Co-operative Workplaces – Workspaces of the Future

vom
Fachbereich Informatik
der Technischen Universität Darmstadt

genehmigte

Dissertation

zur Erlangung des akademischen Grades eines
Doktor-Ingenieur (Dr.-Ing.)

von

Diplom-Informatiker (Univ.) Rolf J. Reinema

geboren am 11.08.1967 in Emden

Darmstädter Dissertation
Darmstadt 2002
Hochschulkennziffer D 17

Erstreferent:	Prof. Dr.-Ing. Ralf Steinmetz	(TU Darmstadt)
Koreferenten:	Prof. Dr. Kurt Geihs	(TU Berlin)
	Prof. Dr.-Ing. Heinz Thielmann	(TU Darmstadt)

Tag der Einreichung: 10. Januar 2002
Tag der Disputation: 08. April 2002

Lebenslauf

Persönliche Daten

Name: Rolf Reinema
Geburtsdatum: 11. August 1967
Geburtsort: Emden, Deutschland
Staatsangehörigkeit: deutsch

Anschrift (privat)
An der Pfeffermühle 11
D-64319 Pfungstadt
Germany

Tel.:+49 6157 911 711
Fax: +49 6157 911 713
Email:

Anschrift (beruflich)
Fraunhofer
Institut für Sichere Telekooperation
Rheinstrasse 75
D-64295 Darmstadt
Germany

+49 6151 869 358
+49 6151 869 224
rolf.reinema@sit.fraunhofer.de

Sprachen: Deutsch (Muttersprache), Englisch, Japanisch (Grundkenntnisse), Latein (gr. Latinum), Griechisch (Graecum)

Ausbildung

Schulbildung

1973-1976 Grundschule Emden-Larrelt
1976-1978 Orientierungsstufe Emden-Wybelsum
1978-1986 Johannes-Althusius Gymnasium, Emden,
Abschluß: Abitur

Studium

1986-1993 Studium an der Friedrich-Alexander-Universität Erlangen-Nürnberg, Fachrichtung Informatik
Studienschwerpunkte: Systemsimulation, Datenbanken, Künstliche Intelligenz, Fertigungstechnik
Studienarbeit: Lehrstuhl für Fertigungsautomatisierung und Produktionssystematik, Prof. Dr.-Ing. K. Feldmann; Thema: Einsatz von Techniken der Wissensverarbeitung zum Aufbau von Simulationsmodellen
Diplomarbeit: Lehrstuhl für Künstliche Intelligenz, Prof. Dr.-Ing. H. Stoyan;
Thema: PEDE-Lab - Aufbau und Entwicklung einer Experimentierumgebung für Multi-Agenten-Systeme
Abschluß: Diplom-Informatiker (Univ.)

Beruflicher Werdegang

08/1989 bis 07/1992: Studentische Hilfskraft am Lehrstuhl für Fertigungs-automatisierung und Produktionssystematik, Erlangen;
09/1992 bis 07/1993: Studentische Hilfskraft am Bayerischen Forschungszentrum für Wissensbasierte Systeme (FORWISS), Erlangen;
08/1993 bis 06/1995: Mitarbeiter der Firma mediatec – Gesellschaft für multimediale Systemlösungen mbH Nürnberg; zuletzt als Leiter des Aufgabengebietes Softwareentwicklung;
07/1995 bis 11/1999: Projektleiter am Institut für Telekooperationstechnik der GMD – Forschungszentrum Informationstechnik GmbH;
02/1999 bis 03/1999: Gastwissenschaftler Electrotechnical Laboratory (ETL), National Institute of Advanced Industrial Science and Technology (AIST), Tskuba, Japan; Stipendium der Japan Industrial Technology Association (JITAC), Tokyo, Japan;
Seit 11/1999: Forschungsbereichsleiter am Fraunhofer Institut für Sichere Telekooperation (vormals GMD Institut für Sichere Telekooperation der GMD – Forschungszentrum Informationstechnik GmbH)

If there is a way of doing it better, find it.
Thomas Alva Edison

Acknowledgements

This work could never have happened without the support and assistance of many people. Only a few of them can be mentioned here.

First of all, I would like to thank Prof. Dr.-Ing. Ralf Steinmetz for his supervision, invaluable advice, and encouragement. My special thanks go also to Prof. Dr.-Ing. Heinz Thielmann and Prof. Dr. Claudia Eckert for their guidance and the fruitful environment, in which this thesis could flourish during the past years. Many thanks to Prof. Dr. Kurt Geihs for acting as a co-reporter.

I would like to thank all those colleagues at the Fraunhofer Institute for Secure Telecooperation (FhG-SIT) in Darmstadt, who are taking care of the pleasant and friendly atmosphere without which the success of such a thesis is hardly impossible. I am also thankful to all those people, in particular the members of the COR research department, who have backed me up during the past few months while I was trying to finish this work.

This thesis has clearly benefited from developments and results in various projects I was involved in at FhG-SIT. Knut Bahr and Heinz-Jürgen Burkhardt do have a very special part in the success of this thesis, as they have stimulated this work and contributed valuable ideas, proposals, and comments throughout many fruitful discussions. I am especially thankful to Mario Hoffmann, Sven Türpe, and Ruben Wolf for the excellent co-operation.

Special thanks go also to the VOTEC and UNITE teams, as the have contributed to the success of this thesis with their constructive criticism and numerous discussions. Daniel Tietze deserves some thanks for the excellent and fruitful collaboration within the context of the Poliwork project and beyond. Many thanks to Michael Zapf and Klaus Herrmann for their great support with respect to their multi-agent development environment AMETAS, from which the Co-operation Platform has benefited. Thanks go to Thomas Kunkelmann for the joint work on scalable partial encryption methods and the implementation of a secure multimedia conferencing gateway.

Many thanks to all students who have contributed to the Co-operation Platform while I was supervising their master's thesis or looking after their practical trainings and internships. In place of all others, I would like to emphasize Thomas Blecher, Matthias Guckler, Jörg Hähner, Peter Hörmann, Max Larsson, Hüsnü Turkac and the software engineering teams Mind &Magic, Lynx, and Multiculture.

Special thanks go to Ludger and Volkmar Hovestadt, Gerd Bonnet, Stefan Steuerwald, Jupp Gauchel, and Eric Giese for the stimulating discussions on the architectural aspects of Co-operative Buildings and their share to the exciting research and development of the RoomComputer in particular.

I am thankful to Natalie Boehm for reading this thesis, correcting linguistic mistakes here a and there, and for making an effort to positively influence my style of writing.

Finally, special thanks go to Marie-Luise for supporting me all the time and in particular for catching all the silly mistakes that managed to slip. She deserves more thanks than anybody else does, because she is the pillar I can always rely on and she always treaded on my toes to get this work finished. Nevertheless, I have benefited from her well-founded knowledge in the area of IT-Security. This thesis is dedicated to her.

Table of Contents

Chapter 1 Introduction 1
1.1 Motivation 1
1.2 Vision 3
1.3 Objectives 4
1.4 Contributions 6
1.4.1 Virtual Project Office 7
1.4.2 Human Centred Communication Services 8
1.4.3 Physical Workspace Integration - The RoomComputer 9
1.4.4 Security 10
1.5 Organisation of the Thesis 10
Chapter 2 Co-operative Workplaces 12
2.1 New Ways of Working 12
2.2 Co-operative Workplaces 15
Chapter 3 Requirements 21
3.1 Document-Centric Work within Distributed Governmental Agencies 23
3.2 Networked Organisations 24
3.3 Requirements 25
3.3.1 Functionality 26
3.3.2 Usability 30
3.3.3 Reliability 33
3.3.4 Performance 33
3.3.5 Supportability 34
3.3.6 Design 34
3.3.7 Implementation 35
3.3.8 User Interface 35
3.3.9 Security 36
Chapter 4 Related Work 38
4.1 Virtual Workspaces 38
4.1.1 Multimedia Conferencing 38
4.1.2 Instant Messaging 39
4.1.3 Support for Document-based Collaboration 40
4.1.4 Object and Application Sharing 41
4.1.5 Virtual Collaborative Environments 43
4.1.6 Intelligent Personal Assistants 44
4.1.7 Ongoing Research Projects 44

