commits to achieve. It is usually declarative in style, emphasising what will happen, rather than how it will be realised. It is common for contracts to be expressed in terms of pre- and post-conditions state changes [GRA95a].

\(^{21}\) For each service operation, a service operation contract is constructed.

\(^{22}\) It is not necessary to include all sections. However, the “Responsibilities” and “Post-conditions” sections are recommended.
Chapter 4: A Methodology for the Development of Telematic Services

statements (instance creation, instance deletion, attribute modification, association formed, association broken, and user interface activation). This section reveals how the telematic service has changed as a result of a specific service operation. The post-conditions are not actions to be performed during the service operation. They are declarations about the service state that are true when the service operation has finished. It has to be noted that post-conditions are expressed in the context of the service conceptual model, and thus it is likely to lead to the enhancement of the service conceptual model.

During the service analysis phase it is difficult (and maybe it is not even necessary) to generate a complete and accurate set of post-conditions for a service operation. However, it is better to create post-conditions early (even if they are incomplete), rather than defer their creation until the service design phase, when service developers should be concerned with the design of a solution, rather than investigating what should be done. Post-conditions will take their final form during the service design phase. Therefore, they will enhance the service analysis work of the following service development cycle.

4.8.6. Refining the Glossary.

The glossary was created in the requirements capture and analysis phase (see Section 4.6.6) and its refinement is an ongoing task that takes place in every service development cycle as new terms are generated.

4.8.7. Definition of Service State Diagrams.

The legal sequence of external service events that are recognised and handled by a telematic service in the context of a specific use case can be successfully described by service state diagrams. These are UML state diagrams, which illustrate the interesting and significant service events and the states of a telematic service, together with the behaviour of the service in reaction to a particular service event. More specifically, as can be seen in Figure 4.16, transitions are indicated by arrows, labelled with the corresponding service event, and states are depicted in rounded rectangles. It is common to include an initial (pseudo) state which automatically causes a transition to another state when an instance of the service state diagram is created.

A service state diagram which depicts the (overall) service events and their desired sequence within a use case is called a use case service state diagram. Such a diagram for the service events of use case X can be seen in Figure 4.16. It illustrates that it is not legal to generate "event 3" if "event 2" has not previously caused a transition of the service to "State 3". It has to be noted that a use case service state diagram is not necessary to illustrate every possible service event of the use case that it examines. If a service event arises that is not represented in the diagram, then it is ignored by the diagram. Thus, a use case service state diagram can be created for a specific use case at varying levels of detail depending on the exact modelling needs.

---

23 Therefore, post-conditions should be expressed in a passive past tense form (was ...), in order to emphasise that they are declarations about a past state change rather than a design of how it is going to be achieved.
Chapter 4: A Methodology for the Development of Telematic Services

The real value of use case service state diagrams is appreciated when they model complex use cases with many service events, because then they help considerably the service developer(s) during the service design to avoid out-of-sequence service events and the corresponding error conditions. Given a set of concise and thorough use case service state diagrams, the service developer(s) can construct a service design that ensures the correct order of service events without missing any of them. However, from this remark is also evident that use case service state diagrams are not necessary if there is no significant service event ordering. Therefore, their definition in the service analysis phase is optional. In such cases, another (optional again) alternative is the creation of a global service state diagram, which illustrates, for the entire telematic service, all the transitions for service events across all the use cases. It is a union of all the use case service state diagrams and is useful as long as the total number of service events is small enough to keep the diagram comprehensible.

Service state diagrams, except from being a service analysis phase artifact, can also be used in the service design phase for modelling the behaviour of specific state-dependent service COs or service classes. More specifically, a service state diagram for a service CO / class can depict internal events (caused by something inside the service boundary) which typically represent messages received from other service COs / classes. However, this is not recommended (and therefore it is not examined in the service design phase), because the paradigm of service COs / classes that collaborate via messages to fulfil tasks is illustrated directly by the service interaction diagrams (see Section 4.9.3.), and therefore it would be somewhat incongruous to use a service state diagram to show a design of service CO / class messaging and interaction.

Taking into account the above discussion, this activity consists mainly of the following steps:

**Step 1:** Draw a service state diagram for each use case or (alternatively) draw a global service state diagram.

**Step 2:** Add to the diagram(s) transition actions, transition guard conditions, and nested states (optionally).

4.8.8. **Concluding Remarks.**

The activities of the service analysis phase can be seen in Figure 4.9. and the artifacts that are produced in Figure 4.10. Service sequence diagrams and service operation contracts are part of the service behaviour model of the service analysis model (see Figure 4.17.). The service behaviour model specifies what service events a telematic service responds to, and what responsibilities and post-conditions the corresponding service operations have. Furthermore, it describes the external interface and behaviour of the overall service.
It has to be noted that in order to be complete (and really object-oriented) the information specification of the service should also take into account the (dynamic) behaviour of individual service IOs. This behaviour is usually defined by allocating operations to the service IOs. However, the issue whether operations should be ascribed to individual service IOs is quite controversial as it actually represents a functional decomposition of the overall service functionality [GASP96]. This clearly implies design decisions that should be better taken at the service design phase.


In the service analysis phase, high level OMT diagrams can be used to represent the service concepts and their relationships [RUMB91]. Additionally, analysis level Message Sequence Chart (MSC) diagrams\(^{24}\) can model the interaction among service parts [ITU92c]. However, both these OMT and MSC models provide a high level overview and an understanding of the service as a whole. They do not focus to the individual service IOs. For this purpose, quasi GDMO (Guidelines for the Definition of Managed Objects) and GRM (General Relationship Model) can be used, with quasi GDMO describing the characteristics of service IOs and GRM specifying the relationships among service IOs. Quasi GDMO and GRM are used for information modelling purposes in TINA-C and originate from the Open Systems Interconnection (OSI) GDMO and GRM [ITU92d] formal techniques that were developed for defining managed objects and their relationships. Finally, for the identification of service concepts and service objects, for the assignment of responsibilities to service classes, and for finding and examining collaborations between service classes, CRC (Class - Responsibility - Collaborator) cards [WIRF90] can be used. This simple but effective technique, assumes the existence of written requirements specifications and is usable at both the analysis and the design levels.

\(^{24}\) OMT and MSCs can also be used for design purposes.
4.9. Service Design Phase.

During this phase the service developer defines the behaviour of the service IOs that were identified in the service analysis phase and structures the telematic service in terms of interacting service computational objects (service components or service objects), which are distributable, multiple interface service objects (see also Section 2.5.2.). They are the units of encapsulation and programming. While service IOs mainly explain how a service is defined, service Computational Objects (COs) reveal what actions have to be performed in order to execute the service. Therefore, the output of this phase is (mainly) the dynamic view of the internal structure of the telematic service.

![Diagram of Service Design Phase](image)

**Figure 4.18: Service design phase activities.**

The activities of the service design phase are depicted in Figure 4.18. As with the previous phases, the linear order that may be inferred from this figure is not strictly the case, as some artifacts may be made in parallel (e.g. the service interaction diagrams and the service design class diagram). The dependencies between the artifacts produced during the service design phase can be seen in Figure 4.19. This figure also shows the way that the service design phase artifacts depend on some of the service analysis phase artifacts. The activities of this phase are examined in the following sections.

4.9.1. Definition of Real Use Cases.

As was mentioned in Section 4.6.4., a real use case describes in a concrete manner the way that a use case will be realised in terms of specific input and output characteristics and implementation details. For example, if the telematic service has a graphical user interface (as is usually happens), real use cases will include diagrams of the windows involved and discussion of the low-level interaction with the interface widgets.
Chapter 4: A Methodology for the Development of Telematic Services

The creation of real use cases is dependent upon the prior creation of the corresponding high level and essential use cases. However, the definition of real use cases may not be necessary. As an alternative, service developers may create rough user interface storyboards and specify all the necessary details during the service implementation phase [BYER93]. Real use cases are useful (and thus their definition mandatory) if excruciatingly detailed interface descriptions are required prior to service implementation.

4.9.2. Definition of User Interface Aspects.

In this activity important characteristics of the user interface of the service are defined by examining the related prototype (see Section 4.6.7.) and taking into account the feedback from the users of the service. The adherence to specific GUI standards [GRA95a] and user interface design principles [CONS99] is also decided in this activity. The overall objective is to ensure that the telematic service will be characterised by a friendly and consistent user interface, designed according to ergonomic criteria, that will present information in convenient and comprehensible formats, without creating information overload to the user, and by providing suitable error and information messages, and help facilities.

It has to be noted that the application of the Model-View separation principle [BUSC96], according to which the service logic should not be bound to a particular user interface, is proposed. More specifically, it is usually desirable that there is no direct coupling from service objects (in the service logic layer, see Figure 4.8.) to user interface objects (in the presentation layer, see Figure 4.8.), because the user interface objects are related to a particular telematic service, while (ideally) the service objects may be reused in new telematic services or attached to a new interface. The application of the model-view separation principle in the service design phase supports the creation of cohesive service design phase artifacts that focus on the service domain processes and not on the satisfaction of user interface requirements, allows the separate development of the service logic from the necessary user interface, and minimises the impact upon the service logic layer from changes of the requirements regarding the user interface [CONS99][LARM98].

Figure 4.19.: Service design phase artifact dependencies.
4.9.3. Definition of Service Interaction Diagrams.

After identifying the service COs, by taking into account the service conceptual model(s) and the TINA-C service architecture, a (separate) service interaction diagram is created for each service operation under development in the current service development cycle. Service interaction diagrams illustrate how service objects communicate in order to fulfil the service requirements. More specifically, initially the expanded use cases suggested the service events which were explicitly shown in service sequence diagrams, then an initial best guess at the effect of these service events was described in service operation contracts, and finally the identified service events represent messages that initiate service interaction diagrams, which illustrate how service objects interact via messages to fulfil the required tasks.

Therefore, service interaction diagrams reveal choices in assigning responsibilities to service objects. The responsibility assignment decisions are reflected in the messages that are sent to different service objects. Responsibilities are related to the obligations that a service object has in terms of its behaviour. In the service implementation phase, methods will be implemented to fulfil responsibilities or alternatively responsibilities will be implemented using methods, which either act alone or collaborate with the methods of other service objects.

The service designer may collect / extract information about what tasks the service interaction diagrams should fulfil by essential or real uses cases, and by the post-conditions of the service operation contracts. Therefore, the creation of service interaction diagrams is (mainly) dependent upon the prior creation of real (or essential) use cases, service conceptual model(s), and service operation contracts (see Figure 4.19.). Regarding the contracts however, it is essential to recognise that the previously (in the service analysis phase) defined post-conditions are merely an initial best guess or estimate of what must be achieved and they may not be accurate.

UML defines two kinds of interaction diagrams, either of which can be used to express similar or even identical message interactions; namely collaboration diagrams, which illustrate object interactions in a graph or network format, and sequence diagrams, which illustrate interactions in a kind of fence format [BOOC98][EVIT00]. The use of collaboration diagrams for the expression of service interaction diagrams is preferred over the use of sequence diagrams, because collaboration diagrams are characterised by expressiveness, an ability to convey more contextual information (such as the kind of visibility between service objects), and a relative spatial economy.

Nevertheless, either notation can express similar constructs. What is really important is that service interaction diagrams is one of the most significant artifacts created during both service analysis and service

---

25 In general, a responsibility is defined as “a contract or obligation of a type or class” [BOOC97].
26 It is possible to express more semantics in a given space with collaboration diagrams, than with sequence diagrams. It is also easier to express (with clarity) conditional logic and concurrency.
design, because the skilful assignment of responsibilities to service objects and the design of collaborations between them are two of the most critical (for the satisfaction of the service requirements and thus for the successful realisation of a service) and unavoidable tasks (which also require the application of design skill) that have to be performed during service creation [LARM98].

This activity of the service design phase consists mainly from the following steps:

**Step 1:** Identify the service COs.

During this step, the service IOs depicted in the service conceptual models (main and ancillary) that were created in the service analysis phase (see Section 4.8.3.) are considered as potential candidates for service COs. In many cases, service IOs are mapped to one corresponding service CO encapsulating the information defined by the service IO and providing an operational interface to access that information. However, the mapping between service IOs and service COs is not necessarily one to one. Furthermore, the existence of a relationship between service IOs, either provides a good rationale for encapsulating them together in the same service CO or indicates the need for a binding between interfaces of their corresponding service COs [DECL97][DEME99].

This *mapping process* is significantly simplified by adopting the use of the generic (access session, service session, and communication session related) service COs, proposed by the TINA-C service architecture (see Section 2.5.2.), in terms of their identified functionality and not in terms of specific interfaces / feature sets. The generic TINA-C service COs that are considered during this step are those listed in Table 2.3. (see Section 2.5.2.). They correspond to the service IOs that are included in the generic TINA-C information models (see Section 4.8.3.2.). Furthermore, by taking into account the related documentation that is available by the TINA-C [TIN95d][TIN95e][TIN96b][TIN97a][TIN98a], Table 4.1. and Table 4.2. are constructed and reveal the way that the functionality of the TINA-C service COs was devised.

<table>
<thead>
<tr>
<th>Service IOs / Session Concepts</th>
<th>TINA-C Service Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Session (AS) with User-Provider Roles</td>
<td>PA and UA</td>
</tr>
<tr>
<td>Access Session (AS) with Peer-to-Peer Roles</td>
<td>PeerA and PeerA</td>
</tr>
<tr>
<td>User Domain Access Session (UD_AS)</td>
<td>PA</td>
</tr>
<tr>
<td>Provider Domain Access Session (PD_AS)</td>
<td>UA</td>
</tr>
<tr>
<td>Peer Domain Access Session (PeerD_AS)</td>
<td>PeerA</td>
</tr>
<tr>
<td>User Profile with User-Provider Roles</td>
<td>UA</td>
</tr>
<tr>
<td>User Profile with Peer-to-Peer Roles</td>
<td>PeerA</td>
</tr>
<tr>
<td>Contract with User-Provider Roles</td>
<td>PA and UA</td>
</tr>
<tr>
<td>Contract with Peer-to-Peer Roles</td>
<td>PeerA and PeerA</td>
</tr>
</tbody>
</table>

Table 4.1.: Mapping between service concepts and TINA-C access session related service COs.

Regarding these two tables, it has to be noted that when a session concept is mapped to a TINA-C service CO, then the service CO supports the functionality and state of the session, and controls the resources which are part of the session. If a session concept is mapped to several TINA-C service COs, then each of them supports part of the functionality and state, and controls some of the resources of the session. When a
service IO is mapped to a TINA-C service CO then the information represented by the service IO is contained within the TINA-C service CO, which may also provide access to that information to other TINA-C service COs. Finally, from Table 4.2. is evident that the service IOs of the SSG information model (Figure 4.15.) may be supported by many of the TINA-C service session related service COs, depending on the design decisions of the service developer(s).

<table>
<thead>
<tr>
<th>Service IOs / Session Concepts</th>
<th>TINA-C Service Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Session (SS)</td>
<td>ss-UAP, USM, SSM</td>
</tr>
<tr>
<td>Usage Service Session (USS)</td>
<td>ss-UAP</td>
</tr>
<tr>
<td>User Domain Usage Service Session (UD_USS)</td>
<td>ss-UAP</td>
</tr>
<tr>
<td>Provider Domain Usage Service Session (PD_USS)</td>
<td>USM</td>
</tr>
<tr>
<td>Provider Service Session (PSS)</td>
<td>SSM</td>
</tr>
<tr>
<td>Composer Domain Usage Service Session (CompD_USS)</td>
<td>CompUSM</td>
</tr>
<tr>
<td>Peer Domain Usage Service Session (PeerD_USS)</td>
<td>PeerUSM</td>
</tr>
<tr>
<td>SSG Information Model IOs</td>
<td>ss-UAP, USM, SSM, CompUSM, PeerUSM</td>
</tr>
</tbody>
</table>

Table 4.2.: Mapping between service concepts and TINA-C service session related service COs.

Considering Table 4.1. and Table 4.2., together with the service requirements and any other artifact produced by the proposed methodology so far, the service IOs depicted in the service conceptual models (main and ancillary) that were created in the service analysis phase (see Section 4.8.3.) are mapped to the appropriate service COs. As a result of this process a table is constructed listing all the service COs that will be used in the service design phase and their corresponding service IO(s).

Considering the previous discussion, the following actions take place during this step:

- Consider the generic TINA-C service COs and their mapping to service IOs (Table 4.1. and Table 4.2.).
- Relate each service IO in the service conceptual models (main and ancillary) to the appropriate service CO.
- Construct a table regarding the service COs that will be used during the service design phase.

**Step 2:** Consider the generic TINA-C service scenarios and select the most appropriate.

After identifying the service COs and before proceeding to the construction of the service interaction diagrams, the computational views of a number of generic TINA-C service scenarios, deduced by the computational modelling guidelines of TINA-C [TIN96b][TIN97a], should be considered. These are useful for improving structure and general comprehension throughout the service design phase, and for offering to the service developer(s) a generic pattern of thought, compatible with fundamental TINA-C concepts, that he / she could use / consider when designing the service interaction diagrams. More specifically, the service developer(s) should select the most appropriate from the following generic TINA-C service scenarios:

**Service scenario of User-Provider roles**

In this service scenario, which is depicted in Figure 4.20., two users are using a service in a provider domain, which acts in both an access provider role and a usage provider role.
Chapter 4: A Methodology for the Development of Telematic Services

Figure 4.20.: A generic TINA-C service scenario involving user-provider roles.

Access session related service COs (as-UAP, IA, PA, UA) provide a framework for offering secure and personalised access to services and for supporting mobility. Service session related service COs (ss-UAP, USM, SSM) provide a framework for defining services which can be accessed and managed across multiple domains. Finally, the communication session related service COs provide end-to-end connectivity and support the establishment of a stream binding between two stream interfaces on the ss-UAPs of the two users (see also Section 2.5.2.).

Service scenario of Peer-to-Peer roles

In this service scenario, which is depicted in Figure 4.21., two domains cooperate in order to provide a service session between two users. The two users are consumers of different retailers. Both of the retailer domains act in provider roles, while the consumer domains act in user roles.

Figure 4.21.: A generic TINA-C service scenario involving peer-to-peer access and usage roles.

The retailers act in peer roles towards each other at the access level, as both domains are able to initiate an access session with the other domain, request the use of services, and forward invitations to consumers of the other domain. The IA is used (again) as the initial contact point for the other domain wishing to establish an access session. A PeerA in one of the domains, wishing to interact with the other domain, contacts the IA and uses it to gain an access session with the PeerA of the other domain. In this service scenario, the retailer
domains are interacting to set-up a federated service session. Therefore, a PeerUSM is used by each of the domains to represent the other domain in the service session. The PeerUSMs interact in a peer usage role, to keep the other service session up-to-date with changes in the local service session.

**Service scenario of Composition**

In this service scenario, which is depicted in Figure 4.22., two service sessions are combined using composition. Access session related service COs are used to establish a relation between the domains involved in the composition in the same manner as in either of the previous two service scenarios.

![Figure 4.22: A generic TINA-C service scenario involving a compound service session.](image)

Furthermore, a service session in one domain initiates the composition by making a request to the access service COs in order to start (and use) a second service in another domain. This causes a SF in the second domain to instantiate a service, in a similar way to the user-provider service scenario. When the initial request is confirmed, a SF in the first domain instantiates a CompUSM, which allows the first service to use (i.e. act as a usage party to) the new service. It has to be noted that the relation between the services could be reversed with the new service adopting a usage party role to the requesting service. This would be reflected in the usage session components instantiated in each domain.

**Step 3: Form the service interaction diagrams.**

A telematic service is composed of a set of service COs interacting with each other via messages with the objective to complete the required service operations. The service operation contracts present an initial best guess at responsibilities and post conditions for the service operations. Service interaction diagrams illustrate the proposed design solution (in terms of service COs) that satisfy these responsibilities and post conditions, and therefore the corresponding service operations.

A service interaction diagram in the form of a UML collaboration diagram is created for each one of the service operations that were identified in the service analysis phase (see Section 4.8.4.). The objective is to fulfil the responsibilities and the post-conditions of the corresponding service operation contracts, recognising however that their accuracy should be questioned.

As was explained in step 1 of this activity the service COs that participate in the service interaction diagrams are drawn from the service conceptual model(s). Therefore, the links between them are actually instances of the associations present in the service conceptual model(s), represent connection paths between service object instances, and indicate that some form of navigation between the instances is possible. More

---

27 In this service scenario the two domains act in peer roles at both the access and usage levels, although this is not mandatory.
specifically, in order for a service object to send a message to another service object it must have visibility to it. Thus, it is important to ensure that the necessary (attribute, parameter, locally declared or global) visibility is present in order to support the required message interaction [JACO99][LARM98][RUMB91].

Finally, as was mentioned in Section 4.6.9., all telematic services have a “Start Up” use case and some initial service operation related to the starting up of the telematic service. Therefore, there should also be a “Start Up” service interaction diagram, which is proposed to be created last. Although the “Start Up” service operation is the earliest one to execute, the development of its service interaction diagram should be delayed until after all other service operations have been considered. This ensures that significant information has been discovered concerning what initialisation activities are required to support the “Start-Up” service operation interaction diagram. The way that a telematic service starts and initialises is affected by related concepts / guidelines in the TINA-C service architecture (e.g. it is assumed that the IA must be present at the provider domain), and is dependent upon the DPE, the programming language, and the operating system that is being used.

Taking into account the previous discussion, the following actions take place during this step:

- Create a separate service interaction diagram for each service operation under development in the current service development cycle. More specifically, for each service operation, create a service interaction diagram with it as the starting point.
- Design a set of interacting service COs with the intention to fulfil the responsibilities and post-conditions of the appropriate service operation contracts. Consider also the corresponding use case descriptions.
- Split each service interaction diagram into smaller diagrams (if it gets complex).

4.9.4. Definition of Service Design Class Diagrams.

Another important artifact created during service design is the service design class diagram, which illustrates the specifications for the software classes of a telematic service using a strict and very informative notation. More specifically, from the service interaction diagrams the service designer identifies the software classes (service classes) that participate in the software realisation of the telematic service under examination, together with their methods, and from the service conceptual model(s) the service designer adds detail to the service class definitions.

A service design class diagram typically includes / illustrates service classes, their attributes and methods, attribute type information, navigability, and associations and dependencies between service classes. It has to be noted that in practice, service design class diagrams and service interaction diagrams are usually created in parallel. Furthermore, in contrast with a service conceptual model, a service design class diagram shows definitions of software entities (service components), rather than real-world concepts.
It has to be noted, that the service design class diagram constructed during this activity can be (optionally) further enhanced by the specification of service classes / service COs using the TINA-C Object Definition Language (ODL) [TIN96a][TIN96b], which is an enhancement (or a superset) of the OMG's IDL, and permits the definition of objects that have multiple interfaces and the definition of stream interfaces among others [KITS95][MERC97].

The following steps are proposed for the creation of a service design class diagram:

Step 1: Identify the service classes that participate in the software realisation of the telematic service by analysing the service interaction diagrams.

Step 2: Draw all the identified service classes in a simple service design class diagram.

Step 3: Duplicate the attributes to the service classes from the associated concepts in the service conceptual model(s). All attributes are assumed to be private by default.

Step 4: Add method names to the service classes by analysing the service interaction diagrams. In general, the set of all messages sent to a service class X across all service interaction diagrams indicates the majority of methods that service class X must define.

Step 5: Add type information to the attributes, method parameters, and method return values. It is only recommended when automatic processing of the service design class diagram is anticipated by a specialised software tool.

Step 6: Add the (different types of) associations necessary to support the required attribute visibility. In general, associations are added in order to satisfy the ongoing memory needs indicated by the service interaction diagrams.

Step 7: Add navigability arrows to the associations to indicate the direction of attribute visibility.

Step 8: Add dependency relationship lines to indicate non-attribute visibility between service classes (i.e. parameter, global, or locally declared visibility).

Step 9: Split the service design class diagram into smaller diagrams (if it gets complex).

4.9.5. Definition of Service Architecture Layers.

This activity starts with the definition of draft service architecture layers, if they haven’t already been defined during the requirements capture and analysis phase (see Section 4.6.8.). Then, the service architecture layers are illustrated using packages, as can be seen in Figure 4.23., which group elements that have related functionality, with relatively high coupling and collaboration. Such a diagram is called a service architecture package diagram and has the advantage of presenting in a structured way all the important constituent parts of the design of a telematic service.

---

28 TINA-C ODL has been issued recently as ITU-T standard Z-130 [BERNO00].
Chapter 4: A Methodology for the Development of Telematic Services

UML provides the package mechanism for the purpose of illustrating groups of elements or subsystems. A package is a set of model elements of any kind, such as classes, use cases, collaboration diagrams or other packages (nested packages). Therefore, an entire telematic service may be considered within the scope of a single top level package (the “Telematic Service” package). It has to be noted that a package defines a nested name space, so elements with the same name may be duplicated within different packages.

A service architecture package diagram can be (optionally) annotated with dependencies between its packages. Furthermore, during this activity, the service design class diagram, which is included in the “Service Domain COs / Classes” package, can be partitioned into smaller parts with the use of packages placed inside the “Service Domain COs / Classes” package. More specifically, the packages that will be created in that way can contain elements of the service design class diagram that belong to the same conceptual area, are in a type hierarchy together, participate in the same use cases, or are generally strongly associated.

![Diagram of service architecture layers expressed in terms of UML packages.](image)

Figure 4.23.: Service architecture layers expressed in terms of UML packages.

This partitioning becomes necessary as the service development proceeds and the service design class diagram is approaching an unwieldy size. In that way, detailed elements of the service design class diagram are placed into larger abstractions, supporting a higher-level view of the diagram in terms of more simple, and thus more easily comprehensible and manageable, functionality groupings. It has to be noted that the packages that are used for the partitioning of the service design class diagram can also be used (optionally) in subsequent service development cycles, for the partitioning of the main service conceptual model with similar benefits.

Taking into account the above mentioned guidelines and remarks, this activity consists mainly of the following steps:
**Step 1:** Define draft service architecture layers (if not yet done).

**Step 2:** Draw a service architecture package diagram.

**Step 3:** Partition the service design class diagram using suitable packages (optionally).

**Step 4:** Annotate the service architecture package diagram with dependencies between its packages (optionally).

### 4.9.6. Definition of a Database Schema.

This activity is performed only when the telematic service under development is storing and retrieving information in a persistent storage mechanism, such as an object or relational database [GOEB95]. In this case, some service COs require persistent storage (persistent objects). If an object database is used to store and retrieve service COs, then there is relatively little effort required on the part of the service developer to interact with the database. However, because of the prevalence of relational databases, their use is often required. Then, a number of problems arise due to the representation mismatch between service COs and the record-oriented way they are stored in relational databases. Because of these problems, the use of a persistence framework, which is a reusable (and usually extendible) set of classes that provide facilities for persistent objects, is necessary [GAMM95][LARM98]. Typically, a persistence framework will translate service COs to records and save them in the (relational) database, and translate records to service COs when retrieving them from the database. It is evident that the use of a persistence framework affects considerably the design of a database schema.

### 4.9.7. Concluding Remarks.

The activities of the service design phase can be seen in Figure 4.18. The artifacts that are produced during this phase can be seen in Figure 4.19. From the service design model depicted in Figure 4.24., is evident that real use cases are members of the service design use case model, service interaction diagrams are members of the service object behaviour model, because they describe the behaviour of service COs, and service design class diagrams compose the service class model.

Furthermore, for reasons of completeness, the service design model includes service state diagrams for service COs / classes as members of the service design state model, although their creation is not recommended as was explained in Section 4.8.7. Such diagrams may be useful to summarise the results of a service design (at the end of the service design phase) or when the service code is to be produced with a code generator that will be driven by the state diagrams.

### 4.9.8. Complementary and Alternative Approaches.

In the service design phase, SDL can be used to describe the behaviour of a telematic service exploiting the finite state machine concept. Then, the SDL specification will serve also as a basis for validation, simulation
and test case generation [COMB99]. In general, for making formal models of telematic services and being able to reason about these models, SDL is undoubtedly the notation of choice, as the tool support for SDL is perhaps the most advanced of all the formal notations existing today. However, adopting an SDL-based approach cannot guarantee that the developed services will be error free and the value of SDL for service creation purposes is questioned [OLSE99], as it may introduce unnecessary complexity in the service design phase. Furthermore, the application of SDL can be difficult (or even problematic) in the case of relatively complex telematic services with many service objects interacting in non-trivial ways, due to the problem of state space explosion.

![Service Design Model](image)

Figure 4.24.: The service design model.

In the case that the use of SDL is considered to be necessary, it has to be noted that SDL can be combined with UML and IDL [ACT99a]. More specifically, SDL constructs can be obtained from UML diagrams, through the use of suitable mapping conventions, and the IDL code can then be generated from the SDL specification. In an alternative approach, the interfaces of the service objects are first described using UML and IDL, and then the IDL specification is used to generate a SDL skeleton to which behaviour is finally added. Instead of using SDL, other formal notations that can be used are LOTOS (Language Of Temporal Ordering Specifications) [BRIN87] and object-Z [STEP92].

In the service design phase, service COs have a dominant role. Their interfaces are the result of the examination of the service IOs and the corresponding information models that they participate in, which reveal the way that service IOs are related to each other. This aggregation of interfaces into a service CO ensures the semantic understanding that operations at one interface may affect the behaviour of other interfaces because they may be linked by a common, underlying information model captured by the service CO. Therefore, such information models influence considerably the parameters and the semantics of the operations found on the interfaces of the service COs.
In order to aid the service development process TINA-C, proposes and prescribes a set of generic interfaces for the generic TINA-C service COs. These interfaces correspond to the interactions that take place between business administrative domains, support a particular session role, and are defined by the appropriate reference point specifications. TINA-C assembles the proposed interfaces into feature sets. A feature set is a group of related interfaces that exposes restricted parts of the appropriate information model(s) for manipulation or examination, defines the details of interactions between service COs, and specifies levels of functionality inside a service (e.g. basic or multiparty session control) [TIN97a]. If a feature set is supported by a domain, the domain supports all of the interfaces in the feature set, which are associated with its session role. In general all domains in a session must support the same feature sets. It has to be noted that the basic feature set (BasicFS) allows business administrative domains to exchange and agree the use of additional non-standardised interfaces. This is essential to obtain service specific interfaces and value added interfaces above or instead of TINA-C standard session control.

The proposed methodology does not suggest the use of the TINA-C feature sets because:

• In order to be applied successfully (i.e. in order to select the appropriate feature sets and customise them according to the specific service requirements) they require deep knowledge of the TINA-C service architecture by the service developer(s) together with considerable related experience. Otherwise, their application is very likely to lead to poor and inefficient service designs. It is evident that this requirement for TINA-C expertise can be very restrictive, especially for small and medium sized service providers.

• Their application can be a complex task especially when they have to be customised according to the specific service requirements. In these cases, it is very likely that they will introduce more complexity than necessary in the service design phase.

• Their use tends to make service developers focus on the service design phase underestimating all the previous phases. In that way, they increase the possibility of having a service design that doesn’t reflect (correct or even at all) the real service requirements.

• Their application is very likely (especially in relatively complex telematic services) to create a “semantic gap” between service analysis and service design. This is mainly due to the fact that TINA-C does not explain sufficiently the way that feature sets were created and their exact relationship with the TINA-C information models [LEW198]. This “semantic gap” can have serious negative implications for the maintainability and future evolution of the telematic service under examination.

• Their specification by TINA-C is not yet complete.

Although not suggested, feature sets can possibly to be applied during the service design phase of the proposed methodology. More specifically, service developers with a TINA-C expertise can critically use them as an aid (by taking from them whatever they consider useful) when devising and constructing the interfaces of the service COs. However, service developers should not use feature sets as an excuse for not
carefully performing the requirements capture and analysis phase and the service analysis phase. Moreover, they should try to fully integrate them in the service design phase, improving as much as possible the consistency of the results of this phase with the results of the previous phases. Finally, it has to be noted that the importance of feature sets is expected to increase when their specification by TINA-C is completed. The application of feature sets will be especially useful for telematic services that span multiple business administrative domains and have to consider composition and federation issues.

4.10. Service Implementation Phase.

During this phase an implementation of the telematic service (service code) is generated from the service specifications and the deployability of the overall implementation on a TINA-C compliant DPE is examined (DPE targeting). It is assumed that at the beginning of this phase a specific (object-oriented) programming language and a specific distributed object platform are chosen (see Section 4.6.1.).

The activities of the service implementation phase can be seen in Figure 4.25. The dependencies between the artifacts produced during this phase can be seen in Figure 4.26. This figure also shows the way that the service implementation phase artifacts depend on some of the service design phase artifacts. The activities of this phase are examined in the following section.

The engineering representation of a service CO (using an object-oriented programming language like C++ or Java) is called an *engineering Computational Object* (eCO). The mapping between service COs and their eCOs is one to one: no eCO represents a composition of service COs nor is a service CO represented by more than one eCOs. The interfaces of an eCO represent the interfaces of its corresponding service CO [KELL95].
However, they may be modified by type conversions for operation parameters or by adding operations needed for the eCO’s interaction with other eCOs (e.g. those that provide required distribution transparencies). Furthermore, a management interface may be added, with operations to be performed by the object instance after its creation (constructor), just before its destruction (destructor), after activation or before deactivation. These modifications and additions depend significantly on the exact characteristics of the selected DPE.

With the completion of the service design phase there is sufficient detail to generate code for the service objects and construct the appropriate eCOs. For this purpose the service design phase artifacts (and especially the service interaction diagrams and the service design class diagram) provide a significant degree of the necessary information and the translation process is relatively straightforward, especially if service classes are implemented (and tested) from the least coupled to the most coupled. More specifically, as a service interaction diagram shows the messages that are sent in response to a method invocation, the sequence of these messages translates to a series of statements in the method definition. Furthermore, from the service design class diagram, a mapping to the basic attribute definitions and method signatures is almost evident.

Despite these facts, the service implementation phase is not a trivial code generation process. The results generated during the service design phase have an approximate nature. During programming and testing many changes will be made and detailed problems will be uncovered and resolved. However, because of the nature and structure of the proposed methodology, the service design phase artifacts will provide a resilient core that scales up with elegance and robustness to meet the new problems encountered during programming. Consequently, change and deviation from the service design phase artifacts during the service implementation phase should be expected and planned for (see also Section 4.6.). After all the spirit of iterative development is to capture a “reasonable” degree of information during the service analysis phase, filling in details during the service design phase. Similarly, it is in the spirit of this process to capture a “reasonable” degree of design results during the service design phase, filling in further details during the service implementation phase. The definition of “reasonable” is, as it was expected, a matter of judgement [HANS97][LARM98].

Figure 4.26.: Service implementation phase artifact dependencies.

The activities of the service implementation phase, which can be seen in Figure 4.25., depend significantly on the selected programming language and on the target DPE. Taking this into account, this phase is structured in the following way:

**Activity 1:** Create IDL Specifications.

Each service class of the service design class diagram is specified using IDL based on the ODL specifications or (in the case that these were not created) on information deduced by the appropriate service interaction diagrams and the service design class diagram. It has to be noted that IDL is capable to express less semantics than the UML artifacts created in the service design phase. Although, there are approaches that add the extra (missing) functionality to IDL (without having to extend it) through the use of structured comments and contract objects [WATK98], the service developer is usually responsible to ensure that the service code is semantically equivalent with the outcome of the service design phase.

**Activity 2:** Implement Service Class and Interface Definitions.

The service class and interface definitions are implemented in the programming language of choice (possibly automatically) by taking into account the IDL specifications. During this process the service design class diagram is also considered in order to avoid errors resulting from possible misinterpretation of the IDL specifications. When DCOM and Visual C++ are used, the service class and interface definitions can be generated automatically using the ActiveX Template Library (ATL), which is a rapid ActiveX development tool that is an integral part of Visual C++ 6.0.

**Activity 3:** Implement Methods of Service Classes.

The methods of the service classes are implemented in the programming language of choice by examining the service interaction diagrams and by considering the service class and interface definitions. Reusability matters are also examined in this activity (see Section 4.10.2.).

**Activity 4:** Implement Graphical User Interface (Windows).

The service GUI is implemented by taking into account the remarks of the users regarding the user interface prototype that is being used (see Section 4.6.7.) and the related design considerations as were stated in activity 2 of the service design phase (see Section 4.9.2.). The user interface prototype can be the basis for the implementation of the service GUI (depending on the technology that was used for its creation), in order to avoid implementing the service GUI from scratch.

**Activity 5:** Implement Reports.

If the generation of reports by the service is desirable according to the service requirements then these are implemented in this activity. Reports can also be considered by the user interface prototype of the service.
Chapter 4: A Methodology for the Development of Telematic Services


The database schema that was designed in activity 6 of the service design phase (see Section 4.9.6.) is implemented in this activity using the technology of choice (object database or relational database with a persistence framework).

Activity 7: Integrate Implementation Work.

In this activity, the separate parts that constitute the service code are assembled and connected together, in order to behave in an integrated manner, and their efficient execution upon the target DPE is ensured. The UML component and deployment diagrams can be used to define mechanisms and functions required to support distributed interactions between the service objects in the target DPE.

Activity 8: Prepare for Testing

The test cases and the test data that will be used in the next phase (service validation and testing phase) are designed in this activity, which is also responsible for the completion of any (usually limited) implementation work that may be needed for starting testing.

4.10.2. Important Reusability Matters.

The service implementation phase is a phase in which reusability matters should be considered seriously (i.e. more seriously than in the other phases). According to the TINA-C service architecture new services can be realised by enhancing already existing components (e.g. with the use of inheritance) or by defining new ones [TIN97a]. Therefore, the service independent components that are specified by TINA-C and other suitably created (and carefully selected) service components can be considered as reusable units in the creation of new services. They may be used in a service implementation as they are, or as the basis for the construction of a service specific component. More specifically, service dependent components may, either inherit or aggregate the characteristics of service independent components, or have a relation with them [ADA00g]. This activity can be greatly facilitated and enhanced by the construction of service independent component libraries, where components can be expressed in UML (various notations), ODL, C++ / Java, etc. [DEME99][VERSO0].

The exploitation of the available service independent components in the service implementation phase, and in previous phases (depending on the nature of the available component libraries), begins with the selection and reuse of the appropriate service independent functionality. Then, the service dependent segment is developed, by exploiting as much as possible the service independent segment. Finally, the two segments are integrated [ACT96a][ECKE97][POLY98]. This process can be expressed with the following series of steps (fine tuning implies a feedback loop):

- Configure the access session related segment:
  - Select the access session related functionality.
Chapter 4: A Methodology for the Development of Telematic Services

- Customise (if necessary) the selected access session related functionality.
- Configure the service session related segment:
  - Select the service generic functionality.
  - Customise (if necessary) the selected service generic functionality.
  - Determine the service specific functionality.
  - Develop the service specific functionality.
  - Fine tune the relations between the service generic and the service specific part.
- Fine tune the relations between the access session and the service session segment.
- Configure the communication session related segment:
  - Select the communication session related functionality.
  - Customise (if necessary) the selected communication session related functionality.
- Fine tune the relations between the service session and the communication session segment.
- Integrate all three segments (access, service, and communication session).
- Prepare the end user system.

Finally, as was mentioned in Section 2.6., the reusable component libraries and the related software tools (reuse infrastructure) that were constructed in the RACE project SCORE [RAC95b] and the ACTS project SCREEN [ACT98a] can be used (with appropriate modifications) by the service developer(s) when creating services according to the proposed methodology.

4.11. Service Validation and Testing Phase.

Validation takes place in this phase by comparing the developed service software against the service specifications produced at the service design phase and by ensuring that it meets the service requirements [ADA98a][GASP96]. Therefore, at the end of every service development cycle, the service code is tested in order to determine whether it behaves like expected, i.e. whether it behaves like specified during the service design phase, and in this way it is also examined whether the right telematic service is being built. Testing is proposed to be done using conventional testing practices (e.g. walkthroughs, condition and data flow testing, boundary value analysis, etc.) [GRA95a][SOMM93] and it is evident that it can initiate the reconsideration of the service code and possibly its modification. A limitation inherent in the testing process is that it cannot be exhaustive, but it can undoubtedly increase the quality of the service implementation and maximise the possibility to have a service that is accepted by its users.

The main types of testing that are proposed to be conducted in this phase are unit testing, integration testing, system testing, and acceptance testing. Only unit testing, which refers to the testing of individual service classes, should be done in every service development cycle. Integration testing may happen less frequently, and system testing and acceptance testing usually take place during the last service development cycle.
Chapter 4: A Methodology for the Development of Telematic Services

Figure 4.27.: The influence of service implementation work in the proposed service development process.

With this phase a service development cycle ends and another one (depending on the exact nature of the specific service requirements) is ready to start. A significant strength of the iterative and incremental service development process adopted by the proposed methodology is that the results of a prior service development cycle can feed into the beginning of the next service development cycle. Thus, subsequent service analysis and service design results are continually being refined and informed from prior service implementation work (see Figure 4.27.). For example, when the code in cycle N deviates from the service design of cycle N (which it inevitably will), the final service design based on the implementation can be input into the service analysis and service design models of cycle N+1. For this reason, as can be seen from Figure 4.3., an early activity within a service development cycle is to synchronise the created artifacts. More specifically, the artifacts of cycle N will not match the final service code of cycle N, and they need to be synchronised before extended with new service analysis and service design results.


As was mentioned in Section 2.6., the service developer(s) decide about the software tools that will support the proposed methodology when creating / customising the SCE that will be used in combination with the methodology. This is an important decision, as tools of any kind (either simple tools that are widespread, such as word processing and graphics construction packages, or more complex specialist tools) can facilitate significantly the application of the proposed methodology, especially for the development of complex telematic services, improving the productivity and efficiency of the service developer(s). However, this is also a difficult decision, because of the wide variety of proprietary tools that are currently available and because the consequences of choosing inadequate or inappropriate tools cannot be ignored.

Therefore, tool selection should be based on a thorough understanding of the proposed methodology, of the capabilities of the evaluated tools, and of the contribution of these tools when used in combination with
the methodology. Furthermore, related standards [IEEE93] should be taken into account and carefully chosen tool selection criteria (e.g. existing organisational expertise, cost, documentation availability, performance, easy of use, ability for integration, etc.) should be used [RAC95c]. Examples of software tools that can be considered are the various modelling tools that support UML (e.g. ROSE from Rational, Paradigm Plus from Platinium, etc.), and the ACE (Application Construction Environment) CASE tool that supports the TINA-C object model [BERS95].


Emerging telecommunications systems promise to offer a wide variety of highly sophisticated, personalised, affordable, high quality, and ubiquitous services over the widest possible coverage area. Several providers are involved in such an ambitious (yet realistic) service provisioning scenario in which competition will mostly focus at the service level, with multiple providers offering new services to the market in a short time over a variety of networks and end systems [DIDR99][MOTA99]. In the light of these challenges and because of the highly increasing complexity of new telecommunications services and the inherent distributed nature of them, a methodology covering the whole service development process, like the one proposed and examined in this chapter, is absolutely necessary.

This chapter starts by recognising (taking also into account the concept of the service life-cycle) that building new telecommunications services in a distributed and heterogeneous environment is very difficult unless there is a methodology to support the entire service creation process in a structured and systematic manner, assure a good quality result, reduce the development time, and assist and constrain service designers and developers by setting out goals and providing specific means to achieve these goals. Then, important related approaches in the service engineering area that attempt to address these needs are briefly presented and it is argued that none of them can qualify as a service development methodology.

After justifying the importance of proposing such a methodology and after providing some initial evidence about its novel character, the benefits from using the UML notation for modelling purposes are identified. Then, an overview of the proposed methodology is given, emphasising on its important characteristics. Therefore, the four initial sections of this chapter (4.1., 4.2., 4.3., and 4.4.) clearly reveal that the proposed service development methodology “covers” the entire service creation process, considers telematic services as distributed object-oriented applications operating upon distributed object platforms, adopts an incremental and iterative use case driven approach, and exploits fundamental object-oriented analysis and design concepts, important TINA-C results, and the UML notation.