4.2 Physical Workspaces 47
4.2.1 Intelligent Rooms 47
4.2.2 Location Awareness 49
4.3 Conclusion 50
Chapter 5 Approach 53
5.1 Virtual Work Environments 55
5.1.1 The Virtual Project Office 55
5.1.2 Human Centred Communication Services 57
5.1.3 Multimedia Conference Reservation System 58
5.1.4 Application Integration 60
5.1.5 User Interface 61
5.2 Physical Work Environments 61
5.2.1 CoMeet - The Cooperative Meeting Room 62
5.2.2 Location Awareness – The Sm@rtLibrary 65
5.2.3 World Wide Facility Management 66
5.2.4 Managing Physical Workspaces – The RoomComputer 69
5.3 Security 71
5.4 Semantic Model 72
5.4.1 The Notion of Work Context 73
5.4.2 Working within Work Contexts 75
5.4.3 Switching Work Contexts 77
5.4.4 Transfer between personal and teamwork contexts 80
5.4.5 Interfaces to personal and teamwork contexts 81
Chapter 6 Architecture 84
6.1 Architectural Goals 84
6.2 Architectural Approach 85
6.3 Integration Approach 88
6.3.1 Component technologies 90
6.3.2 Component interaction 98
6.4 Service-oriented Middleware 101
6.4.1 Service-oriented Middleware Solutions 103
6.4.2 Service Interaction 114
6.4.3 Conclusion 115
6.5 Agent Orientation 115
6.5.1 Agents 117
6.5.2 Mobile Agents for Internet-based System Structures 119
6.5.3 Agent Systems 120
6.5.4 Asynchronous MEssage Transfer Agent System (AMETAS) 123

6.5.5 Suitability of AMETAS 124
6.6 Transparent Data Access 128
6.7 Extensible Mark-up Language (XML) 129
6.8 Platform Architecture 130
6.8.1 General Platform Functions 132
6.8.2 Agent-based architecture 135
6.9 Virtual Project Office 141
6.10 Multimedia Conference Reservation System 143
6.10.1 Web-based Client 144
6.10.2 Reservation Server 145
6.10.3 Directory Service 147
6.10.4 MCU-Server 147
6.11 The RoomComputer 147
6.11.1 Bus Manager 149
6.11.2 Device Manager 150
6.11.3 Service Manager 150
6.11.4 RoomAgents 151
6.11.5 Infrastructure 151
6.11.6 Network Layer 152
6.11.7 Service Layer 153
6.12 Location Awareness 157
Chapter 7 User Interface 161
7.1 Abstract User Interface Description Language 162
7.1.1 User Interface Mark-up Language (UIML) 163
7.1.2 AWT/Swing 164
7.1.3 Java Beans 164
7.1.4 XSL Formatting Objects 165
7.2 Abstract User Interface Mark-up Language (AUI-ML) 166
7.2.1 Layout Elements 167
7.2.2 Abstract Widgets 169
7.2.3 Common Entities and Elements 171
7.2.4 Binding User Interface Elements to Method Calls 173
7.2.5 Links 175
7.2.6 Conditional Rendering 176
7.2.7 AUI-ML Renderer 177
7.2.8 AUI-ML Example 178
7.3 The RoomComputer 178
7.4 Virtual Project Office 179

Chapter 8 Security Aspects 183
8.1 Public Key Infrastructure 185
8.2 Office Identity Card 187
8.3 Confidentiality and Integrity 195
8.4 Non-repudiation and Auditing 198
8.5 Management of Security Policies 199
8.6 Location Management 201
8.7 Virtual Project Office 202
8.7.1 Authentication of Project Members 203
8.7.2 Confidentiality and Integrity of Communication Contents 204
8.7.3 Access Control 204
8.7.4 Attributability of Project Processes 205
8.8 RoomComputer 206
8.8.1 Prevention of External Attacks 207
8.8.2 Access Control 208
Chapter 9 Conclusions and Future Work 210
9.1 Co-operation Platform 210
9.2 Virtual Workspaces 212
9.3 Physical Workspaces 213
9.4 RoomComputer 213
9.5 User interface 215
9.6 Security 215
9.7 Scientific Evaluation 217
Bibliography 218
List of publications 242

Appendix A - Acronyms and Abbreviations 247

Appendix B - Terms and Definitions 255

Appendix C - Abstract User Interface Mark-up Language - DTD 263

Appendix D - AUI-ML Document for the SCLUB Example 270

Appendix E - Co-operation Platform – Software Packages 272

E.1 Co-operation Platform Packages – gmd.sit.vpo
(Co-operation Platform, VPO, Human Centred Communication Services) 273

E.2 Co-operation Platform Packages – com.mcrs
(Multimedia Conference Reservation and Management System) 276

E.3 Co-operation Platform Packages – de.gmd.de.darmstadt.dia
(Office Identification Card) 278

E.4 Co-operation Platform Packages – com.raumcomputer
(RoomComputer) 280

E.5 Co-operation Platform Packages – congo
(Media Control System for the CoMeet Room) 282

E.6 Co-operation Platform Packages – smartlibrary
(Sm@rtLibrary) 286

List of Figures

Figure 1 Symbiosis between virtual and physical workspaces 22
Figure 2 Sample VPO user interface 60
Figure 3 Sample SCLUB office layout 61
Figure 4 CoMeet room 62
Figure 5 CoMeet room media control system 63
Figure 6 Sm@rtLibrary 65
Figure 7 Sm@rtAssistant user interfaces 65
Figure 8 RoomComputer 69
Figure 9 Sample RoomComputer installation 70
Figure 10 Co-operation Platform work context components 73
Figure 11 Overlapping of personal work contexts and teamwork contexts 74
Figure 12 Work context areas 74
Figure 13 Meaning of work context areas 75
Figure 14 Automaton describing work context switching for one user 78
Figure 15 Product Net describing the work context switching for all users 79
Figure 16 Local visibility of global states S_0, S_1 82
Figure 17 Local visibility of global states S_2, S_3 82
Figure 18 Local visibility of global states S_4, S_5 83
Figure 19 Co-operation Platform – layered architecture 88
Figure 20 AMETAS platform 123
Figure 21 Data Access Layer (DAL) 128
Figure 22 High-level architecture of the Co-operation Platform 132
Figure 23 Agent-based architecture of the Co-operation Platform 137
Figure 24 Client-side architecture 139
Figure 25 Server-side architecture 140
Figure 26 VPO communication and conferencing subsystem 142
Figure 27 MCRS architecture 143
Figure 28 MCRS user interface 144
Figure 29 Reservation server 146
Figure 30 RoomComputer architecture 149
Figure 31 RoomComputer devices 151
Figure 32 RoomComputer gateway 152
Figure 33 RoomComputer as an Open Service Gateway 154
Figure 34 Remote configuration 155
Figure 35 Sm@rtLibrary components 157
Figure 36 Sm@rtLabel, RFID reader, and Sm@rtshelf (from left to right) 158

Figure 37 Sm@rtLibrary architecture 159
Figure 38 AUI-ML based user interface rendering 162
Figure 39 Overview of the AUI-ML DTD elements 167
Figure 40 Sample layouts 169
Figure 41 Container example 171
Figure 42 AUI-ML Servlet 177
Figure 43 AUI-ML-based user interface 178
Figure 44 RoomComputer user interface 179
Figure 45 VPO user interface - 2D version 180
Figure 46 VPO user interface – 3D version 181
Figure 47 Security functionality and dependencies 185
Figure 48 Security infrastructure of the Co-operation Platform 187
Figure 49 Persons, roles, and authorisations 191
Figure 50 Delegating authorisations 191
Figure 51 User interfaces of the OIC management system 192
Figure 52 Structure of the OIC file system 193
Figure 53 Architecture of the OIC system 194
Figure 54 Secure conferencing gateway 197
Figure 55 Different security policies 199
Figure 56 Security domains and meta-policies 200
Figure 57 Location management 201
Figure 58 Access control to services and documents within a VPO 205
Figure 59 Extended access control model 216

List of Tables

Table 1 Container element 168
Table 2 Textinput element 169
Table 3 Slider element 170
Table 4 Setselection element 170
Table 5 Output element 170
Table 6 Borderstyle 171
Table 7 Textstyle 172
Table 8 NameId 172
Table 9 Value element 172
Table 10 Javatypes 173
Table 11 Activator element 173
Table 12 Call element 174
Table 13 Constant arguments 174
Table 14 Widget arguments 174
Table 15 Call arguments 174
Table 16 Link element 175
Table 17 URL element 175
Table 18 Device element 175
Table 19 Chose element 176
Table 20 Case element 176
Table 21 Example of a role space 189

Chapter 1

Introduction

1.1 Motivation

There currently exists a fairly well developed awareness that today's organizations are subject to increasingly rapid and significant processes of metamorphosis. The world today is in a transition phase whereby the models underlying business, communication, education, working, and everyday life are changing. Many organizations are no longer strictly hierarchically structured, but go more for vertical integration, networked clusters, project groups, and flattened hierarchies.