The rest of this chapter, after considering some important methodological issues regarding time-box scheduling, the distinction between service analysis and service design and the timing of the creation of certain artifacts, is devoted to the presentation and examination of the proposed methodology.
This methodology enforces the service developer to take into account all the necessary features for the successful design and realisation of a service by exploiting the session-oriented, multi-party, multi-domain nature of the TINA-C service architecture. It provides a step by step approach from problem definition to the realisation of new telecommunications services. For this purpose, the service is studied and described (in a number of service development cycles) at hierarchically related abstraction levels, in the sense that at each level the results achieved at previous levels of abstraction are preserved and refined. Additionally, the use of the object-oriented paradigm is advocated all along the development process. Consequently, this methodology combines the benefits of object-oriented modelling, particularly in terms of scalability and reusability, with those provided by a top-down approach [DEME99][GASP96]. Moreover, as service requirements are emphasised, such an approach ensures a high level of confidence that the users’ expectations on the service will be met.

It is evident that the proposed methodology, which is part of the proposed telecommunications service engineering framework of Figure 1.1, satisfies Requirements 1, 2, 5, 8, 9, 11, and 14 (see Section 1.3.). More specifically, the proposed service development methodology has the following benefits:

- It guides successfully service developers during the entire service creation process offering them a well-defined activity roadmap that enables them to proceed in a step by step fashion from service requirements to the service code.
- It facilitates the recognition and understanding of typical service engineering problems with respect to a particular service, and assists the deduction and expression of solutions for them in an adequate manner.
- It assists and constrains service developers by taking into account all the peculiarities of telecommunications services and by properly structuring the service creation problem domain. More precisely, it improves their reasoning power by structuring important service engineering activities, it increases their efficiency by reusing service specifications, and it promotes a clear separation between “what” the service is and “how” the service is provided.
- It reduces risk, as it increases the likelihood of creating a successful telematic service, by emphasising the careful understanding of the service requirements and by promoting appropriate service analysis and service design strategies.
- It creates a detailed process that can be repeated and duplicated by individuals and especially by service development teams.
- It avoids complexity overload and manages the otherwise overwhelming complexity that usually characterises the development of telematic services, by appropriately modelling services and by abstracting to find and examine only the essential details.
- It creates robust and maintainable object-oriented telematic services by adopting fundamental OOAD principles and by emphasising service analysis and service design, recognising that it is cheaper and easier to make changes during these phases rather than during the service implementation phase.
- It supports reusability at different abstraction levels.
Chapter 4: A Methodology for the Development of Telematic Services

Such a methodology can result in an efficient and effective standardised process for the systematic development of telematic services, at lower cost, higher quality, increased speed, meeting the needs of the users and the enterprise better, and leading to lower maintenance costs. Moreover, a common approach can be used throughout an enterprise, which means that more integrated systems can result, the management of related projects will be significantly facilitated, staff can easily change from project to project without much retraining, the reuse of service specifications and software will be greatly supported, and a base of common experience and knowledge will be achieved [ADA98a][ADA99b][DECL97].

The major research contribution of this chapter is the proposed service development methodology. The importance of this contribution is evident when considering the need for a service creation methodology and the benefits of the proposed methodology, as discussed earlier. Furthermore, Section 4.2. explains the novelty of the contribution when the proposed methodology is considered as a whole. However, despite from being a service development methodology, the proposed methodology has also the following novel characteristics [ADA98a][ADA99b][ADA00b][ADA00c][ADA00d][ADA00f]:

• It addresses in sufficient detail and in a concise, structured, systematic, and unambiguous way all the service creation phases.

• It emphasises the satisfaction of the user requirements which are reflected in the service design and in the service code.

• It adopts an incremental and iterative use case driven approach and provides guidance for its realisation.

• It critically filters the TINA-C service architecture and adopts only certain parts of it that considers useful in a service creation process.

• It promotes the use of the UML notation (suitably customised), with all the corresponding benefits, throughout the service creation process.

• It facilitates reusability, without underestimating the importance of carefully identifying and considering the service requirements.

It has to noted that, before reaching any final conclusions regarding the proposed service development methodology, it is necessary to experiment with it and apply it to the development of a specific telematic service. Such attempts are examined in the following chapter where the proposed methodology is validated and evaluated. Finally, it must be kept in mind, that to obtain the maximum possible productivity gains and to exploit the full potential of the methodology it is not sufficient to apply it in a mechanical manner. On the contrary, an adaptation of the methodology to the service developer's attitude and to the wider organisational mentality and approach regarding telecommunications and information technology in general, is required. In this way, the proposed methodology will be able to support service creation activities even more effectively, without restricting the creativity of service developers, and by utilising fully their prior service development experience.
Chapter 5: Validation and Evaluation of the Proposed Methodology

5.1. Introduction.

Explosive increase in service variety as well as in globalisation and customisation of services produce ever increasing pressures for the efficient support of service engineering activities in an environment open to changes in market and technology. Furthermore, as the complexity of telecommunications services and networks is growing, the creation of high quality telecommunications software is becoming more and more a labour-intensive, costly, and uncertain undertaking. Therefore, responding to the requirements of the emerging open telecommunications service market, in Chapter 4 a service creation methodology was proposed and presented focusing on its essential characteristics, with the intention to assist service developers, in a structured and systematic manner, during the entire service development process.

The proposed methodology adopts an incremental and iterative use case driven approach, recognises the importance of distributed object platforms as service execution environments, exploits fundamental object-oriented analysis and design concepts, incorporates important TINA-C results, and uses the UML notation. As can be seen from Figure 4.3., it can involve several service development cycles and is structured according to a number of phases. The most important of them are the requirements capture and analysis phase, the service analysis phase, the service design phase, the service implementation phase, and the service validation and testing phase.

From the detailed description of these phases in Chapter 4 is evident that devising, constructing, and suitably presenting a service development methodology is a complicated task, as it has a wide scope, requires extended skills for the analysis and synthesis of conceptual structures, and involves the consideration of a variety of factors. Furthermore, this task is impossible to be viewed as a pure theoretical exercise, because the success of such a methodology depends on its ability to be applied, in an efficient and effective manner,
to the development of actual telematic services, and the only way to take into account and examine this ability is by gaining significant *practical experience* on the use of the methodology for various services under different circumstances. For this reason, this chapter, which complements and supports Chapter 4, has a more practical nature and attempts to determine the *applicability*, the *correctness*, and the *completeness* of the proposed service development methodology by performing appropriate *validation* and *evaluation* activities [ADA00]. The significance of these activities has been identified in Section 1.4., as they constitute one of the main objectives of this thesis and an important research contribution of it.

More specifically, Section 5.2. validates the proposed service development methodology and determines its practical value. In the context of this section, validation is the task of checking that the proposed methodology can be used efficiently and effectively for the development of new telecommunications services and is performed by applying the methodology to the development of specific services. Therefore, Section 5.2., after justifying this validation approach and clarifying the expectations from it, describes in a concise manner (in Section 5.2.1.) the development of a complex characteristic telematic service (a MultiMedia Conferencing Service for Education and Training, MMCS-ET), according to the main phases of the proposed methodology. A number of additional simple validation attempts, regarding the application of the methodology to several different simple services, are then outlined (in Section 5.2.2.), focusing on the description of the service scenarios that they incorporate. The conclusions from all these validation attempts are included in Section 5.2.3.

One of the benefits from validating the proposed methodology was the motivation for its *enhancement* from the feedback that was generated. The most important methodology extension that was devised in this way is related to the exploitation of *design patterns* and is examined in Section 5.3., where a number of general purpose design patterns specifically constructed for the proposed methodology are proposed and presented, together with the basic characteristics of service engineering design patterns. The extended service development methodology is then evaluated in Section 5.4. according to the *Normative Information Model-based Systems Analysis and Design (NIMSAD) framework* and the results of this evaluation are highlighted. Finally, Section 5.5. summarises the whole chapter emphasising on the research contributions included in it.

### 5.2. Validation of the Proposed Methodology.

As was mentioned in Section 5.1., the service development methodology that was proposed and presented in Chapter 4, is in need of a proper validation. Information systems development methodologies and general purpose object-oriented software development methodologies are usually validated by applying the methodologies to the development of a number of typical / characteristic systems, illustrating the important models that are created, commenting on decisions that have to be made, offering additional guidelines when required, and pointing out any potential problems that may appear, together with possible ways to encounter them [AVIS95][GRA95][GRA95b][JAC099].
Chapter 5: Validation and Evaluation of the Proposed Methodology

The use of such examples or case studies or test cases for a variety of validation purposes is also a very common practice in the area of service engineering, as is indicated by the vast majority of RACE and ACTS EU projects, and especially by the ACTS project VITAL (Validation of Integrated Telecommunication Architectures for the Long term), which used this approach (very successfully) in order to demonstrate and validate the development, deployment, management and use of heterogeneous service features on an Open Distributed Telecommunication Architecture (ODTA), which is TINA-based, while integrating existing networking concepts, such as TMN, IN, and the Internet [CANA99][MOTA99]. Furthermore, in most of the attempts to define various methodologies in the area of service engineering that were mentioned in Section 4.4., validation is done by using suitable examples or case studies [ACT96b][ACT97a][ACT98d][ACT99a][EUR94b][EUR97c][EUR97d][EUR98b][RAC94a][RACE96]. Therefore, the “validation by example” approach seems to be very successful for methodologies of different origins and different objectives and is also widely acceptable and accredited. For this reason, it is selected for the validation of the proposed service development methodology.

Because of this decision, in order to validate the proposed methodology and examine its usefulness, correctness, consistency, flexibility, effectiveness and efficiency, several simple scenarios, regarding a variety of service creation activities for different simple telecommunications services, were considered. The application of the methodology to these service scenarios confirmed that the methodology has all the above anticipated positive characteristics.

To verify and reinforce these findings under (more) realistic conditions, and to determine also the true practical value and applicability of the proposed methodology, all the phases of the methodology were used for the development of a real telematic service. The chosen service is a MultiMedia Conferencing Service for Education and Training (MMCS-ET), because it is expected to have great demand in the near future [PAP98a][PAP98b] and because it is considered to be a characteristic and easily comprehensible multimedia telematic service, with sufficiently complex and varied functionality, that can be used successfully for validation purposes offering the maximum possible benefit [VOLM97].

All these validation attempts, which are described in the following sections, provided valuable feedback, enlightened many aspects regarding the proposed process (in terms of phases, activities, and steps) and the modelling approach that is followed, revealed different perspectives for considering some matters, pointed out a number of deficiencies, and (in general) resulted to the further improvement of the methodology (see also Section 5.3.). The development of the MMCS-ET is examined first and is described and analysed in significant detail (both in Section 5.2.1. and Appendix A), because it is considered to be the most important validation attempt due to its wide scope and its realistic nature. In addition to validating the proposed service development methodology, the MMCS-ET (as it is being gradually realised in a step by step fashion proceeding through the different phases of the methodology) provides also a representative example of the
use of the methodology to those interested in employing it for the development of new telecommunications services (e.g. service developers, service designers, etc.) and increase in that way their understanding of the methodology and their confidence on its effectiveness [ADA99b][ADA00b][ADA00f].

5.2.1. The Development of a MMCS-ET.

The rich functionality of a MMCS-ET prohibits the detailed presentation and explanation of its development according to the proposed service creation methodology in this section, because such an attempt would require a lot of space, destroying the structure and decreasing the readability of the entire chapter. Therefore, the development of the MMCS-ET is only described briefly in this section, focusing mainly on the most important artifacts that were created during the application of the different phases of the methodology, and considering only part of the service functionality (four characteristic use cases).

However, as the full details of this validation attempt are important for the better comprehension of the proposed methodology and of the way that it can be applied, and also for the appreciation of its practical value, Appendix A ("Detailed Presentation of the Development of a MMCS-ET using the Proposed Methodology") includes more information in a structured manner, taking into account the complete functionality of the MMCS-ET. The intention is thus to use this section (5.2.1.) for offering a general understanding of the way that the methodology is applied to the development of the MMCS-ET (without missing the important points) and for providing the motivated reader with a roadmap capable of guiding him/her through the contents of Appendix A enabling him/her to get the most out of its study.

Finally, it has to be noted that both this section and Appendix A, for reasons of simplicity, do not take into account that the MMCS-ET was developed in several service development cycles, and present everything as if the MMCS-ET was developed in a single service development cycle. Moreover, this section starts with a brief introduction to the family of telematic services that the MMCS-ET belongs, which represents the attempt of the service developer to familiarise himself/herself with the wider application domain that encompasses services like the MMCS-ET (see also Section 4.6.2.).

5.2.1.1. Telematic Services for Education and Training.

Modern post-industrial societies are characterised by a growth and diversification of information and communication needs. As a consequence, a deep innovation in the contents and techniques of the educational and training processes has become a mandatory task. Future education and training is in need of proper practices and tools able to overcome space, time, and performance demands, which are pointed out by the increasing geographical distribution of education and training centres, the need for a continuous updating in technology-related information, and the learning effectiveness provided by the integrated use of multiple forms of information [CHIR95][PAPA97].
Learning of any sort is an active, dynamic, and active feedback process with imagination driving action in a variety of exploratory and interactive activities [MACF94]. Therefore, it can only take place effectively given a suitably supportive environment, able to provide both a rich resource and a flexible stimulus [SCHA94]. Such a supportive environment can be created by the utilisation of multimedia communication systems in education and training (telelearning systems), mainly because they permit, in a flexible and gradually more and more inexpensive way, the storage, transmission, processing, presentation, and in general, manipulation, of virtually all types of information (data, text, pictures, audio, video). Thus, the students / trainees can interact with the instructors and with each other as if they were together in the same classroom. Attempts in this direction have taken the form of interactive distance learning, distributed learning, interactive teletraining, and teleteaching [CHIR95][FOW96b][GIAR92]. These are telelearning systems which promote better quality learning opportunities, a greater quantity of training options, and generally a more enhanced learning process [PAP98b].

The following types of telelearning can be identified [PAP98a]:

- **Audio telelearning**: It supports two-way audio communication between the instructor and learners at remote sites (audioconferencing).
- **Audiographic telelearning**: It is audio telelearning supported by enhanced image or data transmissions (audiographic conferencing).
- **Multimedia telelearning**: It supports two-way audio and video communication between multiple locations using sophisticated computer technology (computer-based interactive videoconferencing) 1.

The most important benefits of such telelearning systems are increased flexibility in the delivery of learning, reduced learning time, reduced cost, instructional consistency, privacy, mastery of learning, increased retention and motivation, the ability to share limited instructor resources, and the fact that students / trainees enjoy learning (edutainment) [GIAR92][PAPA97]. Research comparing telelearning to traditional face-to-face instruction indicates that teaching and studying at a distance can be as effective as traditional instruction, when the method and technologies used are appropriate to the instructional tasks, there is student-to-student interaction, and timely teacher-to-student feedback [FOW96b].

Although telelearning has undeniable benefits, caution is necessary as organisational and management structures need to be adapted along with this new mode of delivery, course design needs to be carefully structured, and appropriate pedagogies thoughtfully chosen. In this manner, interactive telelearning systems will quickly mature and support in a more active, flexible and economic way, highly individualised, just-in-time, and on-demand educational and training processes [PAP98a][PAP98b].

---

1 The MMCS-ET that is being developed using the proposed service development methodology is a representative example of multimedia telelearning.
5.2.1.2. Requirements Capture and Analysis Phase.

At the beginning of the first phase of the proposed methodology the main objective of the MMCS-ET is specified (step 2, activity 2). This is to enable a teacher/trainee to teach a specific course to a number of geographically dispersed/distributed students/trainees. The service should establish an educational session between the teacher and the remote students that is equivalent to the educational session that would have been established between the same people (teacher and students) in a traditional classroom.

Further investigation (step 3, activity 2) revealed that every educational session has at least one teacher and that a teacher can participate in only one educational session at a given time, while a student can't participate in more than one educational sessions simultaneously. The establishment of an educational session involves the creation of the session by the teacher and the invitation of students to participate in the session. A student can't create an educational session and can't invite a new participant (teacher or student) to a session. A student participating in a session can only invite to direct communication with him/her another student, who is also (already) a participant in the same session.

The educational session established in a virtual classroom by the MMCS-ET should be equivalent to the educational session established in a traditional classroom and should have as many characteristics as possible in common with it. More specifically, in a traditional classroom the teacher has the ability to manage the educational session. Besides establishing the session he/she can also modify the session (e.g. by removing a student from the classroom and thus from the session), suspend and resume the session (e.g. by allowing the students to have a break for a few minutes without leaving the classroom), and finally finish the session (e.g. by ringing a bell or by telling it to the students who leave the classroom and end the session). The same capabilities should also characterise the teacher in a virtual classroom.

Furthermore, in a traditional classroom the teacher and the students can interact (and possibly collaborate) during an educational session in the following ways:

- By seeing and talking to/hearing each other. This is the most common way of interaction.
- By writing at their notepads. In this way, a teacher can interact with only one student at a time, and e.g. see/correct his/her answer to an exam question.
- By writing at the blackboard of their classroom. In this case, everyone in the classroom sees what is written at the blackboard and everyone can write something at the blackboard.
- By exchanging course material (e.g. documents, graphs, etc.). This material can be used during or after the educational session, according (usually) to the instructions of the teacher.

---

2 Such statements refer always to the phase of the methodology that is currently examined. More details are given in Chapter 4 where the proposed methodology is presented.

3 For reasons of simplicity, the rest of Section 5.2.1. will refer to teachers and students implying teachers/trainers and students/trainees respectively.
Therefore, in a virtual classroom there is a need for audio / video (A/V) communication among all the session participants (to substitute face to face contact), text communication between only two session participants (as that achieved with the use of notepads), text communication among all the session participants (as that achieved with the use of a blackboard), file communication between the session participants (for the exchange of course material), and collaboration among all the session participants in order to perform a common task.

By analysing all the above mentioned (unstructured) requirements a set of initial service requirements are identified (step 4, activity 2). These are structured in the following way (step 5, activity 2):

- Group communication support requirements:
  - Session related requirements:
    - Session establishment support (support for session creation and user invitation).
    - Session modification support.
    - Session suspension support.
    - Session resumption support.
    - Session termination support.
  - Interaction requirements:
    - Audio / video communication support.
    - Text communication support.
    - File communication support.
- Collaboration support requirements:
  - Synchronous requirements:
    - Chat support.
    - Voting support.
  - Asynchronous requirements:
    - File exchange support.

The business environment (step 6, activity 2) assumed for the MMCS-ET is depicted in Figure 5.1. The users of the service (teacher, students) and a service provider (MMCS-ET service provider) are...
distinguished as stakeholders (independent actors / entities). The teacher and the students correspond to the
generic (business) role of consumers of the service and have the service independent specific role of an end-
user. The MMCS-ET service provider corresponds to the generic (business) role of both a retailer and a
connectivity provider. As can be seen from Figure 5.1., the transport service provided by the connectivity
provider goes through intermediate service retailing. Through this configuration only the Retailer (Ret) inter-
domain reference point becomes relevant. The interactions that should be considered between each pair of
generic (business) roles of the MMCS-ET are access, service usage, and communications usage interactions.
During these interactions, the actors of the MMCS-ET can take either a user / party or a provider session role.

<table>
<thead>
<tr>
<th>Ref. #</th>
<th>MMCS-ET session related functions</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF.1.1.</td>
<td>Contact the MMCS-ET provider domain.</td>
<td>Hidden</td>
</tr>
<tr>
<td>SF.1.2.</td>
<td>Enable a user to log in with an ID (user name) and password in order to use the telematic service.</td>
<td>Evident</td>
</tr>
<tr>
<td>SF.1.3.</td>
<td>Check the length of the password provided by a user during the log in process.</td>
<td>Hidden</td>
</tr>
<tr>
<td>SF.1.4.</td>
<td>Cipher the password provided by a user during the log in process.</td>
<td>Hidden</td>
</tr>
<tr>
<td>SF.1.5.</td>
<td>Check the validity of the ID (user name) provided by a user during the log in process.</td>
<td>Hidden</td>
</tr>
<tr>
<td>SF.1.6.</td>
<td>Decipher the password provided by a user during the log in process.</td>
<td>Hidden</td>
</tr>
<tr>
<td>SF.1.7.</td>
<td>Check the validity of the information (user name and password) provided by a user during the log in process.</td>
<td>Hidden</td>
</tr>
<tr>
<td>SF.1.8.</td>
<td>Provide a persistent storage mechanism.</td>
<td>Hidden</td>
</tr>
<tr>
<td>SF.1.9.</td>
<td>Start a new MMCS-ET session.</td>
<td>Hidden</td>
</tr>
<tr>
<td>SF.1.10.</td>
<td>Determine whether a (potential) user is a subscriber of the MMCS-ET service.</td>
<td>Hidden</td>
</tr>
<tr>
<td>SF.1.11.</td>
<td>Check whether a user has started a new MMCS-ET session or not.</td>
<td>Hidden</td>
</tr>
<tr>
<td>SF.1.12.</td>
<td>Check whether a user is a teacher or a student.</td>
<td>Hidden</td>
</tr>
<tr>
<td>SF.1.13.</td>
<td>Check the status of a user (whether a user is logged-in or active).</td>
<td>Hidden</td>
</tr>
<tr>
<td>SF.1.14.</td>
<td>Change the status of a user.</td>
<td>Hidden</td>
</tr>
<tr>
<td>SF.1.15.</td>
<td>Invite a student to join an existing MMCS-ET session.</td>
<td>Evident</td>
</tr>
<tr>
<td>SF.1.16.</td>
<td>Maintain a list with all the logged-in users.</td>
<td>Evident</td>
</tr>
<tr>
<td>SF.1.17.</td>
<td>Maintain a list with all the users that participate in a MMCS-ET session (active users).</td>
<td>Hidden</td>
</tr>
<tr>
<td>SF.1.18.</td>
<td>Check whether a user participates in a MMCS-ET session or not.</td>
<td>Hidden</td>
</tr>
<tr>
<td>SF.1.19.</td>
<td>Enable a student to accept the invitation to join an existing MMCS-ET session.</td>
<td>Hidden</td>
</tr>
<tr>
<td>SF.1.20.</td>
<td>Invite a student that participates in a MMCS-ET session to direct communication.</td>
<td>Evident</td>
</tr>
<tr>
<td>SF.1.21.</td>
<td>Enable a student that participates in a MMCS-ET session to accept the invitation for direct communication from another student that participates in the same MMCS-ET session.</td>
<td>Evident</td>
</tr>
<tr>
<td>SF.1.22.</td>
<td>Remove a student from a MMCS-ET session in which he / she participates.</td>
<td>Evident</td>
</tr>
<tr>
<td>SF.1.23.</td>
<td>Terminate direct communication between two students that participate in the same MMCS-ET session.</td>
<td>Evident</td>
</tr>
<tr>
<td>SF.1.24.</td>
<td>Terminate an existing MMCS-ET session.</td>
<td>Evident</td>
</tr>
<tr>
<td>SF.1.25.</td>
<td>Terminate the MMCS-ET service.</td>
<td>Evident</td>
</tr>
</tbody>
</table>

Table 5.1.: Session related MMCS-ET functions.

The service requirements regarding the MMCS-ET (activity 3) consist mainly of a set of desirable
service functions that are structured according to the categories used for the structuring of the initial service
requirements. Thus, Tables 5.1., 5.2., and 5.3., depict the most important desirable session related,
interaction related, and collaboration support MMCS-ET functions respectively.

Taking into account the MMCS-ET functions and the initial service requirements, and considering the
teacher and the students as the only external actors of the MMCS-ET (i.e. the MMCS-ET service provider is
considered an internal actor), a number of use cases are deduced (activity 4) by using both the actor-based
and the event-based method (see Section 4.6.4.). These are:
Chapter 5: Validation and Evaluation of the Proposed Methodology

<table>
<thead>
<tr>
<th>Ref. #</th>
<th>MMCS-ET interaction related functions</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF.2.1.</td>
<td>Establish text communication between two users that participate in the same MMCS-ET session.</td>
<td>Evident</td>
</tr>
<tr>
<td>SF.2.2.</td>
<td>Send a (text) message to a user that participates in a MMCS-ET session.</td>
<td>Evident</td>
</tr>
<tr>
<td>SF.2.3.</td>
<td>Maintain a list with all the recently exchanged (text) messages between two users that participate in the same MMCS-ET session.</td>
<td>Evident</td>
</tr>
<tr>
<td>SF.2.4.</td>
<td>Establish A/V communication between two users that participate in the same MMCS-ET session.</td>
<td>Evident</td>
</tr>
<tr>
<td>SF.2.5.</td>
<td>Stop an existing A/V communication between two users that participate in the same MMCS-ET session.</td>
<td>Evident</td>
</tr>
<tr>
<td>SF.2.6.</td>
<td>Start a blocked A/V communication between two users that participate in the same MMCS-ET session.</td>
<td>Evident</td>
</tr>
<tr>
<td>SF.2.7.</td>
<td>Terminate an established A/V communication between two users that participate in the same MMCS-ET session.</td>
<td>Evident</td>
</tr>
<tr>
<td>SF.2.8.</td>
<td>Establish file communication between two users that participate in the same MMCS-ET session.</td>
<td>Evident</td>
</tr>
<tr>
<td>SF.2.9.</td>
<td>Send a file to a user that participates in a MMCS-ET session.</td>
<td>Evident</td>
</tr>
<tr>
<td>SF.2.10.</td>
<td>Maintain for each user that participates in a MMCS-ET session a list with all the other users, that participate in the same MMCS-ET session, and are in direct communication with him.</td>
<td>Evident</td>
</tr>
</tbody>
</table>

Table 5.2.: Interaction related MMCS-ET functions.

<table>
<thead>
<tr>
<th>Ref. #</th>
<th>MMCS-ET collaboration support functions</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF.3.1.</td>
<td>Establish a chat between all users that participate in a MMCS-ET session.</td>
<td>Evident</td>
</tr>
<tr>
<td>SF.3.2.</td>
<td>Send a (text) message to all users that participate in a MMCS-ET session.</td>
<td>Evident</td>
</tr>
<tr>
<td>SF.3.3.</td>
<td>Establish file communication between the teacher and all the students that participate in a MMCS-ET session.</td>
<td>Evident</td>
</tr>
<tr>
<td>SF.3.4.</td>
<td>Send a file to all students that participate in a MMCS-ET session.</td>
<td>Evident</td>
</tr>
<tr>
<td>SF.3.5.</td>
<td>Initiate a voting process between all users that participate in a MMCS-ET session.</td>
<td>Evident</td>
</tr>
<tr>
<td>SF.3.6.</td>
<td>Enable voting between all users that participate in a MMCS-ET session.</td>
<td>Evident</td>
</tr>
<tr>
<td>SF.3.7.</td>
<td>Calculate the outcome of a voting process between all users that participate in a MMCS-ET session.</td>
<td>Hidden</td>
</tr>
<tr>
<td>SF.3.8.</td>
<td>Inform all the users that participate in a MMCS-ET session about the outcome of the voting process.</td>
<td>Evident</td>
</tr>
</tbody>
</table>

Table 5.3.: Collaboration support MMCS-ET functions.

- Contact the MMCS-ET provider (start up the MMCS-ET service).
- Log in to the MMCS-ET provider domain.
- Start a new MMCS-ET session.
- Invite a student to join a MMCS-ET session.
- Join a MMCS-ET session after being invited.
- Invite a student (active user) to direct communication.
- Accept direct communication with a student (active user).
- Engage in text communication with an active user.
- Engage in file communication with an active user.
- Engage in A/V communication with a student (active user).
- Stop A/V communication with a student (active user).
- Start A/V communication with a student (active user).
- Terminate A/V communication with a student (active user).
- Engage in a chat with all active users.
- Engage in file communication with all students (active users).
- Start a voting process between all active users.
- Vote in a voting process between all active users.
- Present the outcome of a voting process to all the involved active users.
- Terminate direct communication between two students (active users).
Chapter 5: Validation and Evaluation of the Proposed Methodology

- Remove a student from a MMCS-ET session.
- Terminate a MMCS-ET session.
- Terminate the MMCS-ET service.

**Use Case:** Invite a student to join an MMCS-ET session.

**Actors:** Teacher (initiator), student.

**Type:** Primary.

**Description:** A teacher (logged in user) invites a student (logged in user) to participate in an educational / training session. On completion, the student is left to decide whether to accept this invitation by the teacher or not.

**Use Case:** Join a MMCS-ET session after being invited.

**Actors:** Student (initiator), teacher.

**Type:** Primary.

**Description:** A student (logged in user) accepts the invitation to participate in an educational / training session by a teacher. The teacher is informed about the decision of the student. On completion, the teacher and the student are ready to engage in direct (text, audio / video, and file) communication with each other.

**Use Case:** Engage in text communication with an active user.

**Actors:** Teacher (initiator), student or student (initiator), teacher or two students (one of them is the initiator; the one that sends the message).

**Type:** Primary.

**Description:** A user, participating in a MMCS-ET session, sends a (text) message to another user, participating in the same MMCS-ET session. Both users are ready to engage in direct communication with each other. On completion, the two users are engaged in text communication.

Figure 5.2.: The high-level form of the target use cases.

These use cases are examined in several iterative service development cycles. However, in Section 5.2.1., for reasons of clarity and simplicity, only three characteristic use cases are examined (target use cases) in the same way as if it was only one service development cycle. These (target) use cases are expressed in a high-level format (step 3, activity 4) in Figure 5.2.

5.2.1.3. Service Analysis Phase.

This phase starts with the definition of essential use cases which are expressed in an expanded format (activity 1). Figures 5.3., 5.4., and 5.5. depict the target use cases in their expanded form.

The middle section of an expanded use case (the "Typical Course of Events") is the most important as it describes in detail the required interaction between the actors and the telematic service (MMCS-ET). A critical aspect of this section is that it describes the most common sequence of activities needed for the successful completion of a service process. Alternative situations or exceptions that may arise with respect to the typical course are included in the final section of an expanded use case (the "Alternative Courses").

All the MMCS-ET use cases, together with their relationships, are illustrated in the (simplified) use case diagram (activity 2) of Figure 5.6.

Based on the expanded use cases, the (main) service conceptual model (step 4, activity 3) of Figure 5.7. is created for the MMCS-ET. More specifically, the noun phrases in the text of the expanded use cases (shown in italics) are considered as candidate service concepts and attributes (step 1 and step 3, activity 3). Furthermore, as an attribute is a logical data value of a service object, the main service conceptual model includes all the attributes for which the service requirements suggest or imply a need to remember
Chapter 5: Validation and Evaluation of the Proposed Methodology

Use Case: Invite a student to join an MMCS-ET session.

Actors: Teacher (initiator), student.

Purpose: Describes the invitation of a student to join an educational / training session by a teacher.

Overview: A teacher (logged in user) invites a student (logged in user) to participate in an educational / training session. On completion, the student is left to decide whether to accept this invitation by the teacher or not.

Type: Primary and essential.


Use Cases: The users involved in the current use case must have completed the use case “Log in to the MMCS-ET provider domain”.

Typical Course of Events:

<table>
<thead>
<tr>
<th>Actor Action</th>
<th>Service Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. This use case begins when a teacher (logged in user) decides to invite a student (logged in user) to an educational / training session (a MMCS-ET session).</td>
<td></td>
</tr>
<tr>
<td>2. The teacher specifies the user name of the student that he / she wants to invite to a MMCS-ET session, and the name of the service (MMCS-ET) that the MMCS-ET session is part of.</td>
<td></td>
</tr>
<tr>
<td>3. Examines whether the teacher has already started a new MMCS-ET session or not, by querying the MMCS-ET provider profile. If the teacher has not already started a new MMCS-ET session, see / initiate use case “Start a New MMCS-ET Session”. If the teacher has already started a new MMCS-ET session, continues.</td>
<td></td>
</tr>
<tr>
<td>4. Locates a list containing information about all the users that participate in the MMCS-ET session (active users) with the help of the MMCS-ET session.</td>
<td></td>
</tr>
<tr>
<td>5. Examines whether the user that initiated the invitation is active or not and whether he / she is a teacher or not, using the active user information list. Finds that the user that initiated the invitation is an active teacher.</td>
<td></td>
</tr>
<tr>
<td>6. Locates the MMCS-ET service profile by querying the MMCS-ET session.</td>
<td></td>
</tr>
<tr>
<td>7. Locates the user profile of the student that the teacher wants to invite (e. g. student A) by querying the MMCS-ET service profile.</td>
<td></td>
</tr>
<tr>
<td>8. Examines whether the user that the teacher wants to invite is a student or not, by querying the appropriate user profile. Finds that the user that the teacher wants to invite is a student.</td>
<td></td>
</tr>
<tr>
<td>9. Examines the status of student A by querying the user profile of student A. The student can be, either a logged in user or an active user (participating already in a session of the MMCS-ET service or of another service). Finds that student A is a logged in user.</td>
<td></td>
</tr>
<tr>
<td>10. Examines the status of student A by querying the user profile of student A. The student can be, either a logged in user or an active user (participating already in a session of the MMCS-ET service or of another service). Finds that student A is a logged in user.</td>
<td></td>
</tr>
<tr>
<td>11. Locates a catalog containing subscription information about all the users that are subscribers of the MMCS-ET service with the help of the user profile of student A.</td>
<td></td>
</tr>
<tr>
<td>12. Examines whether student A is a subscriber of the MMCS-ET service or not, using the user subscription catalog. Finds that student A is a subscriber of the MMCS-ET service.</td>
<td></td>
</tr>
<tr>
<td>13. Informs the MMCS-ET provider profile that student A has been invited by the teacher to join the MMCS-ET session.</td>
<td></td>
</tr>
<tr>
<td>14. Prompts student A to accept or reject the invitation of the teacher to join the MMCS-ET session.</td>
<td></td>
</tr>
</tbody>
</table>

Alternative Courses:

- **Event 5:** The user that initiated the invitation is not active and / or he / she is not a teacher. However, only a teacher can invite a student to join a MMCS-ET session and a teacher after starting a new MMCS-ET session is always active (see use case “Start a New MMCS-ET Session”). Indicate the error to the user.
- **Event 9:** The user that the teacher wants to invite is not a student (i.e. he / she is a teacher). However, only one teacher can participate in a MMCS-ET session at a given time. Indicate the error to the teacher.
- **Event 10:** Student A is an active user (i.e. he / she participates already in a session of the MMCS-ET service or of another service) 2. However, a student can participate in only one session of the MMCS-ET service or of another service at a given time. Indicate the error to the teacher. 2
- **Event 12:** Student A is not a subscriber of the MMCS-ET service. Indicate the error to the teacher.

1 The fact that the user that initiated the invitation has started a new MMCS-ET session ensures that this user is an active teacher. Therefore, events 4 and 5 are not necessary. However, they are included as a means of extra error checking.
2 The MMCS-ET session that student A already participates could be initiated, either by the specific teacher or by another teacher. In the former case, the teacher is already in direct communication with the student and he / she invites him / her again by mistake.

Figure 5.3.: The expanded form of use case “Invite a student to join a MMCS-ET session”.

Information. These attributes are not depicted in Figure 5.7. for reasons of clarity, but can be seen in Appendix A (Figure A.1.). Service concepts are related by associations which indicate some meaningful and interesting connection. Therefore, the main service conceptual model of the MMCS-ET includes all the associations (step 3, activity 3) for which knowledge of the corresponding relationship needs to be preserved.
Chapter 5: Validation and Evaluation of the Proposed Methodology

Use Case: Join a MMCS-ET session after being invited.
Actors: Student (initiator), teacher
Purpose: Describes the way that a student accepts the invitation to join an educational / training session by a teacher.
Overview: A student (logged in user) accepts the invitation from a teacher to participate in an educational / training session. The teacher is informed about the decision of the student. On completion, the teacher and the student are ready to engage in direct (text, audio / video, and file) communication with each other.
Type: Primary and essential.
Use Cases: The users involved in the current use case must have completed the use case "Invite a student to join a MMCS-ET session".

Typical Course of Events:

<table>
<thead>
<tr>
<th>Actor Action</th>
<th>Service Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. This use case begins when a student that was invited by the teacher (e.g. student A) to join a MMCS-ET session decides to accept the invitation.</td>
<td></td>
</tr>
<tr>
<td>2. Student A indicates his / her decision to accept the teacher invitation.</td>
<td>3. Informs the MMCS-ET provider profile for the decision of student A to accept the teacher invitation.</td>
</tr>
<tr>
<td>4. Locates the user profile of student A by querying the MMCS-ET provider profile.</td>
<td></td>
</tr>
<tr>
<td>5. Locates a catalog containing subscription information about all the users that are subscribers of the MMCS-ET service with the help of the user profile of student A.</td>
<td></td>
</tr>
<tr>
<td>6. Examines whether student A is still a subscriber of the MMCS-ET service or not, using the user subscription catalog.¹</td>
<td>Finds that student A is still a subscriber of the MMCS-ET service.</td>
</tr>
<tr>
<td>7. Examines the status of student A (whether he / she is logged in or active) by querying the user profile of student A.²</td>
<td>Finds that student A is a logged in user.</td>
</tr>
<tr>
<td>8. Changes the status of student A from logged in to active with the help of the user profile of student A in order to signify that student A is allowed to participate in the MMCS-ET session.</td>
<td></td>
</tr>
<tr>
<td>9. Informs the MMCS-ET session that student A is allowed to participate in the MMCS-ET session.</td>
<td></td>
</tr>
<tr>
<td>10. Prepares the MMCS-ET session to consider student A as one of its participants.</td>
<td></td>
</tr>
<tr>
<td>11. Locates a list containing information about all the users that participate in the MMCS-ET session (active users) with the help of the MMCS-ET session.</td>
<td></td>
</tr>
<tr>
<td>12. Updates the active user information list with information regarding student A.</td>
<td></td>
</tr>
<tr>
<td>13. Informs the MMCS-ET provider profile that student A is a participant of the MMCS-ET session.</td>
<td></td>
</tr>
<tr>
<td>14. Locates a list containing information about all the users that are in direct communication with student A with the help of the MMCS-ET provider profile.</td>
<td></td>
</tr>
<tr>
<td>15. Updates the directly communicating user information list that refers to student A with information regarding the teacher.</td>
<td></td>
</tr>
<tr>
<td>16. Indicates that student A participates in a MMCS-ET session.</td>
<td></td>
</tr>
<tr>
<td>17. Locates a list containing information about all the students that are in direct communication with the teacher.</td>
<td></td>
</tr>
<tr>
<td>18. Updates the directly communicating user information list that refers to the teacher with information regarding student A.</td>
<td></td>
</tr>
<tr>
<td>19. Informs the teacher that student A accepted his / her invitation and he / she is now a participant of the MMCS-ET service.</td>
<td></td>
</tr>
</tbody>
</table>

**Alternative Courses:**
- **Event 1:** Student A decides to reject the teacher invitation. Indicate the decision of student A to the teacher.
- **Event 6:** Student A is not a subscriber of the MMCS-ET service any more. Indicate the error to the teacher and to student A.
- **Event 7:** Student A is an active user (i.e. he / she participates in another MMCS-ET session initiated by a different teacher or in a session of another service). However, a student can participate in only one session of the MMCS-ET service or of another service at a given time. Indicate the error to the teacher and to student A.

This examination takes place again (see also event 12 of the use case "Invite a student to join a MMCS-ET session") because during the time lapsed between the teacher's invitation and student A's response (acceptance), the subscription of student A to the MMCS-ET service may came to an end.

This examination takes place again (see also event 10 of the use case "Invite a student to join a MMCS-ET session") because during the time lapsed between the teacher's invitation and student A's response (acceptance), student A may became a participant of another MMCS-ET session initiated by a different teacher or of a session of another service.

Figure 5.4.: The expanded form of use case "Join a MMCS-ET session after being invited".

for some duration ("need-to-know" associations), avoiding associations that are not strongly required and which do not increase understanding.

The main service conceptual model of Figure 5.7. is accompanied by the ancillary service conceptual models (step 5, activity 3) depicted in Figure 5.8. These models are the result of the customisation of the
Chapter 5: Validation and Evaluation of the Proposed Methodology

Use Case: Engage in text communication with an active user.

Actors: Teacher (initiator), student or student (initiator), teacher or two students (one of them is the initiator; the one that sends the message).

Purpose: Describes the way that text communication is realised between two users that participate in the same MMCS-ET session.

Overview: A user, participating in a MMCS-ET session, sends a (text) message to another user, participating in the same MMCS-ET session. Both users are ready to engage in direct communication with each other. On completion, the two users are engaged in text communication.

Type: Primary and essential.

Cross References: Service Functions: SF.2.1, SF.2.2, SF.2.3.

Use Cases: The users involved in the current use case must have completed the use case "Join a MMCS-ET session after being invited" (when one of them is a teacher and the other is a student) or the use case "Accept direct communication with a student (active user)" (when both are students).

**Typical Course of Events:**

<table>
<thead>
<tr>
<th>Actor</th>
<th>Action</th>
<th>Service Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>User A</td>
<td>Specifies the message M that he / she wants to send to user B.</td>
<td></td>
</tr>
</tbody>
</table>
3. Informs the MMCS-ET session that text communication is taking place between users A and B, as user A wants to send message M to user B.  
4. Examines whether text communication between user A and user B is allowed or not, by querying the MMCS-ET session.  
5. Locates a list containing information about all the messages exchanged recently between user A and user B. This list refers to user A.  
6. Updates the exchanged message information list that refers to user A with information regarding the message M that user A sends to user B.  
7. Informs user A about all the messages exchanged recently between him / her and user B, using the exchanged message information list that refers to user A.  
8. Transfers the message M from user A to user B, with the help of the MMCS-ET session.  
9. Locates a list containing information about all the messages exchanged recently between user A and user B. This list refers to user B.  
10. Updates the exchanged message information list that refers to user B with information regarding the message M that user A sent to user B.  
11. Informs user B about all the messages exchanged recently between him / her and user A, using the exchanged message information list that refers to user B. In this way user B is also informed about the message M that he / she received from user A. |

**Alternative Courses:**

- Event 4: Text communication between user A and user B is not allowed. The reason for this is that direct communication between user A and user B is being terminated (see use case "Terminate direct communication between two students (active users)" and use case "Remove a student from a MMCS-ET session"). Indicate the error to user A.

Figure 5.5.: The expanded form of use case “Engage in text communication with an active user”.

generic TINA-C information models of Figures 4.11, 4.12., 4.13., 4.14. and 4.15., according to the requirements of the MMCS-ET.

In an attempt to gain an understanding of the service behaviour, a service sequence diagram (activity 4) is created for the typical course of events of each one of the identified use cases (see Figure 5.9. for the diagrams corresponding to the target use cases) in the following way:

- A vertical line is drawn representing the MMCS-ET as a black box.
- Each actor that directly operates on the MMCS-ET is identified and a vertical line is drawn for him / her.
- From the use case typical course of events text, the (external) service events that each actor generates are identified and illustrated in the correct order on the diagram.
Chapter 5: Validation and Evaluation of the Proposed Methodology

The MMCS-ET

Figure 5.6.: The use case diagram of the MMCS-ET (simplified).

Figure 5.7.: The main service conceptual model of the MMCS-ET.
Figure 5.8: The ancillary service conceptual models of the MMCS-ET.
Chapter 5: Validation and Evaluation of the Proposed Methodology

Use case:

**InvReStudent(StudentNim-ne)** a “Invife a student to join a MMCS-ET session

**JoinSession(Teceleqr) “Join a MMCS-ET session after being invtecf

**EngageTextCom(DemtUserNarne, Use case., ’Engage in text communication Message) I with an ad&@ uset’

Figure 5.9.: Service sequence diagrams of the MMCS-ET for the target use cases.

The effect of the service operations that are revealed from the service sequence diagrams is described in *service operation contracts*. For each service operation, a service operation contract is constructed (activity 5). Its “Responsibilities” section informally describes the purpose of the service operation, while its “Post-conditions” section declaratively describes the state changes that occur to service IOs in the main service conceptual model of Figure 5.7., using a number of suitably selected statements (instance creation, instance deletion, attribute modification, association formed, association broken, and user interface activation). The service operation contracts of the service operations suggested by the target use cases can be seen in Figures 5.10., 5.11., and 5.12.

| Name: | InviteStudent(StudentName: String) |
| Responsibilities: | Invite a student (logged in user) to participate in an educational / training session (a MMCS-ET session). |
| Type: | MMCS-ET |
| Use Case: | “Invite a student to join a MMCS-ET session”. |
| Notes: | - Use a list containing information about all the logged in users. |
| - Use a list containing information about all the users that participate in a MMCS-ET session (active users). |
| - Use a catalog containing subscription information about all the users that are subscribers of the MMCS-ET service. |
| Exceptions: | - If the user that initiates the invitation is not active and / or he / she is not a teacher, indicate that it was an error. |
| - If the user that the teacher wants to invite is not a student and / or he / she is already active, indicate that it was an error. |
| - If student A is not a subscriber of the MMCS-ET service, indicate that it was an error. |
| Output: | - |
| Pre-conditions: | - A teacher logged in to the MMCS-ET provider domain. |
| - A student (e. g. student A) logged in to the MMCS-ET provider domain. |
| - The teacher decided to invite student A to a MMCS-ET session. |
| - The teacher specified the user name of student A. |
| Post-conditions: | - The User (corresponding to the teacher) was associated with the MMCS-ETProviderProfile (association formed). |
| - This User was associated with the MMCS-ETSession, based on the confirmation that User.SessionCreation was set to true (association formed). |
| - The MMCS-ETSession was associated with ActiveUserInformation, based on the satisfaction of the following conditions (association formed): |
| - The user that initiated the invitation was active. |
| - The user that initiated the invitation was a teacher. |
| - The MMCS-ETSession was associated with the MMCS-ETServiceProfile (association formed). |
| - The MMCS-ETServiceProfile was associated with LoggedInUserInformation regarding student A (association formed). |
| - The MMCS-ETSession was associated with the UserProfile (corresponding to student A) (association formed). |
| - This UserProfile was associated with the UserSubscriptionCatalog, based on the satisfaction of the following conditions (association formed): |
| - UserProfile.Role was already set to student. |
| - UserProfile.Status was already set to logged in. |
| - The UserSubscriptionCatalog was associated with UserSubscriptionInformation, based on the confirmation that student A was (at that time) a subscriber of the MMCS-ET service (association formed). |
| - The UserProfile (corresponding to student A) was associated with the MMCS-ETProviderProfile (association formed). |
| - The MMCS-ETProviderProfile was associated with the User (corresponding to student A) (association formed). |
| Interaction with student A was prepared (user interface activation). |

Figure 5.10.: Service operation contract of service operation “InviteStudent”. 
Chapter 5: Validation and Evaluation of the Proposed Methodology

### JoinSession(TeacherName: String)

**Responsibilities:** Accept the invitation to participate in an educational / training session (MMCS-ET session) by a teacher.

**Type:** MMCS-ET.

**Cross References:** Service functions: SF.1.8., SF.1.10., SF.1.13., SF.1.14., SF.1.17., SF.1.19.

**Use Case:** "Join a MMCS-ET session after being invited".

**Notes:**
- Use a catalog containing subscription information about all the users that are subscribers of the MMCS-ET service.
- Use a list containing information about all the users that participate in a MMCS-ET session (active users).
- Use a list containing information about all the users that are in direct communication with a specific user.

**Exceptions:**
- If student A is not a subscriber of the MMCS-ET service any more, indicate that it was an error.
- If student A is already active, indicate that it was an error.

**Output:**

**Pre-conditions:**
- A teacher invited a student (e.g. student A) to join a MMCS-ET session.
- Student A decided to accept the teacher invitation.
- Student A indicated his / her decision to accept the teacher invitation.

**Post-conditions:**
- The User (corresponding to student A) was associated with the MMCS-ETProviderProfile (association formed).
- The MMCS-ETProviderProfile was associated with the UserProfile (corresponding to student A) (association formed).
- This UserProfile was associated with the UserSubscriptionCatalog (association formed).
- The UserSubscriptionCatalog was associated with UserSubscriptionInformation, based on the confirmation that student A was (at that time) still a subscriber of the MMCS-ET service (association formed).
- The UserProfile (corresponding to student A) was associated with the MMCS-ETSession based on the confirmation that UserProfile.Status was already set to logged in (association formed).
- UserProfile.Status (corresponding to student A) was set to active (attribute modification).
- The MMCS-ETSession was associated with ActiveUserInformation regarding student A who just joined the MMCS-ET session (association formed).
- The MMCS-ETSession was associated with the User (corresponding to student A) (association formed).
- This User was associated with DirectlyCommunicatingUserInformation regarding the teacher (association formed).
- User.SessionParticipation (corresponding to student A) was set to true (attribute modification).
- Interaction with student A was prepared (user interface activation).
- The MMCS-ETSession was associated with the User (corresponding to the teacher) (association formed).
- This User was associated with DirectlyCommunicatingUserInformation regarding student A (association formed).
- Interaction with the teacher was prepared (user interface activation).

#### Figure 5.11.: Service operation contract of service operation "JoinSession".

### EngageTextCom(DestUserName: String, Message: String)

**Responsibilities:** Send a (text) message to another user participating in the same MMCS-ET session.

**Type:** MMCS-ET.

**Cross References:** Service functions: SF.2.1., SF.2.2., SF.2.3.

**Use Case:** "Engage in text communication with an active user".

**Notes:**
- Use a list containing information about all the messages exchanged recently between two users.

**Exceptions:**
- If text communication between user A and user B is not allowed, indicate that it was an error.

**Output:**

**Pre-conditions:**
- Two users participated in the same MMCS-ET session and were ready to engage in direct communication.
- One of the users (teacher or student, e.g. user A) decided to send a (text) message (e.g. message M) to the other user (teacher or student, e.g. user B).
- User A specified the message M that he / she wanted to send to user B.

**Post-conditions:**
- The User (corresponding to user A) was associated with the MMCS-ETSession (association formed).
- The MMCS-ETSession was associated with TextCommunication (association formed).
- The User (corresponding to user A) was associated with ExchangedMessageInformation regarding message M (association formed).
- Interaction with user A was prepared (user interface activation).
- The MMCS-ETSession was associated with the User (corresponding to user B), based on the confirmation that text communication between users A and B is allowed (association formed).
- The User (corresponding to user B) was associated with ExchangedMessageInformation regarding message M (association formed).
- Interaction with user B was prepared (user interface activation).

#### Figure 5.12.: Service operation contract of service operation "EngageTextCom".

### 5.2.1.4. Service Design Phase

Taking into account all the artifacts produced so far, in the service design phase, a service interaction diagram in the form of a UML collaboration diagram is created (step 3, activity 3) for each one of the identified service operations. The objective is to fulfil the post-conditions of the corresponding service operations.
operation contracts, recognising however that the previously defined post-conditions are only an initial best
guess or estimate of what must be achieved, and therefore their accuracy should be questioned.

Figure 5.13.: Service interaction diagram for inviting a student to join a MMCS-ET session.

Figure 5.14.: Service interaction diagram for joining a MMCS-ET session after being invited.

From these service interaction diagrams (Figures 5.13., 5.14., and 5.15. depict the service interaction
diagrams for the target use cases) the way that the MMCS-ET service COs communicate via messages in
order to fulfil the service requirements is evident. The participating MMCS-ET service objects are drawn
from the main service conceptual model of Figure 5.7. (see Table 5.4. for a mapping between service IOs
and service COs), after taking into account the service components proposed by the TINA-C service
architecture (see Table 4.1. and Section 2.5.2.), the service requirements regarding the MMCS-ET, and
other previously created artifacts (step 1, activity 3).
Chapter 5: Validation and Evaluation of the Proposed Methodology

Figure 5.15.: Service interaction diagram for engaging in text communication with an active user.

Table 5.4.: Mapping of the service IOs of the main service conceptual model to appropriate service COs.

<table>
<thead>
<tr>
<th>Service IOs</th>
<th>Service COs</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveUserInfo</td>
<td>ActiveUserInfo</td>
</tr>
<tr>
<td>AudioVideoCommunication</td>
<td>AVCom</td>
</tr>
<tr>
<td>AudioVideoDevice</td>
<td>AVDevice</td>
</tr>
<tr>
<td>BroadcastedMessageInformation</td>
<td>BroadcastedMessageInfo</td>
</tr>
<tr>
<td>ChatFacility</td>
<td>ChatFacility</td>
</tr>
<tr>
<td>DirectlyCommunicatingUserInfo</td>
<td>DCUserInfo</td>
</tr>
<tr>
<td>ExchangedMessageInformation</td>
<td>ExchangedMessageInfo</td>
</tr>
<tr>
<td>FileCommunication</td>
<td>FileCom</td>
</tr>
<tr>
<td>LoggedinUserInfo</td>
<td>LoggedinUserInfo</td>
</tr>
<tr>
<td>MMCS-ETProviderProfile</td>
<td>PA</td>
</tr>
<tr>
<td>MMCS-ETServiceProfile</td>
<td>IA</td>
</tr>
<tr>
<td>MMCS-ETSession</td>
<td>SSM, USM</td>
</tr>
<tr>
<td>StreamBinder</td>
<td>StreamBinder</td>
</tr>
<tr>
<td>TextCommunication</td>
<td>TextCom</td>
</tr>
<tr>
<td>User</td>
<td>as-UAP, as-UAP</td>
</tr>
<tr>
<td>UserProfile</td>
<td>UserSecurityCatalog</td>
</tr>
<tr>
<td>UserSecurityInformation</td>
<td>UserSecurityInfo</td>
</tr>
<tr>
<td>UserSubscriptionCatalog</td>
<td>UserSubscriptionCatalog</td>
</tr>
<tr>
<td>UserSubscriptionInformation</td>
<td>UserSubscriptionInfo</td>
</tr>
<tr>
<td>Voting</td>
<td></td>
</tr>
</tbody>
</table>

The links between the MMCS-ET service objects are actually instances of the associations present in the main service conceptual model of Figure 5.7., represent connection paths between service object instances, and indicate that some form of navigation and visibility between the instances is possible (attribute, parameter, locally declared or global visibility). In order to illustrate the creation (or deletion) of service object instances (from service classes), a language-independent creation (or deletion) message (create or delete) is shown being sent to the instance being created (or deleted). Finally, it has to be noted that for the construction of the service interaction diagrams the generic TINA-C service scenario of user-provider roles, depicted in Figure 4.20. (see also Section 4.9.3.), is also considered (step 2, activity 3).

By analysing the service interaction diagrams of the MMCS-ET, all the service classes (together with their attributes and methods) participating in the software realisation of the MMCS-ET are identified and illustrated in the service design class diagram of Figure 5.16. (activity 4). The attributes of the service
classes are not depicted in Figure 5.16. for reasons of clarity. The associations present in this diagram satisfy the ongoing "memory needs" revealed by the service interaction diagrams and the navigability arrows on them indicate the direction of attribute visibility (non-attribute visibility is indicated by dependency relationships). Creation (deletion) -related methods are omitted from the service design class diagram; because they can have multiple (implementation specific) interpretations, and because they represent a very common activity. For similar reasons, accessing methods (those which retrieve or set attributes) are also excluded from depiction in the service design class diagram, in order to keep it concise and focused.

5.2.1.5. Service Implementation Phase.

Considering all the artifacts produced in the service design phase, the MMCS-ET was implemented on MS Windows NT 4.0 using Microsoft’s Visual C++ (ver. 6.0) together with Microsoft’s DCOM [BROW98], appropriately extended with a high-level API in order to support continuous media interactions (see Chapter 6), and was executed on a number of workstations connected via a 10 Mbit/s Ethernet LAN [ADA00a][PAP98a]. All the interconnected workstations belong to the same (MS Windows NT) domain and one of them functions as a primary domain controller.

Figure 5.16.: The service design class diagram of the MMCS-ET (simplified).

Before starting to implement the service code (activity 3), each service class of the service design class diagram of the MMCS-ET (see Figure 5.16.) is specified using Microsoft’s Interface Definition Language (MIDL), which is an extension of DCE’s IDL (activity 1). The UserAgent (UA) service class, which
corresponds to the user profile service concept (see Figure 5.7), is used as an example. This service class
has only one interface (IUserAgent), which is defined using MIDL in the following way:

```c
// Definition of an interface
{
    uuid(36ECA947-5DC5-11D1-BD6F-204C4F4F5020),
    helpstring("IUserAgent Interface"),
    pointer_default(unique)
} interface IUserAgent : IUnknown
{
    typedef enum {type1, type2, type3, type4} ErrorType;
    HRESULT PrepareUA([in] BSTR* UserInfo, IUnknown* PA_ref);
    HRESULT RequestNewSession([in] BSTR* ServiceName,
                                [in] IUnknown* ssUAP_ref, [out] ErrorType* ErType);
    HRESULT InviteStudent([in] BSTR* TeacherName,
                           [in] IUnknown* SS_M_ref, [out] ErrorType* ErType);
    HRESULT JoinSession([in] BSTR* TeacherName,
                        [in] IUnknown* ssUAP_ref, [out] ErrorType* ErType);
    HRESULT PrepareRemoval();
    HRESULT NotifyEndSession();
    HRESULT TerminateService();
};
```

The purpose of the interface attribute uuid is to match the 128 bit universally unique identifier (that is
generated by a utility program called uuidgen under MS Windows operating systems) with the interface
that is currently being defined. This IID (Interface ID) distinguishes the IUserAgent interface from all other
interfaces. The interface attribute helpstring is used to introduce a string that helps to identify the interface
(e.g. in interface browsers), while the interface attribute pointer default is related with marshaling
pointers and can hold one of the following three values: ref (reference), unique, and ptr (pointer) [GRIM97].

In MIDL, functions return a 32 bit value called a HRESULT. A zero value for the most significant bit of a
HRESULT indicates that a function has executed successfully, while a one indicates that an error occurred.
The other bits represent the subsystem that generated the error and error information. The parameter list for a
function uses the in attribute to inform the compiler that the parameter is passed from the client to the server
and the out attribute to inform the compiler that the parameter is passed from the server back to the client.
Finally, the IUserAgent interface must inherit, either directly or indirectly, from the IUnknown interface,
which provides functionality required by all COM objects. Thus, all COM interfaces export the three
functions [QueryInterface(), AddRef(), Release()] of IUnknown in their public interface [RUBI99].

The UserAgent service class is defined as a coclass (component class), which is a top-level object in a
COM object hierarchy, in the following way:

```c
[ uuid(543FB20E-6281-11D1-BD74-204C4F4F5020) ]
coclass UserAgent
{
    [default] interface IUserAgent;
};
```
Chapter 5: Validation and Evaluation of the Proposed Methodology

The service code that is produced after implementing all the service classes of the service design class diagram of the MMCS-ET (see Figure 5.16.) has the characteristics of Table 5.5.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>MMCS-ET code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of code lines</td>
<td>7000</td>
</tr>
<tr>
<td>Number of COM objects</td>
<td>34</td>
</tr>
<tr>
<td>Number of user-defined methods / functions</td>
<td>273</td>
</tr>
<tr>
<td>Error checking</td>
<td>Examination of HRESULT</td>
</tr>
<tr>
<td>Threading</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The simple single threaded model,</td>
</tr>
<tr>
<td></td>
<td>The Single Threaded Apartment (STA) model,</td>
</tr>
<tr>
<td></td>
<td>The MultiThreaded Apartment (MTA) model</td>
</tr>
<tr>
<td></td>
<td>Aggregation, Containment</td>
</tr>
<tr>
<td>Reuse of COM objects</td>
<td>Callback interfaces, Connection points</td>
</tr>
<tr>
<td>Notification of asynchronous events</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.5.: Implementation characteristics of the MMCS-ET code.

The first three characteristics (number of code lines, number of COM objects, and number of user-defined methods / functions) reveal the complexity and difficulty of implementing the MMCS-ET, especially when considering that at the beginning of this validation attempt DCOM was a very new technology. The other implementation characteristics included in Table 5.5. are related with the exploitation of certain DCOM capabilities in the MMCS-ET code. These characteristics are:

- **Error checking**: In order to track potential problems, the HRESULT value is examined after every function call, in a way that reveals the error without affecting the clarity and compactness of the code (see also Section A.5.).

- **Threading**: Specific parts of the service code are executed concurrently by specifying threads. Each thread has its own stack and program registers and is managed by the operating system. Multiple threads can be run independently of each other and the operating system schedules their execution. All three COM threading models (the simple single threaded model, the single threaded apartment model, the multithreaded apartment model) are applied and examined (see also Section 6.3.3.).

- **Reuse of COM objects**: COM object specialisation (and thus reuse) is realised by aggregation and containment (see also Section 6.3.1.).

- **Notification of asynchronous events**: Usually a client connects to a COM object (server) and uses its methods. However, in some cases, the server may need to notify the client whenever some asynchronous event takes place. This is done with the use of callbacks, which are functions on the client that are called by the server, or with the use of connection points, which are specialisations of callbacks.

Finally, during the service implementation phase, the user interface of the MMCS-ET is implemented (activity 4). It consists of the following windows:

---

4 The DCOM specifications matured around 1997 and Microsoft started to officially support DCOM from the beginning of 1998. Work on this validation attempt started in 1998 and the first results appeared at the beginning of 1999 [ADA99a].
Chapter 5: Validation and Evaluation of the Proposed Methodology

Figure 5.17.: The login window of the MMCS-ET.

- **Login window:** It prompts the teacher and the students to specify their user name and password. If this data is correct the corresponding user is considered to be logged-in and the following window appears. The login window can be seen in Figure 5.17.

Figure 5.18.: The participant choice window of the MMCS-ET.

- **Participant choice window:** It prompts a logged-in user to choose another logged-in user (from a dynamically updated list) in order to start a MMCS-ET session with him/her. The restrictions mentioned in Section 5.2.1.2. regarding the role of the inviting user apply. In case of success, an MMCS-ET session is started and the following two windows appear. The participant choice window can be seen in Figure 5.18.

Figure 5.19.: The main user window of the MMCS-ET.

- **Main user window:** It refers to the way that two users communicate when they are in direct communication. Thus, its upper part is used for A/V communication and its lower part for text.
communication. This window is also used to terminate the direct communication between two users. It can be seen in Figure 5.19.

![Common activities window](image)

**Figure 5.20.:** The common activities window of the MMCS-ET.

- **Common activities window:** It is used for chat with all the active users, file exchange (file communication), and voting. It can be seen in Figure 5.20.

### 5.2.2. Additional Simple Validation Attempts.

As was explained at the beginning of Section 5.2., the proposed service development methodology is also validated by applying it to several simple (but realistic and characteristic [BYER92][BYER93]) service scenarios ⁵, which can be considered as simple telecommunications services. These simple validation attempts, due to their (relatively) limited scope and focused nature, were repeated a number of times during the construction of the proposed methodology and contributed significantly to its synthesis and structuring. Because of their importance and because they reveal the range of service creation activities that the proposed methodology can be used for, the scope of the simple validation attempts is presented in the following paragraphs by focusing on the description of the service scenarios that they incorporate.

The service scenarios that were considered are the following:

#### Distributed collaborative design

Many design processes involve the participation of a number of different people with particular skills, who may be located in different (geographically distributed) places, according to their specialities. In this service scenario, which is depicted in Figure 5.21., three designers (e.g. in different branches of an industrial engineering company) at three sites collaborate in a design exercise. For this reason, they need to communicate effectively with each other by participating in a multimedia conference, share access to a variety of files (e.g. Computer Aided Design - CAD files), and be able to display and process these files. These capabilities will enable multiple, remote, and iterative contributions to the target design process, which is focused on an objective related to the processing of a shared file.

---

⁵ In order to keep the service scenarios simple, only the core functionality was taken into account with appropriate simplifications whenever necessary. Furthermore, the consideration of user interface matters was kept to a minimum.
Chapter 5: Validation and Evaluation of the Proposed Methodology

This service scenario is likely to be increasingly prevalent in the future. Benefits include shorter design cycles, decreased travel requirements, and reduced isolation of designers. Because of the multimedia requirements and the large data volumes of objects like CAD files, broadband facilities are likely to be required (although perhaps only on demand for large bursts of data) [PAP98c].

Distributed case handling

This service scenario refers to the handling and management of a series of documents (possibly of a multimedia nature) interrelated by a common theme (termed cases) by people (clerks) which are distributed across one or more organisations and domains. More specifically, as can be seen in Figure 5.22., two clerks are linked to a central case management system. A supervisor at a third site is also indicated. The case management system allocates cases to clerks and manages the flow of work. It stores the case files centrally and can automatically forward a file to the appropriate clerk. Thus, this service scenario allows both remote access to centralised records and automated distribution / management of work.

Remote monitoring

Remote monitoring can cover a wide variety of situations, ranging from monitoring of people to monitoring of machines / places, where knowledge of what is happening at a place remote from the inquirer needs to be available. As can be seen in Figure 5.23., in this service scenario a patient at home and his local therapist (or doctor) are supported by a remote therapist that provides periodic check-ups when the local...
therapist is absent from the patient’s home. The local therapist contacts the service centre and arranges for a periodic check-up. This is then provided with the use of a videocamera which allows the remote therapist to observe the patient.

**Figure 5.23.: Remote monitoring.**

**Remote database access**

In this service scenario, which is depicted in Figure 5.24., three different forms of remote database access are considered. Type 1 involves an interactive search / information display session (e.g. similar to the one established during the use of Internet search engines). Type 2 involves transmission of a query to the database and the subsequent downloading of data from the database to the subscriber. Type 3 is similar to Type 2, except that in this case the initial searching / querying is interactive, although it is followed again by a downloading of the required data.

**Figure 5.24.: Remote database access.**

**Remote database utilisation**

This service scenario is conceptually related to the previous one ("Remote database access"). As can be seen in Figure 5.25., two professionals (e.g. trainers or teachers) are creating and using a (multimedia) database containing multimedia training material. The first trainer loads the training material and the second trainer searches for and retrieves particular parts of the training material, annotates them with text (and possibly with sound / voice and / or video) and re-loads them in the database.

**Remote access to expertise**

This service scenario allows people to consult human or machine sources of expertise and enables access to expertise to be interactive and efficiently shared by many people who may be geographically distributed.
Chapter 5: Validation and Evaluation of the Proposed Methodology

It is important in providing access to remote expertise in various business sectors, including medicine, manufacturing and transportation. In Figure 5.26, an example from a manufacturing factory has been chosen [PAPA96]. In this case the target is stationary and the expert can be either stationary or mobile. In other cases, the reverse situation may apply (e.g. in transportation).

Figure 5.25: Remote database utilisation.

In this service scenario, the technician contacts a remote expert for support in diagnosing and repairing a fault in a machine. The expert downloads data about the machine’s performance from the on-board computer. To obtain more information, the expert controls an on-board camera and views moving image sequences of the faulty machine. The expert is able to ask specific questions to another expert (not shown in Figure 5.26), who is located elsewhere.

Figure 5.26: Remote access to expertise.

Remote application running

As can be seen in Figure 5.27, this service scenario is actually a remote typesetting scenario. More specifically, an author makes use of a remote typesetting facility to typeset and print the final copy of his/her document. The first task involves interactive exchange with the typesetting system to agree on the parameters for layout and printing. The file is then transferred and periodic checks are made regarding the progress of printing.

Figure 5.27: Remote application running.
Entertainment on demand (pay-per-view)

With this service scenario, a wide range of entertainment material can be distributed. Rather than being broadcasted, the entertainment material can be ordered specifically, when required. In this way, the range of subject matters on offer can be increased to satisfy specialist audiences. This service scenario is depicted in Figure 5.28., where a domestic subscriber orders and views a film on a pay-per-view basis.

![Image of Figure 5.28: Entertainment on demand (pay per view).]

Remote consultation

According to this service scenario, which can be seen in Figure 5.29., a doctor and/or a patient, involved in a local consultation, decide to call another expert located at another (remote) site, because (for example) another opinion may be required or because of other difficulties encountered.

![Image of Figure 5.29: Remote consultation.]

Social conversation

Figure 5.30. presents two possible realisations of a "simple" social conversation service scenario (possibly) between two domestic subscribers. More specifically, two types of session are presented: one involving full A/V interaction from the start, and another involving an initial audio link with the video channel opened when both parties agree. Both types require the support of continuous media interactions (see Chapter 6).

![Image of Figure 5.30: Social conversation.]

186
5.2.3. Concluding Remarks.

In order to validate the service creation methodology that was proposed and presented in Chapter 4, demonstrate its feasibility, illustrate its utility and gain more insight into the issues involved, the methodology has been applied to the development of an actual telematic service (a MMCS-ET) and to several simple service scenarios.

These validation attempts revealed and provided strong evidence that the proposed methodology can be used efficiently for the development of new telecommunications services and that it is correct and effective as it can lead to the desired outcome, i.e. a successful telematic service that satisfies the requirements of its users. Furthermore, besides ensuring the maturity and practical value of the proposed methodology, the validation attempts, and especially the development of the MMCS-ET as is presented in Section 5.2.1. and Appendix A, illustrate and clarify the way that the methodology can be applied to the development of specific telematic services, and indicate the range of service creation problems that it can solve.

In general, the validation of the proposed service development methodology in this section, reinforced its position and highlighted its significance in the proposed telecommunications service engineering framework of Figure 1.1., as it eliminated any doubt regarding the value of the methodology and proved that it truly satisfies Requirements 1, 2, 5, 8, 9, 11, and 14 (see Section 1.3. and Section 4.13.) and has the benefits identified in Section 4.13. Additionally, the validation efforts resulted to the improvement of the proposed methodology and to its enhancement which will be examined in the following section, focusing on the utilisation of design patterns in the service engineering area.

5.3. Exploitation of Design Patterns.

When the number of service concepts and service COs involved in the development of a telematic service increase significantly, the successful application of the proposed service creation methodology can be a difficult task. Furthermore, there is great variability in the potential quality of responsibility assignment and the design of interaction between service COs (see also Section 4.9.3.). Therefore, this section examines the exploitation of design patterns in the service engineering area as an efficient potential solution to these problems, in an attempt to enhance and improve the proposed methodology [ADA99b][ADA00d][ADA00f].

Closely related to the concept of design patterns and essential for the accurate comprehension of it, is the concept of (object-oriented) frameworks. A framework can be defined as a set of prefabricated functional elements (e.g. service components), together with some architectural concepts that define the constraints to put these elements together. The architecture includes the rules that can be used to integrate the functional elements and to define possible flows of control between them [EVIT00]. The scope of design patterns is more narrow; as they represent abstract solutions for specific problem classes. They capture the static and dynamic structure, and collaboration of a group of objects and can be defined in an abstract, language
independent way [GAMM95][HANS97]. While design patterns can be considered as a horizontal structure over a set of COs, frameworks can be considered as vertical, domain specific (e.g. telecommunications specific) configurations of components.

In the case of TINA-C, design patterns can be defined by identifying groups of interworking service objects, where every group is characterised by a micro-architecture that determines the way the objects interact to provide a solution for the specific aspects of a subproblem that arises during the development of a telematic service. Furthermore, a framework can be defined as the overall architecture, which specifies how the identified configurations of service objects can collaborate to implement a solution for the whole problem. Thus, a framework is a kind of construction kit for complete or semi-complete telematic services that has to be complemented and customised using inheritance techniques [ECKE97]. As an example, the TINA-C service architecture can be considered as a framework, which is also applicable to the proposed service development methodology.

In order to decrease the possibility of poor choices, improve the quality of the design of service interaction diagrams, increase the possibility of a skilful implementation, and assist significantly service developers in this process, a number of general purpose design patterns (specifically constructed for service engineering purposes) are proposed in order to be applied during the creation of service interaction diagrams, when assigning responsibilities to service objects and designing service object collaborations (step 3, activity 3 of the service design phase, Section 4.9.3.). The introduction of these design patterns in the proposed service development methodology implies the establishment of a common vocabulary and the definition of common design structures for all persons involved. Thus, they ensure the effective and efficient communication of architectural knowledge between the service developers. Furthermore, they assist to reduce the scope of the problem solving process in the case of service creation, because they support the identification of similar problems and similar solutions, as they encapsulate service design experience [GAMM95][LARM98].

The proposed general purpose design patterns are general principles and idiomatic solutions that aim to guide service developers in the design of telematic services. They are presented in the following paragraphs in a structured way, as named problem / solution pairs that can be applied in new contexts, with advice / guidance on how to apply them in specific service creation situations, which are derived from the development of the MMCS-ET (see Section 5.2.1.):

**Assign a responsibility to a service class / object**

**Pattern Name:** Assign a responsibility to a service class / object.

**Purpose / Problem:**
During the service design phase, when the interactions between service objects are defined, choices are necessary to be made regarding the assignment of responsibilities to service classes (i.e. which service classes will be responsible for the implementation of which methods).
Chapter 5: Validation and Evaluation of the Proposed Methodology

**Motivation:**
Successful responsibility assignment can result to telematic services that are easier to maintain and extend, increasing the possibility of reusing their service components to other telematic services.

**Solution:**
A responsibility is assigned to the service class that has the necessary information to fulfil the responsibility (information expert).

**Applicability / Example:**
This design pattern is used extensively in all the service interaction diagrams that were created during the development of the MMCS-ET (see Section 5.2.1.4. and Section A.4.) as it represents a basic design principle. For example, it is used for the assignment of responsibilities to all the service COs that appear in the service interaction diagram of Figure 5.13. It has to be noted that all the responsibilities that are to be assigned should be initially clearly and unambiguously stated.

**Discussion / Consequences:**
The fulfilment of a responsibility often requires information that is spread across different service classes, which will have to collaborate by interacting via messages. This pattern has a real-world analogy as in reality responsibility is given to individuals who have the information necessary to fulfil a task.

**Benefits:**
- Encapsulation is ensured, since service classes use their own information to fulfil tasks. Therefore, low coupling (see below) is supported, which leads to more robust and maintainable telematic services.
- Behaviour is distributed across the service classes that have the required information, encouraging in this way more cohesive “lightweight” service class definitions that are easier to understand and maintain. Therefore, high cohesion (see below) is supported.

*Create an instance of a service class*

**Pattern Name:** Create an instance of a service class.

**Purpose / Problem:**
The creation of service objects is a very common task in object-oriented telematic services. Therefore, a general principle for the assignment of creation responsibilities is not only useful but also necessary.

**Motivation:**
The successful assignment of creation responsibilities can result to service designs that support low coupling (see below), increased clarity, encapsulation, and reusability.

**Solution:**
The responsibility to create an instance of service class A (a service object) is assigned to service class B when one of the following sentences is true $^6$:

$^6$ The first two sentences are preferred over the others, when more than one sentence is valid at the same time.
Chapter 5: Validation and Evaluation of the Proposed Methodology

- Service class B aggregates service objects of service class A.
- Service class B contains service objects of service class A.
- Service class B records service objects of service class A.
- Service class B closely uses service objects of service class A.
- Service class B has the necessary initialising data to create a service object of service class A.

![Service Interaction Diagram]

Figure 5.31.: Creating an instance of the ActiveUserInfo list.

**Applicability / Example:**

The (partial) service interaction diagram of Figure 5.31. is considered, which is contained in the service interaction diagram for starting a MMCS-ET session (see Section A.4.). Since a SSM records information about all the users that participate in a MMCS-ET session (active users) in an ActiveUserInfo list (see Section A.4.) and closely uses this list, then the SSM (according to this design pattern) is a good candidate to have the responsibility of creating instances of the ActiveUserInfo list.

**Discussion / Consequences:**

Sometimes a creator service class is found by looking for a service class that has the initialising data that will be passed in during the creation process. This is actually a case where the "Assign a responsibility to a service class / object" design pattern is applied.

**Benefits:**

Low coupling (see below) is supported, which implies lower maintenance dependencies and a higher possibility for reuse. Coupling is probably not increased because the created service class is likely to be already visible to the creator service class, due to the existing associations that motivated its choice as creator.

**Facilitate low coupling**

**Pattern Name:** Facilitate low coupling.

**Purpose / Problem:**

Coupling is a measure of how strongly one service class is connected to, has knowledge of, or relies upon other service classes. A service class with low (or weak) coupling is not dependent on too many other service classes. On the contrary, a service class with high (or strong) coupling relies upon many other service classes. Such service classes are undesirable because it is difficult to understand them in isolation, changes in related service classes force local changes, and it is more difficult to reuse them, as their use requires the

---

7 It is obvious that “too many” is context dependent and varies according to the telematic service under consideration.
additional presence of the service classes that they are dependent upon. Therefore, the support of low coupling in the service design phase is desirable and should be aimed at.

**Motivation:**

Low coupling supports low dependency and increased reuse (see Purpose / Problem above).

**Solution:**

A responsibility is assigned to a service class so that coupling remains low.

![Diagram](image)

**Figure 5.32:** Facilitating low coupling when sending a message from a teacher to a student.

**Applicability / Example:**

The two (partial) service interaction diagrams of Figure 5.32. present two alternative designs for sending a (text) message from a teacher to a student, and can be contained in the service interaction diagram for engaging in text communication with an active user (Figure 5.15.). Taking into account that TextCom will eventually be coupled to knowledge of the ActiveUserInfo list, it is evident that the (partial) service interaction diagram of Figure 5.32.(a) is characterised by an overall lower coupling and therefore is preferable. More specifically, this design does not increase the coupling of the SSM, which is a service class with many responsibilities in the MMCS-ET.

**Discussion / Consequences:**

There is no absolute measure of when coupling is too high. It is left to the service developer to estimate the current degree of coupling and assess if increasing it will lead to problems. In general, service classes which are inherently very generic in nature, and have a high probability for reuse, should be characterised by especially low coupling. It has to be stressed that some moderate degree of coupling between service classes is normal and necessary, as service classes fulfil tasks by collaborating via messages. Very little or no coupling between service classes yields to a poor service design because it leads to a few incohesive, bloated, and complex active service objects that do all the work, with many very passive zero-coupled service objects that act as simple data repositories.
Benefits:
This pattern supports the design of service classes that are simple to understand in isolation, not affected by changes in other service classes, and convenient to reuse.

Facilitate high cohesion

Pattern Name: Facilitate high cohesion.

Purpose / Problem:
Cohesion (or functional cohesion) is a measure of how strongly related and focused the responsibilities of a service class are. A service class with highly related responsibilities, which also does a moderate amount of work, has high cohesion. On the contrary, a service class with low cohesion does many unrelated things or does too much work. Service classes with low cohesion often represent a very "large-grain" of abstraction with responsibilities that should have been delegated to other service classes. Such service classes are undesirable because it is difficult to understand, maintain and reuse them, and because they are constantly affected by change in related service classes. Therefore, the support of high cohesion in the service design phase is desirable and should be aimed at.

Motivation:
High cohesion keeps the complexity in the service design phase manageable. A service class with high cohesion is advantageous because, due to the high degree of related functionality and the small number of operations, it is relatively easy to maintain, understand, and reuse.

Solution:
A responsibility is assigned to a service class so that cohesion remains high.

Applicability / Example:
The (partial) service interaction diagram of Figure 5.32.(a) is preferable over the one of Figure 5.32.(b), because it supports higher cohesion in the SSM. This is important, because due to the central role of the SSM in the MMCS-ET, the possibility of it to become a "bloated" incohesive service object is high.

Discussion / Consequences:
A service class is characterised by very low cohesion when it is solely responsible for many tasks in very different functional areas, and by low cohesion when it has sole responsibility for a complex task in one functional area. Moderate cohesion is the characteristic of a service class that has lightweight and sole responsibilities in a few different areas that are logically related to the service class concept, but not to each other. Finally, a service class is characterised by high cohesion when it has moderate responsibilities in one functional area and collaborates with other service classes to fulfil tasks when the involved effort is significant.

Benefits:
This pattern increases clarity and ease of comprehension of the service design, simplifies the maintenance of service classes, supports low coupling, and facilitates reuse.
Chapter 5: Validation and Evaluation of the Proposed Methodology

Except from the proposed general purpose design patterns which describe and codify fundamental principles of assigning responsibilities to service objects during the service design phase of the proposed methodology, more complicated design patterns, which are directly related to service design aspects, can be devised in order to simplify even more the construction of service interaction diagrams and facilitate considerably service developers, especially when designing complex telematic services. These service engineering design patterns encompass a rich functionality, proved to be useful in the simple validation attempts described in Section 5.2.2., and can be easily constructed by considering the service interaction diagrams that were created during the development of the MMCS-ET (see Section 5.2.1.4. and Section A.4.). More specifically, from each one of these service interaction diagrams a service engineering design pattern can be constructed. As an example, the service engineering design pattern which corresponds to the service interaction diagram of Figure 5.13., is described in the following paragraphs:

**Pattern Name:** Invite a user to join a service session.

**Purpose / Problem:**
The invitation of a logged in user (e.g. user B) by another logged in user (e.g. user A) to participate in a service session.

**Motivation:**
Most new telecommunications services incorporate a (service) session. This session is created / started by an authorised user of the service, who then invites other users to participate.

**Solution:**
This pattern is based on the service interaction diagram for inviting a student to join a MMCS-ET session (Figure 5.13.). It uses a list containing information about all the logged in users (LoggedInUserInfo list),

---

**Figure 5.33.:** Service interaction diagram for inviting a user to join a service session.

---

It is evident that the full understanding of the service interaction diagrams is essential. This can be achieved by considering the previously created artifacts.
Chapter 5: Validation and Evaluation of the Proposed Methodology

a list containing information about all the users that participate in the service session (ActiveUserInfo list), and a catalog containing information about all the users that are subscribers of the service (UserSubscriptionCatalog).

**Applicability / Example:**
The service interaction diagram for inviting a user to join a service session can be seen in Figure 5.33.

**Discussion / Consequences:**
It is assumed that both users A and B are logged in to the service provider domain, user A is already active, user B is not active, user B is a subscriber of the service, and user A who invites user B has the authorisation to do so.

**Benefits:**
This pattern ensures the application of the proposed general purpose design patterns, exploits important TINA-C functionality, and takes into account a number of necessary decisions involved in the invitation of a user to a service session.

Although the proposed general purpose and service engineering design patterns have been successfully applied during the validation attempts of the proposed methodology, it has to be stressed, for those interested in, either devising new specialised design patterns or applying existing ones [GAMM95] to the proposed methodology, that design patterns are abstract concepts. There is no guarantee that their usage will lead to design reusability, design portability, and abstract customisability. Furthermore, good design patterns, like good inheritance hierarchies, can not be invented in an easy way. They have to be chosen and designed very carefully [KOER97]. Otherwise, the introduction of insufficient and wrong chosen design patterns in the service creation process may hinder or even prevent the design and implementation of successful telematic services.

5.4. Evaluation of the Proposed Methodology.

The applicability, correctness, and practical value of the proposed service development methodology have been examined by the validation attempts described earlier in this chapter (Section 5.2.). However, the proposed methodology (together with its extensions that were presented in Section 5.3.), apart from being specialised for service creation purposes in the area of telecommunications service engineering, remains a methodology for the development of systems 9. Therefore, it is necessary to evaluate the proposed methodology and examine its completeness, in order to determine whether it addresses all the issues that a premium methodology for the development of systems should address. Furthermore, such an evaluation improves the critical understanding of the proposed methodology (of both the author and the reader of the evaluation), and provides an abstract framework that can be used as the basis for choosing service development methodologies when more of them appear.

---

9 A system is a set of interrelated elements [CHEC81]. In the case of the proposed methodology, a telematic service is a system.
For the evaluation of the proposed service development methodology the Normative Information Model-based Systems Analysis and Design (NIMSAD) framework [JAYA94] is selected (among other methodology evaluation frameworks [9]), because it is appropriate for the evaluation of object-oriented methodologies for the development of information systems, it examines “soft” and “hard” methodology characteristics in a balanced manner, it has been successfully applied in many cases, and it enjoys a wider acceptance in the information systems community [AVIS95][BERK98]. NIMSAD is a methodology-independent systemic framework that can be used for understanding and evaluating methodologies and aims to promote the comprehension of problem-solving processes in general. It is based on the models and epistemology of systems thinking [CHEC81].

According to the NIMSAD framework, a methodology is considered to be an explicit way of structuring one’s thinking and actions. Methodologies contain model(s) and reflect particular perspectives of “reality” based on a set of philosophical paradigms. A methodology should specify “what” steps are necessary, “how” to perform these steps, and most importantly the reasons “why” these steps should be taken in that particular order. Therefore, any explicitly ordered set of activities / steps with a convincing rationale and ways of performing the steps to bring about a certain transformation can qualify as a methodology. However, those who follow any established methodology structure should understand fully the reasons for following the suggested steps.

![Diagram of the NIMSAD framework](image)

Figure 5.34.: The essential elements of the NIMSAD framework.

As can be seen in Figure 5.34., the NIMSAD framework consists of the following three essential elements, which constitute the main perspectives for evaluating a methodology [JAYA94]:

- **Element 1- The “problem situation” (the methodology context):** This element refers to the type of problems that a methodology can solve and to the owners of these problems.

- **Element 2 - The intended problem solver (the methodology user):** This element refers to the people that apply a methodology to a particular “problem situation”. These problem solvers select some elements of

---

Examples are the Avison and Fitzgerald’s framework, the Tudor and Tudor’s framework, the DESMET (Determining an Evaluation Method for Software Methods and Tools) framework, etc.

A “soft” methodology characteristic considers mainly human factors, while a “hard” one is more technically oriented.
the "problem situation" to be relevant. This selection, although sometimes prompted by the used methodology, is significantly influenced by the "mental construct" of the methodology users. The "mental construct" is one's mental structure and consists of a number of useful parts namely perceptual process, values, ethics, motives and prejudices, reasoning ability, experiences, skills and knowledge sets, structuring process, models, frameworks, and roles.

- **Element 3 - The problem-solving process (the methodology process):** This element refers to the way that problem solving is performed. It is decomposed into the three essential phases below, which are expanded to form eight detailed stages that are applicable to any problem-solving process:

  - **Phase 1 - Problem Formulation:**
    
    - **Stage 1 - Understanding of the "situation of concern":** The "situation of concern" is a situation in which various people perceive problems. The exact boundaries of this problematic situation determines the focus of the investigation and establishes the real "situation of concern".
    
    - **Stage 2 - Performing the diagnosis:** Diagnosis is the explicit projection or expression of the understanding that is gained from the investigation of the "situation of concern". Therefore, this stage attempts to answer the question "Where are we now?".
    
    - **Stage 3 - Defining the prognosis outline:** Prognosis is the expression of a desired situation. Therefore, this stage is concerned with defining a desired state for the current "situation of concern" and attempts to answer the question "Where do we want to be and why?".
    
    - **Stage 4 - Defining problems:** Problems are preventing the realisation of the desired state. Therefore, this stage focuses on what is preventing the diagnosis outline (current state) from changing to the prognosis outline (desired state).
    
    - **Stage 5 - Deriving notional systems:** Notional systems are notional constructs that are considered to be relevant for resolving problems in the "situation of concern". Its parts are expected to become the basis for overcoming problems and therefore for transforming the "situation of concern".

  - **Phase 2 - Solution Design:**
    
    - **Stage 6 - Performing conceptual / logical design:** The outcome of this stage is the production of an agreed and acceptable logical design specification which states the nature and the function of the logical parts of the desired state of the "situation of concern". Logical means that these parts can be argued as being useful and essential to the realisation of the notional systems and thus to the realisation of the desired state of the "situation of concern".
    
    - **Stage 7 - Performing physical design:** Physical design can be considered as the selection of "ways and means" of realising the logical design within a given set of resources, facilities, and other environmental factors.
Chapter 5: Validation and Evaluation of the Proposed Methodology

- Phase 3 - Design Implementation:

Stage 8 - Implementing the designs: Implementation is concerned with the realisation of the notional systems within the context of the "situation of concern".

Table 5.6.: Evaluating the proposed methodology according to the NIMSAD framework: The "problem situation" element.