Dramatic changes in job situations and working conditions are occurring in many areas. In the future, individual and team-based work in organizations will be characterized by a degree of flexibility and dynamics that will go far beyond many of today's developments and examples. On demand and ad-hoc teams, virtual organizations, distributed and mobile workers are only initial examples of organizational innovations to be expected in the future.

In the new technology society, there is a need to form alliances; there is a high competition for skilled workers, and this creates a contention between organizations and high level of fluctuation in the workforce. This problem originates from the fact that in some societies, organizations were formed to provide employees with lifetime employment and there was a stable linkage between the employee and their employers. In the new society, workers move frequently from one employer to another and from one company to another. Companies need to form alliances to be able to create new technologies. They are international and work around the globe providing international solutions. The world is becoming “a global village” in the technology sense as well as in the social and working habits. This new behavioural change presents a need for a new organization model as well as a new workplace model.

Teams have become an increasingly important part of business success [KS93]. One of the most exciting new challenges in modern business life is the need to collaborate

within teams, which are geographically distributed across the country or the world. Teams will affect our daily activities, our workspaces and will change our social interactions. The creation of so-called virtual teams of members who are geographically dispersed but electronically connected has already set many companies on fire [LS97]. In spite of all euphoria, team-based organizations are looking for workspaces that enable and support team behaviour and processes. Teams ask for a new range of tools and support, providing more flexibility in adapting dynamically to special needs. Virtual teams face greater obstacles and ask for more flexibility and information technology support than traditional teams do. New paradigms, technologies, information systems and communication infrastructures are required to provide virtual teams with the necessary means in order to strengthen the power of an enterprise. Much more research needs to be done on how to link people and teams by technology to enhance collaboration and increase productivity.

There is an increasing trend to dynamically form project teams in order to gain more flexibility in reacting to rapidly changing situations in the working world. Such teams are tailored to the task in hand and often consist of people from distinct organisational units. They are specially assembled for a project, coordinated, adapted where necessary and broken up at the end of the project or re-formed for another task. In the form of virtual organisations based on modern information and communication technologies, experts, who are scattered over different or frequently changing locations, collaborate intra- or inter-organisationally, nationally or internationally, fast, cost effectively and securely. Their information and communication infrastructure needs to be progressed towards a collaboration infrastructure.

On the technological side, we are experiencing the convergence of data, voice, wired, and wireless networks; the convergence of physical and virtual places; convergence of applications and telecommunication services; convergence of the home and the workplace, and others. So, solutions are needed to assist this transition, help the convergence, and allow the migration towards new forms of collaborative work and new tools and technologies, enabling them.

The role of information and communication technology in these developments is manifold. On the one hand, it initiates and triggers new forms of communication and cooperation processes. On the other hand, it provides the infrastructure and application software needed to meet the requirements, which result from new ideas on organising work practices. Colleagues, contents as well as contexts, processes and structures will be changing frequently. It is time for these developments to be reflected in the design of equally flexible and dynamic workspaces.

Solutions, which exist today, are essentially centred on individual work and on widely isolated tools and services. To support teams, particularly geographically dispersed ones, they have to evolve towards collaboration infrastructures, which are dedicated to teamwork within virtual organizations. This does not call for the development of very new technologies, but for the integration of available technologies and a functional enrichment to support context-oriented collaboration under unified user interfaces, which follow an intuitively comprehensible metaphor.

The basic question to be answered is: How will we be able in the future to dynamically construct workspaces with distributed people, resources, real and virtual rooms and customize them for a given task or project?

This thesis acknowledges the growing need for collaborative work environments supporting teams working on specific projects, in particular when team members are located at different and, over time, changing places, possibly in different time zones. It is essential that such environments allow team members to switch between different projects, make it easy for them to preserve and restore the work context of any given project, allows them to reserve and tailor physical workspaces as team members move and change places [DM92], and last but not least be sufficiently secured.

The following three key observations are being taken into account:

- Different people (and organisations) typically work with different local tools and services, be it for reasons of culture, installed base, or legacy. For them to collaborate, it is necessary to hide the specifics of e.g. communication or presentation tools under a more unified and intuitive user interface, and at the same time provide built-in adaptability to and transcoding between basic services, e.g. to achieve unified messaging.
- There already exist a number of collaboration tools and group-ware services and more will emerge. The answer of this thesis is to use them as a reservoir of building blocks and to encapsulate them in a particular unified shell so that they can be integrated into the Co-operation Platform for use of higher-level services in terms of a project's application domain.
- Many physical work places already tend to be non-territorial; team members use them according to room availability. Thus, facility management services need to be made available across networks, to administer physical rooms and their interiors. That makes it possible to remotely reserve physical workspaces, to personalise and restore their equipment and furnishings, to account for usage, to track and locate people and resources.

It will be shown, that today’s computer-based technology together with efficient communications has the potential for superior solutions that maintain the information base needed by the team and the relations among team members, while overcoming the physical barriers among them [RBB+98], [BBH+99] [Schi98].

1.2 Vision

The vision that is carried with this thesis is the paradigm of so-called Co-operative Workplaces. They serve as an environment for high performance of ever more geographically dispersed teams and enable people to migrate from project to project in a highly flexible working environment. There they are able to focus on the context of their work and are being freed from locations and time zones. People are the most expensive and valued resource in today's economy; thus, the need to use them in the most efficient way and to utilize their expertise and skills as well as enabling them to work from any

location and participate in meetings either physically or virtually is critical to the future business world.

Co-operative Workplaces provide shared virtual workspaces, customised to support teamwork, and seamlessly integrated with the physical workspaces where team members perform their work, whether on companies, premises, at home or on the move.

Just like traditional physical workspaces, virtual workspaces can be structured into areas designated for specific activities: individual offices, meeting rooms, conference rooms, show rooms, boards, and a library of the documents shared by the team. In the virtual workspace, team members can switch activities by moving from one area to the other as naturally as in the traditional workspace. Team members are informed of the presence of their colleagues in the virtual workspace, and they can easily communicate among each other in real-time when in the same area such as an individual office area or a meeting area, without the burden of managing their communications themselves through telephones, faxes, e-mail, or videoconference studios. Team members can access the team repository, which holds shared resources such as e.g. documents [Ven91] [GW91] used and produced by the team. They can share boards and other posting areas without the necessity to physically move from their work location [RBB+00].

As a smooth and effective working environment, the Co-operative Workplace promises to be cost-effective, to be rapidly set up and tailored to the needs of a specific team [Sie98], and to be dynamically adapted to possible changes of the project. This concept primarily applies to project work within or across enterprises, and is scalable from small and medium size enterprises (SMEs) to large organisations. It can also easily extend to call centres with employees alternating between working at home or on their company premises, insurance companies with employees on the move, customer care centres, education centres, or exhibition centres.

This thesis addresses any collaboration, which is characterized by a group of geographically distributed people teaming up to work on a specific project for a fixed time. Their work environments consist of both physical and virtual workspaces. The main topics covered are: creating so-called Virtual Project Offices, accessing physical locations from them, customizing such collaboration environments to specific projects, running them within a given project and (re)configuring them dynamically as need arises.

1.3 Objectives

This thesis presents research results on the potentials of Co-operative Workplaces, which are being seen an alternative to traditional, work-sharing organizations, which are not well-suited to reacting dynamically, flexibly, fast and economically enough to constantly changing customer demands. Since after all we as physical beings are dependent on physical rooms to live and work in, a sensible symbiosis of physical and virtual workspaces is being advocated, to make the best use of information and communication technology advancements. The objective is to design and develop a framework for such

future workplaces by bringing together information and communication technology, architecture, design and management of buildings, and new flexible forms of individual and joint work.