<table>
<thead>
<tr>
<th>Questions and Answers about the &quot;Problem Situation&quot; (the Methodology Context)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q: Who are the clients?</td>
</tr>
<tr>
<td>A: Step 2, activity 1, requirements capture and analysis phase.</td>
</tr>
<tr>
<td>Q: What degree of commitment do the clients have?</td>
</tr>
<tr>
<td>A: Step 2, activity 1 and step 1, activity 2, requirements capture and analysis phase.</td>
</tr>
<tr>
<td>Q: How the clients are brought within the problem-solving process?</td>
</tr>
<tr>
<td>A: Adoption of an iterative and incremental approach.</td>
</tr>
<tr>
<td>Step 1, activity 2 and activity 7, requirements capture and analysis phase.</td>
</tr>
<tr>
<td>Q: Are the concerns of the clients legitimate?</td>
</tr>
<tr>
<td>A: Step 2, activity 1 and step 3, activity 2, requirements capture and analysis phase.</td>
</tr>
<tr>
<td>Q: Which are the needs of the clients regarding the &quot;problem situation&quot;?</td>
</tr>
<tr>
<td>A: Step 2 and step 3, activity 2, requirements capture and analysis phase.</td>
</tr>
<tr>
<td>Q: What types of &quot;problem situation&quot; are the methodology users facing?</td>
</tr>
<tr>
<td>A: Related to service development / service creation.</td>
</tr>
<tr>
<td>Q: What &quot;problem situation&quot; does the methodology claim is the type for which it is suitable?</td>
</tr>
<tr>
<td>A: Related to service development / service creation.</td>
</tr>
<tr>
<td>Q: What evidence and explanations are offered?</td>
</tr>
<tr>
<td>A: Specific guidelines, consideration of TINA-Q, proper validation.</td>
</tr>
<tr>
<td>Q: What are the characteristics of the &quot;problem situation&quot;?</td>
</tr>
<tr>
<td>A: Activity 2, requirements capture and analysis phase.</td>
</tr>
<tr>
<td>Q: What type of culture (i.e. special requirements) affect the methodology?</td>
</tr>
<tr>
<td>A: Step 2 and step 6, activity 1, requirements capture and analysis phase.</td>
</tr>
<tr>
<td>Q: What type of &quot;reality&quot; do the clients perceive?</td>
</tr>
<tr>
<td>A: Step 2, activity 1 and step 1, step 2, and step 3, activity 2, requirements capture and analysis phase.</td>
</tr>
<tr>
<td>Q: Which are the dominant perceptions in the &quot;problem situation&quot;?</td>
</tr>
<tr>
<td>A: Requirements capture and analysis phase.</td>
</tr>
</tbody>
</table>

Table 5.7.: Evaluating the proposed methodology according to the NIMSAD framework: The problem solver element.

<table>
<thead>
<tr>
<th>Questions and Answers about the Problem Solver (the Methodology User)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q: Is true participation of the methodology users facilitated?</td>
</tr>
<tr>
<td>A: Yes, by offering specific guidelines, by discussing the proposed process, and by explaining important decisions.</td>
</tr>
<tr>
<td>Q: Are methodology users guided to find more information about specific matters?</td>
</tr>
<tr>
<td>A: Yes. References are given throughout the methodology.</td>
</tr>
<tr>
<td>Q: Is it possible for the methodology users to use appropriate software tools during their work?</td>
</tr>
<tr>
<td>A: Yes. See also Section 4.12.</td>
</tr>
<tr>
<td>Q: Does the methodology inform its intended users about the nature and type of concepts they need to be familiar with before applying the methodology?</td>
</tr>
<tr>
<td>A: Yes. It is a service development methodology. Familiarity with TINA-C and UML is desirable.</td>
</tr>
<tr>
<td>Q: What level of abstract and technical thinking, and what knowledge sets and skills does the methodology demand from its users before they are able to practice it?</td>
</tr>
<tr>
<td>A: They should be service developers or service designers (service engineers). Familiarity with TINA-C and UML is desirable.</td>
</tr>
<tr>
<td>Q: Is it possible for the methodology users to customise the methodology?</td>
</tr>
<tr>
<td>A: Yes. See also step 1, activity 1, requirements capture and analysis phase.</td>
</tr>
<tr>
<td>Q: Are the motives, the value sets, and the ethical behaviour of the methodology users important for the &quot;problem situations&quot; considered by the methodology? If yes, how they are taken into account?</td>
</tr>
<tr>
<td>A: No, they are not important.</td>
</tr>
<tr>
<td>Q: What models does the methodology offer to the methodology users in order to deal with the &quot;problem situation&quot;?</td>
</tr>
<tr>
<td>A: See Figures 4.11., 4.25., and 4.32.</td>
</tr>
<tr>
<td>Q: Is it necessary for the methodology users to have experience in the domain of the &quot;problem situation&quot;?</td>
</tr>
<tr>
<td>A: Yes. They should be service developers or service designers (service engineers). It is a specialised methodology.</td>
</tr>
</tbody>
</table>

Table 5.7.: Evaluating the proposed methodology according to the NIMSAD framework: The problem solver element.
Chapter 5: Validation and Evaluation of the Proposed Methodology

Questions and Answers about the Problem-Solving Process (the Methodology Process)

### Problem Formulation

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>What assistance does the methodology offer for identifying the independent actors involved in the “situation of concern”?</td>
<td>A: Step 6, activity 2, requirements capture and analysis phase.</td>
</tr>
<tr>
<td>Does the methodology alert its users to boundary construction?</td>
<td>A: Yes. Step 6, activity 2 and step 1, activity 4, requirements capture and analysis phase.</td>
</tr>
<tr>
<td>What assistance does the methodology offer for boundary construction?</td>
<td>A: Step 6, activity 2 and step 1, activity 4, requirements capture and analysis phase. Important is also the comprehension from previous steps.</td>
</tr>
<tr>
<td>What is the role of the clients with respect to the methodology?</td>
<td>A: Clients participate from the beginning. Adoption of an iterative and incremental approach.</td>
</tr>
<tr>
<td>Step 1, activity 2 and activity 7, requirements capture and analysis phase.</td>
<td></td>
</tr>
<tr>
<td>Does the methodology help its users to identify the sources of information?</td>
<td>A: Yes. Step 1, activity 2, requirements capture and analysis phase.</td>
</tr>
<tr>
<td>Does it suggest / discuss any particular method(s) of investigation?</td>
<td>A: Yes. Step 1, activity 2, requirements capture and analysis phase.</td>
</tr>
<tr>
<td>What skills does it highlight as being relevant or useful for conducting such an investigation?</td>
<td>A: Step 1, activity 2, requirements capture and analysis phase.</td>
</tr>
<tr>
<td>Does the methodology help with the formulation of the “situation of concern”?</td>
<td>A: Yes. Step 1, activity 2, requirements capture and analysis phase.</td>
</tr>
<tr>
<td>What are the implications of these for the subsequent steps of the methodology?</td>
<td>A: Their use in the creation of subsequent artifacts is explained.</td>
</tr>
<tr>
<td>Does the methodology facilitate the filtering of the gathered information?</td>
<td>A: Yes. Step 3 and step 4, activity 2, requirements capture and analysis phase.</td>
</tr>
<tr>
<td>Is it necessary to update regularly the characteristics of the “situation of concern”?</td>
<td>A: No, these are assumed to be stable. However changes are possible due to the iterative and incremental nature of the methodology.</td>
</tr>
<tr>
<td>Does the methodology alert its users about the importance of accurately describing the “situation of concern”?</td>
<td>A: Step 3, requirements capture and analysis phase.</td>
</tr>
</tbody>
</table>

### Performing the diagnosis

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>What modelling notions and techniques does the methodology offer for expressing the characteristics of the “situation of concern”?</td>
<td>A: Activity 2, requirements capture and analysis phase.</td>
</tr>
<tr>
<td>What are the implications of these for the subsequent steps of the methodology?</td>
<td>A: Their use in the creation of subsequent artifacts is explained.</td>
</tr>
<tr>
<td>Does the methodology facilitate the filtering of the gathered information?</td>
<td>A: Yes. Step 3 and step 4, activity 2, requirements capture and analysis phase.</td>
</tr>
<tr>
<td>Is it necessary to update regularly the characteristics of the “situation of concern”?</td>
<td>A: No, these are assumed to be stable. However changes are possible due to the iterative and incremental nature of the methodology.</td>
</tr>
<tr>
<td>Does the methodology alert its users about the importance of accurately describing the “situation of concern”?</td>
<td>A: Step 3, requirements capture and analysis phase.</td>
</tr>
</tbody>
</table>

### Defining the prognosis outline

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>What help does the methodology offer in defining the desired state regarding the “situation of concern”?</td>
<td>A: The accurate description of the “situation of concern” and suitable concepts (see the answer to the next question).</td>
</tr>
<tr>
<td>What modelling notions and techniques does the methodology offer for expressing the desired state?</td>
<td>A: Activity 3 and step 2, step 3, activity 4, requirements capture and analysis phase.</td>
</tr>
<tr>
<td>What are the implications of misinterpreting the desired state?</td>
<td>A: The validity of the service requirements is examined in subsequent service development cycles. Thus, any problems will be eventually discovered and corrected.</td>
</tr>
<tr>
<td>Does the methodology record the assumptions that are made in order to deal with the problems that prevent the desired state?</td>
<td>A: Yes. Activity 6, requirements capture and analysis phase and activity 6, service analysis phase.</td>
</tr>
<tr>
<td>Are all the defined problems considered to be of equal importance?</td>
<td>A: No. Step 3, activity 9, requirements capture and analysis phase.</td>
</tr>
<tr>
<td>Is it necessary to define all the problems from the beginning?</td>
<td>A: No, due to the iterative and incremental nature of the methodology. Use cases are scheduled to service development cycles.</td>
</tr>
</tbody>
</table>

### Defining problems

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Does the methodology explain what is preventing the desired state?</td>
<td>A: Yes, by accurately describing the desired state. Activity 4, requirements capture and analysis phase.</td>
</tr>
<tr>
<td>How does it help its users to make such an explanation?</td>
<td>A: Activity 1, activity 2, activity 3, and step 2 of activity 4, requirements capture and analysis phase.</td>
</tr>
<tr>
<td>Does the methodology record the assumptions that are made in order to deal with the problems that prevent the desired state?</td>
<td>A: Yes. Activity 6, requirements capture and analysis phase and activity 6, service analysis phase.</td>
</tr>
<tr>
<td>Are all the defined problems considered to be of equal importance?</td>
<td>A: No. Step 3, activity 9, requirements capture and analysis phase.</td>
</tr>
<tr>
<td>Is it necessary to define all the problems from the beginning?</td>
<td>A: No, due to the iterative and incremental nature of the methodology. Use cases are scheduled to service development cycles.</td>
</tr>
</tbody>
</table>

### Deriving notional systems

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Does the methodology derive notional systems from the problems identified?</td>
<td>A: Yes. Activities 1, 2, 3, 4, 5, and 7 of service analysis phase.</td>
</tr>
<tr>
<td>Does it offer any guidance for formulating the notional systems?</td>
<td>A: Yes. Activities 1, 2, 3, 4, 5, and 7 of service analysis phase.</td>
</tr>
<tr>
<td>Does the methodology consider the effect on the formulation of the notional systems by the characteristics of the “problem situation”?</td>
<td>A: Yes. It offers specific guidelines and considers TINA-C models. Step 5, activity 3, service analysis phase.</td>
</tr>
<tr>
<td>Does the methodology alert its users to the important decisions involved?</td>
<td>A: Yes. Activities 1, 2, 3, 4, 5, and 7 of service analysis phase.</td>
</tr>
<tr>
<td>Are changes to the notional systems possible?</td>
<td>A: Yes, due to the iterative and incremental nature of the methodology.</td>
</tr>
<tr>
<td>Does the methodology clarify the purpose of the notional systems?</td>
<td>A: Yes. Activities 1, 2, 3, 4, 5, and 7 of service analysis phase.</td>
</tr>
</tbody>
</table>

Table 5.8.: Evaluating the proposed methodology according to the NIMSAD framework:

The problem-solving process element (problem formulation).

The NIMSAD framework is designed to be used for asking questions to the methodologies as to what elements of the framework they address, in what order and how they address them. Thus, the role of the
framework is to help question the methodologies as to what they attempt to transform, why they try to transform it and how they facilitate the methodology users to undertake the transformation, and provide in this way the means for debating the rationale of the steps of the methodologies and their structure. This kind of evaluation measures the effectiveness of the problem-solving process and the problem solver in the "problem situation", but it has to be noted that its application requires to consciously consider the rational basis and the relationships of the activities suggested by the steps of the methodology that is going to be evaluated.

### Questions and Answers about the Problem-Solving Process (the Methodology Process)

#### Performing conceptual / logical design:

**Q:** Does the methodology accept the notional systems as the starting point of the logical design?

**A:** Yes.

**Q:** Does it rely on its users’ experience or expertise for the design of the solutions?

**A:** It requires detailed guidance but design skill is required. Activity 3 of service design phase.

**Q:** Does it distinguish between logical and physical design stages?

**A:** Yes. Activity 8 of requirements capture and analysis phase and activity 5 of service design phase.

**Q:** What techniques does the methodology suggest for the formulation of solutions?

**A:** Activities 1, 2, 3, 4, 5, and 6 of service design phase.

**Q:** Are the application of these techniques explained sufficiently through a series of steps?

**A:** Yes. Activities 1, 2, 3, 4, 5, and 6 of service design phase.

**Q:** How does the methodology ensure that the clients agree with the logical design?

**A:** Adoption of an iterative and incremental approach.

**Q:** Are there any aspects of the "situation of concern" that are excluded by the methodology in its design support?

**A:** No.

**Q:** Does the methodology consider / support the modelling of behavioural issues?

**A:** Yes. Activity 1 and activity 3, service design phase.

#### Performing physical design:

**Q:** What specific technology does the methodology brings to the design process?

**A:** Step 4, activity 1, requirements capture and analysis phase.

**Q:** Does the methodology alert its users to the importance of making careful decisions regarding technology?

**A:** Step 4 and step 5, activity 1, requirements capture and analysis phase.

**Q:** Does the methodology help to accommodate the views of the clients regarding the technology that is necessary to be used?

**A:** Step 2 and step 5, activity 1, requirements capture and analysis phase.

**Q:** Does it offer support for considering both physical and logical design in an integrated manner?

**A:** Yes. Activity 8 of requirements capture and analysis phase and activity 5 of service design phase.

**Q:** Is the logical design independent from specific technologies?

**A:** Yes.

#### Implementing the designs:

**Q:** What help does the methodology offer for the implementation of the design solutions?

**A:** Service implementation phase.

**Q:** Does the methodology support testing activities?

**A:** Yes, Service validation and testing phase.

**Q:** Does the methodology alert its users to the possibility of changes to the logical design due to problems discovered during implementation?

**A:** Yes. Service implementation phase.

**Q:** Does the methodology consider user interface issues?

**A:** Yes. Activity 2, service design phase and activities 4 and 5, service implementation phase.

### Table 5.9.: Evaluating the proposed methodology according to the NIMSAD framework:

The problem-solving process element (solution design and design implementation).

According to these remarks, the proposed service development methodology is evaluated by extracting, from the description of the NIMSAD framework [JAYA94], a number of questions regarding each one of the elements of the framework and by attempting to answer them taking into account the properties, the phases / activities / steps, the models, and the structure of the proposed methodology as are described (mainly) in Chapter 4. These questions correspond to the characteristics that the proposed methodology should have according to the NIMSAD framework and can be considered as a much more structured and
concise expression of the framework itself. Tables 5.6., 5.7., 5.8., and 5.9. present questions and (pointers to) answers regarding the “problem situation” (the methodology context), the problem solver (the methodology user) and the problem-solving process (the methodology process), and collectively constitute the evaluation of the proposed service development methodology according to the NIMSAD framework.

From Tables 5.6., 5.7., 5.8., and 5.9. it is evident that the proposed service development methodology satisfies the requirements imposed by the NIMSAD framework and has all the desired characteristics regarding its three elements; although it must be noted that the characteristics related to the problem-solving process are the most significant. In this way, the completeness and the overall quality of the proposed methodology are ensured, and the confidence regarding its applicability and its practical value for a variety of service creation activities is increased. Furthermore, the objective, the features, the nature, and the philosophy of the proposed methodology are now better explained, offering to service developers a broader view of its capabilities before applying it, increasing their appreciation of the strength of its features, and enabling them to customise it according to the requirements of a particular circumstance in the best possible way. Finally, the evaluation of the proposed methodology pointed out the need for the service developers to use their intellectual reasoning capabilities and apply the methodology in a critical manner, as the success of any methodology is significantly affected by the “mental construct” of the methodology user and his/her true understanding of the guidelines and the structure of the methodology.

5.5. Summary and Conclusions.

The telecommunications industry is currently facing a growing need of making telecommunications services more versatile, easier to develop, interoperable, consistent, manageable, and independent of the underlying network infrastructure. Additionally, the demand for new sophisticated telematic services with multimedia characteristics, that require more flexible access, management and charging mechanisms, is increasing.

Therefore, as was explained in Section 4.1., it is necessary to assist service developers when performing service creation activities. During this effort, the easier development of services together with service personalisation, portability, interoperability, and reuse are of prime importance. For this reason, a service development methodology was proposed and presented in Chapter 4 with the intention to support the service creation process in a highly competitive environment of service provisioning. In order to determine the real value of this methodology, this chapter attempts to validate and evaluate it.

Initially, Section 5.2. validates the proposed methodology by applying it to the development of an actual telematic service (a MMCS-ET) and to several simple service scenarios. More specifically, regarding the development of the MMCS-ET, a variety of use cases are considered throughout the main phases of the proposed methodology (see also Appendix A), involving the support of session management requirements (session establishment, modification, and shutdown), interaction requirements (audio / video, text, and file communication), and collaboration support requirements (chat facility, file exchange facility, and voting).
Chapter 5: Validation and Evaluation of the Proposed Methodology

<table>
<thead>
<tr>
<th>Service Development Cycles (SDCs)</th>
<th>Examined Use Cases</th>
<th>Number of artifacts created</th>
<th>Duration (in months)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDC 1</td>
<td>- Start up the MMCS-ET service.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Log in to the MMCS-ET provider domain.</td>
<td>24</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>- Start a new MMCS-ET session.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDC 2</td>
<td>- Invite a student to join a MMCS-ET session.</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>SDC 3</td>
<td>- Join a MMCS-ET session after being invited.</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>SDC 4</td>
<td>- Invite a student (active user) to direct communication.</td>
<td>14</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>- Accept direct communication with a student (active user).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDC 5</td>
<td>- Engage in text communication with an active user.</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>- Engage in file communication with an active user.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDC 6</td>
<td>- Engage in A/V communication with a student (active user).</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>- Stop A/V communication with a student (active user).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Start in A/V communication with a student (active user).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Terminate A/V communication with a student (active user).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDC 7</td>
<td>- Engage in a chat with all active users.</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>- Engage in file communication with all students (active users).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDC 8</td>
<td>- Start a voting process between all active users.</td>
<td>26</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>- Vote in a voting process between all active users.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Present the outcome of a voting process to all the involved active users.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDC 9</td>
<td>- Terminate direct communication between two students (active users).</td>
<td>29</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>- Remove a student from a MMCS-ET session.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Terminate a MMCS-ET session.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Terminate the MMCS-ET service.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.10.: Essential metrics related to the development of the MMCS-ET (during the formation of the proposed methodology).

Due to the incremental and iterative nature of the proposed methodology these use cases were examined (from requirements elicitation up to actual implementation) in nine (9) service development cycles covering a time period of almost two (2) years. The exact distribution of use cases in service development cycles, the number of main artifacts that were created / changed in each of them, and their duration can be seen in Table 5.10. It has to be noted that this table is greatly affected by the fact that during the development of the MMCS-ET the proposed service creation methodology was under continual refinement and was augmented several times in order to take its final form. With this (stationary) form of the proposed methodology, which is described in Chapter 4, it is estimated that less development cycles are required, covering a significantly shorter time period (see Table 5.11.).

These validation attempts enlightened many aspects regarding the structure and the use of the proposed service development methodology, ensured its applicability and correctness, proved its practical value, and offered confidence that it can enable the fast and efficient creation of telematic services. Additionally, they illustrated the way that the methodology can be applied to the development of specific telematic services and provided important feedback which resulted to the improvement and enhancement of the proposed methodology.

One important methodology extension, which can be considered as an enhancement of the basic service development methodology as it complements it without affecting its main characteristics, is related to the exploitation of design patterns in the service engineering area and is examined in Section 5.3. The starting point of this section is the definition of the concept of design patterns, the explanation of its relation to the
concept of (object-oriented) frameworks, and the recognition of its importance in the proposed methodology. Then, a number of general purpose design patterns, specifically constructed for service engineering purposes, are proposed and presented, in order to (among other benefits) reduce the scope of the problem solving process in service creation activities and facilitate service developers in the design of service interaction diagrams. From the application of these patterns together with the proposed methodology, several more complicated service engineering patterns (which are directly related to service design aspects) are devised and their basic characteristics together with an example of them, are also described in this section.

<table>
<thead>
<tr>
<th>Service Development Cycles (SDCs)</th>
<th>Examined Use Cases</th>
<th>Number of artifacts created</th>
<th>Duration (in months)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDC 1</td>
<td>- Start up the MMCS-ET service.</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>- Log in to the MMCS-ET provider domain.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Start a new MMCS-ET session.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDC 2</td>
<td>- Invite a student to join a MMCS-ET session.</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>- Join a MMCS-ET session after being invited.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Invite a student (active user) to direct communication.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Accept direct communication with a student (active user).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDC 3</td>
<td>- Engage in text communication with an active user.</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>- Engage in file communication with an active user.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDC 4</td>
<td>- Engage in A/V communication with a student (active user).</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>- Stop A/V communication with a student (active user).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Start in A/V communication with a student (active user).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Terminate A/V communication with a student (active user).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDC 5</td>
<td>- Engage in a chat with all active users.</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>- Engage in file communication with all students (active users).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Start a voting process between all active users.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Vote in a voting process between all active users.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDC 6</td>
<td>- Present the outcome of a voting process to all the involved active users.</td>
<td>35</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>- Terminate direct communication between two students (active users).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Remove a student from a MMCS-ET session.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Terminate a MMCS-ET session.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Terminate the MMCS-ET service.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.11.: Essential metrics related to the development of the MMCS-ET (after the formation of the proposed methodology).

Finally, Section 5.4. evaluates the proposed service development methodology (in its final form) using the NIMSAD framework. After explaining the need for evaluating the proposed methodology, this section presents the rationale of the NIMSAD framework and describes the main characteristics of its essential elements (the "problem situation", the intended problem solver, and the problem-solving process). These elements, structured appropriately, are used for the evaluation of the proposed methodology, which ensured its competeness, its applicability and its overall quality, and clarified a number of important issues regarding its use.

The main research contribution of this chapter, which corresponds to one of the main objectives of this thesis (see Section 1.4.), is the validation and evaluation of the proposed service development methodology [ADA00b][ADA00f][PAPA97][PAP98a][PAP98b]. It is thus in direct relation with the research contribution of Chapter 4 (see Section 4.12.) and increases its importance, as it ensures that the proposed methodology satisfies Requirements 1, 2, 5, 8, 9, 11, and 14 (see Section 1.3.) and addresses all the matters.
that should be specified by such a methodology. Other secondary research contributions in this chapter, which are among the secondary objectives of this thesis (see Section 1.4.), are the following [ADA99b][ADA00c][ADA00d][PAP98c][PAPA99]:

- The identification, specification, and description of the functionality of a conferencing telematic service with multimedia characteristics that can be used for education and training purposes (a MMCS-ET). It is thus evident that Appendix A presents also a research interest.

- The description of several simple scenarios, regarding a variety of telecommunications services, that can be used for validation purposes in the service engineering area.

- The exploitation of design patterns in the service engineering area by proposing a number of general purpose and service engineering design patterns that communicate “best practice” idioms and solutions for the creation of telematic services and can significantly facilitate service developers when performing service creation activities.

- The structuring of the (otherwise abstract, verbose and difficult to use) NIMSAD framework in a concise but detailed manner, using its essential elements as a guide. This structuring improves significantly the applicability of the framework and increases its value.

This chapter provides confidence on the capabilities and the quality of the service development methodology that was proposed and presented in Chapter 4, and push forward its great potential to increase the momentum of TINA-C and enforce it to pave the way towards an open integrated broadband telematic infrastructure populated by a virtually limitless variety of telematic services. Therefore, with this chapter all the main constituent parts of the proposed telecommunications service engineering framework of Figure 1.1. are examined in considerable detail with the only exception of the service execution environment. This will be done in the next chapter that will enhance Microsoft’s DCOM (that was selected as a DPE in Chapter 3) with the ability to support continuous media interactions, making it thus capable to be used for the implementation of telematic services that are developed according to the proposed service creation methodology (such as the MMCS-ET that was presented in this chapter). It is thus evident that Chapter 6 is a prerequisite for the validation of the proposed methodology, but it is placed after it in order to examine all issues related to the methodology in a cohesive manner that facilitates consistency and a better comprehension.
Chapter 6:
Continuous Media Support in the Distributed Component Object Model

6.1. Introduction.

Driven by technological advances, market growth and deregulation, the global telecommunications industry is rapidly adopting a highly dynamic and open character, which, in combination with the evolving synergy between information and telecommunication technologies, provides a wide range of opportunities for the delivery of advanced and sophisticated telematic services with multimedia characteristics. Due to recent developments in object orientation and distributed computing, these telematic services are increasingly designed, realised, and deployed as multimedia applications operating on distributed object platforms (see also Chapter 3) [ADA00g][MUHL96].

These platforms are object-oriented Distributed Processing Environments (DPEs) that provide a uniform distributed computational model, isolating service developers from the heterogeneity of the various underlying systems, and thus hiding many of the complexities encountered in building distributed software [ADA99c]. However, there are key application areas in which distributed object platforms have lagged behind ad-hoc approaches to building distributed applications. In particular, support for distributed multimedia functionality is weak or non existent in the most important of today’s commercial object-oriented DPEs [COUL98].

Despite the fact that multimedia support has been considered in general terms in the ISO’s/ITU-T’s RM-ODP [GAY97][ITU95a], it has not yet been examined in Microsoft’s DCOM [BROW98] and is not yet mature in OMG’s CORBA [IONA98][OMG98a][OMG98d]. Recently, a wide range of new telecommunications services are becoming increasingly popular by employing video and audio to convey information and to enhance communication among human users (e.g. videoconferencing, video on-demand, interactive teletraining, etc.). Therefore, in the emerging multi-vendor, multi-stakeholder telecommunications environment,
Chapter 6: Continuous Media Support in DCOM

it is necessary to facilitate the rapid and flexible deployment of a great diversity of multimedia, multi-party telematic services by providing support for continuous media in DPEs. In this environment, the role of DCOM is expected to be important as it is a very promising distributed object platform for service engineering activities [ADA99c] that can be used as a DPE technology in the service execution environment of the proposed telecommunications service engineering framework of Figure 1.1. (see Section 3.6.).

Recognising the importance of DCOM and its significant potential in the area of telecommunications service engineering, Chapter 3 selected it as the distributed object platform that will be used in this thesis (see Section 3.6.) and this chapter presents a structured approach which extends DCOM to an environment suitable for the development of advanced multimedia telecommunications services. As was mentioned in Section 1.4., proposing and examining such an approach that enhances DCOM with the capability of handling continuous media interactions is one of the main objectives of this thesis and constitutes one of its main research contributions. In this way, as the rest of this chapter will illustrate and explain, DCOM is enabled to fully satisfy Requirements 3, 5, and 7 (see Section 1.3.), it is transformed to a TINA-C compliant DPE that is capable to support the construction of multimedia telecommunications services, and it becomes suitable for use with the proposed service development methodology. For this reason, DCOM is used during the validation of the proposed methodology (see Section 5.2.) according to its role in the service execution environment of the proposed telecommunications service engineering framework.

Taking into account these remarks, after this introductory section (Section 6.1.), Section 6.2. starts by identifying the main control aspects that should be considered when modelling multimedia telecommunications services. After examining briefly important state of the art activities related to the handling of multimedia streams, this section emphasises the necessity to support the object-oriented development of distributed multimedia applications in a flexible manner and outlines the way that this chapter attempts to model continuous media interactions in DCOM.

This modelling attempt is examined in detail in Section 6.3. More specifically, after considering briefly the capability of COM and DCOM to handle multimedia streams, Section 6.3.1. discusses central issues associated with the provision of object-oriented support in DCOM for the handling of continuous media in terms of representation, transmission, and management. For this purpose, it proposes and examines a multimedia support platform that consists of primitive COM objects (multimedia support services), which can facilitate the construction of new telecommunications services with multimedia characteristics. Furthermore, Section 6.3.2. presents the main steps of the stream communication algorithm that is used for the establishment and control of stream communication in DCOM, and Section 6.3.3. discusses some important implementation details.

Finally, Section 6.4. validates the proposed modelling approach through the design and implementation of a multimedia conferencing service, that is also used for the validation of the proposed service development
methodology (see Section 5.2.1.), and assess its flexibility and efficiency by conducting two types of experiments and reasoning about their results. The main findings of this chapter are summarised in Section 6.5., which also highlights the research contributions included in it.


Multimedia computing is concerned with the integration of a variety of media types (text, graphics, still images, animation, motion video, voice, sound) into a single coherent computing environment, while multimedia communication involves the interaction of devices which can deal with networked suppliers and consumers of various types of digitally represented information [LUDE96][PAP98b]. The tasks broadly involved in this process can be divided into the coding and transport of the different media, and into related control aspects, such as how to locate services 1 (in the sense of required functionality), request transfer, establish and maintain connections, ensure integrity and timeliness, and handle presentation issues during the delivery of multimedia information. These control aspects are the concern of this chapter, since they are particularly important for the realisation of the full potential of distributed object platforms in telecommunications service engineering [ALSA96][JHA97][MAR98a]. Another important requirement is the ability to hide the heterogeneous low-level aspects of dealing with streams through high-level Application Programming Interfaces (APIs) and to provide abstractions which could be easily dealt with by non-network programmers. In the rest of this section, important research and standardisation work related to the flexible handling of multimedia streams is examined.

The model of object interaction conventionally adopted in distributed object platforms (i.e. remote method invocation) is inappropriate for continuous or dynamic media, i.e. media which contain a temporal element, such as real-time audio or video. Information from a microphone or a video camera is an unlimited continuous stream of information that needs to be handled in real time. For these media types, a streaming, i.e. continuous mode of interaction is required rather than a request / response method invocation model. The main difference from discrete data interaction is that continuous interaction is not atomic since it models the exchange of continuous data (an on-going communications activity) between multimedia objects [COUL92][KINA96].

This difference is also reflected by the RM-ODP’s multimedia computational model that builds streaming interaction over the primitive notion of a signal, which is defined as the emission / reception of a data item from / to an interface. A stream interface is modelled as a sequence of signal emissions from a producer interface together with an associated sequence of signal receptions at a consumer interface. In RM-ODP the emission or reception point of such a sequence of signals is known as a flow (as opposed to an operation in

1 In this chapter, the term “service” has a more general meaning and it is not used as a synonym with the terms “telematic service” and “telecommunications service”.

207
the traditional request / response style of interaction), and an interface containing flows rather than operations is known as a stream interface. A stream interface may contain many flows with varying types and directionalities. It is specified by a stream interface template that consists of a finite set of action templates, one for each flow type in the stream interface. Each action template contains the flow name, its information type, and an indication of its causality, since flows are unidirectional (producer or consumer but not both) [COUL98][GAY97][ITU95a].

Streams are also present in several other distributed computing architectures. Initially, they appeared in the Multimedia Systems Services (MSS) architecture, which was proposed by the Interactive Multimedia Association (IMA) [IMA94]. In this approach, the objects producing the streams are "special" and inaccessible to applications; the latter can control, but not directly access, the real-time media. The IMA MSS is currently being adopted and extended by ISO in its PR esentation Environments for Multimedia Objects (PREMO) standard [ISO96]. Furthermore, much of the initial work on streams, which influenced greatly the RM-ODP, took place trying to address the requirements of multimedia support in the Advanced Network Systems Architecture (ANSA) [REES94], either within the context of the existing computational model [COUL92] or by changing it [NICO90]. The current ANSA Phase III Distributed Interactive Multimedia Architecture (DIMMA) project is pursuing the latter approach and is based on the ANSAware distributed systems platform, which has been enhanced with a modular protocol stack and a flexible multiplexing structure [LI97].

OMG has recently addressed the need for streams and real-time functionality in CORBA by issuing a request for proposals (RfP) for the control and management of audio / video streams, and by summarising submissions in [OMG98d]. However, this RfP does not examine the implementation of streams in CORBA. Such implementation issues were addressed by specific ORB vendors [SCHM98] and by the ACTS ReTINA project, which designed a specialised distributed object platform based on CORBA, and enhanced with streams and Quality of Service (QoS) extensions [DANG96]. More specifically, the architecture proposed by the ACTS ReTINA project was based on a clear separation between ORB support mechanisms (such as interface reference management, threads, buffers, etc.) and stream binding classes, which provide communication services tailored to the needs of particular applications through the use of a generic binding protocol. Another CORBA version 2.0 compliant implementation considering multimedia support is the TAO ORB, which runs on real-time operating system platforms and is primarily designed for strict real-time applications [SCHM99].

In all the above architectures, the modelling of continuous media communications through a flexible, high-level but efficient infrastructure are crucial. More specifically, modelling mainly involves the choice of suitable and sufficient abstractions, their implementation on a target DPE, and the adoption of appropriate interaction patterns and semantics. Flexibility is a general property referring to the way that modelling
Chapter 6: Continuous Media Support in DCOM

concepts and artifacts are used for the design, development, and deployment of open telecommunications services. The great variety, inherent complexity, and the increasing demand for customisation of such services raise the importance of flexibility.

Based on these assumptions, a generic platform (a multimedia support platform) for the handling of continuous media in DCOM is proposed [ADA99a][ADA00a][ADA00g]. It consists of primitive COM objects or services (multimedia support services), which can facilitate the construction of new telecommunications services with multimedia characteristics. More specifically, the multimedia support services, and the associated COM objects, are compatible with RM-ODP in the sense that they adopt related concepts and functionality, and thus enable a wide degree of information sharing and application interoperability. Furthermore, these services can be reused and customised, and their interfaces have been designed to allow flexibility and efficiency in achieving their implementation. This is important for telecommunications services which manipulate multimedia objects, where performance is critical.

While this chapter deals with the flexible modelling of multimedia streams in a DCOM-based DPE, it should be noted that besides supporting modelling aspects, the control and management software of new telecommunications infrastructures needs also to support a range of QoS characteristics, the synchronisation of continuous media, and the careful management of underlying resources [COUL92][MUHL96][WADD97]. These aspects are outside the scope of this chapter.

6.3. Enhancing DCOM for the Support of Continuous Media.

As is mentioned in Section 3.2.4., DCOM is the distributed extension to COM (Component Object Model) that builds an Object Remote Procedure Call (ORPC) layer on top of DCE RPC to support remote objects. In general, DCOM provides all the necessary facilities for the integration of heterogeneous components in a distributed environment (see also Sections 3.2.4. and 3.4.) [BROW98][GRIM97].

However, DCOM does not satisfy the more complicated and stringent requirements of handling multimedia streams. To enable DCOM to be the basis for new telecommunications services which require the handling and control of continuous media, extra features are necessary. The most obvious requirement is that the concept of streams should be added to the DCOM object model through the introduction of stream interfaces, since at present only operational interfaces are defined.

Before focusing on DCOM, it has to be noted that COM handles multimedia information through the Microsoft DirectShow architecture ² (previously Microsoft ActiveMovie architecture), which incorporates the notion of streams [MICR95][PINN98]. Apparently, the use of this notion is restricted to the environment

² DirectShow enables the playback of multimedia streams from files and the capture of multimedia streams from devices. More specifically, it enables the playback of video and audio content compressed in various formats including Motion Picture Experts Group (MPEG), Apple QuickTime, Audio-Video Interleaved (AVI) and WAV, and both Video for Windows-based capture and WDM-based (Windows Driver Model) capture.
Chapter 6: Continuous Media Support in DCOM

of stand-alone multimedia capable computers with Microsoft Windows operating systems (9x, NT 4.0, 2000), i.e. DirectShow is not a distributed architecture.

For the rest of this section, in subsection 6.3.1. the key modelling abstraction is presented and examined, in 6.3.2. the stream communication algorithm is described and explained, and in 6.3.3. some important implementation details are discussed.

6.3.1. The Proposed Approach.

In order to facilitate the development of new telecommunications services with multimedia characteristics in DCOM a multimedia support platform is proposed with the introduction of a number of support services in the DCOM architecture. These services (which are used in conjunction with existing DCOM services) provide new functionality without requiring any changes to the basic underlying DCOM architectural model. The new support services consist principally of two types of COM objects: devices and stream binders. These are both seen by the higher layers of (software) abstraction as normal services with standard abstract data type interfaces, but they encapsulate the control and transmission of continuous media.

COM object devices are an abstraction of physical devices, stored continuous media or software processes. They may be either sources, sinks or transformers of continuous media data ("modules"). A source is a media producer and is normally an abstraction of a media-generating hardware device, such as a camera or a microphone. A sink is a media consumer and is normally an abstraction of a media-rendering hardware device, such as a framebuffer / VDU or a loudspeaker. Finally, a module is both a media producer and consumer, as it accepts incoming data, processes it in some way, and produces output. Devices are "virtual" entities in the sense that there may be multiple logical devices per physical device depending on the requirements of the specific application.

As can be seen from the inheritance structure of Figure 6.1., most devices present a device dependent interface, a generic control or chain interface (IChain), and an endpoint interface (IEndpoint). The device dependent interface contains operations specific to the device modelled and is used for the management of the device. For example, a camera might have operations such as focus, pan or tilt. Furthermore, it has to be noted that all devices have to inherit from the IUnknown interface, which provides functionality required by
all COM objects (see also Section 5.2.1.5.). This is common in most distributed object platforms; for example in CORBA and Java-RMI.

A piece of continuous media can be visualised as a chain comprising a sequence of segments or links, each of which represents an atomic unit specific to the media type in question (e.g. a frame of video) [COUL92]. Thus, a chain is an abstraction over a continuous media source or sink that focuses on the control of the production and consumption of continuous media data. Based on this abstraction, the IChain interface provides generic operations for controlling continuous media devices and managing continuous media transmissions. It is a device independent interface which is common to all continuous media devices. The IChain interface is summarised in Figure 6.2. using (a simplified variation of) Microsoft's IDL (M-IDL).

```
interface IChain : IUnknown
{
    typedef enum {in, out, inout} DeviceType;
    HRESULT GetDeviceType([out] DeviceType* DType);
    HRESULT Start();
    HRESULT StartEx([in] int NumberOfSegments);
    HRESULT Stop();
    HRESULT Suspend();
    HRESULT SuspendEx([in] int Time);
    HRESULT SuspendEx1([in] int NumberOfSegments);
    HRESULT Resume();
    HRESULT Skip([in] int NumberOfSegments);
    HRESULT GetPosition([out] int* SegmentNumber);
};
```

Figure 6.2.: The IChain Interface.

More specifically, the GetDeviceType operation returns the type of the device under examination (producer, consumer or module), while the Start and Stop operations switch the device's information flow on and off accordingly. The functionality of the two last operations, which are the most important of the IChain interface, is based on the use of a virtual pointer (CurrentSegment) that moves through the media chain as it is played or recorded. The value of this pointer in a producer device reveals the number of segments that have been transmitted, while in a consumer device represents the number of segments that have been received. The Start and Stop operations make also use of the fact that a producer device places the outgoing segments on the OutputSegment buffer, while a consumer device places the incoming segments (before processing them) on the InputSegment buffer.

Therefore, the Start operation, after initialising the CurrentSegment pointer to 0, on a producer device fills continuously the OutputSegment buffer and increases each time the value of CurrentSegment by 1, while on a consumer device empties continuously the InputSegment buffer and increases each time the value of CurrentSegment by 1. On the other hand, after a Stop operation a device no longer produces or consumes segments, and the value of the CurrentSegment pointer is preserved. It has to be noted, that the StartEx operation is a variation of the Start operation, which does not initialise the CurrentSegment pointer and produces / consumes a specific number of segments (NumberOfSegments).
There are also operations for suspending and resuming the activity of a device (Suspend and Resume respectively). After a Suspend operation the production / consumption of segments in a device stops until the Resume operation is called or in the case of SuspendEx / SuspendExI for the time period specified (explicitly or implicitly) by the parameters of the operation. In both cases the value of the CurrentSegment pointer is preserved. Finally, this pointer may be located and moved using the GetPosition and Skip operations. More specifically, GetPosition returns the current value of the CurrentSegment pointer, while Skip ignores NumberOfSegment segments that were to be transferred (in the case of a producer device) or that were already transferred (in the case of a consumer device) preserving the value of the CurrentSegment pointer.

Another interface which is common to all continuous media devices (device independent interface) is the IEndpoint interface. An endpoint is a connection point (a port) for a stream and the IEndpoint interface is thus the “stream interface” of a device. It has to be stressed though, that the term “stream” in this chapter is not strictly used in accordance with the RM-ODP terminology, in which a stream is related to a set of flows. On the contrary, the more accepted meaning of the term “stream” is used to denote one particular unidirectional flow of a series of messages of a pre-defined type, such as an audio flow or a video flow between two devices [JACO96]. If a device participates in more than one streams, multiple instances of that device have to be used.

```c
interface IEndpoint : IUnknown {
    HRESULT GetSegment([out] BSTR* Segment);
    HRESULT PutSegment([in] BSTR* Segment);
    HRESULT SetCharacteristics([in] long ChrSize,
                               [in, size_is(ChrSize)] long* ChrArray);
    HRESULT GetCharacteristics([in, out] long* ChrSize,
                               [out, size_is(ChrSize)] long* ChrArray);
};
```

**Figure 6.3.: The IEndpoint interface.**

The IEndpoint interface abstracts over all aspects of a device which are concerned with the transport of continuous media. Essentially, as it can be seen in Figure 6.3., it presents a pair of operations, GetSegment and PutSegment through which segments can be read (from the OutputSegment buffer of a producer device) or written (to the InputSegment buffer of a consumer device) respectively. With this approach, the content of a stream is not considered and it is viewed purely as a byte transport mechanism. The two other operations of the interface (SetCharacteristics and GetCharacteristics) refer to a number of transmission related characteristics used for QoS issues.

The operations inside the IChain and the IEndpoint interfaces of a specific device must take place in an acceptable and semantically correct order (e.g. for the same device a Stop operation can not be followed by a Suspend operation) to avoid unexpected results / errors. To ensure such an order, each device has a state
(DeviceState), which is checked before an operation is executed. The proposed and anticipated transitions between the states of a device can be seen at the state transition diagram of Figure 6.4., which also depicts the possible states of a device (idle, ready, active, and suspended).

![State Transition Diagram](image)

**Figure 6.4.:** The state transition diagram of a device.

In order to be able to control streams the *binding process* must be made explicit. The term “binding” is used in a general sense to mean both the process of associating and interconnecting different sources and sinks according to specific communication semantics, and the end result of this process. Binding implies setting up an access path between the involved COM objects (devices), which in turn typically comprises of locating the desired COM objects, setting up appropriate data structures to enable communication between them, and using suitable communication resources to support remote object interactions [COUL92][DANG96].

```c
interface IStreamBinder : IUnknown
{
    HRESULT StartSource([in] IUnknown* SourceGroup);
    HRESULT StartSink([in] IUnknown* SinkGroup);
    HRESULT ConnectAndTransfer([in] IUnknown* SourceGroup,
                                [in] IUnknown* SinkGroup);
    HRESULT StopSource([in] IUnknown* SourceGroup);
    HRESULT StopSink([in] IUnknown* SinkGroup);
    HRESULT SuspendSource([in] IUnknown* SourceGroup);
    HRESULT SuspendSink([in] IUnknown* SinkGroup);
    HRESULT ResumeSource([in] IUnknown* SourceGroup);
    HRESULT ResumeSink([in] IUnknown* SinkGroup);
    HRESULT DestroyConnection([in] IUnknown* SourceGroup,
                               [in] IUnknown* SinkGroup);
};
```

**Figure 6.5.:** The IStreamBinder interface.

The binding process is made explicit through the introduction of a binding COM object (StreamBinder). StreamBinder represents the connection between bound COM objects and provides an operational interface (IStreamBinder) through which the bindings required for the realisation of streams can be
created, monitored, and controlled by other COM objects. More specifically, as can be seen in Figure 6.5.,
the IStreamBinder interface contains operations which allow the client of a StreamBinder to start and stop
the flow of continuous media information, connect and disconnect devices via their IEndpoint interfaces
(and thus create and destroy stream connections), and suspend / resume the activity of the involved devices.
With these operations the StreamBinder hides continuous media transmissions, which can be optimised by
using dedicated transport protocols entirely distinct from those used to convey control messages.