The overall objective of this thesis can be summarized as follows:

This thesis develops the design of an architecture together with a prototype serving as a proof of concept for an open, teamwork-oriented, and unified Co-operation Platform. It focuses on users, their work contexts, and usage scenarios by providing a unified and worldwide accessible platform based on widespread open standards and Internet technologies. The Co-operation Platform and its components have been prototypically implemented in order to support the creation of Co-operative Workplaces, which can be flexibly built, customised and run according to user needs. They bind dynamically, project information, team members, their physical workspaces, and information tools in a context that is pervasive, secure, and transparent.

The architecture and framework of the Co-operation Platform together with the underlying concepts and the implemented prototype, serving as a proof of concept, are key results of this thesis. The Co-operation Platform provides the facilities for devices, components, and networks to fully interact, despite the possible original inherent heterogeneity. As such, it takes care that a uniform and ubiquitous view is presented to all team members regardless of their physical location. System requirements and an architecture for the seamless integration of virtual and physical workspaces will be addressed. It is built on technology convergence among communication and collaboration media as well as different network technologies. It exploits the ability of appliances to notify their existence and characteristics to the workspace environment. Rather than aiming at a collection of tools and services in the abstract, it is focusing on supporting the context of work of its users. The objective is to provide virtual workspace that can be adapted as a project moves along. Therefore, the Co-operation Platform offers a set of configurable basic building blocks and services with which Co-operative Workplaces can be flexibly built and customized to user needs. Developing such a platform is more effective than developing a new monolithic system. It will provide means and ends to incorporate well-established tools and services from an installed base into a value-added project support environment.

Furthermore, this thesis pursues a strategy aimed at integrating resources, tools, and services. To this end, an architecture and platform will be developed which then allows to construct customised workspaces out of tools that are, for the most part, already available. In addition, this involves taking first steps toward the integration of real and virtual rooms. Thirdly, this thesis aims at developing a new way of re-presenting co-operative workspaces on user interfaces, in particular on computer screens. With respect to this representation, the focus, again, is on the context of work and the structure of projects and teamwork rather than the set of tools available for work. The aim is to provide comprehensive and ubiquitous co-operation platforms, built from a set of (sometimes) heterogeneous basic tools and services, allowing users to easily switch between activities and projects, developing and dissolving with the progress of a project and, finally, integrating the management of physical workspaces and resources.

The Virtual Project Office (VPO) forms the common working base in virtual workspaces for distributed members of a team within a virtual organisation. The term Virtual Project Office is a metaphor referring to the room as a place where people meet and work together on a common subject. It presents to its users a context oriented view of the contents of work and the people involved, and supplies the tools, information objects, collaboration and communication services needed to support team formation and team collaboration as well as communication with external partners.

In the Virtual Project Office, real-time media rich interaction is one of the key factors to running it successfully. Real-time interaction must be carried out at any place and any time by using any kind of media so that both visual and audio communication will be able to complement and to some degree replace real office interactions. Therefore, the easy-to-use provision of communication channels between the people involved in such collaborative work processes, at any time and at any place, is specifically addressed by this thesis. The major objective is to provide the framework of so-called human centred communication services to the users, which means that they may use different networks and protocols but still join in the same communicative session, without the need to deal with the different underlying communication protocols or network infrastructure. Human centred communication will support the ability to establish connections between end-points that use different communication protocols as well as blend different media streams. Since collaborative processes often span multiple sites, multi-party distributed communication is necessary. Furthermore, by providing a unified way of communication for all types of real-time multimedia interaction, it will be possible for a developer to develop applications for the Virtual Project Office (VPO) with the ability to focus on a specific application domain rather than on communication protocols or underlying network infrastructures. By this, the VPO will become flexible and work over various disciplines, will use any available communication channel and enable media rich real-time interaction within geographical dispersed teams.

Another, equally important objective is to provide human centred, natural, and intuitive user interfaces to the people who are supposed to work in any of the envisioned new work environments. Therefore, the familiar ways people meet and orientate themselves in physical rooms will be taken as a model for a new general metaphor applicable to Co-operative Workplaces as a combination of real and virtual workspaces. Here too, a mechanism for assembling interfaces from basic building blocks will be devised, thus allowing as much commonality as possible among different application cases.

Work on these principal objectives is detailed in several basic building blocks, each of which deals with a specific aspect of Co-operative Workplaces. In the following, some of the main ideas and key contributions will be introduced.

1.4 Contributions

The key contribution by this thesis is the architecture and prototypical implementation of a Co-operation Platform that supports tele-collaboration among members of a team, communication between teams and their outside world, as well as dynamic formation of

inter-organizational teams. The envisaged area of tele-collaboration is the one, which is typically found in projects and tasks, which have to be completed in a given period of time, and which are based on the division of production and cannot be precisely pre-planned but require continual coordination. The platform architecture and specifications defines and structures the interactions among the platform functional components that co-operate to provide the Co-operation Platform services. The overall architecture identifies the core components together with their interfaces, and defines the security architecture.

In contrast with the architecture of CSCW platforms available on the market, the architecture of the Co-operation Platform aims to feature openness to third-party components while contributing to the establishment of an open interoperable standard that supports sharing of collaborative work contexts within geographically dispersed teams. Another key aspect is the provision of supporting methods and examples of software modules, for the implementation of the reference architecture. This includes flexible methodologies for the mapping of the software modules onto the Co-operation Platform.

The developed prototype offers the essence of the Co-operation Platform as a proof of concept of the underlying ideas, overall framework, and architecture. It forms the basis to conduct user trials, and based on their results, to consolidate the requirements and to evaluate both the paradigm of Co-operative Workplaces and the platform architecture. The platform is an excellent generic basis on which domain-specific enhancements can be built.

1.4.1 Virtual Project Office

The Virtual Project Office (VPO) is an application that addresses collaboration of distributed teams, sets up a virtual workspace for that purpose, and is structured like a real team office. This preserves some of the essential advantages of a local shared office, such as transparency, ease of interaction, availability of project data etc. Team members are projected into the VPO (and possible represented by avatars) as if they were all physically present in a single office. By including a project's input and output, its resources, tools and services, the VPO captures the context of the project. The key objective of the VPO is to support the dynamic creation of teams together with configuration and customisation of their workspace.

A VPO offers different so-called work zones, each supporting a different type of work and work context; for example zones for individual work and zones for collaborative work such as conference sessions or informal information exchange are provided. Anyone "in office" is shown in a zone depending on what s/he currently does. Users can easily switch between different work zones and therefore work contexts. They can thus get in touch with a colleague, enter the conference zone, or go to the project file zone; simply by going there, and without having to select or start the proper phone, conference or work space tool. They can communicate with their colleague, participate in the conference session or inspect and update files, respectively.

The VPO presents its users a context oriented view of its contents and members. This is achieved by importing the necessary basic tools, services, and resources into the Co-operation Platform, relating them to each other where meaningful, embedding them in the

project context, surrounding them with a context oriented user interface, and thus creating value-added project services. Phone and fax calls in a VPO for example, would only be those related to this project, conference sessions would automatically be tied to the context of project data and cost, any collaboration input and output would continually be matched against the project database, cost file, security rules etc.

1.4.2 Human Centred Communication Services

To support teams and in particular geographically dispersed teams, current information and communication infrastructures have to be evolved towards collaboration infrastructures, which are dedicated to teamwork within virtual organizations. Multimedia conferencing is a powerful technology, which enables people to collaborate in ways never before possible. While multimedia conferencing solutions provides many important features, easy usage and control of this technology by the end user is currently still a big problem. Additionally, administrating the necessary infrastructure, setting up and scheduling conferences as well as controlling them can be very difficult, time consuming, and costly [Sha97].

Therefore, this thesis presents an enhanced approach to multimedia conference reservation, which integrates conference scheduling and the reservation of resources necessary to perform a multi-point multimedia conference. It provides a powerful extension in a way that allows even end-users to schedule, manage, and control their multimedia conferences without any need for additional help (e.g. by an operator). This speeds up the scheduling process, allows spontaneous and ad-hoc collaboration and reduces the likelihood of scheduling errors. All the steps required to set-up the necessary communication infrastructure are automated in a way that multimedia collaboration between distributed users and groups of users can be just a mouse-click away.