The binding action can be initiated by a COM object involved in the binding or by a completely separate
object. In general, client COM objects wishing to initiate continuous media transfer, request from the
StreamBinder to start the appropriate source and sink devices (StartSource, StartSink). Then, the
StreamBinder establishes a stream connection between these devices and activates the transmit function
(Connect&Transfer). The resulting stream can be managed by suspending (SuspendSource,
SuspendSink), resuming (ResumeSource, ResumeSink) and stopping (StopSource, StopSink) the
participating devices, and it can be destroyed when desired (DestroyConnection). It is evident that this
approach is working best in a multi-threaded DCOM environment. In such an environment, other methods
can be invoked on the StreamBinder object (and thus other streams can be activated), whilst data is
streamed via an existing stream connection. Additionally, this approach ensures that an operation will not be
executed on a device with an incorrect, i.e. semantically unintended, role (producer / consumer). For
example, the StartSource operation before calling Start on the device specified by its parameter, checks
whether this device is a producer (using the GetDeviceType operation).

The StreamBinder, in the most general case, supports multiple stream connections, as it allows M
sources to be connected to N sinks (without necessarily M=N), by establishing the appropriate streams
between them. When it is desirable to start, stop, establish, and generally perform control operations to a
number of streams simultaneously, the notion of object groups simplifies greatly the necessary code (calls to
the StreamBinder operations). Additionally, it eases considerably the process of ensuring that the code
reflects the correct / intended semantics, as it decreases the possibility of missing, wrong, or out of order
operations on devices. This is due to the fact that errors can now appear only during the formation of object
groups; an activity which corresponds to a relatively small and well structured piece of code that can easily
be examined. Two typical errors that can be avoided without difficulty through the use of object groups is
the execution of an operation on a device that belongs to a different stream than the one intended, and the
execution of Connect&Transfer and / or DestroyConnection on two devices that (are intended to) participate
in different streams.

Conceptually, object groups are modelled using the COM class ObjectGroup, which collects in a group
a set of related COM objects. Actually, it maintains a list of the interface references (REFIID)s of the COM
objects that belong to a specific group. The IObjectGroup interface can be seen in Figure 6.6. Join and
Leave operations allow new members to join the group and existing members to leave the group respectively, while Use and Reset provide access to the group's current membership list.

```c
interface IObjectGroup : IUnknown
{
    HRESULT Join([in] IUnknown* refiid);
    HRESULT Leave();
    HRESULT Use([out] IUnknown* refiid);
    HRESULT Reset();
};
```

**Figure 6.6.:** The IObjectGroup interface.

In a typical scenario, two instances of the ObjectGroup COM class are used: a `SourceGroup` and a `SinkGroup` (which are actually the interface references of the two instances). The two lists that are maintained by these two groups, contain at corresponding positions the interface references of the sources and sinks that are going to be engaged in stream communication. Thus, the use of (the interface references of) these two groups as parameters in the operations of the IStreamBinder interface allows the invocation of (corresponding) operations on a number of COM objects (sources / sinks) at the same time. However, it must be noted that in order to increase the flexibility and support application semantics where the simultaneously establishment and control of multiple streams is not desirable, the use of object groups in the operations of the IStreamBinder interface is not mandatory. Interface references to simple COM objects (sources / sinks) can also be used as parameters.

The COM objects examined so far constitute the proposed multimedia support services for DCOM and should be reused during the development of specific multimedia services. They may therefore have to be customised according to the specific service requirements. This activity, which is very important as it determines the practical value of the proposed approach is supported through the use of either containment or aggregation, as DCOM allows only interface and not implementation inheritance (see also Section 3.4.1) [GRIM97].

More specifically, under *containment* one COM object contains another, with the outer COM object (e.g. representing a "new" enhanced multimedia device) accessing the inner COM object (representing an already used and tested "old" device) through its interfaces. Generally, clients of the outer COM object are unaware of the relationship, except in the case where the outer COM object chooses to expose an interface that is supported by the inner COM object.

On the other hand, *aggregation* occurs when the outer COM object exposes the interfaces of the inner COM object directly to clients. One important characteristic of this technique is that it can only be used for in-process COM object servers (i.e. DLL server modules). However, this particularity of aggregation can become a significant restriction when using the proposed API, because some of its COM objects (devices, StreamBinder) may be, as far as their clients are concerned and depending on application requirements,
either local servers (implemented as EXEs) or remote servers (executed on a remote server machine). Therefore, when reusing or customising COM objects from the proposed API, the containment method is the preferred way since it enables the resulting service component to operate under all possible COM server types.

6.3.2. The Stream Communication Algorithm.

The proposed multimedia support services described in the previous section can be used for the establishment and control of stream communication in DCOM in a structured fashion. To illustrate this approach, a possible scenario is examined. According to Figure 6.7., which depicts the configuration of the COM objects involved in the example scenario, two source devices (e.g. video cameras) are connected via a StreamBinder to two sink devices (e.g. VDUs), and two different streams are established between the source and sink devices.

![Figure 6.7.](image)

Figure 6.7.: An example scenario for the proposed multimedia support services.

The necessary steps that have to be followed in order to realise the two video connections (stream 1 and stream 2) between the sources and sinks of Figure 6.7. using the proposed multimedia support services are the following:

**Step 1:** Obtain the necessary interface references.

The interface references (REFIIDs) of the two sources (Source1UserA and Source2UserB) and the two sinks (Sink1UserC and Sink2UserC) involved in stream communication are obtained. Device dependent operations are also performed if necessary.

**Step 2:** Create new instances of required services (COM objects).

A StreamBinder instance is created and the related interface reference is obtained. Additionally (if required), two ObjectGroup instances are created and the related interface references are also obtained (SourceGroup and SinkGroup).
Step 3: Form the appropriate object groups (if required).

Taking into account the streams that is desirable to be established (or actually considering the source and sink devices that need to be connected by streams), the REFIIDs of the sources become members of the SourceGroup \( \text{Join(Source1UserA), Join(Source2UserB)} \), and the REFIIDs of the sinks become members of the SinkGroup \( \text{Join(Sink1UserC), Join(Sink2UserC)} \).

Step 4: Start the devices.

The sink and source devices are started \( \text{StartSink(SinkGroup), StartSource(SourceGroup)} \).

Step 5: Establish connections between source and sink devices.

Associate the appropriate sources and sinks and initiate continuous media transfer between them \( \text{Connect&Transfer(SourceGroup, SinkGroup)} \). Steps 4 and 5 can also take place in the opposite order.

Step 6: Stop the devices.

When the interaction is finished the sink and source devices are stopped \( \text{StopSink(SinkGroup), StopSource(SourceGroup)} \).

Step 7: Destroy connections and services.

The connections established between the appropriate sources and sinks are destroyed \( \text{DestroyConnection(SourceGroup, SinkGroup)} \). Then, the StreamBinder and the ObjectGroup instances created in step 2 are also destroyed.

The above described steps constitute a kind of algorithm, i.e. a stream communication algorithm for establishing and controlling stream connections in DCOM. Two more steps can be added to this algorithm depending on the functionality required by some applications. More specifically, between steps 5 and 6 (i.e. while all the devices are active), the sink and source devices can be suspended \( \text{SuspendSink(SinkGroup), SuspendSource(SourceGroup)} \) and then, on a consecutive (mandatory) step, they can be resumed \( \text{ResumeSink(SinkGroup), ResumeSource(SourceGroup)} \).

The stream communication algorithm utilises the operations of the IStreamBinder interface to create and manage bindings between the appropriate sources and sinks. These bindings do not have to be controlled directly by the COM objects involved in the binding (i.e. the sources and the sinks), but may instead be created by third party COM objects which obtain references to interfaces owned by those COM objects. This facility eases considerably the configuration and structuring of potentially complex multimedia telecommunications services containing many per-media COM objects.

A similar situation is described in Figure 6.7., where the StreamManager COM object interacts with the StreamBinder and performs all the steps of the stream communication algorithm. In the general case, the
StreamManager can call directly operations, both on (source / sink) devices and on the StreamBinder, and is responsible for the "encapsulation" of the (control) logic that is related with streams. To avoid errors and unexpected results, caution is needed to ensure that when an operation is executed on a device, the same or a (semantically) compatible operation is also executed on the device with which the former device is (will be) connected by a stream.

From these remarks is evident that the structure and the behaviour of the StreamManager depends on the requirements of a specific application, and on the way that this application handles streams. On the contrary, the interfaces and the functionality of the (COM objects used to model) devices and the StreamBinder are application independent and thus suitable for reuse. Actually, these interfaces (and the corresponding multimedia support services that they provide) can be considered as a high level API for the handling of continuous media in DCOM.

The advantages of this high level API are highlighted when taking into account that the main alternative approach for stream handling in DCOM requires the use of low level native Windows APIs (such as Win32), which is characterised by [GRIM97][SCHM99]:

- Excessive low level details that:
  - Divert the attention of the developers from the more crucial (broader) application-related semantics and the program structure.
  - Raise the potential for errors.
  - Increase the learning effort required.
  - Hinder the development of complex applications.
- Continuous re-discovery and re-invention, in an ad hoc manner, of incompatible higher-level programming abstractions that seriously hampers programming productivity and code compatibility.

Therefore, the development of multimedia telecommunications services in DCOM benefits greatly from the use of the proposed API, because it isolates the application domain semantics from the complexities of multimedia devices and continuous media communications, by providing services based on abstract data type interfaces. Additionally, it reduces and simplifies the required programming effort, by locating all the code related with the handling of streams inside easily extensible reusable components, preventing thus developers from "reinventing the wheel" using elementary capabilities and functionalities.

6.3.3. Important Implementation Considerations.

There are a few DCOM related issues that affect considerably the implementation of the proposed multimedia support services. These issues, which will be examined briefly in this subsection, include class factories, access to remote COM objects, and the available threading models.
In order to be able to use a (device or a StreamBinder) COM object, an instance must be created. This is done through a special COM object called a class factory, which implements the IClassFactory interface and has knowledge on how to “manufacture” (one or more) COM objects of a particular class (this is again typical in distributed object platforms). More specifically, the CoGetClassObject() call using the Interface ID (IID) of IClassFactory and the class ID (CLSID) of a class (e.g. a source / sink device), returns an interface pointer to a class factory which implements a CreateInstance() method that creates COM objects (instances) of that particular class. In this way, the COM runtime gains efficiency as it is not necessary to know about all the possible object types, that might have to create, ahead of time. This functionality is based on the design pattern of a “Factory Method”, according to which, when a client wishes to instantiate a server object, a request is sent to a “Factory Object” for the corresponding class [GAMM95].

Taking into account this design pattern, a class factory has to be created for every server component specified by the proposed multimedia support services. It has to be noted, that for optimisation reasons in some (not very common) cases (e.g. when a device has a large number of device specific interfaces and depending also on their intended use), a custom implementation of the IClassFactory interface is allowed, but caution is needed to avoid possible compatibility conflicts / problems.

After performing all the necessary instantiations, a client that wishes to call operations on a (device or a StreamBinder) COM object has to obtain a pointer to a suitable interface of that object. When the desired COM object is remote (which is common during the proposed stream communication algorithm) the CoCreateInstanceEx() function has to be used in a suitable manner to locate the server machine, create (an instance of) the appropriate COM object on that machine, and finally return the desired interface pointer. This function is called with an array of MULTI_QI structures as one of its parameters:

```c
typedef struct _MULTI_QI {
const IID* pIID; // pointer to an interface identifier
IUnknown * pItf; // returned interface pointer
HRESULT hr; // result of the operation
} MULTI_QI;
```

As can be seen in Figure 6.8., each pIID member of this array is given an IID of an interface of the remote COM object. If the CoCreateInstanceEx() succeeds, the desired interfaces can be obtained through the pointers in the pItf members. If there is an error, the hr member will receive the error code. Thus, except from the status of CoCreateInstanceEx(), the status of each element in the MULTI_QI array should also be checked, before a (valid) interface pointer can be extracted from the array.

When a client requires access to a particular remote COM object, and this object has more than one interface to which the client needs pointers, an array of MULTI_QI structures should be created, containing as many pIIDs as necessary to keep all the IIDs of the interfaces that the client will (or intends to) use on the COM object. In this way, the CoCreateInstanceEx() will be called only once and multiple calls to it, due
to an incomplete (in terms of requested IIDs) MULTI_QI array, will be avoided. This tactic reduces the
number of necessary RPC calls across the network, and improves the efficiency of the code especially when
remote COM objects exhibit more than one interface (e.g. as in the case of COM objects used to model
continuous media devices) and/or the network performance is or becomes slow.

// initialise the MULTI_QI structure
MULTI_QI qi[2]; // create an array of e.g. 2 structures
memset(&qi, 0, sizeof(qi)); // prepare the array for use
qi[0].pIID = &IID_IChain; // add the 1st interface
qi[1].pIID = &IID_IEndpoint; // add the 2nd interface

// create a server COM object on the server machine
HRESULT hr=CoCreateInstanceEx(
    CLSID_CMyServer, // COM class id
    NULL, // outer unknown
    CLSCTX_SERVER, // server object scope
    &ServerInfo, // name of the server machine
    2, // length of the MULTI_QI array
    qi); // pointer to the 1st element of this array

// check the qi codes
if (SUCCEEDED(hr))
{
    // also check qi hresult
    hr=qi[0].hr;
}
if (SUCCEEDED(hr))
{
    // extract interface pointers from MULTI_QI structure
    m_pComServer=(ICPServer*)qi[0].pItf;
}

Figure 6.8.: Using an interface of a remote server object in DCOM.

Finally, shifting the focus to the internal structure of COM objects, the way that threading is performed needs
to be examined. Threading involves specifying code segments that will be executed concurrently by creating,
somewhere within a program, more than one thread, and ensuring the protection of shared resources, the
provision of thread synchronisation, and the avoidance of deadlocks and race conditions. In DCOM, threads
are established to improve performance (minimise execution time), to simplify the code, and to avoid the
blocking of COM objects (e.g. to prevent the blocking of the StreamBinder when executing an operation
on a device). Therefore, in the proposed multimedia support services threading is used in the implementation
of the StreamManager and the IChain interface of the COM objects used to represent devices, in the
interfacing with physical devices, and in the realisation of stream connections using transport protocols.

When programming using DCOM, and therefore in the proposed API, except from the thread handling
functions of Win32 (e.g. CreateThread(), ExitThread(), etc.), the following COM threading models
can be applied [BROW98][GRIM97]:

- **The simple single threaded model:** All COM usage in a client must be performed in the same thread; the
  one that called CoInitialize(). Only this single thread can use COM objects that are created by the
  client. COM objects that support this model do not have to protect any shared variables / data.
Chapter 6: Continuous Media Support in DCOM

- **The Single Threaded Apartment (STA) model:** Multiple threads in a client can call CoInitializeEx() (forming separate apartments) and create (instances of) COM objects. However, the COM objects created on a particular thread can only be used on that thread. Any interaction between COM objects in different apartments (threads), even apartments in the same process, has to be marshaled by the COM runtime through a proxy. Thus, any shared variables / data used by these COM objects do not have to be protected, since they are actually accessed only via one single thread.

- **The MultiThreaded Apartment (MTA) model:** This is the free threading model. A client may create as many threads as it wishes (that all are part of the same apartment), create any (instances of) COM objects on any of the threads, and use the COM objects on the same or other threads. The client always calls the methods of the COM objects directly, without doing any marshaling. COM objects that support this model must protect all shared variables / data.

6.4. Validation and Experimentation.

The proposed multimedia support infrastructure and the related API have been tested in several simple scenarios (like the one depicted in Figure 6.7.) involving different configurations of source and sink devices associated by various stream connections. It has been found that they constitute a viable, flexible, consistent, coherent, and relatively intuitive way of building multimedia telecommunications services in DCOM.

To verify and reinforce these results under (more) realistic conditions, and to determine also the true practical value and applicability of the proposed API, it has been used during the validation of the proposed service creation methodology, where a MultiMedia Conferencing Service for Education and Training (MMCS-ET) has been developed (see Section 5.2.) [ADA00f][ADA00g]. This service is implemented (see also Section 5.2.1.5.) using MS Visual C++ 6.0 and DCOM on MS Windows NT 4.0, and is executed on a number of workstations connected via a 10 Mbit/s Ethernet LAN. All the interconnected workstations belong to the same (MS Windows NT) domain and one of them functions as a primary domain controller.

As was mentioned in Section 5.2.1.2., the main objective of the MMCS-ET is to facilitate the establishment of an educational / training session between one teacher / trainer and a number of remote students / trainees, which is equivalent to the educational / training session that would have been established between the same people (teacher / trainer and students / trainees) in a traditional classroom. More specifically, in a virtual classroom the teacher still has the need to manage the educational / training session. Additionally, there is also a need for audio / video (A/V) communication among all the session participants (to substitute face to face contact), text communication between only two session participants (as that achieved with the use of notepads), text communication among all the session participants (as that achieved with the use of a blackboard), file communication between the session participants (e.g. for the exchange of course material), and collaboration among all the session participants in order to perform a common task. For this
reason, the MMCS-ET implements a variety of use cases supporting session management requirements (session establishment, modification, and shutdown), interaction requirements (audio / video, text, and file communication), and collaboration support requirements (chat facility, file exchange facility, and voting).

The computational view of the MMCS-ET in the simple case where one teacher interacts with only one student can be seen in Figure 6.9. This figure emphasises on the way that A/V communication is achieved between the teacher and the student by the establishment of two streams of opposite directions, presenting the position of the interfaces of the proposed multimedia support services for DCOM. It has to be noted that the Communication Session Manager (CSM), which is at the boundary with the resource layer, incorporates the functionality of the StreamManager, and that the GetFromProducer and PutToConsumer COM objects are used for the realisation of the stream connections.

The MMCS-ET, not only validated the proposed API and confirmed the results of the initial tests with the simple scenarios, but also gave an insight, through several experiments, for the optimisation of the proposed API in terms of its use and its more efficient implementation in DCOM. More specifically, two types of experiments were conducted. The first type involved the application of object groups in the stream
communication algorithm, examining the complexity of the resulting code (which is actually the main piece of code written by the service developer when using the proposed API) in terms of the number of necessary calls of operations to other COM objects. The number of such calls (with and without the use of object groups) for an increasing number of stream connections can be seen in Figure 6.10. It has to be noted that different stream connections are established between different source and sink devices. Thus, for example, 4 stream connections imply the existence of 4 sources and 4 sinks. From Figure 6.10, it is evident that the use of object groups, as the number of stream connections is increasing, reduces considerably the number of operation calls that have to be made, and therefore simplifies the code of the stream communication algorithm (together with the task of the service developer) and increases its efficiency.

![Graph showing the number of calls with and without object groups](image)

**Figure 6.10.:** Experimenting with the use (or not) of object groups in the stream communication algorithm.

The second type of experiment involved the examination of the performance (in terms of execution time) of the different COM threading models when applied to the proposed API. A choice between these models becomes especially important when the StreamManager creates separate threads for the instantiation of the (COM objects representing the) devices and the execution of device specific actions, for the instantiation and initialisation of the GetFromProducer and PutToConsumer COM objects, and for the instantiation of the StreamBinder and the execution of the stream communication algorithm. It has to be noted that only the STA and MTA models are considered because the single threaded model is really just a special type of the STA model.

When each of the STA and MTA models are applied to all of the (COM objects) of the proposed API the time needed (in ms) to start (Time 1) and stop (Time 2) a stream connection between one source and one sink device is measured for each of them. Time 1 corresponds to steps 4 and 5 of the stream communication algorithm, while Time 2 corresponds to steps 6 and 7. The results of the measurements can be seen at Table 6.1. From this table is evident that the MTA model is at least as good as the STA model. Therefore, for the proposed API, taking also into account that there are no synchronisation issues, the MTA model is the
preferred choice. Its performance superiority is mainly due to the fact that inter-thread access is direct (as all
the threads are in the same apartment), requiring no proxy intervention as in the STA model.

<table>
<thead>
<tr>
<th>COM Threading Models</th>
<th>STA</th>
<th>MTA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>19.6 ms</td>
<td>12.6 ms</td>
</tr>
<tr>
<td></td>
<td>18.19 ms</td>
<td>11.66 ms</td>
</tr>
</tbody>
</table>

Table 6.1.: Comparison of COM threading models using the proposed API.

Finally, to place the proposed API for DCOM in a more general context and increase in this way the
confidence in its use, a (high level) comparison with the approach followed by OMG for the handling of
continuous media [OMG98d] is attempted, because OMG’s CORBA is considered to be the main
(commercial) alternative to DCOM. The results of this comparison, which focuses on how continuous media
communication is modelled, can be seen in Table 6.2.

<table>
<thead>
<tr>
<th>Important Concepts</th>
<th>Modelling in OMG’s A/V Spec. (CORBA)</th>
<th>Modelling in the proposed API (DCOM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multimedia device</td>
<td>MDevice interface/object</td>
<td>Device COM object</td>
</tr>
<tr>
<td>Device specific aspects</td>
<td>VDev interface/object</td>
<td>Device dependent interface</td>
</tr>
<tr>
<td>Device control aspects</td>
<td>StreamExpPoint interface/object</td>
<td>TXChain interface</td>
</tr>
<tr>
<td>Stream endpoint</td>
<td>StreamExpPoint interface/object</td>
<td>TXPoint interface</td>
</tr>
<tr>
<td>Stream binding</td>
<td>StreamCtrl interface/object</td>
<td>StreamBinder COM object</td>
</tr>
<tr>
<td>Stream</td>
<td>StreamCtrl interface/object</td>
<td>Connect &amp; Transfer operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(StreamBinder COM object)</td>
</tr>
<tr>
<td>Stream flows</td>
<td>FlowConnection interface/object</td>
<td>A stream has only one flow</td>
</tr>
</tbody>
</table>

Table 6.2.: Comparison of modelling approaches for handling continuous media in CORBA and DCOM.

From this table it can be easily deduced that the proposed API and the OMG’s A/V streams specification
have the same scope as they are modelling the same basic concepts (due to their common influence by the
RM-ODP), albeit in different ways. Therefore, they are “conceptually compatible”, although their target
 technological domains are divergent, facilitating thus service developers to map their designs regarding
continuous media interactions easily to either a DCOM or a CORBA DPE. It has to be noted that the
comparison of Table 6.2. wasn’t extended to cover (related) performance issues, because performance
depends greatly on the actual implementation of the OMG A/V streams specification by the different ORB
vendors, and because the result of such a comparison wouldn’t be very important as the decision for the
 adoption of one of the approaches depends almost entirely on the choice of the base DPE technology e.g.
DCOM or CORBA; a choice which is relatively difficult as detailed in Chapter 2 (Sections 3.4. and 3.6.).

6.5. Summary and Conclusions.

There is a technology push in the area of multimedia communications, which is acting as a catalyst for the
specification and development of new multimedia telecommunications services. These services is very likely
to be deployed in a distributed object environment [REDL98][SALE99]. Therefore, there is an increasingly
important need for distributed object platforms to support continuous media interactions in a flexible manner.
Recognising this need, OMG attempts to promote the use of continuous media in CORBA DPEs, by enhancing CORBA with the ability to control and manage continuous media streams together with standard CORBA interactions in an integrated way [OMG98d]. Central role in this approach has the exploitation of easy to use and comprehend programming abstractions for the simplification and improvement of the work of service designers/developers. Therefore, the OMG’s A/V streams specification reveals also the more general trend towards the use of high level APIs in a variety of telecommunications service engineering activities, as for example is the JAIN set of integrated network APIs for the Java platform, which provides a framework to build and combine services that span across different (packet, PSTN, and wireless) networks [KEIJ00].

Taking into account these approaches, this chapter starts by recognising Microsoft’s DCOM as a key potential technology in the area of service engineering and by highlighting the need to facilitate the efficient creation of distributed programs with multimedia data exchanges that execute on distributed object platforms (Section 6.1). Then, the importance of considering control aspects in a flexible manner when modelling multimedia telecommunications services is explained, and significant research and standardisation work related to the flexible handling of multimedia streams is briefly examined (Section 6.2).

The rest of this chapter focuses on the modelling of continuous media communications in DCOM. More specifically, in Section 6.3. a structured approach to enhance DCOM for the handling of continuous media streams is proposed and presented by designing and implementing a number of suitable RM-ODP compliant multimedia support services (which correspond to primitive COM objects) together with a related API, that collectively constitute a multimedia support platform for constructing telematic services with multimedia characteristics in DCOM. This platform, which offers an abstraction over stream communications and multimedia devices, does not affect the core DCOM architecture, but only adds the necessary functionality in terms of additional services (DPE services). Crucial, for the correct and efficient use of the proposed multimedia services, is the role of the stream communication algorithm, which is also discussed in this section, together with a few important implementation considerations pertaining class factories, access to remote COM objects, and the use of the available threading models.

The viability of the proposed modelling approach is evaluated in Section 6.4. (mainly) by the implementation of the MMCS-ET (that took place when validating the proposed service development methodology), which demonstrated that DCOM’s features can be successfully extended to address multimedia requirements in such a way that a substantial amount of software reuse can be achieved. Furthermore, in this section it is argued, by conducting two types of experiments, that the use of object groups simplifies the code of the stream communication algorithm and increases its efficiency, and that the MTA threading model has a better performance and thus it is the preferred choice for the proposed API. The examination of the proposed modelling approach is completed by comparing it with the approach followed by OMG for the handling of continuous media [OMG98d] and commenting on their conceptual relation.
The major research contribution of this chapter, that is related to one of the main objectives of this thesis (see Section 1.4.), is the enhancement of DCOM with the capability to handle continuous media interactions by designing and implementing a set of multimedia support services. In this way, DCOM becomes a TINA-C compliant DPE that fully satisfies Requirements 3, 5 and 7 (see Section 1.3.), and can be used efficiently together with the proposed service development methodology. It is evident that this research contribution includes the stream communication algorithm, the important related implementation issues, the validation of the proposed modelling approach, the experimentation with the use of object groups and different threading models, and the comparison with the OMG’s A/V streams specification [ADA99a][ADA00a][ADA00g]. It has to be noted that although the proposed modelling approach has used DCOM as the target platform, the concepts, the principles, and the design rational presented are general enough to be used for realising stream interface support in other distributed object platforms, such as CORBA and Java RMI.
Chapter 7:

Summary and Conclusions

7.1. Introduction.

This chapter is the epilogue of the thesis that presents the conclusions from the research work that was undertaken and indicates some directions for related future research activities. More specifically, after this brief introductory section (Section 7.1.), Section 7.2. provides a concise overview of the thesis, emphasising on the realisation of its objectives. Then, Section 7.3. highlights the research contributions of the thesis and summarises its main findings. Finally, Section 7.4. identifies and briefly discusses a number of areas in which more research is necessary, and Section 7.5. concludes with some last remarks.

7.2. Thesis Overview.

Over the last decade, telecommunications has dramatically changed from a relatively simple technology in a stable monopolistic era to a fast changing scientific and business field of strategic importance. Recognising this change, Chapter 1 started by identifying and explaining the main trends that characterise the world-wide telecommunications environment, and continued by reasoning about the central role of telecommunications services (and telecommunications software in general) in this emerging environment. It was then argued that in this context information networking is gradually gaining momentum making telecommunications service engineering an important scientific discipline.

After defining this new discipline and commenting about its scope, the need to enable service engineering activities to satisfy a number of carefully selected requirements (see Section 1.3.) was identified as the motivation for the research work presented in this thesis; the telecommunications service engineering framework of Figure 1.1. was proposed to fulfil the expectations created by the identified motivation. For this reason, the main constituent parts of the proposed framework (a service development methodology, a Service Creation Environment (SCE), and a service support environment consisting mainly of service engineering principles and a service execution environment) were then briefly presented, their role was discussed, and their association with the objectives of the thesis was clarified. During this process, the main
Chapter 7: Summary and Conclusions

objectives of the thesis, i.e. to propose and present a service development methodology, to validate and evaluate this methodology, and to enhance Microsoft’s DCOM with the capability to handle continuous media interactions, were identified.

The consideration of the objectives of the thesis started in Chapter 2, which revealed and explained the rationale for proposing the framework of Figure 1.1. by examining advanced approaches in telecommunications service engineering. More specifically, this chapter after attempting to define the term “telematic service”, focused on the object-oriented development of telematic services motivated by the increased penetration of object orientation in the telecommunications world. In particular, it analysed basic concepts of object-oriented technology, it identified the main requirements associated with the design of telematic services, it examined ways in which object orientation can support these requirements, and it proposed the application of object orientation for the complete life cycle of telematic services. It was also argued that current general purpose object-oriented software development methodologies are inadequate for service development purposes, and this led to the decision to propose a specialised service creation methodology.

The next focal point of this chapter was the reasoning about the emergence of Multi-Service Networks (MSNs), the presentation of state of the art descriptions of the technologies of IN and TMN, and the consideration of their possible future integration. Then, it highlighted the necessity for an architectural framework beyond IN and TMN, and examined OSA and TINA-C for this purpose. After presenting these two frameworks focusing on their essential characteristics, it compared and contrasted them emphasising on important approaches followed pertaining their service architecture and component models. TINA-C was finally chosen to be the service support environment of the proposed telecommunications service engineering framework of Figure 1.1. The rationale for this choice was then presented, the way in which TINA-C satisfies Requirements 2, 3, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, and 16 (see Section 1.3.) was explained, and the practical value and future perspectives of TINA-C were discussed. After selecting TINA-C and commenting on the difficulty of applying it, the rest of this chapter was devoted to SCEs. More specifically, it identified their main characteristics, it examined important related approaches, it highlighted their relationship with the service development methodology, it reasoned about their role and purpose in the proposed framework, and it explained under which circumstances a SCE is able to satisfy Requirement 4 (see Section 1.3.).

As can be seen in Figure 1.1. the service support environment of the proposed framework has two main constituent parts. Although the service engineering principles are specified by the TINA-C service architecture, the service execution environment (the most important part of which is the DPE) is only influenced by the TINA-C approach regarding the DPE, as TINA-C does not suggest a particular DPE. For this reason, Chapter 3 examined in considerable detail important issues and technologies regarding DPEs, justified their central role in the service execution environment, and explained the way in which they satisfy.
Chapter 7: Summary and Conclusions

Requirement 3 (see Section 1.3.). More specifically, this chapter started by introducing open distributed processing and examining the most important de jure and de facto DPEs (ISO's / ITU-T's RM-ODP, Open Group’s DCE, OMG’s CORBA, Microsoft’s DCOM, and Sun’s Java RMI) from the perspective of telecommunications service engineering. After presenting a state of the art description of these DPEs, it evaluated them, and proposed a potential path to convergence, which revealed that DCOM and CORBA are currently the most significant DPE technologies for service engineering activities.

Recognising the importance of DCOM and CORBA, and considering the benefits from engineering telematic services as distributed object applications, the rest of Chapter 3 attempted to compare and contrast DCOM and CORBA with the objective to clarify important issues regarding their capabilities and use in the area of service engineering, and also guide service developers during a possible selection process between them. In particular, it proposed a decision framework, constructed by a set of basic, core and service engineering related properties, it specified the way that DCOM and CORBA support these properties, it examined the performance of DCOM and CORBA by conducting a suitably structured experiment, and it considered their interworking with the WWW. Then, it discussed the most significant results of this comparison attempt, it presented key issues regarding the interworking between DCOM and CORBA, and after considering some wider service development matters, together with the main TINA-C requirements from a DPE, it selected DCOM as the DPE technology that would be used and examined in the rest of the thesis, explaining also the rationale for this choice.

The first main objective of the thesis, that constitutes an important and central research contribution of it, was examined in Chapter 4, which proposed and presented a service development methodology, illustrating its ability to enable service engineering activities to satisfy Requirements 1, 2, 5, 8, 9, 11, and 14 (see Section 1.3.). More specifically, this chapter initially explained the necessity for such a methodology, by introducing the concept of the service life cycle, and highlighted its significance and novel character, by summarising and critically examining related approaches found in the area of service engineering. After fully justifying the importance of proposing a service creation methodology, it discussed a number of service modelling issues together with the decision to use the UML notation, and it presented an overview of the proposed methodology, considering also significant matters related to the service development process. Then, it presented and examined in detail all the phases of the proposed methodology focusing on their essential characteristics and artifacts, and concluded by highlighting the importance of supporting the methodology with appropriate and carefully selected software tools.

The proposed service development methodology was validated and evaluated (according to the second main objective of the thesis) in Chapter 5, which ensured the practical value, the applicability, the correctness and the completeness of the proposed methodology, and illustrated that it is able to satisfy Requirements 1, 2, 5, 8, 9, 11, and 14 (see Section 1.3.). More specifically, the starting point of this chapter...
was the validation of the proposed methodology, which was attempted by applying it to the development of telematic services. In particular, the chosen validation approach was justified, the development of a complex representative telematic service (a MultiMedia Conferencing Service for Education and Training, MMCS-ET), according to the main phases of the proposed methodology, was described and discussed, and a number of additional simple validation attempts were outlined, focusing on the explanation of the service scenarios that they incorporate. Then, based on the validation results, a number of general purpose design patterns (specifically constructed for the proposed methodology) were proposed and presented, together with the basic characteristics of service engineering design patterns. Chapter 5 ended by evaluating the service development methodology according to the NIMSAD framework and by highlighting the results of this evaluation.

Figure 7.1.: A detailed version of the proposed telecommunications service engineering framework.

Considering that Chapter 3 selected DCOM as the DPE of choice, Chapter 6 was devoted to the last main objective of the thesis, and proposed and examined an approach for enhancing DCOM with the capability of handling continuous media interactions. In this way it enabled DCOM to transform to a TINA-C compliant DPE, to become suitable for use with the proposed service development methodology, and to fully satisfy Requirements 3, 5, and 7 (see Section 1.3.). More specifically, this chapter initially identified the main issues related to the modelling of multimedia telecommunications services and reviewed briefly
important state of the art activities regarding the handling of continuous media. It then emphasised the importance of flexibility when supporting continuous media communications in DCOM and considered briefly the capability of COM and DCOM to handle multimedia streams. The rest of this chapter focused on the examination of a multimedia support platform for DCOM that was proposed to facilitate the construction of multimedia telecommunications services. In particular, it presented and explained a number of multimedia support services, in the form of COM objects, used for this purpose, it proposed and analysed an appropriate stream communication algorithm, and it discussed some important implementation details. The proposed modelling approach was then validated through the design and implementation of the MMCS-ET, and its flexibility and efficiency was assessed by conducting two types of experiments, reasoning about the results, and comparing it with the OMG's A/V streams specification.

Taking into account all the chapters of the thesis, it is evident that the main and secondary objectives of it are realised. Furthermore, the proposed telecommunications service engineering framework of Figure 1.1. has been transformed to a more precise and concrete framework (depicted in Figure 7.1.) that fully satisfies all the requirements identified in Section 1.3. as the motivation of the research work presented in this thesis.

The detailed version of the proposed framework strengthens the belief that telecommunication networks and data networks are converging and reveals that this convergence is also manifested in the service engineering realm. With the use of the proposed framework, paradigms of software development and associated practices can more easily be diffused between telecommunications and data networks. Since the latter have from their inception been digital and have relied heavily on software due to their greater processing needs, this diffusion is not actually symmetric. It is rather that practices and procedures cultivated in data networks are beginning to make their way into telecommunications networks as the latter become more similar to data networks. The proposed framework facilitates this transition from the legacy telecommunications world since it proposes the use of distributed object platforms as "signalling" infrastructures for the realisation of unified service control and management.

7.3. Research Contributions and Main Findings of the Thesis.

The research contributions of the thesis were identified and explained at the last section of Chapters 2, 3, 4, 5, and 6. In this section, these contributions are highlighted once again by discussing in a concise manner the main findings of the thesis that are related to them.

The most important research contributions of the thesis, which correspond to its main objectives (see Section 1.4.), are the following:

1) The proposal and presentation of a service development methodology.

A complete methodology for the development of telematic services was devised, structured, proposed, and examined in detail (see Chapter 4) [ADA98a][ADA99b][ADA00b][ADA00c][ADA00d][ADA00f]. This
methodology "covers" the entire service creation process through a requirements capture and analysis phase, a service analysis phase, a service design phase, a service implementation phase and a service validation and testing phase, considers telematic services as distributed object-oriented applications operating upon distributing object platforms, exploits fundamental object-oriented analysis and design concepts, and has the following novel characteristics (see also Section 4.13.):

- It addresses in sufficient detail all the service creation phases.
- It emphasises the satisfaction of the user requirements.
- It adopts an incremental and iterative use case driven approach.
- It considers the TINA-C service architecture in a critical manner.
- It promotes the use of the UML notation.
- It exploits design patterns.
- It facilitates reusability.

2) The validation and evaluation of the proposed methodology.

The proposed service creation methodology was validated by its application to the development (from requirements elicitation up to actual implementation) of a complex representative telematic service (a MMCS-ET) and to several simple service scenarios (see Section 5.2.) [ADA00b][ADA00f][PAPA97] [PAP98a][PAP98b]. These validation attempts (see also Section 5.5.):

- Enlightened many aspects regarding the structure and the use of the proposed methodology.
- Ensured its applicability, correctness, consistency, flexibility, and effectiveness.
- Proved its usefulness and its practical value.
- Offered confidence that it can enable the fast and efficient creation of telematic services.
- Illustrated the way that the methodology can be applied to the development of specific telematic services.
- Revealed different perspectives for considering some matters and provided important feedback which resulted to the improvement and enhancement of the proposed methodology.

For the evaluation of the proposed methodology, the NIMSAD framework was used structured appropriately according to its essential elements (the "problem situation", the intended problem solver, and the problem-solving process) (see Section 5.4.). In this way (see also Sections 5.4. and 5.5.):

- The completeness of the proposed methodology was proved as it was found that it addresses all the issues that a premium methodology for the development of systems should address.
- Its applicability and its overall quality were ensured.
- A number of important issues regarding its use were clarified.
- Its objectives, features, nature, and philosophy were better explained.
• Service developers were enabled to customise the methodology according to the requirements of a particular circumstance in the best possible way.

• The necessity to apply the methodology in a critical manner was highlighted.

3) The enhancement of DCOM with the capability of handling continuous media interactions.

A structured approach that enables DCOM to handle continuous media streams was proposed and presented by designing and implementing a number of suitable RM-ODP compliant multimedia support services (which correspond to primitive COM objects) together with a related API, that collectively constitute a multimedia support platform for constructing telematic services with multimedia characteristics in DCOM (see Chapter 6) [ADA99a][ADA00a][ADA00g]. The proposed modelling approach, which includes also a stream communication algorithm, was validated through the design and implementation of the MMCS-ET, went through experimentation regarding the use of object groups and different threading models, and was compared with the OMG's A/V streams specification. In this way (see also Sections 6.4. and 6.5.):

• The flexibility, consistency, and intuitive character of the proposed modelling approach were ensured.

• Its applicability and its true practical value were proved.

• The importance of using object groups, as the number of stream connections is increasing, was justified.

• The rationale for preferring the use of the MultiThreaded Apartment (MTA) threading model was explained.

• The conceptual compatibility of the proposed modelling approach with the OMG's A/V streams specification was illustrated.

Except from the above central research contributions, several other parts of the thesis constitute a secondary research contribution, because they clarify some issues, examine in depth certain properties, reason about some decisions or structure / synthesise some concepts / processes into something basically new. These secondary research contributions correspond to the secondary objectives of the thesis (see Section 1.4.), and are highlighted in the following paragraphs by presenting the related findings of the thesis structured according to the constituent parts of the proposed telecommunications service engineering framework of Figure 1.1. (in the order that they are examined in the thesis).

The main findings related to the service support environment are the following (see Section 2.7.):

• A telematic service is defined as a geographically distributed entity providing a number of people a predefined, carefully selected, set of capabilities / facilities regarding the integrated coverage of a wide range of needs, utilising the resources of telecommunication networks (see also Section 2.2.) [ADA98a][ADA99d].

• The application of object orientation to the development of telematic services is more likely to start from the service design phase. When designing telematic services it is necessary to support quality, modularity, the management of complexity, and distributed processing. All these requirements can be efficiently supported by the object-oriented paradigm (see also Section 2.3.2.) [ADA98c].

233
Chapter 7: Summary and Conclusions

- The full potential of object orientation can only be realised when it is applied to the complete life-cycle of a telematic service (see also Section 2.3.3.) [ADA98c][ADA98d].

- General purpose object-oriented software development methodologies are unable to efficiently support service development mainly because, due to their (intentionally) abstract nature, their customisation for service creation purposes is generally a difficult and time consuming task that involves a significant amount of work. Furthermore, the customisation of such a methodology increases the possibility of inheriting its deficiencies, while it does not guarantee the preservation of its advantages (see also Section 2.3.3.) [ADA00f].

- The two most important architectural frameworks that support the control and management of telecommunications services in an integrated manner and promise to make the future open information market a reality are OSA and TINA-C. Although they have many similarities and present a surprising commonality in the technical approach that they follow, they also have important differences in scope and their architectures are expressed in different ways. Between them, TINA-C appears to be more robust, more complete and more promising for the future, as various problems, weaknesses and pitfalls will be tackled by the ongoing efforts of the TINA consortium (see also Section 2.5.) [ADA98d].

- TINA-C is already a mature architectural framework that is flexible in adapting new and existing approaches and technologies (e.g. IN, TMN, the Internet, mobile communications). It is gaining acceptance at a steady rate, through validation activities, the appearance of related products, and the examination of interworking concepts and migration strategies from current and emerging technologies. Therefore, TINA-C is a very interesting and promising solution for the middle and long term. Furthermore, TINA-C is not an "all-or-nothing" approach and the principles and concepts of TINA-C will continue to be very valuable as a conceptual guideline in any context, irrespective of the penetration of the complete TINA-C architectural framework (see also Section 2.5.4.) [ADA98d][ADA00f].

The main findings related to the service creation environment are the following (see Section 2.7.):

- A SCE is a logical framework incorporating a collection of appropriate, carefully designed and tested, customisable, and user-friendly software tools together with a reuse infrastructure (see also Section 2.6.).

- A SCE is in close cooperation with the service development methodology being used and aims to increase its applicability and not to substitute it. For this reason, a SCE is proposed to be constructed (or customised) by the service developer(s), prior to the start of a specific service development project, by considering a number of factors related to the project (see also Section 2.6.).

The main findings related to the service execution environment (except from those that constitute a central research contribution) are the following (see Section 3.7.):

- The evaluation of the most significant existing de jure and de facto DPEs (ISO's/ITU-T's RM-ODP, Open Group's DCE, OMG's CORBA, Microsoft's COM / DCOM, Sun's Java RMI) from the perspective of telecommunications service engineering, revealed that DCOM and CORBA are the only two DPE
Chapter 7: Summary and Conclusions

technologies that have currently the capabilities to support service engineering activities in an efficient and effective manner. However, irrespective of this fact, the various initiatives in the field of open distributed processing are complementary with a potential path to convergence (see also Section 3.2) [ADA98d].

- The most important benefits offered by distributed object technology to telematic services engineered in this way are ease of development and maintenance, abstraction, modularity, reusability, and granularity flexibility (see also Section 3.3) [ADA99c].

- There is no doubt that both DCOM and CORBA are important for the realisation of telematic services in today's heterogeneous information networking environment. A selection between them can be facilitated by a decision framework, including a number of basic, core and service engineering related properties, and examining the way that DCOM and CORBA support these properties (see also Section 3.4) [ADA99c].

- DCOM and CORBA have a comparable performance under the MS Windows operating system platform, although DCOM appears to be more flexible and with a significant potential for improved performance due to its extensible and customisable remoting architecture. Furthermore, in both DCOM and CORBA remote method calls are slower than local method calls, and many single method calls are five to ten times slower than a single multiple method call (see also Section 3.4.3) [ADA00e].