The human centred communication services are being made available by a communication middleware, which provides a generic common implementation and dynamically loading of different services. This middleware provides a low-level interface for the different stacks as plug-ins and presents a single interface to the applications. Currently, the following features are being provided:

- People Awareness – every team member within the Co-operation Platform needs to be recognised and identified by a unique identification scheme. People want/need to be aware about the presence of e.g. their teammates. The middleware of the Co-operation Platform provides an identification scheme that uses a common registration/identification tool shared by all components of the Co-operation Platform.
- Device Awareness – every device within the Co-operation Platform needs to be recognised and identified by its address and capabilities using a common registry managing the specific capabilities of each device. The middleware of the Co-operation Platform is able to recognize the available resources and dynamically loads and uses the appropriate protocols.
- Hybrid Conferencing –multi-party conferencing needs to be supported within the Co-operation Platform, where participants may use different networks and

protocols but still join the same conference. Hybrid conferencing supports the ability to establish connections between end-points that use different communication protocols as well as blend different media streams.

- Capability Exchange – is based on the devices and tools available to users within their physical workspaces, who are participating in virtual workspaces. The middleware of the Co-operation Platform provides a mechanism to exchange capabilities between users so that resolution on available and preferred media channels, bandwidth, compression algorithms or other communication dependent capabilities are being resolved by the middleware.

1.4.3 Physical Workspace Integration - The RoomComputer

One of the key concepts for achieving the symbiosis between virtual and physical workspaces is the development of an innovative device, called RoomComputer. It serves to bridge the gap between physical and virtual workspaces. Its development is motivated by the realisation that modern forms of work and organisation require frequent and dynamic reconfiguration of physical workspaces and easy adaptation of rooms to varying requirements of users and their work contexts.

Present-day office buildings often house a larger number of companies and organisations and thus must be capable of adapting the floor plan, technical facilities, and organisation to changing uses and changing tenants on short notice and at no great expenses. With the RoomComputer, it is possible to equip physical rooms with the necessary intelligence to give access to the full set of their services.

Due to increasing economy pressure, the use of office space becomes more diverse and intense in the future. This makes it necessary that buildings take over services, which are presently rendered by traditional facility management systems, information systems, and various specialists. For the building and its management, this means to break with the traditional services and technologies. Hence, based on the RoomComputer integrated information and building management systems can be built, which will provide efficient and productive access and usage of the facilities within the provided physical workspaces. The RoomComputer allows these services be offered to users in an integrated and intuitively usable form.

The RoomComputer is an embedded system and as such offers unprecedented chances to administer distributed physical workspaces of people from virtual ones and vice versa. It is an autonomous installation unit consisting of a tiny embedded PC to which with a touch sensitive LCD-display as local interface, loud speakers, microphone, camera, smart card reader, and an Intranet/Internet connection can be attached. RoomComputers can be easily installed in any room in order to give access to a full set of services for that room. With the introduction of technologies such as e.g. JINI (JAVA Intelligent Network Infrastructure), the development of a distributed system like the RoomComputer and therefore the usage of interconnected embedded devices and ubiquitous services became more simplified.

1.4.4 Security

Integrated security features within the Co-operation Platform provide means for the secure identification and authentication of people; e.g., access rights to Co-operative Workplaces can be defined by means of role based access control. A flexible spectrum of security services makes it possible to tailor Co-operative Workplaces so that they can assure confidentiality for commercial reasons and guarantee individual privacy, as needed. For example, security policies for Virtual Project Offices would specify their appearance to the outside world, access to them, and rules for signatures and confidential documents. Much of this is based on existing security tools, such as e.g. smartcards and public key cryptography. Security functions will be made available to provide confidentiality, authentication, and security profiles, to define responsibilities, rights, and roles within a project and in the end to allow contracting of experts for a team over the network. One of the key contributions within this thesis with respect to security is the Office Identification Card (OIC).

1.5 Organisation of the Thesis

This remainder of this thesis is organised as follows.

Chapter 2 elaborates on a new paradigm for computerised and networked collaboration, the Co-operative Workplace. It introduces the Co-operation Platform, developed within the context of this thesis, as a unified, integrated, and open platform aiming at providing Co-operative Workplaces and therefore the collaborative work within geographically dispersed teams.

In chapter 3, the most important requirements of future team and project-oriented teamwork are being identified and documented. The requirements analysis is based on the user requirements gathered in two user communities. Requirements are categorised in accordance to the FURPS+ model [Gra92] [IEEE-610.12-1990].

Chapter 4 gives an overview on the state-of-the-art in this field and tries to explain why some of the existing solutions have not succeeded or not accepted by users. Reasons are given why it is necessary to find a new approach for future work environments.

By introducing new metaphors, an integrated approach to establish a closer relationship between modern information and communication technology, physical workspaces and new flexible forms of work organization is being followed. An overview of the approach being followed is given in chapter 5. Additionally, chapter 5 introduces the various components developed within the course of this thesis together with the Semantic Model of the Co-operation Platform, a formal model on which the design and architecture of the platform are being based on.

Based on this approach, chapter 6 presents a suitable architectural approach for Co-operative Workplaces, realised by the architecture of the Co-operation Platform, which addresses the creation, operation, administration, and disposal of Co-operative Workplaces. As the overall objective of this thesis is to prototypically implement a framework of the Cooperation Platform, ready for trials and further evaluation involving

real users and teams in the course of their daily work, different facets of the platform architecture are being examined. Key interfaces, including those to users, applications, tools, building blocks, and physical workspaces are being identified.

Because of their heterogeneous nature, Co-operative Workplaces have to hide the underlying complexity and present a unified and easy-to-use user interface. The user should be able to access the provided functionality from virtually any device. For that purpose an abstract user interface mark-up language has been developed, which is presented in chapter 7. Chapter 7 also elaborates on a room based user interface metaphor for virtual workspaces.

Besides the simple intuitive handling of the user interfaces, security is decisive for the success of the Co-operation Platform. As the objective of the Co-operation Platform is to provide a platform, which allows the integration of different resources, collaboration and communication services in a heterogeneous environment into an Internet-based system, this constitutes a special challenge for a security architecture. Chapter 8 addresses the security objectives, which have been addressed by this thesis in connection with the services offered by the Co-operation Platform

A conclusion of the achieved results and an outlook to future work is given in chapter 9.

Chapter 2

Co-operative Workplaces

2.1 New Ways of Working

The advent of modern information and communication technology has already changed work processes and the contents and routines encompassed by the employee's normal duties significantly. Traditional work is increasingly being carried out at any place and at any time while offices are increasingly being regarded primarily as places for face-to-face exchange. In many locations, workplaces are no longer allocated to a specific person but are flexibly assigned as required. There is an increased focus on teamwork worldwide and societies without a previously well-developed teamwork ethic are joining this trend. For this reason, in addition to the emergence of mobile and tele-working forms of occupation, offices will continue to gain in importance as meeting points. Yet, in spite of these developments, the design of workplaces, especially of physical workplaces in offices and buildings, has remained almost unchanged. Present workplaces are not very well prepared for supporting teamwork-oriented activities nor are they well equipped for integrating the IT-infrastructure needed for work processes with the infrastructure needed for managing and operating the building (facility management).

Current workplaces still reflect the pyramidal organizational structure established during the industrial revolution of the 19th century [Ste98]. These have not significantly evolved over the last 50 years, but have primarily been simply embellished through the addition of technological devices, such as telephones, computers, or videoconferencing systems. However, the overall concept and architecture of the office have never been thoroughly revisited to consider how to accommodate the needs of modern organizational structures of work (such as networked structures, projects, and virtual organizations), to address increased employees' responsibility and diversity of tasks, and to fully reap the benefits of new technologies. The results are impairments with respect to the global economy, in particular: reducing production cycles, coping with high competition for skilled people around the globe, and fulfilling the need of enterprises to form cross-

boundary, temporary alliances for the duration of a project, no matter where they are geographically located.