- DCOM and CORBA are equally capable to interoperate with the WWW and the Internet, and there are a number of possible approaches for such an interoperation (see also Section 3.4.4) [ADA98b].

- DCOM and CORBA have much in common and continue to converge in several aspects, although each architecture has different origins, with consequent strengths and weaknesses. Both of them provide a solid distributed object infrastructure and there are specific scenarios in which each excels over the other. However, neither technology provides a complete solution for service engineering activities (see also Section 3.4.5) [ADA99c][ADA00e].

- Selecting between DCOM and CORBA for a specific service engineering activity can be a complicated task that requires experience, an accurate appreciation of the main parameters of the problem domain, and an ability to consider the merits of these technologies from a less technical and more strategic perspective. Nevertheless, DCOM is more suitable for the development of telematic services in Windows-based environments, particularly by medium and small organisations and departments, while CORBA is preferable when there is a requirement for multi-platform support or for a choice with the least technological risk (see also Section 3.6) [ADA99c][ADA00e].

- The main requirements that TINA-C has from a DPE are satisfied by DCOM, after its enhancement with the capability to model continuous media interactions, and by an ORB implementing CORBA 3.0 specification (see also Section 3.6) [ADA98d].

- Control aspects are important when modelling multimedia telecommunications services and flexibility is a desired property when modelling continuous media interactions (see also Section 6.2) [ADA99a][ADA00a][ADA00g].
Chapter 7: Summary and Conclusions

The main findings related to the service development methodology (except from those that constitute central research contributions) are the following (see Sections 4.13. and 5.5.):

- A service development methodology is necessary because service creation is not only one of the most important stages of the service life-cycle, but also one of the most abstract and general. It is also necessary because of the inefficiencies and the shortcomings of the related approaches that are found in the service engineering literature, and because general purpose object-oriented methodologies and RM-ODP fail to support efficiently the service creation process (see also Section 4.2.) [ADA99b][ADA00c][ADA00f].

- As is revealed by the main validation attempt of the proposed service creation methodology (i.e. the development of the MMCS-ET), a conferencing telematic service with multimedia characteristics that can be used for education and training purposes, involves the support of various session management, interaction, and collaboration support requirements (see also Section 5.2.1. and Appendix A) [ADA00b][ADA00f][PAP98a].

- The simple service scenarios used together with the MMCS-ET for validating the proposed methodology can also be used for similar validation purposes in the service engineering area (see also Section 5.2.2.) [PAP98c][PAPA99].

- The general purpose and the service engineering design patterns that enhanced the proposed methodology decrease the possibility of poor choices, improve the quality of the design of service interaction diagrams, increase the possibility of a skilful implementation, and assist significantly service developers in this process. Furthermore, they ensure the effective and efficient communication of architectural knowledge between the service developers, and reduce the scope of the problem solving process in the case of service creation, because they communicate “best practice” idioms and solutions for the development of telematic services (see also Section 5.3.) [ADA99b][ADA00c][ADA00f].

- The structuring of the (otherwise abstract, verbose and difficult to use) NIMSAD framework in a concise but detailed manner, using its essential elements as a guide, improves significantly its applicability and increases its value (see also Section 5.4.).

7.4. Possible Future Work.

It is commonly the case with research that, as progress is made, a substantial amount of further work is generated. This section therefore highlights a number of areas in which the research presented in this thesis may be continued and extended. These areas, which correspond to potential future research directions, are:

The seamless integration of mobile agents into the proposed service development methodology

Agents are software entities that have certain properties such as autonomy, mobility, intelligence, etc. There is no concrete and agreed definition of agents; rather agents are defined by the properties they have. Agents provide a new design and programming paradigm, and more specifically mobile agent technology has been proposed to address complexity problems related with service creation and deployment. However, the inte-
Integration of mobile agents into the service creation process should not be decided on an ad-hoc basis. The proposed methodology should be carefully adapted and specialised to take into account agent-related concepts, and should provide guidelines on which service components are beneficial for implementation as mobile agents, and on how parts of a service may be modelled and implemented using agents [HAGE98].

The consideration of mobility requirements by the proposed methodology

The increased expectations from the rapid evolution of mobile communication systems, in combination with the current proliferation of mobile telephony (and other related supplementary services), make evident the need to take into account the different facets of mobility (terminal, personal, and service mobility) in the proposed service development methodology. The consideration of the requirements and features involved in mobility will affect all the phases of the proposed methodology, and should be done in alignment with related research work regarding the TINA-C service architecture (see Section 2.5.4.) and with standardisation work pertaining the Universal Mobile Telecommunication System (UMTS), developed by ETSI, and the International Mobile Telecommunications 2000 (IMT2000), developed by the ITU-T.

The enhancement of the proposed methodology for the detailed support of federation and composition

Currently, the proposed methodology offers only limited support for service federation and composition (see Section 4.9.3.). However, in the not so distant future, situations in which consumers have access sessions with different retailers and would like to engage in the same service session, are expected to occur more often. Then, service federation and composition mechanisms are needed between the different retailers. For this reason, the phases of the proposed methodology should be tuned to take into account issues of service composition and federation. Necessary is also the examination of several related matters that are still open in TINA-C, such as the support for more than two domains, on-line and off-line contract negotiation, handling of session invitations, and session model negotiation.

The consideration of integrity requirements during service creation

As modern telematic services are becoming increasingly complex, and due to regulatory demands for operator interworking, the ability of services to retain their operational state of high integrity is crucial. The number of integrity issues in such services are vast (e.g. robustness, resilience, availability, performance, scalability, data coherence, liveness, safety, feature interaction, complexity, reliability, security) and these issues must be understood and managed throughout service development, testing, deployment, and maintenance. Therefore, in parallel with the application of the service development methodology, service developers should ensure that integrity requirements are satisfied by adopting carefully selected and structured integrity-preserving policies.

The further exploitation of design patterns in the service engineering area

Design patterns can provide a powerful means of representing proven solutions to problems which arise in course of service development, as they can capture knowledge gained during the service creation process,
Chapter 7: Summary and Conclusions

which might be usefully reused in future service development activities. Therefore, the careful construction, in alignment with the proposed methodology, of more service engineering design patterns (see also Section 5.3.) regarding a variety of service requirements, will undoubtfully facilitate the task of service developers. Furthermore, design patterns can provide important contextual information about the intention / purpose, the appropriate uses, the previous uses, and the design trade-offs embodied in service components. The effective use of this information during service component reuse is also a significant open research issue.

The experimentation with the use of DCOM for implementing a variety of telematic services

Chapter 3 revealed that the importance of DCOM in service engineering is expected to increase. Therefore, further experience on the use of DCOM for implementing various telematic services is needed, as it is a prerequisite for extending DCOM in order to provide real-time functionality through an appropriate API, for examining the optimum way that DCOM's performance can be improved by customising its remoting architecture under different service provisioning conditions, and for creating a library of reusable DCOM service components and considering component adaptation issues and the structuring / functionality of reuse infrastructures for DCOM. Finally, the efficient interoperability between DCOM and CORBA is necessary to be more carefully examined under various conditions, possibly with the help of the proposed service creation methodology, as its application is independent from a specific distributed object technology.

7.5. Concluding Remarks.

The telecommunications industry is currently facing a number of challenges imposed by changes in the telecommunications market. Deregulation, liberalisation and competition, fuelled by rapid technological developments, imply requirements for higher utilisation of the network infrastructure, shorter time to market for new telecommunications services, much higher degree of customisation of these services, cost reduction of service development, open network provision, global connectivity, and global information access. Furthermore, both telecommunications services and networks are ever growing in sophistication and complexity with a tendency to become large-scale, highly decentralised and heterogeneous systems involving numerous users and resources. All these changes require more complex software systems and thus make evident the necessity to accelerate the integration of information technology and telecommunications.

The research work undertaken in this thesis, together with the proposed telecommunications service engineering framework (depicted in Figures 1.1. and 7.1.), promise to facilitate such an integration and provide the basis for the evolution of the service engineering discipline according to the main trends that characterise the world-wide telecommunications environment (see Section 1.1.). Therefore, this thesis can be considered as a necessary foundation for the provision of a great variety of new sophisticated telematic services with multimedia characteristics (in a highly competitive environment of service provisioning, consisting of a multiplicity of cooperating and competing providers), that will eventually realise the open services market, creating a ubiquitous information and communication system available to, and usable by, all.
Bibliography

[ACT96a] ACTS Project AC227 (SCREEN), "Basic Object-Oriented Technology for Service Creation", CEC Deliverable D21, November 1996.


239
Bibliography


Bibliography


<table>
<thead>
<tr>
<th>Reference</th>
<th>Title and Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>[EUR99a]</td>
<td>Eurescom, &quot;TelCos Users Guide to TINA&quot;, Deliverable 1 of Project P847 &quot;What is TINA and is it Useful for the TelCos?&quot;, February 1999.</td>
</tr>
</tbody>
</table>


Bibliography


Bibliography


Creation and Deployment of Telecommunication Services in Heterogeneous Distributed Processing

for Reuse Based Production", Proceedings of IS&N '94, LNCS, Vol. 851, Springer-Verlag, Berlin,

[PROZ97] Prozeller, P., "TINA as the Software Infrastructure of the Telecom Network of the Future",

[RAC92b] RACE Project R2019 (CASSIOPEIA), "Initial Statement of Requirements on Methods,

[RAC94a] RACE Project R2017 (SCORE), "SCORE Service Creation Methods and Tools Set",

MAR/MCS/DR/R/017/b0, Brussels, Belgium, September 1994.

[RAC95a] RACE Project R2019 (CASSIOPEIA), "Open Services Architectural Framework for Integrated

[RAC95b] RACE Project R2017 (SCORE), "The SCORE Component Model and its Framework"


[RAC95d] RACE Project R2017 (SCORE), "Service Creation Environment Concepts", Deliverable D26,
August 1995.


C/FR/b2, November, 1996.

124-127.

[RALP96] Ralph, B., Reid, A., "Convergence - Synthesising a New Industry", British Telecommunications


[REEN92] Reenskaug, T., et al., "OORASS: Seamless Support for the Creation and Maintenance of

256


Bibliography


Bibliography


Appendix A:

Detailed Presentation of the Development of a MMCS-ET using the Proposed Methodology

A.1. Introduction.

As was mentioned in Section 5.2, the service creation methodology that is proposed and presented in Chapter 4 is validated (mainly) by using it for the development of a MultiMedia Conferencing Service for Education and Training (MMCS-ET). This validation attempt is described in considerable detail in this appendix, complementing Section 5.2.1., as it takes into account the entire functionality of the MMCS-ET. The focus is on the important artifacts that were created during the application of the main phases of the proposed methodology. In order to avoid overlapping with Section 5.2.1., appropriate references to this section are used in this appendix, whenever it is considered to be necessary.

A.2. Requirements Capture and Analysis Phase.

The main objective of the MMCS-ET (step 2, activity 2) is to enable a teacher / trainer to teach a specific course to a number of geographically dispersed / distributed students / trainees. The service should establish an educational session between the teacher and the remote students that is equivalent to the educational that would have been established between the same people (teacher and students) in a traditional classroom.

The results of the analysis of the constituent parts of the main objective of the MMCS-ET (step 3, activity 2) are:

- "a teacher / trainer": Every educational session has at least one teacher who can participate in only one educational session at a given time.

---

1 Such statements refer always to the phase of the methodology that is currently examined. More details are given in Chapter 4 where the proposed methodology is presented.

2 For reasons of simplicity, the rest of Appendix A will refer to teachers and students implying teachers / trainers and students / trainees respectively.
• "teach": Teaching is the core action that the MMCS-ET should enable and facilitate. It is the main objective of an educational session and the reason for its establishment.

• "specific course": In every educational session a course is being taught by the teacher to the students. This course is selected by the teacher before the establishment of the educational session. The selection of a course (e.g. from a course database) is the responsibility of the teacher and is not facilitated by the MMCS-ET.

• "a number of geographically dispersed / distributed students / trainees": The students can be located in different places, as long as they have available a personal computer and network connectivity. In the most simple case, only a teacher and one student participate in an educational session. A student can’t participate in more than one educational sessions simultaneously.

• "traditional classroom": In a traditional classroom the teacher and the students are all (physically) located in the same classroom. When the teacher and the students are geographically distributed they constitute a virtual classroom (or virtual class).

• "The service should establish an educational session between the teacher and the remote students..." / "... the educational session that would have been established between the same people in a traditional classroom": Every time a teacher teaches a number of (local or remote) students an educational session takes place. The establishment of such a session involves the creation of the session by the teacher and the invitation of students to participate in the session. For example, in a traditional classroom the teacher creates the educational session by entering a classroom and invites the students either verbally or by ringing a bell. It is obvious that in order to have a session at least one student has to accept the invitation (and not be a participant to another session). Only a teacher can create an educational session. A student can’t create an educational session and can’t invite a new participant (teacher or student) to a session. A student participating in a session can only invite to direct communication with him / her another student, who is also (already) a participant in the same session.

• "equivalent to": The educational session established in a virtual classroom by the MMCS-ET should be equivalent to the educational session established in a traditional classroom and should have as many characteristics as possible in common with it. More specifically, in a traditional classroom the teacher has the ability to manage the educational session. Besides establishing the session he / she can also modify the session, suspend and resume the session, and finally finish the session. The same capabilities should also characterise the teacher in a virtual classroom. Furthermore, in a traditional classroom the teacher and the students can interact (and possibly collaborate) during an educational session by seeing and talking to / hearing each other, by writing at their notepads, by writing at the blackboard of their classroom, and by exchanging course material (e.g. documents, graphs, etc.). Therefore, in a virtual classroom there is a need for audio / video (A/V), text, and file communication between the session participants, and collaboration among all the session participants in order to perform a common task.
As was explained in Section 5.2.1.2, the following use cases are considered (activity 4):

- Contact the MMCS-ET provider (start up the MMCS-ET service).
- Log in to the MMCS-ET provider domain.
- Start a new MMCS-ET session.
- Invite a student to join a MMCS-ET session.
- Join a MMCS-ET session after being invited.
- Invite a student (active user) to direct communication.
- Accept direct communication with a student (active user).
- Engage in text communication with an active user.
- Engage in file communication with an active user.
- Engage in A/V communication with a student (active user).
- Stop A/V communication with a student (active user).
- Start A/V communication with a student (active user).
- Terminate A/V communication with a student (active user).
- Engage in a chat with all active users.
- Engage in file communication with all students (active users).
- Start a voting process between all active users.
- Vote in a voting process between all active users.
- Present the outcome of a voting process to all the involved active users.
- Terminate direct communication between two students (active users).
- Remove a student from a MMCS-ET session.
- Terminate a MMCS-ET session.
- Terminate the MMCS-ET service.

These use cases (except from the target use cases) are expressed in a high-level format (activity 4) in the following way:

**Use Case:** Contact the MMCS-ET provider domain.
**Actors:** Teacher (initiator) or student (initiator).
**Type:** Primary.
**Description:** A user (teacher or student) starts the MMCS-ET service. This is done by contacting the MMCS-ET provider domain and preparing the MMCS-ET service for use.

**Use Case:** Log in to the MMCS-ET provider domain.
**Actors:** Teacher (initiator) or student (initiator).
**Type:** Primary.
**Description:** A user (teacher or student) logs in to the MMCS-ET provider domain by specifying an appropriate user name and password. On completion, the MMCS-ET service can correctly identify the user by a user name.

**Use Case:** Start a new MMCS-ET session.
**Actors:** Teacher (initiator).
**Type:** Primary.
**Description:** A teacher (logged in user) starts a new educational / training session. On completion, an educational / training session is created and the teacher is ready to invite students (logged in users) to join the session.
Appendix A: Detailed Presentation of the Development of a MMCS-ET

Use Case: Invite a student (active user) to direct communication.
Actors: Two students (one of them is the initiator) - the one that makes the invitation.
Type: Primary.
Description: A student participating in a MMCS-ET session invites another student participating in the same MMCS-ET session to direct (text, audio / video, file) communication. On completion, the latter student is left to decide whether to accept this invitation by the former student.

Use Case: Accept direct communication with a student (active user).
Actors: Two students (one of them is the initiator - the one that replies to the invitation).
Type: Primary.
Description: A student participating in a MMCS-ET session accepts the invitation from another student participating in the same MMCS-ET session to engage in direct (text, audio / video, file) communication with him / her. The latter student is informed about the decision of the former student. On completion, the two students are ready to engage in direct communication with each other.

Use Case: Engage in file communication with an active user.
Actors: Teacher (initiator), student or student (initiator), teacher or two students (one of them is the initiator; the one that sends the file).
Type: Primary.
Description: A user, participating in a MMCS-ET session, sends a file to another user, participating in the same MMCS-ET session. Both users are ready to engage in direct communication with each other. On completion, the two users are engaged in file communication.

Use Case: Engage in A/V communication with a student (active user).
Actors: Teacher (initiator), student or two students (one of them is the initiator).
Type: Primary.
Description: A user participating in a MMCS-ET session initiates the establishment of two A/V stream connections with another user participating in the same MMCS-ET session. Both users are ready to engage in direct communication with each other. On completion, the two users are engaged in A/V communication with each other.

Use Case: Stop A/V communication with a student (active user).
Actors: Teacher (initiator), student or two students (one of them is the initiator).
Type: Primary.
Description: A user participating in a MMCS-ET session stops to communicate using audio and video with another user participating in the same MMCS-ET session. On completion, the A/V communication between the two users is blocked.

Use Case: Start A/V communication with a student (active user).
Actors: Teacher (initiator), student or two students (one of them is the initiator).
Type: Primary.
Description: A user participating in a MMCS-ET session starts a blocked A/V communication between him / her and another user participating in the same MMCS-ET session. On completion, the two users are engaged in A/V communication with each other.

Use Case: Terminate A/V communication with a student (active user).
Actors: Teacher (initiator), student or two students (one of them is the initiator).
Type: Primary.
Description: A user participating in a MMCS-ET session terminates an established A/V communication between him / her and another user participating in the same MMCS-ET session. On completion, the two users can't communicate using audio and video.

Use Case: Engage in a chat with all active users.
Actors: All active users (teacher, students) - one of them is the initiator (the one that sends the message which is broadcasted).
Type: Primary.
Description: A user, participating in a MMCS-ET session, sends a (text) message to all the other users participating in the same MMCS-ET session. On completion, all these users are engaged in a chat.

Use Case: Engage in file communication with all students (active users).
Actors: Teacher (initiator), all active students.
Type: Primary.
Description: A teacher, participating in a MMCS-ET session, sends a file to all the students, participating in the same MMCS-ET session. On completion, all these users are engaged in file communication with all the students.

Use Case: Start a voting process between all active users.
Actors: Teacher (initiator), all active users.
Type: Primary.
Description: A teacher initiates a voting process between all the users that participate in the MMCS-ET session that he / she created. On completion, all users are asked for their vote on a specific matter.

Use Case: Vote in a voting process between all active users.
Actors: Teacher (initiator) or student (initiator).
Type: Primary.
Description: A user (teacher or student) participating in a voting process started by the teacher, expresses his / her voting decision by sending a vote. On completion, the user is informed that his / her vote has been received.

Use Case: Present the outcome of a voting process to all the involved active users.
Actors: Teacher (initiator), all active users.
Type: Primary.
Description: The teacher requires the calculation of the outcome of the voting process that he / she started and the presentation of this outcome to all the involved active users. On completion, the voting process is considered to be finished.
Appendix A: Detailed Presentation of the Development of a MMCS-ET

Use Case: Terminate direct communication between two students (active users).
Actors: Two students (one of them is the initiator - the one that requests the termination).
Type: Primary.
Description: A student participating in a MMCS-ET session requests the termination of his / her direct (text, audio / video, file) communication with another student participating in the same MMCS-ET session. On completion, the two students are not engaged any more in direct communication with each other.

Use Case: Remove a student from a MMCS-ET session.
Actors: Teacher (initiator), student.
Type: Primary.
Description: A teacher removes, from the MMCS-ET session that he / she created, a student who is participating in this MMCS-ET session. On completion, the student is not engaged in direct (text, audio / video, file) communication with the teacher, and he / she is not a participant of the MMCS-ET session any more.

Use Case: Terminate a MMCS-ET session.
Actors: Teacher (initiator).
Type: Primary.
Description: A teacher terminates the MMCS-ET session that he / she created. On completion, all the students that were participants of the MMCS-ET session are removed and the teacher is ready to start a new MMCS-ET session.

Use Case: Terminate the MMCS-ET service.
Actors: Teacher (initiator) or student (initiator).
Type: Primary.
Description: A user (teacher or student) terminates the MMCS-ET service that he / she wanted to use when he / she logged in to the MMCS-ET provider domain. On completion, the user is ready to start again the MMCS-ET service.

Important terms, related to the MMCS-ET, that are used during the service development process are defined in the glossary (activity 6). The glossary also records important assumptions regarding the MMCS-ET. These are the following:

- A teacher can only create MMCS-ET sessions and not sessions of another service. In the same manner, a student can participate only in MMCS-ET sessions (after being invited by a teacher) and not in sessions of another service.

- The subscription of a user (teacher or student) can’t end while he / she is active, i.e. while he / she participates in a MMCS-ET session.

- A student joins a MMCS-ET session after being invited by a teacher. The student should not refuse such an invitation and should not terminate the service without being removed by the teacher first. Furthermore, an active student should not refuse an invitation for direct communication by another active student of the same MMCS-ET session. Finally, a student should always respond to invitations.

- Two different teachers and two different group of students can participate in two different MMCS-ET sessions at the same time (with two different active user information lists). However, the MMCS-ET has been tested with one teacher and two students participating in a MMCS-ET session, without the presence of another MMCS-ET session at the same time.

- Instead of a videocamera, for the A/V interaction between users of the service a number of .avi files are interchanged continuously between the users.

- The terminal configuration of the users is suitable for the MMCS-ET.

- There are no specific policies that apply on invitations of students by a teacher.
Appendix A: Detailed Presentation of the Development of a MMCS-ET

- The subscription profile of a student permits the direct communication with another student when both students participate in the same MMCS-ET session.
- A student (active user) can't be removed from a MMCS-ET session by the teacher before answering all the invitations for direct communication by other active students.
- A student (active user) can't be removed from a MMCS-ET session by the teacher while there is voting activity. The student must vote first. He / she must also be informed about the result of the voting process.
- The list containing information about all the messages exchanged recently between two users can store up to ten (10) messages.
- The list containing information about all the messages broadcasted recently to all the users that participate in the MMCS-ET session can store up to ten (10) messages.

A.3. Service Analysis Phase.

The MMCS-ET use cases are expressed in an expanded format (activity 1) in the following way (except from the target use cases, see Section 5.2.1.3):

1) Use case “Contact the MMCS-ET provider (start up the MMCS-ET service)”:  

<table>
<thead>
<tr>
<th>Use Case:</th>
<th>Contact the MMCS-ET provider domain.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>Teacher (initiator) or student (initiator).</td>
</tr>
<tr>
<td>Purpose:</td>
<td>Start up the MMCS-ET service.</td>
</tr>
<tr>
<td>Overview:</td>
<td>A user (teacher or student) starts the MMCS-ET service. This is done by contacting the MMCS-ET provider domain and preparing the MMCS-ET service for use.</td>
</tr>
<tr>
<td>Type:</td>
<td>Primary and essential</td>
</tr>
<tr>
<td>Cross References:</td>
<td>Service Functions: SF.1.1.</td>
</tr>
</tbody>
</table>

Use Case: -

Typical Course of Events:

<table>
<thead>
<tr>
<th>Actor Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. This use case begins when a user (teacher or student) decides to start a MMCS-ET service.</td>
</tr>
<tr>
<td>2. The user specifies the name of the service (MMCS-ET) that he / she wants to start.</td>
</tr>
</tbody>
</table>

Service Response

| 3. Locates a MMCS-ET service profile in the MMCS-ET provider domain by querying a MMCS-ET provider profile with the name of the desired service (MMCS-ET). |
| 4. Informs the user that a MMCS-ET service profile has been located successfully. |
| 5. Prompts the user to log in to the MMCS-ET provider domain. |

Alternative Courses:
None

2) Use case “Log in to the MMCS-ET provider domain”:

<table>
<thead>
<tr>
<th>Use Case:</th>
<th>Log in to the MMCS-ET provider domain.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>Teacher (initiator) or student (initiator).</td>
</tr>
<tr>
<td>Purpose:</td>
<td>Describes the log in process for a teacher or a student.</td>
</tr>
<tr>
<td>Overview:</td>
<td>A user (teacher or student) logs in to the MMCS-ET provider domain by specifying an appropriate user name and password. On completion, the MMCS-ET service can correctly identify the user by a user name.</td>
</tr>
<tr>
<td>Type:</td>
<td>Primary and essential</td>
</tr>
<tr>
<td>Cross References:</td>
<td>Service Functions: SF.1.2., SF.1.3., SF.1.4., SF.1.5., SF.1.6., SF.1.7., SF.1.8., SF.1.16.</td>
</tr>
<tr>
<td>Use Cases:</td>
<td>The user involved in the current use case must have completed the use case “Contact the MMCS-ET provider domain”.</td>
</tr>
</tbody>
</table>

Typical Course of Events:

<table>
<thead>
<tr>
<th>Actor Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. This use case begins when a user (teacher or student) decides to log in, in order to be able to use the MMCS-ET service.</td>
</tr>
</tbody>
</table>

Service Response
2. The user specifies his/her user name and the corresponding password (security information).

3. Examines the length of the password specified by the user and finds it to be equal or greater than six.

4. Ciphers the password.

5. Locates the MMCS-ET service profile by querying the MMCS-ET provider profile.

6. Locates a catalog containing security information about potential users of the MMCS-ET service with the help of the MMCS-ET service profile.

7. Deciphers the password of the user just before starting to use the user security catalog.

8. Examines the validity of the user name specified by the user using the user security catalog.

9. Examines the validity of the security information specified by the user using the user security catalog.

10. Creates a user profile (teacher profile or student profile) for the user with the given user name in order to signify that the specific user is considered to be logged in.

11. Locates a list containing information about all the logged in users with the help of the MMCS-ET service profile.

12. Updates the logged in user information list with information regarding the user that just logged in.

13. Informs the MMCS-ET provider profile for the newly created user profile.

14. Customises the newly created user profile for the specific user that just logged in with the help of the MMCS-ET provider profile.

15. Prompts the user to invite another user in order to interact with him/her using the capabilities provided by the MMCS-ET service.

Alternative Courses:

- Event 3: The length of the password specified by the user is less than six (6). Indicate the error to the user.

- Event 4: The user security information catalog does not contain the user name specified by the user (i.e. the user name is not valid). Indicate the error to the user.

- Event 9: The user security information catalog does not contain the given password or the given password is contained in the user security information catalog but it does not correspond to the given user name. Indicate the error to the user.

3) Use case “Start a new MMCS-ET session”:

Use Case: Start a new MMCS-ET session.

Actors: Teacher (initiator).

Purpose: Describes the attempt of a teacher to start a new educational/training session.

Overview: A teacher (logged in user) starts a new educational/training session. On completion, an educational/training session is created and the teacher is ready to invite students (logged in users) to join the session.

Type: Primary and essential.


Use Cases: The teacher involved in the current use case must have completed the use case “Log in to the MMCS-ET provider domain”.

Typical Course of Events:

<table>
<thead>
<tr>
<th>Actor</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teacher</td>
<td>Indicates that they want to start a new educational/training session (MMCS-ET session).</td>
</tr>
</tbody>
</table>

2. The teacher specifies the name of the service (MMCS-ET) that the MMCS-ET session is part of.

3. Locates the user profile of the teacher (teacher profile) by querying the MMCS-ET provider profile.

4. Examines whether the teacher participates in a MMCS-ET session or not, by querying the MMCS-ET provider profile.

5. Examines whether the user that attempts to start a new MMCS-ET session is a teacher or not, by querying the appropriate user profile.

6. Examines the status of the teacher by querying the user profile of the teacher. The teacher can be either a logged in user or an active user (participating already in the session of another service).

7. Locates a catalog containing subscription information about all the users that are subscribers of the MMCS-ET service with the help of the user profile of the teacher.

8. Examines whether the teacher is a subscriber of the MMCS-ET service or not, using the user subscription catalog.

9. Creates a session (MMCS-ET session) using the name of the service (MMCS-ET) specified by the teacher.

10. Creates a list containing information about all the users that participate in the MMCS-ET session (active users).

11. Updates the active user information list with information regarding the teacher that started the MMCS-ET session.

12. Informs the user profile of the teacher for the newly created MMCS-ET session.

13. Changes the status of the teacher from logged in to active (as the teacher is now an active user) with the help of the user profile of the teacher.
Appendix A: Detailed Presentation of the Development of a MMCS-ET

Alternative Courses:

Event 4: The teacher participates in a MMCS-ET session. Taking into account that the teacher didn’t start the MMCS-ET session that he / she participates, that only a teacher can start a MMCS-ET session, and that only one teacher can participate in a MMCS-ET session at a given time, it is evident that the user that initiated the invitation is a student, and not a teacher. Follow the use case “Invite a student (active user) to direct communication” from event 5 onwards.

Event 5: The user that attempts to start a new MMCS-ET session is not a teacher (i.e. he / she is a student). However, only a teacher can start a new MMCS-ET session. Indicate the error to the user.

Event 6: The teacher is an active user (i.e. he / she participates already in the session of another service). However, a teacher can participate in only one session of a service at a given time. Indicate the error to the teacher.

Event 8: The teacher is not a subscriber of the MMCS-ET service. Indicate the error to the teacher.

4) Use case “Invite a student (active user) to direct communication”:

Use Case: Invite a student (active user) to direct communication.

Actors: Two students (one of them is the initiator) - the one that makes the invitation.

Purpose: Describes the way that a student participating in a MMCS-ET session with another student proposes the establishment of direct communication between them.

Overview: A student participating in a MMCS-ET session invites another student participating in the same MMCS-ET session to direct (text, audio / video, file) communication. On completion, the latter student is left to decide whether to accept this invitation by the former student.

Type: Primary and essential.


Use Cases: Each of the students involved in the current use case must have completed the use case “Join a MMCS-ET session after being invited”.

Typical Course of Events:

Actor Action

1. This use case begins when a student (e.g. student A), participating in a MMCS-ET session, decides to invite another student (e.g. student B), participating in the same MMCS-ET session, to direct communication.

2. Student A specifies the user name of the student (student B) that he / she wants to invite to direct communication.

3. Examines whether student A has started a new MMCS-ET session or not, by querying the MMCS-ET provider profile.

4. Examines whether student A participates in a MMCS-ET session or not, by querying the MMCS-ET provider profile.

5. Locates a list containing information about all the users (teacher, other students) that are in direct communication with student A.

6. Examines whether student B is already in direct communication with student A or not, using the directly communicating user information list that refers to student A.

7. Locates a list containing information about all the active users with the help of the MMCS-ET session.

8. Examines whether the user that student A wants to invite is active or not and whether he / she is a student or not, using the active user information list.

9. Examines whether student B has already a pending invitation (an invitation not answered yet) for direct communication from another student.

10. Examines whether student B participates in a MMCS-ET session or not.

11. Indicates that student B has a pending invitation for direct communication from another student.

12. Prompts student B to accept or reject the invitation of student A to engage in direct communication with him / her.

Alternative Courses:

Event 3: Student A has started a new MMCS-ET session. Taking into account that only a teacher can start a new MMCS-ET session, it is evident that the user that initiated the invitation (user A) is a teacher, and not a student. Follow the use case “Invite a student (active user) to join a MMCS-ET session” from event 4 onwards.

Event 4: Student A does not participate in a MMCS-ET session. There is a possibility that the user that initiated the invitation (user A) is a teacher, and not a student. Follow section “Start a New MMCS-ET Session” (from event 3 onwards) of the use case “Invite a student to join a MMCS-ET session”.

Event 5: The user that attempts to start a new MMCS-ET session is not a teacher (i.e. he / she is a student). However, only a teacher can start a new MMCS-ET session. Indicate the error to student A.

Event 8: The user that student A wants to invite is not active and / or he / she is a teacher. However, a student can’t invite a logged in student or a teacher to direct communication. Indicate the error to student A.
Appendix A: Detailed Presentation of the Development of a MMCS-ET

- Event 9: Student B has already a pending invitation for direct communication from another student. Therefore, the (current) invitation from student A has to be rejected as a student can have only one pending invitation for direct communication. Indicate the error to student A.
- Event 10: Student B does not participate in a MMCS-ET session although he/she is active. This can happen when the process for removing student B has started (by the teacher) but not finished yet. Indicate the error to student A.

5) Use case "Accept direct communication with a student (active user)"

**Use Case:** Accept direct communication with a student (active user).

**Actors:** Two students (one of them is the initiator - the one that replies to the invitation).

**Purpose:** Describes the way that a student participating in a MMCS-ET session with another student accepts the establishment of direct communication between them.

**Overview:** A student participating in a MMCS-ET session accepts the invitation from another student participating in the same MMCS-ET session to engage in direct (text, audio/ video, file) communication with him/her. The latter student is informed about the decision of the former student. On completion, the two students are ready to engage in direct communication with each other.

**Type:** Primary and essential.

**Cross References:** Service Functions: SF.1.17, SF.1.21.

**Use Cases:** The students involved in the current use case must have completed the use case "Invite a student (active user) to direct communication".

**Typical Course of Events:**

<table>
<thead>
<tr>
<th>Actor</th>
<th>Action</th>
<th>Service Response</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1. This use case begins when a student (e.g. student B), participating in a MMCS-ET session, that was invited to direct communication by another student (e.g. student A), participating in the same MMCS-ET session, decides to accept the invitation.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. Student B indicates his/her decision to accept the invitation from student A.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Informs the MMCS-ET provider profile for the decision of student B to accept the invitation from student A.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Examines whether student B participates in a MMCS-ET session or not, by querying the MMCS-ET provider profile.</td>
<td>Finds that student B participates in a MMCS-ET session &quot;t&quot;.</td>
</tr>
<tr>
<td></td>
<td>5. Locates a list containing information about all the active users with the help of the MMCS-ET session.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6. Examines whether student A is active or not, using the active user information list.</td>
<td>Finds that student A is active.</td>
</tr>
<tr>
<td></td>
<td>7. Locates a list containing information about all the users (teacher, other students) that are in direct communication with student B.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8. Updates the directly communicating user information list that refers to student B with information regarding student A.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9. Indicates that student B has no pending invitations for direct communication from other students.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10. Locates a list containing information about all the users (teacher, other students) that are in direct communication with student A.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11. Updates the directly communicating user information list that refers to student A with information regarding student B.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12. Informs student A that student B accepted his/her invitation and he/she is now ready to engage in direct communication with him/her.</td>
<td></td>
</tr>
</tbody>
</table>

**Alternative Courses:**
- **Event 1:** Student B decides to reject the invitation for direct communication from student A. Indicate the decision of student B to student A.
- **Event 6:** Student A is not active (i.e. he/she does not participate in the MMCS-ET session any more). During the time lapsed between the invitation from student A and the response (acceptance) from student B, the teacher removed A from the MMCS-ET session. Indicate the error to student B.

"t"The acceptance of the invitation from student A isn’t yet complete. This fact ensures that student B participates in a MMCS-ET session, as a teacher can’t remove a student from a MMCS-ET session if there are pending invitations for direct communication (initiated by other students) regarding this student. Therefore, event 4 is not necessary. However, it is included as a means of extra error checking.

6) Use case “Engage in file communication with an active user”:

**Use Case:** Engage in file communication with an active user.

**Actors:** Teacher (initiator), student or student (initiator), teacher or two students (one of them is the initiator; the one that sends the file)

**Purpose:** Describes the way that file communication is realised between two users that participate in the same MMCS-ET session.

**Overview:** A user, participating in a MMCS-ET session, sends a file to another user, participating in the same MMCS-ET session. Both users are ready to engage in direct communication with each other. On completion, the two users are engaged in file communication.

**Type:** Primary and essential.

**Cross References:** Service Functions: SF.2.8, SF.2.9, SF.2.10.

**Use Cases:** The users involved in the current use case must have completed the use case “Join a MMCS-ET session after being invited” (when one of them is a teacher and the other is a student) or the use case “Accept direct communication with a student (active user)” (when both are students).
Appendix A: Detailed Presentation of the Development of a MMCS-ET

Typical Course of Events:

Actor Action Service Response

1. This use case begins when a user (teacher or student, e.g. user A), participating in a MMCS-ET session, decides to send a file (e.g. file F) to another user (teacher or student, e.g. user B), participating in the same MMCS-ET session. Both users are ready to engage in direct communication with each other.

2. User A specifies the file F that he / she wants to send to user B.

3. Informs the MMCS-ET session that file communication is taking place between users A and B, as user A wants to send file F to user B.

4. Examines whether file communication between user A and user B is allowed or not, by querying the MMCS-ET session.

5. Transfers the file F from user A to user B, with the help of the MMCS-ET session.

6. Examines whether the transfer of file F from user A to user B is completed successfully or not, by querying the MMCS-ET session.

7. Informs user A that he / she sent file F to user B.

8. Informs user B that user A sent file F to him / her.

Alternative Courses:

• Event 4: File communication between user A and user B is not allowed. The reason for this is that direct communication between user A and user B is being terminated (see use case "Terminate direct communication between two students (active users)" and use case "Remove a student from a MMCS-ET session"). Indicate the error to user A.

• Event 6: The transfer of file F from user A to user B is not completed successfully. One possible reason is that user B is a student who has been removed from the MMCS-ET session by the teacher. In this case, indicate the error to user A (the teacher).

7) Use case “Engage in A/V communication with a student (active user)”: Use Case: Engage in A/V communication with a student (active user).

Actors: Teacher (initiator), student or two students (one of them is the initiator).

Purpose: Describes the way that A/V communication is realised between two users that participate in the same MMCS-ET session.

Overview: A user participating in a MMCS-ET session initiates the establishment of two A/V stream connections with another user participating in the same MMCS-ET session. Both users are ready to engage in direct communication with each other. On completion, the two users are engaged in A/V communication with each other.

Type: Primary and essential.

Cross References: Service Functions: SF.1.17., SF.2.4.

Use Cases: The users involved in the current use case must have completed the use case “Join a MMCS-ET session after being invited” (when one of them is a teacher and the other is a student) or the use case “Accept direct communication with a student (active user)” (when both are students).

Typical Course of Events:

Actor Action Service Response

1. This use case begins when a user (teacher or student, e.g. user A), participating in a MMCS-ET session, is ready to engage in direct communication with another user (a student, e.g. user B), participating in the same MMCS-ET session.

2. Informs the MMCS-ET session that A/V communication is being established between users A and B, as user A initiated the related process.

3. Prepares for the binding of the A/V streams that will be created between user A and user B, with the help of the MMCS-ET session.

4. Starts the sink A/V devices that are associated with user A and user B.

5. Starts the source A/V devices that are associated with user A and user B.

6. Establishes two A/V stream connections between the appropriate source and sink A/V devices of user A and user B.

7. Makes evident to user A that he / she is engaged in A/V communication with user B.

8. Makes evident to user B that he / she is engaged in A/V communication with user A.

9. Locates a list containing information about all the active users with the help of the MMCS-ET session.

10. Examines whether user A (who initiated the A/V communication establishment) is a teacher or not, using the active user information list.

11. If user A is a teacher, indicates to user A that he / she is allowed to stop the existing A/V communication between users A and B, and indicates to user B that he / she is not allowed to stop the existing A/V communication between users A and B.

12. If user A is a student, indicates to both users A and B that they are allowed to stop the existing A/V communication between users A and B.

Alternative Courses:
None.
8) Use case “Stop A/V communication with a student (active user)”: 

**Use Case:** Stop A/V communication with a student (active user).

**Actors:** Teacher (initiator), student or two students (one of them is the initiator).

**Purpose:** Describes the way that an existing A/V communication between two users that participate in the same MMCS-ET session is stopped by one of them.

**Overview:** A user participating in a MMCS-ET session stops to communicate using audio and video with another user participating in the same MMCS-ET session. On completion, the A/V communication between the two users is blocked.

**Type:** Primary and essential.

**Cross References:** Service Functions: SF.1.17, SF.2.5.

**Use Cases:** The users involved in the current use case must have completed the use case “Engage in A/V communication with a student (active user)”. 

**Typical Course of Events:**

<table>
<thead>
<tr>
<th>Actor Action</th>
<th>Service Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>User A indicates his / her decision to stop an existing A/V communication between him / her and user B.</td>
<td>1. Informs the MMCS-ET session that the existing A/V communication between users A and B is being stopped from user A.</td>
</tr>
<tr>
<td>2.</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>3. Stops the sink A/V devices that are associated with user A and user B.</td>
</tr>
<tr>
<td>4.</td>
<td>4. Stops the sink A/V devices that are associated with user A and user B.</td>
</tr>
<tr>
<td>5.</td>
<td>5. Stops the source A/V devices that are associated with user A and user B.</td>
</tr>
<tr>
<td>6.</td>
<td>6. Makes evident to user A that the A/V communication with user B is stopped.</td>
</tr>
<tr>
<td>7.</td>
<td>7. Makes evident to user B that the A/V communication with user A is stopped.</td>
</tr>
<tr>
<td>8.</td>
<td>8. Locates a list containing information about all the active users with the help of the MMCS-ET session.</td>
</tr>
<tr>
<td>9.</td>
<td>9. Examines whether user A (who stopped the A/V communication between users A and B) is a teacher or not, using the active user information list.</td>
</tr>
<tr>
<td>If user A is a teacher, indicates to user A that he / she is allowed to start the blocked A/V communication between users A and B, and indicates to user B that he / she is not allowed to start the blocked A/V communication between users A and B.</td>
<td></td>
</tr>
<tr>
<td>If user A is a student, indicates to both users A and B that they are allowed to start the blocked A/V communication between users A and B.</td>
<td></td>
</tr>
</tbody>
</table>

**Alternative Courses:** None.

9) Use case “Start A/V communication with a student (active user)”: 

**Use Case:** Start A/V communication with a student (active user).

**Actors:** Teacher (initiator), student or two students (one of them is the initiator).

**Purpose:** Describes the way that a blocked A/V communication between two users that participate in the same MMCS-ET session is started by one of them.

**Overview:** A user participating in a MMCS-ET session starts a blocked A/V communication between him / her and another user participating in the same MMCS-ET session. On completion, the two users are engaged in A/V communication with each other.

**Type:** Primary and essential.

**Cross References:** Service Functions: SF.1.17, SF.2.6.

**Use Cases:** The users involved in the current use case must have completed the use case “Stop A/V communication with a student (active user)”.

**Typical Course of Events:**

<table>
<thead>
<tr>
<th>Actor Action</th>
<th>Service Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>User A indicates his / her decision to start a blocked A/V communication between him / her and user B.</td>
<td>1. This use case begins when a user (teacher or student, e.g. user A), participating in a MMCS-ET session, decides to start a blocked A/V communication between him / her and another user (a student, e.g. user B), participating in the same MMCS-ET session.</td>
</tr>
<tr>
<td>2.</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>3. Informs the MMCS-ET session that the blocked A/V communication between users A and B is being started from user A.</td>
</tr>
<tr>
<td>4.</td>
<td>4. Starts the sink A/V devices that are associated with user A and user B.</td>
</tr>
<tr>
<td>5.</td>
<td>5. Starts the source A/V devices that are associated with user A and user B.</td>
</tr>
<tr>
<td>6.</td>
<td>6. Makes evident to user A that the blocked A/V communication with user B is started.</td>
</tr>
</tbody>
</table>

| Alternative Courses: None. | |

A-11
Appendix A: Detailed Presentation of the Development of a MMCS-ET

7. Makes evident to user B that the blocked A/V communication with user A is started.
8. Locates a list containing information about all the active users with the help of the MMCS-ET session.
9. Examines whether user A (who started the blocked A/V communication between users A and B) is a teacher or not, using the active user information list.
   - If user A is a teacher, indicates to user A that he / she is allowed to stop the existing A/V communication between users A and B, and indicates to user B that he / she is not allowed to stop the existing A/V communication between users A and B.
   - If user A is a student, indicates to both users A and B that they are allowed to stop the existing A/V communication between users A and B.

Alternative Courses:
None.

10) Use case “Terminate A/V communication with a student (active user)”:  

Use Case: Terminate A/V communication with a student (active user).
Actors: Teacher (initiator), student or two students (one of them is the initiator).
Purpose: Describes the way that an established A/V communication between two users that participate in the same MMCS-ET session is terminated by one of them.
Overview: A user participating in a MMCS-ET session terminates an established A/V communication between him / her and another user participating in the same MMCS-ET session. On completion, the two users can't communicate using audio and video.
Type: Primary and essential.
Cross References: Service Functions: SF.2.7.
Use Cases: The users involved in the current use case must have completed the use case “Engage in A/V communication with a student (active user)”.

Typical Course of Events:

<table>
<thead>
<tr>
<th>Actor Action</th>
<th>Service Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. This use case begins when a user (teacher or student, e.g. user A), participating in a MMCS-ET session, is terminating an established A/V communication between him / her and another user (a student, e.g. user B), participating in the same MMCS-ET session.</td>
<td>2. Informs the MMCS-ET session that the established A/V communication between users A and B is being terminated from user A.</td>
</tr>
<tr>
<td>3. Destroys the two A/V stream connections that are established between the appropriate source and sink A/V devices of user A and user B.</td>
<td>4. Makes evident to user A that the A/V communication with user B is terminated.</td>
</tr>
<tr>
<td>5. Makes evident to user B that the A/V communication with user A is terminated.</td>
<td></td>
</tr>
</tbody>
</table>

Alternative Courses:
None.

11) Use case “Engage in a chat with all active users”:

Use Case: Engage in a chat with all active users.
Actors: All active users (teacher, students) - one of them is the initiator (the one that sends the message which is broadcasted).
Purpose: Describes the way that chat is realised between all the users that participate in the same MMCS-ET session.
Overview: A user, participating in a MMCS-ET session, sends a text message to all the other users, participating in the same MMCS-ET session. On completion, all these users are engaged in a chat.
Type: Primary and essential.
Use Cases: The users involved in the current use case must have completed the use case “Join a MMCS-ET session after being invited” (in pairs consisting of the teacher and one of the students each time).

Typical Course of Events:

<table>
<thead>
<tr>
<th>Actor Action</th>
<th>Service Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. This use case begins when a user (teacher or student, e.g. user A) participating in a MMCS-ET session, decides to send a text message (e.g. message M) to all the other users, participating in the same MMCS-ET session.</td>
<td>2. User A specifies the message M that he / she wants to send to all the other users.</td>
</tr>
<tr>
<td>3. Informs the MMCS-ET session that user A wants to send message M to all the other users.</td>
<td>4. Examines whether message M indicates the start of a voting process or not, with the help of the MMCS-ET session.</td>
</tr>
<tr>
<td>If message M indicates the start of a voting process, initiate use case “Start a voting process between all active users”.</td>
<td>If message M does not indicate the start of a voting process, continues.</td>
</tr>
</tbody>
</table>

A-12
Appendix A: Detailed Presentation of the Development of a MMCS-ET

5. Examines whether message M indicates a vote from a user or not, with the help of the MMCS-ET session.
   If message M indicates a vote from a user, initiate use case "Vote in a voting process between all active users".
   If message M does not indicate a vote from a user, continues.

6. Examines the voting activity in the MMCS-ET session by querying the MMCS-ET session.
   Finds that there is no voting activity in the MMCS-ET session.

7. Locates a list containing information about all the active users with the help of the MMCS-ET session.

8. Examines whether user A is active or not, using the active user information list.
   Finds that user A is active.

9. Examines all the users included in the active user information list.
   For each user in this list (e.g. user X) performs events 10 to 13.

10. Sends the message M from user A to user X, with the help of the MMCS-ET session.

11. Locates a list containing information about all the messages broadcasted recently to all the users that participate in the MMCS-ET session. This list refers to user X.

12. Updates the broadcasted message information list that refers to user X with information regarding the message M that user A sent to all the other users.

13. Informs user X about all the messages broadcasted recently to all the users that participate in the MMCS-ET session, using the broadcasted message information list that refers to user X.

Alternative Courses:

* Event 6: There is voting activity in the MMCS-ET session (i.e. a voting process has started). Therefore, chat is not allowed as users have to send their vote. Indicate the error to user A.

* Event 8: User A is not active, because he / she has been removed by the teacher. Therefore, he / she is not allowed to send message M to all the other users.

1 The active user information list includes also information about user A. Therefore, user X can also be user A.

12) Use case "Engage in file communication with all students (active users)"

Use Case: Engage in file communication with all students (active users).

Actors: Teacher (initiator), all active students.

Purpose: Describes the way that file communication is realised between a teacher and all the students that participate in the MMCS-ET session that he / she created.

Overview: A teacher, participating in a MMCS-ET session, sends a file to all the students, participating in the same MMCS-ET session.

Typical Course of Events:

<table>
<thead>
<tr>
<th>Actor Action</th>
<th>Service Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The teacher specifies the file F that he / she wants to send to all the students.</td>
<td>3. Informs the MMCS-ET session that the teacher wants to send file F to all the students.</td>
</tr>
<tr>
<td>2. Transfers the file F from the teacher to user X with the help of the MMCS-ET session.</td>
<td>4. Locates a list containing information about all the active users with the help of the MMCS-ET session.</td>
</tr>
<tr>
<td>5. Examines all the students included in the active user information list. For each student in this list (e.g. student X) performs events 6 to 8.</td>
<td>6. Transfers the file F from the teacher to user X, with the help of the MMCS-ET session.</td>
</tr>
<tr>
<td>7. Examines whether the transfer of file F from the teacher to student X is completed successfully or not, by querying the MMCS-ET session.</td>
<td>7. Finds that the transfer of file F from the teacher to student X is completed successfully.</td>
</tr>
<tr>
<td>8. Informs user X that the teacher sent file F to him / her.</td>
<td>9. Informs the teacher that he / she sent file F to all the students.</td>
</tr>
<tr>
<td>9. Informs the teacher that he / she sent file F to all the students.</td>
<td></td>
</tr>
</tbody>
</table>

Alternative Courses:

* Event 7: The transfer of file F from the teacher to student X is not completed successfully. One possible reason is that student X has been removed from the MMCS-ET session by the teacher. In this case, indicate the error to the teacher.

13) Use case "Start a voting process between all active users"

Use Case: Start a voting process between all active users.

Actors: Teacher (initiator), all active users.

Purpose: Describes the way that a teacher can start a voting process between all the users that participate in the same MMCS-ET session.

Overview: A teacher initiates a voting process between all the users that participate in the MMCS-ET session that he / she created. On completion, all users are asked for their vote on a specific matter.

Type: Primary and essential.
Appendix A: Detailed Presentation of the Development of a MMCS-ET

Cross References: Service Functions: SF.1.17., SF.3.5.
Use Cases: The users involved in the current use case must have completed the use case “Join a MMCS-ET session after being invited” (in pairs consisting of the teacher and one of the students each time).

Typical Course of Events:

<table>
<thead>
<tr>
<th>Actor Action</th>
<th>Service Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. This use case begins when a teacher, participating in a MMCS-ET session, decides to start a voting process between all the users, participating in the MMCS-ET session that he/she created.</td>
<td>3. Informs the MMCS-ET session that the teacher wants to start a voting process between all active users.</td>
</tr>
<tr>
<td>2. The teacher indicates that he/she wants to start a voting process between all active users, and he/she specifies in a message (e.g. message M) the subject of the voting process.</td>
<td>4. Examines whether message M is a simple message unrelated with the voting process or not, with the help of the MMCS-ET session. If message M is a simple message unrelated with the voting process, initiate use case “Engage in a chat with all active users”. If message M is related with the voting process, continues.</td>
</tr>
<tr>
<td>2. User A specifies this voting decision in a vote which has the form of a message (e.g. message M).</td>
<td>5. Examines whether message M indicates a vote from a user or not, with the help of the MMCS-ET session. If message M indicates a vote from a user, initiate use case “Vote in a voting process between all active users”. If message M does not indicate a vote from a user, continues.</td>
</tr>
<tr>
<td>2. User A specifies this voting decision in a vote which has the form of a message (e.g. message M).</td>
<td>6. Examines the voting activity in the MMCS-ET session by querying the MMCS-ET session. FInds that there is no voting activity in the MMCS-ET session.</td>
</tr>
<tr>
<td>2. User A specifies this voting decision in a vote which has the form of a message (e.g. message M).</td>
<td>7. Locates a list containing information about all the active users with the help of the MMCS-ET session.</td>
</tr>
<tr>
<td>2. User A specifies this voting decision in a vote which has the form of a message (e.g. message M).</td>
<td>8. Examines whether the teacher is truly a teacher or not, using the active user information list. Finds that the teacher is truly a teacher.</td>
</tr>
<tr>
<td>2. User A specifies this voting decision in a vote which has the form of a message (e.g. message M).</td>
<td>9. Indicates that there is voting activity in the MMCS-ET session, with the help of the MMCS-ET session.</td>
</tr>
<tr>
<td>2. User A specifies this voting decision in a vote which has the form of a message (e.g. message M).</td>
<td>10. Examines all the users included in the active user information list. For each user in this list (e.g. user X) performs events 11 and 12.</td>
</tr>
<tr>
<td>2. User A specifies this voting decision in a vote which has the form of a message (e.g. message M).</td>
<td>11. Sends the message M from the teacher to user X, with the help of the MMCS-ET session.</td>
</tr>
<tr>
<td>2. User A specifies this voting decision in a vote which has the form of a message (e.g. message M).</td>
<td>12. Informs user X about message M and the voting process that the teacher started.</td>
</tr>
</tbody>
</table>

Alternative Courses:

- Event 6: There is voting activity in the MMCS-ET session (i.e. a voting process has already started). Therefore, the teacher attempts to start a second voting process without waiting for the outcome of the first one. However, this is not allowed. Indicate the error to the teacher.
- Event 8: The teacher is not truly a teacher (i.e. a student attempted to start a voting process). However, only a teacher is allowed to start a voting process between all active users. Indicate the error to the student.

The active user information list includes also information about the teacher. Therefore, user X can also be the teacher.

14) Use case “Vote in a voting process between all active users”:

Use Case: Vote in a voting process between all active users.
Actor:
Teacher (initiator) or student (initiator).
Purpose: Describes the way that a user (teacher or student) participating in a voting process expresses his/her voting decision.
Overview: A user (teacher or student) participating in a voting process started by the teacher, expresses his/her voting decision by sending a vote. On completion, the user is informed that his/her vote has been received.
Type: Primary and essential.
Use Cases: The user involved in the current use case must have completed the use case “Start a voting process between all active users” together with the teacher.

Typical Course of Events:

<table>
<thead>
<tr>
<th>Actor Action</th>
<th>Service Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. This use case begins when a user (teacher or student, e.g. user A), participating in a voting process started by the teacher, decides to express his/her voting decision.</td>
<td>3. Informs the MMCS-ET session that user A specified his/her voting decision in a vote which has the form of message M.</td>
</tr>
<tr>
<td>2. User A specifies this voting decision in a vote which has the form of a message (e.g. message M).</td>
<td>4. Examines whether message M is a simple message unrelated with the voting process or not, with the help of the MMCS-ET session. If message M is a simple message unrelated with the voting process, initiate use case “Engage in a chat with all active users”. If message M is related with the voting process, continues.</td>
</tr>
</tbody>
</table>
Appendix A: Detailed Presentation of the Development of a MMCS-ET

5. Examines whether message M indicates the start of a voting process or not, with the help of the MMCS-ET session.
   If message M indicates the start of a voting process, initiate use case “Start a voting process between all active users.”
   If message M does not indicate the start of a voting process, continues.
6. Examines the voting activity in the MMCS-ET session by querying the MMCS-ET session.
   Finds that there is voting activity in the MMCS-ET session.
7. Extracts the voting decision of user A from his/her vote by examining message M, with the help of the MMCS-ET session.
8. Informs user A that his/her vote is received.

Alternative Courses:
* Event 6: There is no voting activity in the MMCS-ET session (i.e. a voting process has not yet been started). Indicate the error to user A.

15) Use Case “Present the outcome of a voting process to all the involved active users”:

Use Case: Present the outcome of a voting process to all the involved active users.
Actors: Teacher (initiator), all active users.
Purpose: Describes the way that all active users participating in a voting process learn about its outcome.
Overview: The teacher requires the calculation of the outcome of the voting process that he/she started and the presentation of this outcome to all the involved active users. On completion, the voting process is considered to be finished.
Type: Primary and essential.
Use Cases: The users involved in the current use case must have completed (each of them) the use case “Vote in a voting process between all active users”.

Typical Course of Events:

<table>
<thead>
<tr>
<th>Actor</th>
<th>Action</th>
<th>Service Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teacher</td>
<td>This use case begins when a teacher indicates that he/she wants the presentation of the outcome of a voting process.</td>
<td></td>
</tr>
<tr>
<td>Teacher</td>
<td>Informs the MMCS-ET session that the teacher wants the presentation of the outcome of the voting process.</td>
<td></td>
</tr>
<tr>
<td>All active users</td>
<td>Locates a list containing information about all the active users with the help of the MMCS-ET session.</td>
<td></td>
</tr>
<tr>
<td>All active users</td>
<td>Examines all the users included in the active user information list.</td>
<td></td>
</tr>
<tr>
<td>Teacher</td>
<td>Calculates the outcome of the voting process started by the teacher, with the help of the MMCS-ET session. The outcome has the form of a message (e.g. message M).</td>
<td></td>
</tr>
<tr>
<td>Teacher</td>
<td>Examines all the users included in the active user information list.</td>
<td></td>
</tr>
<tr>
<td>Teacher</td>
<td>Sends the message M to user X, with the help of the MMCS-ET session.</td>
<td></td>
</tr>
<tr>
<td>Teacher</td>
<td>Informs user X about the outcome of the voting process that the teacher started, with the help of message M.</td>
<td></td>
</tr>
<tr>
<td>Teacher</td>
<td>Informs user X that the voting process has finished.</td>
<td></td>
</tr>
<tr>
<td>Teacher</td>
<td>Indicates that there is no voting activity in the MMCS-ET session, with the help of the MMCS-ET session.</td>
<td></td>
</tr>
</tbody>
</table>

Alternative Courses:
None.

1) The active user information list includes also information about the teacher. Therefore, user X can also be the teacher.

16) Use Case “Terminate direct communication between two students (active users)”:

Use Case: Terminate direct communication between two students (active users).
Actors: Two students (one of them is the initiator - the one that requests the termination).
Purpose: Describes the termination of direct communication between two students participating in the same MMCS-ET session, after a request of one of them.
Overview: A student participating in a MMCS-ET session requests the termination of his/her direct (text, audio/video, file) communication with another student participating in the same MMCS-ET session. On completion, the two students are not engaged any more in direct communication with each other.
Type: Primary and essential.
Cross References: Service Functions: SF.1.23.
Use Cases: The students involved in the current use case must have completed the use case “Accept direct communication with a student (active user)”.

Typical Course of Events:

<table>
<thead>
<tr>
<th>Actor</th>
<th>Action</th>
<th>Service Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student A</td>
<td>This use case begins when a student (e.g. student A), participating in a MMCS-ET session, decides to terminate his/her direct communication with another student (e.g. student B), participating in the same MMCS-ET session.</td>
<td></td>
</tr>
</tbody>
</table>

A-15
Appendix A: Detailed Presentation of the Development of a MMCS-ET

2. Student A indicates his / her decision to terminate direct communication with student B.

3. Informs the MMCS-ET session that student A wants to terminate his / her direct communication with student B.

4. Informs student B that his / her direct communication with student A is being terminated.

5. Prevents text communication between student A and student B, with the help of the MMCS-ET session.

6. Prevents file communication between student A and student B, with the help of the MMCS-ET session.

7. Terminates the established A/V communication between student A and student B with the help of the MMCS-ET session.

8. Locates a list containing information about all the users (teacher, other students) that are in direct communication with student A.

9. Removes from the directly communicating user information list that refers to student B, the information regarding student A.

10. Makes evident to student B that his / her direct communication with student A has been terminated.

11. Locates a list containing information about all the users (teacher, other students) that are in direct communication with student A.

12. Removes from the directly communicating user information list that refers to student A, the information regarding student B.

13. Makes evident to student A that his / her direct communication with student B has been terminated.

14. Permits text communication between student A and student B, with the help of the MMCS-ET session.

15. Permits file communication between student A and student B, with the help of the MMCS-ET session.

Alternative Courses:
None.

17) Use case "Remove a student from a MMCS-ET session":

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Remove a student from a MMCS-ET session.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors</td>
<td>Teacher (initiator), student.</td>
</tr>
<tr>
<td>Purpose</td>
<td>Describes the way that a teacher stops a student from participating in a MMCS-ET session.</td>
</tr>
<tr>
<td>Overview</td>
<td>A teacher removes, from the MMCS-ET session that he / she created, a student who is participating in this MMCS-ET session. On completion, the student is not engaged in direct (text, audio / video, file) communication with the teacher, and he / she is not a participant of the MMCS-ET session any more.</td>
</tr>
<tr>
<td>Type</td>
<td>Primary and essential.</td>
</tr>
<tr>
<td>Use Cases</td>
<td>The users involved in the current use case must have completed the use case &quot;Join an existing MMCS-ET session after being invited&quot;.</td>
</tr>
</tbody>
</table>

Typical Course of Events:

1. This use case begins when a teacher decides to remove, from the MMCS-ET session that he / she created, a student (e.g. student A) who is participating in this MMCS-ET session.

2. The teacher indicates his / her decision to remove student A from the MMCS-ET session.

3. Informs the MMCS-ET session that the teacher wants to remove student A from the MMCS-ET session.

4. Examines the voting activity in the MMCS-ET session by querying the MMCS-ET session. Finds that there is no voting activity in the MMCS-ET session.

5. Examines whether student A has a pending invitation (an invitation not answered yet) for direct communication from another student. Finds that student A has no pending invitations for direct communication from other students.

6. Indicates that student A does not participate in a MMCS-ET session.

7. Informs student A that he / she is being removed from the MMCS-ET session by the teacher.

8. Locates a list containing information about all the active users with the help of the MMCS-ET session.

9. Removes from the active user information list, the information regarding student A.

10. Prevents text communication between the teacher and student A, with the help of the MMCS-ET session.

11. Prevents file communication between the teacher and student A, with the help of the MMCS-ET session.

12. Terminates the established A/V communication between the teacher and student A with the help of the MMCS-ET session (see / initiate use case "Terminate A/V communication with a student (active user)". |

13. Locates a list containing information about all the users (teacher, other students) that are in direct communication with student A.

14. Examines all the users included in the directly communicating user information list that refers to student A: If a user is a student, initiate use case "Terminate direct communication between two students (active users)". If a user is a teacher, removes from the directly communicating user information list that refers to student A, the information regarding the teacher.
15. Makes evident to student A that he / she is removed from the MMCS-ET session by the teacher.

16. Locates the user profile of student A by querying the MMCS-ET provider profile.

17. Changes the status of student A from active to logged in with the help of the user profile of student A in order to signify that student A does not participate in a MMCS-ET session any more.

18. Locates a list containing information about all the students that are in direct communication with the teacher.

19. Removes from the directly communicating user information list that refers to the teacher, the information regarding student A.

20. Makes evident to the teacher that student A is removed from the MMCS-ET session.

21. Permits text communication between the teacher and student A, using the MMCS-ET session.

22. Permits file communication between the teacher and student A, using the MMCS-ET session.

Alternative Courses:

• Event 4: There is voting activity in the MMCS-ET session, because either voting is not finished or users are not informed of the voting results. However, the teacher is not allowed to remove a student from a MMCS-ET session when there is voting activity in this MMCS-ET session. Indicate the error to the teacher.

• Event 5: Student A has a pending invitation for direct communication from another student. However, a teacher is not allowed to remove a student from a MMCS-ET session if there are pending invitations for direct communication (initiated by other students) regarding this student. Indicate the error to the teacher.

18) Use case "Terminate a MMCS-ET session":

Use Case: Terminate a MMCS-ET session.

Actors: Teacher (initiator).

Purpose: Describes the termination of an educational / training session by a teacher.

Overview: A teacher terminates the MMCS-ET session that he / she created. On completion, all the students that were participants of the MMCS-ET session are removed and the teacher is ready to start a new MMCS-ET session.

Type: Primary and essential.


Use Cases: The teacher involved in the current use case must have completed the use case "Start a new MMCS-ET session".

Typical Course of Events:

1. This use case begins when a teacher decides to terminate the session that he / she created (current MMCS-ET session).

2. The teacher indicates his / her decision to terminate the current MMCS-ET session.

3. Locates a list containing information about all the students that are in direct communication with the teacher.

4. Examines all the students included in the directly communicating user information list that refers to the teacher:

   For each student, initiate use case "Remove a student from a MMCS-ET session".

5. Destroys the MMCS-ET session that was created by the teacher (current MMCS-ET session).

6. Indicates that the teacher does not participate in a MMCS-ET session.

7. Indicates that the teacher has not started a new MMCS-ET session.

8. Locates the user profile of the teacher by querying the MMCS-ET provider profile.

9. Changes the status of the teacher from active to logged in with the help of the user profile of the teacher in order to signify that the teacher is ready to start a new MMCS-ET session.

10. Informs the teacher that the current MMCS-ET session is being terminated.

11. Makes evident to the teacher that the current MMCS-ET session is terminated.

Alternative Courses:

None.

19) Use case "Terminate the MMCS-ET service":

Use Case: Terminate the MMCS-ET service.

Actors: Teacher (initiator) or student (initiator).

Purpose: Describes the way that a user (teacher or student) can terminate the MMCS-ET service.

Overview: A user (teacher or student) terminates the MMCS-ET service that he / she wanted to use when he / she logged in to the MMCS-ET provider domain. On completion, the user is ready to start again the MMCS-ET service.

Type: Primary and essential.

Cross References: Service Functions: SF.1.16, SF.1.25.

Use Cases: The user involved in the current use case must have completed the use case "Log in to the MMCS-ET provider domain".

Typical Course of Events:

1. This use case begins when a user (teacher or student) decides to terminate the MMCS-ET service that he / she started.

   Service Response

   The user involved in the current use case must have completed the use case "Log in to the MMCS-ET provider domain".
2. The user (e.g. user A) indicates his / her decision to terminate the MMCS-ET service.

3. Locates the MMCS-ET service profile by querying the MMCS-ET provider profile.

4. Locates a list containing information about all the logged in users with the help of the MMCS-ET service profile.

5. Removes from the logged in user information list, the information regarding user A.

6. Destroys the user profile of user A.

7. Informs the MMCS-ET provider profile that the MMCS-ET service is being terminated.

8. Informs user A that the MMCS-ET service is being terminated.

9. Makes evident to user A that the MMCS-ET service is terminated.

Alternative Courses:
None.

The use case diagram of the MMCS-ET (activity 2) can be seen in Figure 5.6. and the main service conceptual model of the MMCS-ET (step 4, activity 3) in Figure 5.7. The service concepts of the main service conceptual model of the MMCS-ET, together with their attributes, are depicted in Figure A.1. The ancillary service conceptual models of the MMCS-ET (step 5, activity 3) can be seen in Figure 5.8.

Figure A.1: The main service concepts of the MMCS-ET and their attributes.

The service sequence diagrams (activity 4) that are created for the typical course of events of each one of the MMCS-ET use cases (except from the service sequence diagrams for the target use cases which are depicted in Figure 5.9.), can be seen in Figure A.2.
The service operation contracts (activity 5) of the service operations suggested by the MMCS-ET use cases are the following (except from those corresponding to the target use cases, see Section 5.2.1.3.):

1) For service operation "StartService":

**Name:** StartService(ServiceName:String)

**Responsibilities:** Contact the MMCS-ET provider domain and prepare the MMCS-ET service for use.

**Type:** MMCS-ET.

**Cross References:** Service functions: SF.I.I.

**Use Case:** "Contact the MMCS-ET provider domain".

**Notes:**

- **Exceptions:**
- **Output:**

**Pre-conditions:**

- A user (teacher or student) decided to start a MMCS-ET service.
- The user specified the name of the service (MMCS-ET) that he/she wanted to start.

**Post-conditions:**

* A User (corresponding either to a teacher or a student) and a MMCS-ETProviderProfile have been created (instance creation).
Appendix A: Detailed Presentation of the Development of a MMCS-ET

- The User was associated with the MMCS-ETSProfileProfile (association formed).
- A MMCS-ETSProfileProfile was already created (instance creation).
- The MMCS-ETSProfileProfile was associated with the MMCS-ETSProfileProfile (association formed).
- Interaction with the user was prepared (user interface activation).

2) For service operation “Logln”:

Name: Logln(UserName: String, Password: String)
Responsibilities: Log in to the MMCS-ET provider domain by specifying an appropriate user name and password.
Type: MMCS-ET.
Cross References: Service functions: SF.1.2., SF.1.3., SF.1.4., SF.1.5., SF.1.6., SF.1.7., SF.1.8.
Notes: — Use a catalog containing security information about potential users of the MMCS-ET service.
— Use a list containing information about all the logged in users.
Exceptions: — If the user security information catalog does not contain the user name specified by the user, indicate that it was an error.
— If the user security information catalog does not contain the given password or the given password is contained in the user security information catalog but does not correspond to the given name, indicate that it was an error.
Output: — A user (teacher or student) started a MMCS-ET service.
— The user decided to log in, in order to be able to use the MMCS-ET service.
— The user specified his/her user name and the corresponding password (security information).

Post-conditions:
— The User (corresponding either to a teacher or a student) was associated with the MMCS-ETSProfileProfile (association formed).
— The MMCS-ETSProfileProfile was associated with the MMCS-ETSProfileProfile (association formed).
— A UserSecurityCatalog was created (instance creation).
— The MMCS-ETSProfileProfile was associated with the UserSecurityCatalog (association formed).
— The UserSecurityCatalog was associated with UserSecurityInformation, based on the satisfaction of the following conditions (association formed):
  — The user name specified by the user was already in the UserSecurityCatalog.
  — The password specified by the user was already in the UserSecurityCatalog and corresponded to the user name that was also specified by the user.
— A UserProfile (corresponding either to the teacher or the student) was created (instance creation).
— The MMCS-ETSProfileProfile was associated with the UserProfile (association creation).
— The MMCS-ETSProfileProfile was associated with LoggedinUserInformation regarding the user that just logged in (association formed).
— Interaction with the user was prepared (user interface activation).

3) For service operation “StartSession”:

Name: StartSession()
Responsibilities: Start a new educational/training session (MMCS-ET session).
Type: MMCS-ET.
Notes: — Use a catalog containing subscription information about all the users that are subscribers of the MMCS-ET service.
— Use a list containing information about all the users that participate in a MMCS-ET session (active users).
Exceptions: — If the user that attempts to start a new MMCS-ET session is not a teacher, indicate that it was an error.
— If the teacher that attempts to start a new MMCS-ET session is already an active user, indicate that it was an error.
— If this teacher is not a subscriber of the MMCS-ET service, indicate that it was an error.
Output: — A teacher logged in to the MMCS-ET provider domain.
— The teacher indicated that he/she wanted to start a new MMCS-ET session.

Post-conditions:
— The User (corresponding to the teacher) was associated with the MMCS-ETSProfileProfile, based on the confirmation that UserSessionParticipation was already set to false (association formed).
— The MMCS-ETSProfileProfile was associated with the UserProfile (corresponding to the teacher) (association formed).
— A UserSubscriptionCatalog was created (instance creation).
— The UserProfile was associated with UserSubscriptionCatalog, based on the satisfaction of the following conditions (association formed):
  — UserProfile.Role was already set to teacher.
  — UserProfile.Status was already set to logged in.
— The UserSubscriptionCatalog was associated with UserSubscriptionInformation, based on the confirmation that the teacher was (at that time) a subscriber of the MMCS-ET service (association formed).
— A MMCS-ETSession was created (instance creation).
— The UserProfile was associated with the MMCS-ETSession (association formed).
— MMCS-ETSessionNetworkingActivity was set to false (attribute modification).
— The MMCS-ETSession was associated with ActiveUserInformation regarding the teacher that just started the MMCS-ET session (association formed).
— TextCommunication was created (instance creation).
— The MMCS-ETSession was associated with TextCommunication (association formed).
— TextCommunication.PermitTextCommunication was set to true (attribute modification).
— TextCommunication.PreventUser1 was set to null (attribute modification).
— TextCommunication.PreventUser2 was set to null (attribute modification).
— FileCommunication was created (instance creation).
— The MMCS-ETSession was associated with FileCommunication (association formed).

A-20
Appendix A: Detailed Presentation of the Development of a MMCS-ET

FileCommunication.PermiFileCommunication was set to true (attribute modification).
FileCommunication.PreventUser1 was set to null (attribute modification).
FileCommunication.PreventUser2 was set to null (attribute modification).
AudioVideoCommunication was created (instance creation).
The MMCS-ETSession was associated with AudioVideoCommunication (association formed).
A StreamBinder was created (instance creation).
AudioVideoCommunication was associated with the StreamBinder (association formed).
A ChatFacility was created (instance creation).
Voting was created (instance creation).
The MMCS-ETSession was associated with Voting (association formed).
Voting.Total was set to 0 (attribute modification).
Voting.CounterYes was set to 0 (attribute modification).
Voting.CounterNo was set to 0 (attribute modification).
UserProfile.Status was set to active (attribute modification).
The User was associated with the MMCS-ETSession (association formed).
User.SessionCreation was set to true (attribute modification).
User.SessionParticipation was set to true (attribute modification).

4) For service operation "InviteDCSStudent":

Name: InviteDCSStudent(DestStName: String)
Responsibilities: Invite another student participating in the same MMCS-ET session to direct (text, audio / video, and file) communication.
Type: MMCS-ET.
Use Case: "Invite a student (active user) to direct communication".
Notes: - Use a list containing information about all the users that are in direct communication with a specific user.
- Use a list containing information about all the users that participate in a MMCS-ET session (active users).
Exceptions: - If student B is already in direct communication with student A, indicate that it was an error.
- If the user that student A wants to invite is not active and / or he / she is a teacher, indicate that it was an error.
- If student B has already a pending invitation for direct communication from another student, indicate that it was an error.
- If student B does not participate in a MMCS-ET session although he / she is active, indicate that it was an error.
Output:
Pre-conditions: - One student (e.g. student A) joined a MMCS-ET session after being invited by a teacher.
- Another student (e.g. student B) joined the same MMCS-ET session after being invited by the same teacher.
- Student A decided to invite student B to direct communication.
- Student A specified the user name of student B.
Post-conditions: - The User (corresponding to student A) was associated with the MMCS-ETProviderProfile (association formed).
- This User was associated with the MMCS-ETSession, based on the satisfaction of the following conditions (association formed):
  - User.SessionCreation was already set to false.
  - User.SessionParticipation was already set to true.
  - Student B is not in direct communication with student A.
- The MMCS-ETSession was associated with ActiveUserInformation, based on the satisfaction of the following conditions (association formed):
  - The user that student A wanted to invite was active.
  - The user that student A wanted to invite was a student.
- The MMCS-ETSession was associated with the User (corresponding to student B), based on the satisfaction of the following conditions (association formed):
  - User.DCPendingInvitation was already set to false.
  - User.SessionParticipation was already set to true.
- User.DCPendingInvitation (corresponding to student B) was set to true (attribute modification).
- Interaction with student B was prepared (user interface activation).

5) For service operation "AcceptDCInvitation":

Name: AcceptDCInvitation(SourceStName: String)
Responsibilities: Accept the invitation from another student participating in the same MMCS-ET session to engage in direct (text, audio / video, file) communication with him / her.
Type: MMCS-ET.
Use Case: "Accept direct communication with a student (active user)"
Notes: - Use a list containing information about all the users that participate in a MMCS-ET session (active users).
- Use a list containing information about all the users that are in direct communication with a specific user.
Exceptions: If student A is not active, indicate that it was an error.
Output:
Pre-conditions: - One student (e.g. student A) invited another student (e.g. student B) to direct communication with him / her.
- Student B decided to accept the invitation from student A.
- Student B indicated his / her decision to accept the invitation from student A.
Post-conditions: - The User (corresponding to student B) was associated with the MMCS-ETProviderProfile (association formed).
- This User was associated with the MMCS-ETSession, based on the confirmation that User.SessionParticipation was already set to true (association formed).
Appendix A: Detailed Presentation of the Development of a MMCS-ET

- The MMCS-ETSessiion was associated with ActiveUserInformation, based on the confirmation that student A was still active (association formed).
- The User (corresponding to student B) was associated with DirectlyCommunicatingUserInformation regarding student A (association formed).
- UserDCPendingInvitation (corresponding to student B) was set to false (attribute modification).
- Interaction with student B was prepared (user interface activation).
- The MMCS-ETSessiion was associated with the User (corresponding to student A) (association formed).
- This User was associated with DirectlyCommunicatingUserInformation regarding student B (association formed).
- Interaction with student A was prepared (user interface activation).

6) For service operation "EngageFileCom":

Name: EngageFileCom(DestUserName: String, FileName: String)
Responsibilities: Send a file to another user participating in the same MMCS-ET session.
Type: MMCS-ET.
Cross References: Service functions: SF.2.8., SF.2.9., SF.2.10.
Use Case: "Engage in file communication with an active user".

Notes: -
Exceptions: - If file communication between user A and user B is not allowed, indicate that it was an error.
- If the transfer of file F from user A to user B is not completed successfully, indicate that it was an error.
Output: -
Pre-conditions: - Two users participated in the same MMCS-ET session and were ready to engage in direct communication.
- One of the users (teacher or student, e.g. user A) decided to send a file (e.g. file F) to the other user (teacher or student, e.g. user B).
- User A specified the file F that he/she wanted to send to user B.

Post-conditions:
- The User (corresponding to user A) was associated with the MMCS-ETSessiion (association formed).
- The MMCS-ETSessiion was associated with the User (corresponding to user B), based on the confirmation that file communication between users A and B is allowed (association formed).
- The MMCS-ETSessiion was associated with FileCommunication (association formed).
- Interaction with user A was prepared (user interface activation).
- Interaction with user B was prepared (user interface activation).

7) For service operation "EngageAVCom":

Name: EngageAVCom(DestUserName: String)
Responsibilities: Initiate the establishment of two A/V stream connections with another user participating in the same MMCS-ET session.
Type: MMCS-ET.
Cross References: Service functions: SF.1.17., SF.2.4.
Use Case: "Engage in A/V communication with a student (active user)".

Notes: -
Exceptions: -
Output: -
Pre-conditions: A user (teacher or student, e.g. user A), that was participating in a MMCS-ET session, was ready to engage in direct communication with another user (teacher or student, e.g. user B), that was participating in the same MMCS-ET session.

Post-conditions:
- The User (corresponding to user A) was associated with the MMCS-ETSessiion (association formed).
- The MMCS-ETSessiion was associated with AudioVideoCommunication (association formed).
- The AudioVideoCommunication was associated with the StreamBinder (association formed).
- An AudioVideoDevice (corresponding to the source A/V device of user A) was created (instance creation).
- An AudioVideoDevice (corresponding to the source A/V device of user B) was created (instance creation).
- An AudioVideoDevice (corresponding to the sink A/V device of user A) was created (instance creation).
- An AudioVideoDevice (corresponding to the sink A/V device of user B) was created (instance creation).
- The StreamBinder was associated with the AudioVideoDevice (corresponding to the source A/V device of user A) (association formed).
- The StreamBinder was associated with the AudioVideoDevice (corresponding to the sink A/V device of user B) (association formed).
- The StreamBinder was associated with the AudioVideoDevice (corresponding to the sink A/V device of user B) (association formed).
- The MMCS-ETSessiion was associated with ActiveUserInformation regarding user A (association formed).
- Interaction with user A was prepared (user interface activation).
- The MMCS-ETSessiion was associated with the User (corresponding to user B) (association formed).
- Interaction with user B was prepared (user interface activation).

8) For service operation "StopAVCom":

Name: StopAVCom(DestUserName: String)
Responsibilities: Stop to communicate using audio and video with another user participating in the same MMCS-ET session.
Type: MMCS-ET.
Cross References: Service functions: SF.1.17., SF.2.5.
Use Case: "Stop A/V communication with a student (active user)".

Notes: Use a list containing information about all the users that participate in a MMCS-ET session (active users).
Exceptions: -
Output: -
Pre-conditions:
- A user (teacher or student, e.g., user A), that was participating in a MMCS-ET session, was engaged in A/V communication with another user (teacher or student, e.g., user B), that was participating in the same MMCS-ET session.
- User A decided to stop the A/V communication between him/her and user B.
- User A indicated his/her decision to stop the A/V communication between him/her and user B.

Post-conditions:
- The User (corresponding to user A) was associated with the MMCS-ETSesson (association formed).
- The MMCS-ETSesson was associated with AudioVideoCommunication (association formed).
- AudioVideoCommunication was associated with the StreamBinder (association formed).
- The StreamBinder was associated with the AudioVideoDevice (corresponding to the source A/V device of user A) (association formed).
- The StreamBinder was associated with the AudioVideoDevice (corresponding to the sink A/V device of user B) (association formed).
- The StreamBinder was associated with the AudioVideoDevice (corresponding to the source A/V device of user B) (association formed).
- The StreamBinder was associated with the AudioVideoDevice (corresponding to the sink A/V device of user B) (association formed).
- The MMCS-ETSesson was associated with ActiveUserInformation regarding user A (association formed).
- Interaction with user A was prepared (user interface activation).
- The MMCS-ETSesson was associated with the User (corresponding to user B) (association formed).
- Interaction with user B was prepared (user interface activation).

9) For service operation "StartAVCom":

Name: StartAVCom(DestUserName: String)
Responsibilities:
Start a blocked A/V communication with another user participating in the same MMCS-ET session.
Type: MMCS-ET.
Cross References: Service functions: SF.1.17, SF.2.6.
Use Case: "Start A/V communication with a student (active user)".
Notes:
Use a list containing information about all the users that participate in a MMCS-ETSesson (active users).
Exceptions:
Output:
Pre-conditions:
- A user (teacher or student, e.g., user A), that was participating in a MMCS-ET session, stopped the A/V communication with another user (a student, e.g., user B), that was participating in the same MMCS-ET session.
- User A decided to start the blocked A/V communication between him/her and user B.
- User A indicated his/her decision to start the blocked A/V communication between him/her and user B.

Post-conditions:
- The User (corresponding to user A) was associated with the MMCS-ETSesson (association formed).
- The MMCS-ETSesson was associated with AudioVideoCommunication (association formed).
- AudioVideoCommunication was associated with the StreamBinder (association formed).
- The StreamBinder was associated with the AudioVideoDevice (corresponding to the source A/V device of user A) (association formed).
- The StreamBinder was associated with the AudioVideoDevice (corresponding to the sink A/V device of user B) (association formed).
- The StreamBinder was associated with the AudioVideoDevice (corresponding to the source A/V device of user B) (association formed).
- The StreamBinder was associated with the AudioVideoDevice (corresponding to the sink A/V device of user B) (association formed).
- The MMCS-ETSesson was associated with ActiveUserInformation regarding user A (association formed).
- Interaction with user A was prepared (user interface activation).
- The MMCS-ETSesson was associated with the User (corresponding to user B) (association formed).
- Interaction with user B was prepared (user interface activation).

10) For service operation "TerminateAVCom":

Name: TerminateAVCom(DestUserName: String)
Responsibilities:
Terminate an established A/V communication with another user participating in the same MMCS-ET session.
Type: MMCS-ET.
Cross References: Service functions: SF.2.7.
Use Case: "Terminate A/V communication with a student (active user)".
Notes:
Exceptions:
Output:
Pre-conditions:
- A user (teacher or student, e.g., user A), that was participating in a MMCS-ET session, was engaged in A/V communication with another user (a student, e.g., user B), participating in the same MMCS-ET session.
- The established A/V communication between user A and user B is to be terminated.

Post-conditions:
- The User (corresponding to user A) was associated with the MMCS-ETSesson (association formed).
- The MMCS-ETSesson was associated with AudioVideoCommunication (association formed).
- AudioVideoCommunication was associated with the StreamBinder (association formed).
- An AudioVideoDevice (corresponding to the source A/V device of user A) was destroyed (instance deletion).
- The StreamBinder destroyed the association with the AudioVideoDevice (corresponding to the source A/V device of user A) (association broken).
- An AudioVideoDevice (corresponding to the sink A/V device of user B) was destroyed (instance deletion).
- The StreamBinder destroyed the association with the AudioVideoDevice (corresponding to the sink A/V device of user B) (association broken).
- An AudioVideoDevice (corresponding to the source A/V device of user B) was destroyed (instance deletion).
- The StreamBinder destroyed the association with the AudioVideoDevice (corresponding to the source A/V device of user B) (association broken).
- An AudioVideoDevice (corresponding to the sink A/V device of user A) was destroyed (instance deletion).
- The StreamBinder destroyed the association with the AudioVideoDevice (corresponding to the sink A/V device of user A) (association broken).
- The MMCS-ETSesson was associated with the User (corresponding to user B) (association formed).
- Interaction with user B was prepared (user interface activation).

Appendix A: Detailed Presentation of the Development of a MMCS-ET
Appendix A: Detailed Presentation of the Development of a MMCS-ET

11) For service operation “EngageChat”:

Name: EngageChat(Message: String)
Responsibilities: Send a (text) message to all the (other) users participating in the same MMCS-ET session.
Type: MMCS-ET.
Cross References: Service functions: SF.1.17., SF.3.1., SF.3.2.
Use Case: “Engage in a chat with all active users”.
Notes: - Use a list containing information about all the users that participate in a MMCS-ET session (active users).
- Use a list containing information about all the messages broadcasted recently to all the users that participate in a MMCS-ET session.
Exceptions: - If there is voting activity in the MMCS-ET session, indicate that it was an error.
- If user A is not active, indicate that it was an error.
Output: -
Pre-conditions: - A number of users participated in the same MMCS-ET session, and were engaged in direct communication with each other.
- One of the users (teacher or student, e.g. user A) decided to send a (text) message (e.g. message M) to all the other users.
- User A specified the message M that he / she wanted to send to all the other users.
- Message M did not indicate a vote or the start of a voting process.
Post-conditions: - The User (corresponding to user A) was associated with the MMCS-ETSession (association formed).
- The MMCS-ETSession was associated with ActiveUserInformation, based on the confirmation that user A was still active (association formed).
- The MMCS-ETSession was associated with the Chatfacility, based on the confirmation that MMCS-ETSession.VotingActivity was already set to false (association formed).
- The Chatfacility was associated with ActiveUserInformation regarding each user (e.g. user X) that was active in the MMCS-ET session (association formed).
- The MMCS-ETSession was associated with the User (corresponding to user X) (association formed).
- Interaction with user X was prepared (user interface activation).

12) For service operation “EngageFileComAll”:

Name: EngageFileComAll(FileName: String)
Responsibilities: Send a file to all the students participating in a MMCS-ET session.
Type: MMCS-ET.
Cross References: Service functions: SF.1.17., SF.3.4.
Use Case: “Engage in file communication with all students (active users)”.
Notes: - Use a list containing information about all the users that participate in a MMCS-ET session (active users).
Exceptions: - If the transfer of file F from the teacher to each one of the students is not completed successfully, indicate that it was an error.
Output: -
Pre-conditions: - A number of users participated in the same MMCS-ET session, and were engaged in direct communication with each other.
- The teacher decided to send a file (e.g. file F) to all the students.
- The teacher specified the file F that he / she wanted to send to all the students.
Post-conditions: - The User (corresponding to the teacher) was associated with the MMCS-ETSession (association formed).
- The MMCS-ETSession was associated with FileCommunication (association formed).
- FileCommunication was associated with ActiveUserInformation regarding each student (e.g. student X) that was active in the MMCS-ET session (association formed).
- The MMCS-ETSession was associated with the User (corresponding to student X) (association formed).
- Interaction with user X was prepared (user interface activation).
- Interaction with the teacher was prepared (user interface activation).