The only changes in the working environment that have occurred have resulted from the need of telecommuters for home offices. Today, people create home offices by purchasing a pile of technologies that are either centred on a computer, e.g. printers, scanners, multimedia equipment; or centred on communication lines such as phone, fax, modems, and pagers. The only capability that these technology collections are capable of providing is a connectivity capability, either to the office network or to a service provider. This connectivity enables workers to communicate using several types of communication software and networks, from dial-up networking, through ATM, Cable access or xDSL. More advanced technologies can be purchased which then provide collaboration and videoconference capabilities over data networks. However, these tools are usually used in an ad-hoc fashion, and are not well integrated with the office services.

To appropriately implement new methods of work, a holistic environment is needed, which can fully support these methods effectively while giving people at work the resources that they need to perform their duties, and convenient means for interacting with each other [Fai98]. Today, the deficiencies of the traditional office environment vis-à-vis the working structures divert people from their real work. A significant amount of time (a scarce and expensive resource) is spent bridging the gap arising from inadequacies of the work environment with respect to individual's needs. While technology could reliably satisfy many of these needs, there continues to be a large overhead for, e.g., travelling, managing, and configuring various resources. So, the traditional workplace must be revolutionized by making available integrated facilities that are appropriate to the work context at any given time. Such an achievement requires a thorough analysis of working needs, a matching and powerful model of relevant workplaces, and the application of this model in the form of well-suited technological solutions.

To facilitate the evolution of new forms of work and organizational structures at a higher level of quality, it is necessary to integrate modern information and communication technology (hardware, software, networks) with innovative spatial and physical structures. Challenges that architectural design must address include more flexible and adaptable building structures, dynamic reconfiguration of offices and work places, computer-based models for planning, operating, and lifetime maintenance of buildings.

New information and communications systems also allow new directions of entrepreneurial activities. Traditional, work-sharing organisations are not suitable for reacting dynamically, flexibly, quickly and economically enough for dealing adequately with permanently changing customer demands. New paradigms of work organization ask for room structures and furnishings which meet the requirements of teams and which contain dynamically re-configurable teamwork environments, individual workplaces, conference rooms, and communication islands.

So far, technology dominates developing workspaces and work organizations, affects buildings, and puts new demands on people. Among the technological areas that need more advanced solutions are:

- computer supported cooperative work environments,
- Internet/Intranet-based groupware tools,
- media rich communication support for distributed teams by audio, video- and data-communication,
- meeting support for virtual teams, and
- mobile work.

In addition, there are many other important factors, which have to be considered in developing future workspaces. Some of the organizational issues are:

- driving the transition to virtual team-based structures,
- managing virtual project-teams,
- facilitating distributed team interaction,
- managing and supporting computer-supported collaborative work,
- implications of new models of flexible work organization, and
- dynamic sharing of resources, such as e.g. office spaces.

With respect to the physical workspaces, issues, which need to be considered, do include:

- intelligent buildings with flexible building structures and technical systems,
- integrated facility management,
- flexible and ergonomic interior design, and
- sustainable concepts for urban design and architecture.

Last but not least, social impacts have to be carefully investigated, for example:

- increasing demands on private time and space,
- the blurring of boundaries between when and where work ends and play begins, and
- integration and coordination of distributed work across boundaries.

In a future networked and distributed organisation, persons will be involved in different contexts of work. While these will intertwine to some extent, it is useful to differentiate between three major contexts:

- individual work
- project related work within teams
- work related to organisational units and social networks

Individual work refers to all kinds of work being done in order to organise and secure one's capacity to work. This may comprise, for example, maintaining one's

communication infrastructure (e.g., communication software and one's address book), organising one's time and commitments with respect to the different tasks one has to perform or taking courses in order to learn about new developments and methods that one may be expected to use in a future project.

Project related teamwork is another such basic work context. This refers to all the work being performed as a member of a particular project team. More often, this involves team members being geographically distributed and originating from different organisations.

Work related to organisational units and social networks serves securing one's membership in these social entities. For example, a freelance quality management consultant may feel the need to be integrated in a network of peers pursuing the same business (e.g. by participating in joint meetings and on-line discussion groups) in order to be aware of the latest developments in this particular business. Similarly, a member of an organisation or organisational unit may be involved in activities such as the maintenance of a research lab that is used by all people belonging to that particular unit.

2.2 Co-operative Workplaces

The availability of a paradigm supporting these needs has become vital for companies that wish to adapt their operations and resources frequently and efficiently to ever changing environments.

One of the objectives of this thesis is to define a new paradigm, namely the Co-operative Workplace, aiming at supporting communication and collaboration in distributed project teams and to put it into practice by developing a Co-operation Platform. The term 'paradigm', as used in this thesis, refers to an outstandingly clear or typical example of a particular class of items, i.e. an archetype. The approach taken is distinctive in several ways:

- the focus is on the structural properties of project work done within teams an therefore on work contexts,
- resources (including physical ones, such as e.g. rooms and devices), tools and services will be seamlessly integrated,
- Co-operative Workplaces are ubiquitous in the sense that the services they are offering are accessible from virtually anywhere, at any time, and by using any device, although not every service might be accessible from everywhere.

The ultimate goal is to strive for a comprehensive Co-operation Platform, which is based on basic building blocks and services provided by heterogeneous tools. Such a Co-operative Workplaces:

- develops and dissolves with the progress of a project,
- is ubiquitous, i.e. accessible from virtually anywhere,
- is integrated with respect to

        - different contexts of work
        - virtual and physical workspaces
        - established and emerging tools;
    - allows for awareness with respect to
        - the progress of project related work
        - the current activities of other project members
        - the availability of other project members;
    - allows for adaptation and personalization not only through a personalized portal but also through
        - the arrangement of tools and resources, and
        - opportunities to adapt the look and feel of the workspace.

Thereby, allowing for the creation of an emotional context or ambience particular to any project or personal work environment.

The paradigm of Co-operative Workplaces comprises

- a shared virtual workspaces, customised to teamwork,
- a set of physical workspaces, where team members perform their work, and
- the dynamic and seamless integration of the both.

The virtual workspace is where co-operation among project team members occur as a whole. The physical workspace is where individual or group work physically takes place. Hence, there is exactly one virtual workspace per project while physical workspaces may be assigned to different projects at different times.

Virtual workspaces offer virtual work environments as the common base for dispersed as well as local project team members. They focus on capturing the project context, and on offering human centred collaborative user interfaces with embedded security. They support the dynamic creation of teams together with configuration and customisation of their work environments. Conceptually, each project has a virtual workspace of its own, and anyone who is a member of several projects would go from one virtual workspace to another as he or she goes to work from one project to another. The ability to dynamically create virtual workspaces becomes an important factor in rapidly creating virtual organisations and facilitating their operations in cyberspace. A virtual workspace is actually the archetype of a virtual enterprise.

The user interface of virtual workspaces may adopt a room metaphor that represents activity-specific areas, by analogy with a real office, where space is distributed according to activities such as individual work, joint work, conference sessions, presentations, informal exchange, and others. In such spaces or so-called work-zones, team-members do share their project resources like for example the project schedule and current status, the team directory, the project and resource calendars, documents, and organisational directives.

Anyone who enters the virtual workspace can wander around and, for example, approach a colleague working at his or her desk, walk up to the project calendar, or enter the conference area. Automatically, and without the need to explicitly open a software tool, he or she could talk to a colleague, inspect documents, make entries, or participate in a conference session. The required communication and the use of processing tools, together with security measures for access control and authorisation would be implicitly initiated and terminated upon entering and leaving specific areas.

Group-aware repositories and collaboration services are being utilised for capturing and maintaining a project's work context. Personal agents automatically authorise and log in project team members from wherever they are. Security policies, e.g. specifying access to a virtual workspace, its appearance to the outside world, or rules for signatures and confidential documents, rely on existing security tools such as smartcards and public key infrastructure. General-purpose tools, such as phone, e-mail, conference systems, databases, and project management software are made project-specific constituents of a virtual workspace. Effectively, the virtual workspace works like a well-organised and well-informed project secretariat.

Project team members are working as individuals and from time to time as groups or subgroups in physical workspaces with a set of stationary or mobile physical devices. Since physical workspaces tend to be non-territorial, which means that they are used as common resources at different times by different people, they have to provide simple means for locating and controlling their physical objects and adjusting their services to a given working context.