13) For service operation “StartVoting”:

Name: StartVoting(VotingSubject: String)
Responsibilities: Start a voting process between all the users that participate in a MMCS-ET session.
Type: MMCS-ET.
Cross References: Service functions: SF.1.17., SF.3.5.
Use Case: “Start a voting process between all active users”.
Notes: - Use a list containing information about all the users that participate in a MMCS-ET session (active users).
Exceptions: - If there is already voting activity in the MMCS-ET session, indicate that it was an error.
- If the teacher is not truly a teacher, indicate that it was an error.
Output: -
Pre-conditions: - A number of users participated in the same MMCS-ET session, and were engaged in direct communication with each other.
- The teacher decided to start a voting process between all the users.
- The teacher specified in a message (e.g. message M) the subject of the voting process.
- Message M did not indicate a vote or a simple message unrelated with the voting process.
Post-conditions: - The User (corresponding to the teacher) was associated with the MMCS-ETSession (association formed).
- The MMCS-ETSession was associated with ActiveUserInformation, based on the confirmation that the teacher was truly a teacher (association formed).
- The MMCS-ETSession was associated with Voting, based on the confirmation that MMCS-ETSession.VotingActivity was already set to false (association formed).
Appendix A: Detailed Presentation of the Development of a MMCS-ET

14) For service operation “Vote”:

Name: Vote(VotingDecision: String)
Responsibilities: Express a voting decision by sending a vote.
Type: MMCS-ET.
Cross References: Service functions: SF.3.6. Use Case: “Vote in a voting process between all active users”.
Notes: If there is no voting activity in the MMCS-ET session, indicate that it was an error.
Exceptions: -
Output: -
Pre-conditions: - A teacher started a voting process between all the users that were participating in a MMCS-ET session.
- One of these users (teacher or student, e.g. user A) decided to express his/her voting decision.
- User A specified his/her voting decision in a vote which had the form of a message (e.g. message M).
- Message M did not indicate the start of a voting process or a simple message unrelated with the voting process.

Post-conditions: -
- The User (corresponding to user A) was associated with the MMCS-ETSession (association formed).
- The MMCS-ETSession was associated with VotingActivity, based on the confirmation that MMCS-ETSession.VotingActivity was already set to true (association formed).
- Voting.Total was increased by 1 (attribute modification).
- Voting.CounterYes was increased by 1 or Voting.CounterNo was increased by 1 (attribute modification).
- Interaction with user A was prepared (user interface activation).

15) For service operation “PresentVotingOutcome”:

Name: PresentVotingOutcome()
Responsibilities: Calculate the outcome of a voting process and present it to all the involved active users.
Type: MMCS-ET.
Cross References: Service functions: SF.1.17, SF.3.7, SF.3.8. Use Case: “Present the outcome of a voting process to all the involved active users”.
Notes: Use a list containing information about all the users that participate in a MMCS-ET session (active users).
Exceptions: -
Output: -
Pre-conditions: - A teacher started a voting process between all the users that were participating in a MMCS-ET session.
- Each of these users expressed his/her voting decision by sending a vote.
- The teacher indicated that he/she wanted the presentation of the outcome of the voting process to all the involved users.

Post-conditions: -
- The User (corresponding to the teacher) was associated with the MMCS-ETSession (association formed).
- The MMCS-ETSession was associated with VotingActivity (association formed).
- Voting.Total was set to 0 (attribute modification).
- Voting.CounterYes was set to 0 (attribute modification).
- Voting.CounterNo was set to 0 (attribute modification).

16) For service operation “TerminateDC”:

Name: TerminateDC(DestStName: String)
Responsibilities: Terminate direct (text, audio/video, file) communication with another student participating in the same MMCS-ET session.
Type: MMCS-ET.
Cross References: Service functions: SF.1.23. Use Case: “Terminate direct communication between two students (active users)”.
Notes: Use a list containing information about all the users that are in direct communication with a specific user.
Exceptions: -
Output: -
Pre-conditions: - Two students (e.g. students A and B) were engaged in direct communication.
- Student A decided to terminate his/her direct communication with student B.
- Student A indicated his/her decision to terminate his/her direct communication with student B.

Post-conditions: -
- The User (corresponding to student A) was associated with the MMCS-ETSession (association formed).
- The MMCS-ETSession was associated with the User (corresponding to student B) (association formed).
- Interaction with student B was prepared (user interface activation).
- The MMCS-ETSession was associated with TextCommunication (association formed).
- TextCommunication.PermitTextCommunication was set to false (attribute modification).
- TextCommunication.PreventUser1 was set to the user name of student A (attribute modification).
Appendix A: Detailed Presentation of the Development of a MMCS-ET

- TextCommunication.PreventUser2 was set to the user name of student B (attribute modification).
- The MMCS-ETSession was associated with FileCommunication (association formed).
- FileCommunication.PermitFileCommunication was set to false (attribute modification).
- FileCommunication.PreventUser1 was set to the user name of student A (attribute modification).
- FileCommunication.PreventUser2 was set to the user name of student B (attribute modification).
- The MMCS-ETSession was associated with AudioVideoCommunication (association formed).
- AudioVideoCommunication was associated with the StreamBinder (association formed).
- An AudioVideoDevice (corresponding to the source A/V device of student A) was destroyed (instance deletion).
- The StreamBinder destroyed the association with the AudioVideoDevice (corresponding to the source A/V device of student A) (association broken).
- An AudioVideoDevice (corresponding to the sink A/V device of student B) was destroyed (instance deletion).
- The StreamBinder destroyed the association with the AudioVideoDevice (corresponding to the sink A/V device of student B) (association broken).
- An AudioVideoDevice (corresponding to the sink A/V device of the teacher) was destroyed (instance deletion).
- The StreamBinder destroyed the association with the AudioVideoDevice (corresponding to the sink A/V device of the teacher) (association broken).
- The User (corresponding to student B) was associated with DirectlyCommunicatingUser-Information regarding student A (association formed).
- Interaction with student B was prepared (user interface activation).
- The User (corresponding to student A) was associated with DirectlyCommunicatingUser-Information regarding student B (association formed).
- Interaction with student A was prepared (user interface activation).
- TextCommunication.PermitTextCommunication was set to true (attribute modification).
- TextCommunication.PreventUser1 was set to null (attribute modification).
- TextCommunication.PreventUser2 was set to null (attribute modification).
- FileCommunication.PermitFileCommunication was set to true (attribute modification).
- FileCommunication.PreventUser1 was set to null (attribute modification).
- FileCommunication.PreventUser2 was set to null (attribute modification).

17) For service operation "RemoveStudent":

**Name:** RemoveStudent(StudentName: String)

**Responsibilities:** Remove a student from a MMCS-ET session.

**Type:** MMCS-ET.

**Cross References:** Service functions: SF.1.14., SF.1.17., SF.1.22., SF.1.23.

**Use Case:** "Remove a student from a MMCS-ET session"

**Notes:**
- Use a list containing information about all the users that participate in a MMCS-ET session (active users).
- Use a list containing information about all the users that are in direct communication with a specific user.

**Exceptions:**
- If there is voting activity in the MMCS-ET session, indicate that it was an error.
- If student A has a pending invitation for direct communication from another student, indicate that it was an error.

**Output:** -

**Pre-conditions:**
- A teacher and one student (e.g. student A) participated in the same MMCS-ET session and were engaged in direct communication.
- The teacher decided to remove from the MMCS-ET session that he/she created student A.
- The teacher indicated his/her decision to remove student A from the MMCS-ET session.

**Post-conditions:**
- The User (corresponding to the teacher) was associated with the MMCS-ETSession (association formed).
- The MMCS-ETSession was associated with the User (corresponding to student A), based on the confirmation that MMCS-ETSession.VotingActivity was already set to false (association formed).
- User.SessionParticipation (corresponding to student A) was set to false, based on the confirmation that User.DCPendingInvitation was already set to false (attribute modification).
- Interaction with student A was prepared (user interface activation).
- The MMCS-ETSession was associated with ActiveUserInformation regarding student A who is being removed from the MMCS-ET session (association formed).
- The MMCS-ETSession was associated with TextCommunication (association formed).
- TextCommunication.PermitTextCommunication was set to false (attribute modification).
- TextCommunication.PreventUser1 was set to the user name of the teacher (attribute modification).
- TextCommunication.PreventUser2 was set to the user name of student A (attribute modification).
- The MMCS-ETSession was associated with FileCommunication (association formed).
- FileCommunication.PermitFileCommunication was set to false (attribute modification).
- FileCommunication.PreventUser1 was set to the user name of the teacher (attribute modification).
- FileCommunication.PreventUser2 was set to the user name of student A (attribute modification).
- The MMCS-ETSession was associated with AudioVideoCommunication (association formed).
- AudioVideoCommunication was associated with the StreamBinder (association formed).
- An AudioVideoDevice (corresponding to the source A/V device of the teacher) was destroyed (instance deletion).
- The StreamBinder destroyed the association with the AudioVideoDevice (corresponding to the source A/V device of the teacher) (association broken).
- An AudioVideoDevice (corresponding to the sink A/V device of student A) was destroyed (instance deletion).
- The StreamBinder destroyed the association with the AudioVideoDevice (corresponding to the sink A/V device of student A) (association broken).
- An AudioVideoDevice (corresponding to the sink A/V device of student B) was destroyed (instance deletion).
- The StreamBinder destroyed the association with the AudioVideoDevice (corresponding to the sink A/V device of student B) (association broken).
- An AudioVideoDevice (corresponding to the source A/V device of the teacher) was destroyed (instance deletion).
- The StreamBinder destroyed the association with the AudioVideoDevice (corresponding to the source A/V device of the teacher) (association broken).
- The User (corresponding to student A) was associated with DirectlyCommunicatingUser-Information regarding the teacher and all the students that were in direct communication with student A (association formed).
Appendix A: Detailed Presentation of the Development of a MMCS-ET

**18) For service operation “TerminateSession”:**

- **Name:** TerminateSession()
- **Responsibilities:** Terminate a MMCS-ET session.
- **Type:** MMCS-ET.
- **Cross References:** Service functions: SF.1.14., SF.1.22., SF.1.24.
- **Notes:** Use Case: “Terminate a MMCS-ET session”.
- **Exceptions:**
- **Output:**
  - - A teacher started a MMCS-ET session.
  - - The teacher decided to terminate the MMCS-ET session that he/she started.
  - - The teacher indicated his/her decision to terminate the MMCS-ET session that he/she started.
- **Pre-conditions:**
  - - The User (corresponding to the teacher) was associated with DirectlyCommunicatingUserInformation regarding all the students that were in direct communication with the teacher (association formed).
- **Post-conditions:**
  - - The User (corresponding to the teacher) was associated with DirectlyCommunicatingUserInformation regarding student A (association formed).
  - - The User (corresponding to student A) was associated with DirectlyCommunicatingUserInformation regarding student A (association formed).
  - - The UserProfile.Status (corresponding to student A) was set to logged in (attribute modification).
  - - Interaction with the teacher was prepared (user interface activation).
- **TextCommunication.PermitTextCommunication** was set to true (attribute modification).
- **TextCommunication.PreventUser1** was set to null (attribute modification).
- **TextCommunication.PreventUser2** was set to null (attribute modification).
- **FileCommunication.PermitFileCommunication** was set to true (attribute modification).
- **FileCommunication.PreventUser1** was set to null (attribute modification).
- **FileCommunication.PreventUser2** was set to null (attribute modification).

**19) For service operation “TerminateService”:**

- **Name:** TerminateService(ServiceName: String)
- **Responsibilities:** Terminate the MMCS-ET service.
- **Type:** MMCS-ET.
- **Cross References:** Service functions: SF.1.16., SF.1.25.
- **Notes:** Use Case: “Terminate the MMCS-ET service”.
- **Exceptions:**
- **Output:**
  - - A user (teacher or student) started a MMCS-ET service.
  - - The user logged in to the MMCS-ET provider domain.
  - - The user decided to terminate the MMCS-ET service that he/she started.
- **Pre-conditions:**
  - - The User (corresponding either to a teacher or a student) was associated with DirectlyCommunicatingUserInformation regarding the user that wanted to terminate the MMCS-ET service (association formed).
- **Post-conditions:**
  - - The User (corresponding either to a teacher or a student) was associated with DirectlyCommunicatingUserInformation regarding the user that wanted to terminate the MMCS-ET service (association formed).
  - - The UserProfile.Status was set to logged in (attribute modification).
  - - Interaction with the teacher was prepared (user interface activation).
Appendix A: Detailed Presentation of the Development of a MMCS-ET

- The MMCS-ETServiceProfile destroyed the association with the UserProfile (corresponding either to the teacher or the student) (association broken).
- The MMCS-ETProviderProfile destroyed the association with the UserProfile (corresponding either to the teacher or the student) (association broken).
- The MMCS-ETProviderProfile destroyed the association with the MMCS-ETServiceProfile (association broken).
- The MMCS-ETProviderProfile was destroyed (instance deletion).
- The User (corresponding either to a teacher or a student) destroyed the association with the MMCS-ETProviderProfile (association broken).
- This User was destroyed (instance deletion).
- Interaction with the user was prepared (user interface activation).


A service interaction diagram is defined for every service operation of the MMCS-ET (activity 3). Therefore, the following service interaction diagrams are created (except from those that correspond to the target use cases, see Section 5.2.1.3.):

1) For service operation “StartService”:

![StartService Diagram]

2) For service operation “LogIn”:

![LogIn Diagram]

3) For service operation “StartSession”:

![StartSession Diagram]
4) For service operation "InviteDCStudent":

5) For service operation "AcceptDCInvitation":

6) For service operation "EngageFileCom":
Appendix A: Detailed Presentation of the Development of a MMCS-ET

7) For service operation "EngageAVCom":

8) For service operation "StopAVCom":

9) For service operation "StartAVCom":

A-30
10) For service operation "TerminateAVCom":

11) For service operation "EngageChat":

12) For service operation "EngageFileComAll":
13) For service operation "StartVoting":

14) For service operation "Vote":

15) For service operation "PresentVotingOutcome":

[Diagram images of the service operations are shown here.]
16) For service operation "TerminateDC":

17) For service operation "RemoveStudent":

18) For service operation "TerminateSession":

Appendix A: Detailed Presentation of the Development of a MMCS-ET
Appendix A: Detailed Presentation of the Development of a MMCS-ET

19) For service operation "TerminateService":

In the service interaction diagrams of the MMCS-ET, a multiobject represents a logical set of instances, it is implemented as a group of instances stored in a container or collection object, and is shown by a stack icon. A message sent to a multiobject indicates that it is a message sent to the collection object itself, rather than an implicit broadcast to the collection's members. This is especially obvious for generic operations such as find and add. It has to be noted that from UML 1.1, messages to a multiobject are not broadcasted to each element (as was the case in prior versions of the UML). The multiobjects that are used in the service interaction diagrams of the MMCS-ET are the following:

- **UserSecurityCatalog**: This is a catalog containing security information (UserSecurityInfo) about potential users of the MMCS-ET. The contents of UserSecurityInfo are: user name, password, first name, surname, and role.

- **UserSubscriptionCatalog**: This is a catalog containing subscription information (UserSubscriptionInfo) about all the users that are subscribers of the MMCS-ET. The contents of UserSubscriptionInfo are: user name, first name, and surname.

- **LoggedInUserInfo list**: This is a list containing information about all the logged in users. The contents of LoggedinUserInfo are: user name and UA reference.

- **ActiveUserInfo list**: This is a list containing information about all the users that participate in a MMCS-ET session (active users). The contents of ActiveUserInfo are: user name, user role, and USM reference.

- **DCUserInfo list**: This is a list containing information about all the users that are in direct communication with a specific user. The contents of DCUserInfo are: user name and user role.

- **ExchangedMessageInfo list**: This is a list containing information about all the messages exchanged recently between two users. The contents of ExchangedMessageInfo are: message and sender.

---

1 A catalog is used for security and subscription information, because they are stored in a file and retrieved on demand.
Appendix A: Detailed Presentation of the Development of a MMCS-ET

- BroadcastedMessageInfo list: This is a list containing information about all the messages broadcasted recently to all the users that participate in the MMCS-ET session. The contents of BroadcastedMessageInfo are: message and sender.

The service design class diagram of the MMCS-ET (activity 4) can be seen in Figure 5.16.

A.5. Service Implementation Phase.

MIDL has a central role in the service implementation phase. Except from the specification of service (COM) objects and their interfaces (see Section 5.2.1.5.), MIDL is also used for the definition of type libraries. A type library is a compiled version of an IDL file that can be accessed programmatically. DCOM allows programmers to dynamically access interface definitions and query for information on interfaces, components, methods, and parameters. To provide this facility, DCOM provides type libraries, which are generated during an IDL compilation, in the following way:

```c
[uuid(543FB200-6281-11D1-BD74-204C4F4F5020),
 helpstring("Type Library")]
library MMCSETLib
{
 importlib("stdole32.tlb");
 importlib("stdole2.tlb");
 [uuid(543FB20E-6281-11D1-BD74-204C4F4F5020)]
 coclass UserAgent
 {
  [default] interface IUserAgent;
  }
}
```

When processing this IDL file, the statement library will force a type library to be generated. The coclass statement is used to identify the COM component UserAgent and to indicate that it implements the IUserAgent interface. A client can ask a server at runtime if it supports the ITypeInfo interface. Through this interface, the client can interrogate the server to find out what methods and properties are provided. This facility is used to invoke methods on COM objects that were unknown when the client was compiled (e.g. when the MMCS-ET is initialised) and is also beneficial for scripting languages (e.g. Visual Basic). A type library can be included in an executable program as a resource. However, since type libraries are compiled, they are platform-specific [GRIM97][RUBI99].

As was mentioned in Section 5.2.1.5. error checking in the MMCS-ET code is performed by examining the value of HRESULT after every function call. The following function displays detailed information contained in an HRESULT:

```c
A-35
```
BOOL ShowStatus(HRESULT hr)
{
    // Construct a _com_error using the HRESULT
    _com_error e(hr);

    // Show the hr as a decimal number
    cout << "hr as decimal: " << hr << endl;

    // Show the first 16 bits (information code)
    // This code describes the specific error
    cout << "Information code: " << HRESULT_CODE(hr) << endl;

    // Show facility code as a decimal number
    // This code reveals the subsystem that created the error;
    cout << "Facility code: " << HRESULT_FACILITY(hr) << endl;

    // Show the severity bit
    cout << "Severity: " << HRESULT_SEVERITY(hr) << endl;

    // Use the _com_error object to format a message string
    cout << "Message string: " << e.ErrorMessage() << endl;
    return TRUE;
}

Finally, regarding the user interface of the MMCS-ET, a typical usage scenario is described in order to illustrate the use of the main windows of the service. More specifically, it is assumed that a teacher and two students (A and B) are ready (i.e. logged in) to participate in an educational session (MMCS-ET session). Initially, the teacher starts a MMCS-ET session and invites student A to join. After the acceptance of the invitation from student A, the following windows will be visible to each one of the three users:

- **Teacher**: participant choice window, main user window (referring to student A), and common activities window.
- **Student A**: participant choice window, main user window (referring to the teacher), and common activities window.
- **Student B**: participant choice window.

Then, the teacher invites student B to join the MMCS-ET session. After the acceptance of the invitation from student B, the following windows will be visible to each one of the three users:

- **Teacher**: participant choice window, main user window (referring to student A), main user window (referring to student B), and common activities window.
- **Student A**: participant choice window, main user window (referring to the teacher), and common activities window.
- **Student B**: participant choice window, main user window (referring to the teacher), and common activities window.

Finally, student A invites student B to direct communication. After the acceptance of the invitation from student B, the following windows will be visible to each one of the three users:
Appendix A: Detailed Presentation of the Development of a MMCS-ET

- **Teacher**: participant choice window, main user window (referring to student A), main user window (referring to student B), and common activities window.

- **Student A**: participant choice window, main user window (referring to the teacher), main user window (referring to student B), and common activities window.

- **Student B**: participant choice window, main user window (referring to the teacher), main user window (referring to student A), and common activities window.

It is pointed out that only the teacher can terminate the MMCS-ET session by removing from it students A and B.
Appendix B

Overview of the Proposed Service Development Methodology

B.1. Introduction.

This appendix presents in a concise manner the most important activities and steps of the main phases of the proposed service development methodology that are examined in detail in Chapter 4.

B.2. Overview of the Requirements Capture and Analysis Phase.

This phase is structured in the following way:

Activity 1: Define Service Development Plan.

Step 1: Specify the exact service development methodology that will be used.

Step 2: Provide important information about the client organisation / enterprise.

Step 3: Specify the service developer(s) that will participate in the service development process.

Step 4: Specify the required computing and network infrastructure, and the required software tools.

Step 5: Initiate a selection process for a programming language and a DPE that will be used for the implementation of the service.

Step 6: Mention and examine all other matters that affect the service development process.

Activity 2: Create Preliminary Investigation Report.

Step 1: Gather useful information regarding the telematic service under development.

Step 2: State in a concise manner the main objective of the service.

Step 3: Analyse the constituent parts of the main service objective.

Step 4: Identify the initial service requirements.

Step 5: Structure the identified initial service requirements.

Step 6: Consider the TINA-C Business Model. More specifically:
Appendix B: Overview of the Proposed Service Development Methodology

- Identify the generic (business) roles that the actors of the service can take.
- Construct a business model for the service by adapting the model proposed by TINA-C.
- Identify the service independent specific role(s) that the actors of the service can take.
- Identify the service dependent specific role(s) that the actors of the service can take.
- Identify the actors of the service.
- Specify the type of interactions that should be considered between each pair of generic (business) roles.
- Identify the session roles that the actors of the service can take.

**Activity 3: Define Service Requirements.**

**Step 1:** Identify the (desirable) service functions.

**Step 2:** Structure the identified service functions by categorising them.

**Step 3:** Specify the type of the service functions.

**Step 4:** Identify the (desirable) service properties.

**Step 5:** Specify the attribute details or the boundary conditions for the identified service attributes.

**Step 6:** Specify the type of the service attributes.

**Step 7:** Relate service functions and service attributes.

**Step 8:** Include all necessary information in the appropriate document (specifications of service requirements).

**Activity 4: Define Use Cases (high level and essential).**

**Step 1:** Specify the boundaries of the telematic service.

**Step 2:** Identify use cases using either the actor-based or the event-based method.

**Step 3:** Write all use cases in the high-level format.

**Step 4:** Draw a (draft) use case diagram.

**Step 5:** Write the most critical and influential use cases in the expanded essential format.

**Step 6:** Create some real use cases (optionally).

**Activity 5: Define Draft Service Conceptual Model (optionally).**

**Activity 6: Record Terms in Glossary.**

**Activity 7: Consider the Application of Rapid Prototyping (optionally).**

**Activity 8: Define Draft Service Architecture Layers (optionally).**

**Step 1:** Identify the service architecture layers.

**Step 2:** Comment on their importance for the telematic service under examination.

**Step 3:** Record the necessary information in the service development plan.

**Activity 9: Refine Service Development Plan.**

**Step 1:** Reconsider the contents of the service development cycles.
Appendix B: Overview of the Proposed Service Development Methodology

Step 2: Devise an appropriate ranking scheme.

Step 3: Rank the use cases according to this ranking scheme.

Step 4: Allocate the use cases to service development cycles.

Step 5: Create a schedule for the proposed methodology.

B.3. Overview of the Service Analysis Phase.

This phase is structured in the following way:

Activity 1: Define Essential Use Cases.

Activity 2: Refine the Use Case Diagrams.

Activity 3: Define Service Conceptual Models.

Step 1: Identify the service concepts. More specifically:
- Apply the service concept category list technique.
- Apply the noun phrase identification technique.
- Draw an initial service conceptual model by representing graphically only the service concepts.
- Identify useful type hierarchies (optionally).

Step 2: Identify associations between the service concepts. More specifically:
- Find the need-to-know associations.
- Consider the common associations list.
- Consider aggregation (composite or shared) associations, derived associations, qualified associations, and recursive or reflexive associations (optionally).
- Select the desirable associations that will be included in the main service conceptual model.

Step 3: Identify attributes of the service concepts.

Step 4: Draw the main service conceptual model.

Step 5: Specify the ancillary service conceptual models. More specifically:
- Customise the session information model according to the service requirements.
- Customise the session role information model according to the service requirements.
- Customise the access session related information models according to the service requirements.
- Customise the service session related information models according to the service requirements.
- Customise the Service Session Graph information model according to the service requirements.

Activity 4: Define Service Sequence Diagrams.

Step 1: Draw a vertical line representing the telematic service as a black box.

Step 2: Identify each actor that directly operates on (or interacts with) the telematic service.

Step 3: Draw a vertical line for each actor.
Appendix B: Overview of the Proposed Service Development Methodology

**Step 4:** Identify the (external) service events that each actor generates.
**Step 5:** Illustrate the identified service events in the correct order on the diagram.
**Step 6:** Include (fragments of) the use case text to the left of the diagram (optionally).

**Activity 5:** Define Service Operation Contracts.

**Activity 6:** Refine Glossary.

**Activity 7:** Define Service State Diagrams (optionally).

**Step 1:** Draw a use case service state diagram for each use case or (alternatively) draw a global service state diagram.
**Step 2:** Add to the service state diagram(s) transition actions, transition guard conditions, and nested states (optionally).

### B.4. Overview of the Service Design Phase.

This phase is structured in the following way:

**Activity 1:** Define Real Use Cases (optionally).

**Activity 2:** Define User Interface Aspects.

**Activity 3:** Define Service Interaction Diagrams.

**Step 1:** Identify the service COs.
**Step 2:** Consider the generic TINA-C service scenarios and select the most appropriate.
**Step 3:** Form the service interaction diagrams. More specifically:
- Create a separate service interaction diagram for each service operation under development in the current service development cycle.
- Design a set of interacting service COs with the intention to fulfill the responsibilities and post-conditions of the appropriate service operation contracts.
- Split each service interaction diagram into smaller diagrams (if it gets complex).

**Activity 4:** Define Service Design Class Diagrams.

**Step 1:** Identify the service classes by analysing the service interaction diagrams.
**Step 2:** Draw all the identified service classes in a simple service design class diagram.
**Step 3:** Duplicate the attributes to the service classes from the associated concepts in the service conceptual model(s).
**Step 4:** Add method names to the service classes by analysing the service interaction diagrams.
**Step 5:** Add type information to the attributes, method parameters, and method return values (optionally).
**Step 6:** Add the associations necessary to support the required attribute visibility.
**Step 7:** Add navigability arrows to the associations to indicate the direction of attribute visibility.
Appendix B: Overview of the Proposed Service Development Methodology

Step 8: Add dependency relationship lines to indicate non-attribute visibility between service classes.
Step 9: Split the service design class diagram into smaller diagrams (if it gets complex).

Activity 5: Define Service Architecture Layers.
Step 1: Define draft service architecture layers (if not yet done).
Step 2: Draw a service architecture package diagram.
Step 3: Partition the service design class diagram using suitable packages (optionally).
Step 4: Annotate the service architecture package diagram with dependencies between its packages (optionally).


B.5. Overview of the Service Implementation Phase.

This phase is structured in the following way:

Activity 1: Create IDL Specifications.
Activity 2: Implement Service Class and Interface Definitions.
Activity 3: Implement Methods of Service Classes.
Activity 4: Implement Graphical User Interface (Windows).
Activity 5: Implement Reports.
Activity 7: Integrate Implementation Work.
Activity 8: Prepare for Testing.
# Appendix C:

## Acronyms and Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>3pty</td>
<td>Third Party</td>
</tr>
<tr>
<td>A/V</td>
<td>Audio / Video</td>
</tr>
<tr>
<td>ACE</td>
<td>Application Construction Environment</td>
</tr>
<tr>
<td>ACTS</td>
<td>Advanced Communications Technologies and Services</td>
</tr>
<tr>
<td>ADSI</td>
<td>Active Directory Service Interface</td>
</tr>
<tr>
<td>AIN</td>
<td>Advanced Intelligent Network</td>
</tr>
<tr>
<td>anonUA</td>
<td>anonymous User Agent</td>
</tr>
<tr>
<td>ANSA</td>
<td>Advanced Network Systems Architecture</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>as-UAP</td>
<td>access session User Application</td>
</tr>
<tr>
<td>ATL</td>
<td>ActiveX Template Library</td>
</tr>
<tr>
<td>ATM</td>
<td>Asynchronous Transfer Mode</td>
</tr>
<tr>
<td>AVI</td>
<td>Audio-Video Interleaved</td>
</tr>
<tr>
<td>BCP</td>
<td>Basic Call Process</td>
</tr>
<tr>
<td>Bkr</td>
<td>Broker</td>
</tr>
<tr>
<td>BOA</td>
<td>Basic Object Adapter</td>
</tr>
<tr>
<td>BOOST</td>
<td>Broadband Object-Oriented Service Technology (RACE Project No. R2076)</td>
</tr>
<tr>
<td>CAD</td>
<td>Computer Aided Design</td>
</tr>
<tr>
<td>CASE</td>
<td>Computer-Aided Software Engineering</td>
</tr>
<tr>
<td>CCS7</td>
<td>Common Channel Signalling no. 7</td>
</tr>
<tr>
<td>CDS</td>
<td>Cell Directory naming Service</td>
</tr>
<tr>
<td>CGI</td>
<td>Common Gateway Interface</td>
</tr>
<tr>
<td>CLSID</td>
<td>CLaSs Identifier</td>
</tr>
<tr>
<td>CMIS/P</td>
<td>Common Management Information Service and Protocol</td>
</tr>
<tr>
<td>CO</td>
<td>Computational Object</td>
</tr>
</tbody>
</table>
**Appendix C: Acronyms and Abbreviations**

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COG</td>
<td>Computational Object Group</td>
</tr>
<tr>
<td>COM</td>
<td>Component Object Model</td>
</tr>
<tr>
<td>CompD_USS</td>
<td>Composer Domain Usage Service Session</td>
</tr>
<tr>
<td>CompUSM</td>
<td>Composer Usage Session Manager</td>
</tr>
<tr>
<td>ConS</td>
<td>Connection Service</td>
</tr>
<tr>
<td>CORBA</td>
<td>Common Object Request Broker Architecture</td>
</tr>
<tr>
<td>CRC</td>
<td>Class-Responsibility-Collaborator cards</td>
</tr>
<tr>
<td>CSCC</td>
<td>International Multiconference on Circuits, Systems, Communications and Computers</td>
</tr>
<tr>
<td>CSM</td>
<td>Communication Session Manager</td>
</tr>
<tr>
<td>D-AS</td>
<td>Domain Access Session</td>
</tr>
<tr>
<td>D_USS</td>
<td>Domain Usage Service Session</td>
</tr>
<tr>
<td>DCE</td>
<td>Distributed Computing Environment</td>
</tr>
<tr>
<td>DCOM</td>
<td>Distributed Component Object Model</td>
</tr>
<tr>
<td>DHTML</td>
<td>Dynamic HyperText Markup Language</td>
</tr>
<tr>
<td>DII</td>
<td>Dynamic Interface Invocation</td>
</tr>
<tr>
<td>DIMMA</td>
<td>Distributed Interactive MultiMedia Architecture</td>
</tr>
<tr>
<td>DLL</td>
<td>Dynamic Link Library</td>
</tr>
<tr>
<td>DME</td>
<td>Distributed Management Environment</td>
</tr>
<tr>
<td>DNA</td>
<td>Distributed interNet Applications architecture</td>
</tr>
<tr>
<td>DOOS</td>
<td>Design Object-Oriented Software</td>
</tr>
<tr>
<td>DPE</td>
<td>Distributed Processing Environment</td>
</tr>
<tr>
<td>DSI</td>
<td>Dynamic Skeleton Interface</td>
</tr>
<tr>
<td>DSOM</td>
<td>Distributed Systems Operations &amp; Management (workshop)</td>
</tr>
<tr>
<td>DTP</td>
<td>Distributed Transaction Processing</td>
</tr>
<tr>
<td>ECMA</td>
<td>European Computer Manufacturers Association</td>
</tr>
<tr>
<td>eCO</td>
<td>engineering Computation Object</td>
</tr>
<tr>
<td>ECOOP</td>
<td>European Conference on Object-Oriented Programming</td>
</tr>
<tr>
<td>EDOC</td>
<td>Enterprise Distributed Object Computing (conference)</td>
</tr>
<tr>
<td>EII</td>
<td>European Information Infrastructure</td>
</tr>
<tr>
<td>ETSI</td>
<td>European Telecommunications Standards Union</td>
</tr>
<tr>
<td>EU</td>
<td>European Union</td>
</tr>
<tr>
<td>EURESCOM</td>
<td>European Institute for Research and Strategic Studies in Telecommunications</td>
</tr>
<tr>
<td>FCAPS</td>
<td>Fault Configuration Accounting Performance Security</td>
</tr>
<tr>
<td>FTDCS</td>
<td>Future Trends of Distributed Computing Systems (workshop)</td>
</tr>
<tr>
<td>GDMO</td>
<td>Guidelines for the Definition of Managed Objects</td>
</tr>
<tr>
<td>GFP</td>
<td>Global Functional Plane</td>
</tr>
</tbody>
</table>

C-2
Appendix C: Acronyms and Abbreviations

GII  Global Information Infrastructure
GRM  General Relationship Model
GUI  Graphical User Interface
GUID  Globally Unique IDentifier
HCI  Human-Computer Interaction
HTML  HyperText Markup Language
IA  Initial Agent
IBC  Integrated Broadband Communications
IBTE  The Institution of British Telecommunications Engineers
ICC  International Conference on Communications
ICIN  International Conference on Intelligent Networks
ICM  Integrated Communications Management (RACE Project No. R2059)
ICSE  International Conference on Software Engineering
ICT  International Conference on Telecommunications
IDL  Interface Definition Language
IDMS  Interactive Distributed Multimedia Systems (workshop)
IID  Interface IDentifier
IIOP  Internet Inter-ORB Protocol
IMA  Interactive Multimedia Association
IMT2000  International Mobile Telecommunications 2000
IN  Intelligent Network
IN CS  Intelligent Network Capability Set
IN LTA  Intelligent Network Long-Term Architecture
INA  Information Networking Architecture
INCM  IN Conceptual Model
IO  Information Object
IPng  Internet Protocol next generation
IS&N  Intelligence in Services & Networks (conference)
ISCC  IEEE Symposium on Computers & Communications
ISDN  Integrated Services Digital Network
ISI  Integrated Services Internet
ISO  International Standards Organisation
IT  Information Technology
ITU-T  International Telecommunications Union
IXIT  Implementation eXtra Information for Testing
JAD  Joint Application Development
### Appendix C: Acronyms and Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KTN</td>
<td>kernel Transport Network</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>LDAP</td>
<td>Lightweight Directory Access Protocol</td>
</tr>
<tr>
<td>LNCS</td>
<td>Lecture Notes in Computer Science</td>
</tr>
<tr>
<td>LOTOS</td>
<td>Language Of Temporal Ordering Specification</td>
</tr>
<tr>
<td>MCS</td>
<td>Microsoft Cluster Server</td>
</tr>
<tr>
<td>MIDL</td>
<td>Microsoft Interface Definition Language</td>
</tr>
<tr>
<td>MMC</td>
<td>Microsoft Management Console</td>
</tr>
<tr>
<td>MMCS-ET</td>
<td>Multimedia Conferencing Service for Education and Training</td>
</tr>
<tr>
<td>MPEG</td>
<td>Motion Picture Experts Group</td>
</tr>
<tr>
<td>MSC</td>
<td>Message Sequence Chart</td>
</tr>
<tr>
<td>MSMQ</td>
<td>MicroSoft Message Queue server</td>
</tr>
<tr>
<td>MSN</td>
<td>Multi-Service Network</td>
</tr>
<tr>
<td>MSS</td>
<td>Multimedia Systems Services</td>
</tr>
<tr>
<td>MTA</td>
<td>MultiThreaded Apartment model</td>
</tr>
<tr>
<td>MTS</td>
<td>Microsoft Transaction Server</td>
</tr>
<tr>
<td>namedUA</td>
<td>named User Agent</td>
</tr>
<tr>
<td>NCCE</td>
<td>Native Computing and Communication Environment</td>
</tr>
<tr>
<td>NDS</td>
<td>Netware Directory Service</td>
</tr>
<tr>
<td>NIMSAD</td>
<td>Normative Information Model-based System Analysis and Design</td>
</tr>
<tr>
<td>NTLM</td>
<td>NT LAN Manager</td>
</tr>
<tr>
<td>OAD</td>
<td>Object Activation Daemon</td>
</tr>
<tr>
<td>ODL</td>
<td>Object Definition Language</td>
</tr>
<tr>
<td>ODTA</td>
<td>Open Distributed Telecommunication Architecture</td>
</tr>
<tr>
<td>OLE</td>
<td>Object Linking and Embedding</td>
</tr>
<tr>
<td>OMA</td>
<td>Object Management Architecture</td>
</tr>
<tr>
<td>OMG</td>
<td>Object Management Group</td>
</tr>
<tr>
<td>OMT</td>
<td>Object Modelling Technique</td>
</tr>
<tr>
<td>ONP</td>
<td>Open Network Provision</td>
</tr>
<tr>
<td>OOA</td>
<td>Object-Oriented Analysis</td>
</tr>
<tr>
<td>OOAD</td>
<td>Object-Oriented Analysis and Design</td>
</tr>
<tr>
<td>OOD</td>
<td>Object-Oriented Design</td>
</tr>
<tr>
<td>OOPSLA</td>
<td>Object-Oriented Programming, Systems, Languages &amp; Applications (conference)</td>
</tr>
<tr>
<td>OORAM</td>
<td>Object Oriented Role Analysis Method</td>
</tr>
<tr>
<td>OOSA</td>
<td>Object-Oriented Systems Analysis</td>
</tr>
<tr>
<td>OOSE</td>
<td>Object-Oriented Software Engineering</td>
</tr>
<tr>
<td>Acronym</td>
<td>Abbreviation</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ORB</td>
<td>Object Request Broker</td>
</tr>
<tr>
<td>ORPC</td>
<td>Object Remote Procedure Call</td>
</tr>
<tr>
<td>OSA</td>
<td>Open Services Architectural framework</td>
</tr>
<tr>
<td>OSF</td>
<td>Operations System Function and Open Software Foundation</td>
</tr>
<tr>
<td>OSI-SM</td>
<td>Open Systems Interconnection - Systems Management</td>
</tr>
<tr>
<td>PA</td>
<td>Provider Agent</td>
</tr>
<tr>
<td>PD_AS</td>
<td>Provider Domain Access Session</td>
</tr>
<tr>
<td>PD_USS</td>
<td>Provider Domain Usage Service Session</td>
</tr>
<tr>
<td>PeerA</td>
<td>Peer Agent</td>
</tr>
<tr>
<td>PeerD_AS</td>
<td>Peer Domain Access Session</td>
</tr>
<tr>
<td>PeerD_USS</td>
<td>Peer Domain Usage Service Session</td>
</tr>
<tr>
<td>PeerUSM</td>
<td>Peer Usage Session Manager</td>
</tr>
<tr>
<td>PEM</td>
<td>Performance Evaluation Methodology</td>
</tr>
<tr>
<td>PNO</td>
<td>Public Network Operator</td>
</tr>
<tr>
<td>PREMO</td>
<td>PRresentation Environment for Multimedia Objects</td>
</tr>
<tr>
<td>PREPARE</td>
<td>PRe-Pilot in Advanced Resource Management (RACE Project No. R2004)</td>
</tr>
<tr>
<td>PSS</td>
<td>Provider Service Session</td>
</tr>
<tr>
<td>PSTN</td>
<td>Public Switched Telephone Network</td>
</tr>
<tr>
<td>PTT</td>
<td>Post Telegraph and Telephone</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>RACE</td>
<td>Research into Advanced Communication in Europe</td>
</tr>
<tr>
<td>RDS</td>
<td>Remote Data Service</td>
</tr>
<tr>
<td>Ret</td>
<td>Retailer</td>
</tr>
<tr>
<td>RetINA</td>
<td>Real-time TINA (ACTS Project No. AC048)</td>
</tr>
<tr>
<td>RfP</td>
<td>Request for Proposal</td>
</tr>
<tr>
<td>RM-ODP</td>
<td>Reference Model for Open Distributed Processing</td>
</tr>
<tr>
<td>RMI</td>
<td>Remote Method Invocation</td>
</tr>
<tr>
<td>ROSA</td>
<td>RACE Open Services Architecture (RACE Project No. R1093)</td>
</tr>
<tr>
<td>ROT</td>
<td>Running Object Table</td>
</tr>
<tr>
<td>RPC</td>
<td>Remote Procedure Call</td>
</tr>
<tr>
<td>RIR</td>
<td>Retailer-to-Retailer</td>
</tr>
<tr>
<td>SCE</td>
<td>Service Creation Environment</td>
</tr>
<tr>
<td>SCORE</td>
<td>Service Creation in an Object-oriented Reuse Environment (RACE Project No. R2017)</td>
</tr>
<tr>
<td>SCP</td>
<td>Service Control Point</td>
</tr>
<tr>
<td>SCREEN</td>
<td>Service CREation Engineering Environment (ACTS Project No. AC227)</td>
</tr>
<tr>
<td>SDL</td>
<td>Specification and Description Language</td>
</tr>
</tbody>
</table>
### Appendix C: Acronyms and Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDP</td>
<td>Service Data Point</td>
</tr>
<tr>
<td>SF</td>
<td>Service Factory</td>
</tr>
<tr>
<td>SIB</td>
<td>Service Independent Building block</td>
</tr>
<tr>
<td>SII</td>
<td>Static Interface Invocation</td>
</tr>
<tr>
<td>SMS</td>
<td>Service Management System</td>
</tr>
<tr>
<td>SNMP</td>
<td>Simple Network Management Protocol</td>
</tr>
<tr>
<td>ss-UAP</td>
<td>service session User APplication</td>
</tr>
<tr>
<td>SSG</td>
<td>Service Session Graph</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Socket Layer</td>
</tr>
<tr>
<td>SSM</td>
<td>Service Session Manager</td>
</tr>
<tr>
<td>SSP</td>
<td>Service Switching Point</td>
</tr>
<tr>
<td>STA</td>
<td>Single Threaded Apartment model</td>
</tr>
<tr>
<td>TCon</td>
<td>Terminal Connection</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>Transmission Control Protocol / Internet Protocol</td>
</tr>
<tr>
<td>TCSM</td>
<td>Terminal Communication Session Manager</td>
</tr>
<tr>
<td>TINA-C</td>
<td>Telecommunications Information Networking Architecture Consortium</td>
</tr>
<tr>
<td>TMN</td>
<td>Telecommunication Management Network</td>
</tr>
<tr>
<td>TN</td>
<td>Transport Network</td>
</tr>
<tr>
<td>TOOLS</td>
<td>Technology of Object-Oriented Languages and Systems</td>
</tr>
<tr>
<td>TOSCA</td>
<td>TINA Open Service Creation Architecture (ACTS Project No. AC237)</td>
</tr>
<tr>
<td>UA</td>
<td>User Agent</td>
</tr>
<tr>
<td>UBMTS</td>
<td>Universal Broadband Mobile Telecommunication System</td>
</tr>
<tr>
<td>UD_AS</td>
<td>User Domain Access Session</td>
</tr>
<tr>
<td>UD_USS</td>
<td>User Domain Usage Service Session</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modelling Language</td>
</tr>
<tr>
<td>UMTS</td>
<td>Universal Mobile Telecommunication System</td>
</tr>
<tr>
<td>USCM</td>
<td>Universal Service Component Model</td>
</tr>
<tr>
<td>USM</td>
<td>User service Session Manager</td>
</tr>
<tr>
<td>USS</td>
<td>Usage Service Session</td>
</tr>
<tr>
<td>UUID</td>
<td>Universal Unique Identifier</td>
</tr>
<tr>
<td>VDU</td>
<td>Video Display Unit</td>
</tr>
<tr>
<td>VHE</td>
<td>Virtual Home Environment</td>
</tr>
<tr>
<td>VITAL</td>
<td>Validation of Integrated Telecommunication Architectures for the Long-term (ACTS Project No. AC187)</td>
</tr>
<tr>
<td>WWW</td>
<td>World Wide Web</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
</tbody>
</table>
Appendix D:

Publications

The following papers have been published on work related to the thesis:


Appendix D: Publications