In particular they have to support the control of light and air conditioning, the usage of communication facilities, e.g. video equipment, telephone, or fax, the access to Internet/Intranet, the reservation of rooms and required resources, the localisation of persons and equipment within rooms and buildings, the organisation of maintenance and house keeping, and charging and billing.

On this behalf, physical rooms need to be equipped with the necessary intelligence to give access to the full set of their services. The central component provided by this thesis is the RoomComputer, an autonomous installation unit (typically located in the doorframe of the room) consisting of an embedded PC with a touch sensitive LCD-display as local interface, loud speakers, microphone, camera, smart card reader, and an Intranet/Internet connection. Through standard Internet-technologies, any RoomComputer may be accessed remotely in the same way as locally. Several of these distributed RoomComputers can be networked, which makes it possible to administer a set of rooms, or buildings, and to cluster rooms into a virtual building.

The Co-operative Workplace integrates physical and virtual workspaces under a unified intuitive user interface. On the one hand, the virtual workspace gets input from and adapts to physical rooms as locations of people and resources. On the other hand, physical rooms can be reserved and configured from within the virtual workspace. Physical and virtual workspaces are combined under a common metaphor, so that one can act equally in either one and make a seamless transition from one to the other.

Co-operative Workplaces, thus, are environments that facilitate and support collaborative work. They provide for easy communication between co-workers and for

easy co-ordination (for example by way of supporting peripheral awareness) of their individual activities. In addition, they provide for collaborative work among co-workers (for example through desk space for jointly discussing a series of photo prints). Ideally, a Co-operative Workplace would allow co-workers to benefit from co-presence and its affordances with respect to easy communication and coordination while, at the same time, allowing to avoid the disadvantages that go along with this (for example, all those sounds that are not currently relevant to one's own activity and therefore are perceived as "noise"). This is, however, difficult to achieve. Co-operative Workplaces may allow progress in that respect.

The paradigm’s focus is on work context, project structure, and teamwork, rather than on the tools needed during work. The Co-operative Workplace is one that can be created and disposed as the project moves from initiation to completion and adapted dynamically and flexibly during project lifetime. It models the need to form, expand, contract, and disband virtual teams and organisations relying on resources and expertise from different companies and individuals, as enterprises tend to concentrate more on their core business and co-operate among each other for the virtue and duration of projects. Beyond this, Co-operative Workspaces are ubiquitous, integrated, allow for tailorability, and, last but not least, providing extensive awareness information.

The Co-operation Platform addresses the development of this paradigm for presenting Co-operative Workplaces on computers. It is realised as an open platform application, which allows the dynamic creation of teams together with configuration and customisation of their work environments and which supports teamwork in such a collaboration environment. The design is centred on the requirements of users in their work context. By a proper combination of document archives, communication and security services, it allows flexible information presentation and scalable communication. It is aimed to give users "universal" access to their (shared) work context anywhere and anytime.

Using this environment, different people (and organisations) typically work at different locations and with different local tools and services, be it for reasons of culture, installed base, or legacy. For them to collaborate, it is necessary to hide the specifics of e.g. communication or presentation tools under a more unified and intuitive user interface, and at the same time provide built-in adaptability to and transcoding between basic services, e.g. to achieve unified messaging.

The Co-operative workplace paradigm is reflected within the Co-operation Platform by user interfaces that are context oriented, present a unified view to the underlying tools and services, and thus are intuitive to use. This means:

- The interfaces allow users to act as though in a familiar, natural office surrounding (e.g. walking up to somebody/thing, speak up, join a meeting, go to specific work zones, watch), it will do away with tool specific details (e.g. dial a phone number, use a database query language, start audio/ video/ and data sharing applications, determine compatibility mode, etc).
- Given a project, any work done by its team members in the Co-operative Workplace is considered project related. It keeps track of the context in which actions occur, and makes sure things are stored in and restored from the proper context. Context switching between different projects is supported. Project specific data, such as data on teams, people, work plans, calendars, activities, progress, customer base, etc. is accessible for effective project management. XML technology is applied for this purpose throughout the Co-operation Platform.

At the user interface level, the Co-operation Platform uses the concept of so-called Virtual Project Offices (VPOs), which are structured like a real team office. This preserves some of the essential advantages of a local shared office, such as transparency, ease of interaction, availability of project data etc. Team members are projected into the Virtual Project Office (and represented by so-called avatars) as if they were all physically present in a single office. By including a project's input and output, its resources, tools and services, the Virtual Project Office captures the context of the project.

Such a VPO offers different so-called work zones, each supporting a different type of work and work context; for example zones for individual work and zones for collaborative work such as conference sessions or informal information exchange can be provided. Anyone "in office" is being shown in a zone depending on what s/he currently does. Users can easily switch between different work zones and therefore work contexts. They can thus get in touch with a colleague, enter the conference zone, or go to the project file zone; simply by going there, and without having to select or start the proper phone, conference or work space tool. They can communicate with their colleagues, participate in conferences session or inspect and update e.g. documents, respectively.

The VPO presents its users a context oriented view of its contents and members. This is achieved by importing the necessary basic tools, services, and resources into the Co-operation Platform, relating them to each other where meaningful, embedding them in the project context, surrounding them with a context oriented user interface, and thus creating value-added project services. Phone and fax calls in a Virtual Project Office for example, are only be those related to this project, conference sessions are automatically be tied to the context of project data and cost, any collaboration input and output can continually be matched against the project database, cost file, security rules etc.

Co-operative Workplaces may be equipped with an inherent security policy to make the confidentiality, reliability, trust, commitments, etc. in business work possible. Security policy based management is an area that promises innovative and productive improvements. Security functions are not be visible to users on the surface, but they affect

user actions. A security scheme has been developed, which allows administrators to define high-level policies and offers them a visual tool to design, maintain and distribute access policies with a significant reduction in time and effort.

A flexible spectrum of security services makes it possible to tailor a Co-operative Workplace so that it can assure confidentiality for commercial reasons and guarantee individual privacy, as needed. For example, security policies for Virtual Project Offices specify their appearance to the outside world, access to them, and rules for signatures and confidential documents. Much of this is based on existing security tools, such as Smartcards and public key cryptography. Security functions are made available to provide confidentiality, authentication, and security profiles, to define responsibilities, rights, and roles within a project and in the end to allow contracting of experts for a team over the network.

Chapter 3

Requirements

Imagine that a contractor is willing to respond to a request for a proposal to design and build a new information system for a large hospital and clinics distributed throughout the city. To fully address the request, one needs to quickly assemble a high-quality, creative, and knowledgeable team of experts. They will cover a variety of different areas: systems architecture and development, application procurement and integration for managing medical and administrative records, accounting, billing, development of specific tools, deployment of the network, etc. Therefore, the contractor inquires, negotiates, selects and collaborates with specialized firms, hires consultants, and allocates some key resources to the task [Sie96]. The contractor may also assign an administrative assistant in charge of arranging meetings, making contacts, managing calendars, keeping project files, etc.

Ideally, one would like all the team members to become co-located in a workspace fitted with the necessary operating environment: meeting rooms, offices, furnished with desks and file cabinets, telephones, fax machines, and the supporting IT infrastructure and needed tools to build a creative and highly productive atmosphere. There, the team members would find all the tools, services, and documents that are needed to accomplish their work. When at the workplace, they naturally understand who is available, what is going on, who is working on which part, the progress made to date in other parts of the project. In addition to their own work for the project, they are able to pick up questions, enter into discussions with their colleagues, get and give help to solve problems, etc.

In the real world, physical constraints, costs and availability of people usually prohibit this ideal situation: such a matching workspace may just be unavailable, the costs incurred for a short period of time may be extravagant, or it is likely not acceptable for consultants and people from specialized firms to move to the place for a period longer than a meeting, due to other commitments and constraints that they also have to handle. Traditional ways of coping with this problem have been to remedy physical constraints by the use of telephone, fax, email, and teleconferencing. However, these communication tools alone cannot sufficiently substitute co-location when a team-oriented project needs consistent close collaboration and coordination of the team and the objects they work with. The situation generates significant cost and time overhead, e.g. when people resort

to travel and physical meetings or place multiple phone calls if a single short informal meeting would suffice.

As has already been stated earlier, today's work environments can be characterized by a high degree of diversity, flexibility, and dynamics. Local work is supplemented by tele-cooperative collaboration, and cooperation within organizations is supplemented by cooperation across organizations. The traditional local office work went together with various forms of fixed and mobile tele-work. Thus, modern work environments have to constitute so called Co-operative Workplaces, in which virtual workspaces supported by information and communication technology together with physical workspaces are highly integrated and can dynamically be re-configured in response to alternating uses and different work situations.

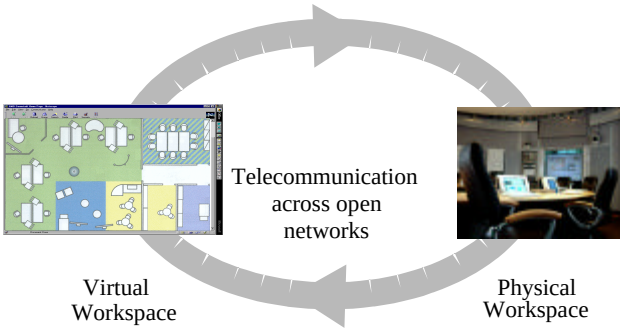


Figure 1 Symbiosis between virtual and physical workspaces

There are a number of deficiencies, which need to be responded to. Firstly, current information and communication infrastructures are essentially centred on individual work. Nevertheless, to support teams and in particular geographically dispersed teams, they must evolve towards a cooperation infrastructure dedicated to teamwork. This does not necessarily mean that very new technologies have to be developed, but that available technologies need to be functionally enriched and integrated under a unified, more application oriented, and easy-to-comprehend user interface.

Secondly, the design of workspaces, especially in offices and buildings, has not kept pace with the change of offices from territorial to non-territorial rooms and from individual work cells to places for face-to-face group meetings. Thus, workspaces have to be created - both in reality and virtual - which are well prepared for teamwork and which are well equipped for integrating the infrastructures needed for IT based work and for the administration of rooms and buildings (facility management). Architects and information scientists have to work together to design real rooms and virtual rooms respectively, for more flexible and adaptable structures, dynamic reconfigurability, and computer-based models for planning, operating, and lifetime maintenance of buildings.

Furthermore, the need to integrate both, the virtual workspaces together with the physical ones has to be addressed. On the one hand, the virtual workspaces have to get input from and adapt to physical ones as locations of people and resources. On the other hand, physical workspaces have to be reserved and configured from within the virtual ones. Virtual and physical workspaces have to be combined under a common metaphor, so that one can act equally in either one and make a seamless transition from one to the other.

The requirements, which are presented in this chapter, were partly gathered in two larger user communities through an analysis of the cooperative work processes found as well as through interviews with the users. The resulting work process scenarios were analysed in order to determine how to adequately support the arising tasks by introducing a collaboration support system combining state-of the-art communication tools and services with advanced groupware technology. Additionally, the existing infrastructure found within the user communities was examined in order to determine in which way it could be integrated into a collaboration support environment.

3.1 Document-Centric Work within Distributed Governmental Agencies

The German unification and the resulting move of governmental agencies from Bonn to the new capital Berlin resulted in a partial relocation and distribution of administrative units between the two cities. This distribution has been both along vertical as well as horizontal lines, meaning that some ministries have been moved entirely to Berlin, some still remained in Bonn, while others have been distributed between Bonn and Berlin. As a result, work groups and task forces (both within a ministry as well as between ministries), which were previously co-located, often became also geographically distributed and required technical support in order to continue their work. To address the challenges of this distribution and to provide technical solutions and organizational suggestions for the resulting distributed work processes, the German government instituted the POLIKOM research program in 1994, within which four projects were created to address the different issues arising from the close cooperation between distributed governmental agencies [TBR98].

One of these projects was the project *POLIWORK – Tele-cooperation and cooperative document management at the personal workspace* – that began in 1994 and last until the end of 1998. The focus of this project was to support the cooperative tasks of the governmental agencies, specifically those collaborative activities centring on document management and document processing, both in synchronous as well as in asynchronous settings, through the development and installation of appropriate support technology. The user community of the project consisted of two German federal ministries, the Ministry of the Interior (BMI) and the Ministry of Economics (BMWi).

In the scope of the POLIWORK project, concentration was on two main scenarios for cooperative activities in the ministerial environment: joint text creation by a work group as well as presentation and discussion including the higher levels of the ministerial hierarchy.

A typical task in the application field is the preparation of a legislative draft. Usually, such a text is created jointly by a small group of people. Different parts of the document are created by different authors and then integrated into one common document. The various intermediate versions of the draft are passed around among the members of the work group. Paper documents are distributed using fax or regular mail while documents in electronic form are sent via e-mail. To coordinate the integration of the various parts, bilateral communication (e.g. telephone call) as well as multilateral communication (e.g. meeting) takes place. Meetings include a certain amount of preparation (setting a date, distributing the necessary documents, etc.) as well as wrapping up (writing minutes etc.).

Another scenario is closely related to the one described above. At the end of the joint text creation, the resulting draft needs to be approved by the higher levels in the ministerial hierarchy. For this step, the final draft is passed to the higher level and then, in a meeting with the superior, the draft is presented, discussed, and changes to be made are annotated. Afterwards, the document is revised accordingly.

3.2 Networked Organisations

Penta Scope is a Dutch consultancy company in which the great majority of activity is carried out either by consultants who work individually, or in small teams together with their clients. Penta Scope has a number of mechanisms for building and transferring core knowledge and expertise, for providing support and for developing new markets.

Typically, teams comprise between 10 and 20 members within a local area. The primary activity of a team is to hold team meetings, taking place frequently on a regular basis. Such team meetings provide an opportunity for consultants to socialise and share experience, knowledge, and expertise, and to find out the latest company developments and directions. Each team has a team manager who schedules and structures the content of meetings. All teams from the same location do form a cluster. Cluster meetings occur less frequent but also regularly. The membership of a consultant at Penta Scope to a specific team is mandatory and normally determined by his or her location.

So-called Intervision Groups are another example, where people at Penta Scope collaborate in teams, which usually comprise about 6 people. Most of the activity within these groups is being performed within group meetings, which sometimes often occur in members' homes on a rotating basis. Intervision Groups are tightly knit social groups in which members are discussing ideas, issues, problems, and experiences in a supportive environment, confidence of constructive criticism and helpful advice and confidentiality. Essentially, these are self-help and self-learning groups where learning can be shared with Intervision Group members. Thus, insight, understanding, and practice can be passed from one member to another, and individual members can benefit from the collective experience of the group when searching for ways of dealing with the problems and

uncertainties that might be experienced, say, in their current projects. The Intervision Groups form the ‘internal Penta Scope family’ of a consultant. The members of an Intervision Group usually originate from different locations. Within Penta Scope, it is mandatory to be part of at least one Intervision Group, but people are free to choose, which one the want to join.

Penta Scope’s core business is realised through projects. Most projects are external, which means, they are initiated by a client and largely realised at the client’s premises. However, in addition to external projects, Penta Scope is usually supporting up to 70 internal projects. Internal projects are initiated by Penta Scope staff members that have to obtain the support of a project sponsor who will fund and be responsible for the overall performance of the project and other staff members willing to participate in the project team. Special internal projects are strategy teams or temporary boards organising special events like the annual Penta Scope meeting.

Besides working on one or more external projects for their clients, a consultant at Penta Scope is normally involved in at least 2-3 internal projects and a member of one ore more Intervision Groups. Thus, consultants’ activity is not only distributed over time and space (i.e., time at client premises and different Penta Scope locations) but also across projects and other business related events.

3.3 Requirements

Future workspaces and in general new co-operation platforms and applications must meet several fundamental criteria in order to allow an organisation to extend its business activities and to participate in a highly dynamic market environment. New services, applications, and underlying infrastructures are needed in order to satisfy the major business drivers and requirements. In order to support the new forms of work and work organisation, there needs to be a flexible, secure, and easy way to provide a team-oriented co-operation platform, which will run across organisations.

Today, the current situation in many cases is a collection of multi-vendor products and services, which do not always easily inter-operate. In other words, major problems exist with respect to integration, inflexibility, low level of scalability and interoperability of tools and services. Current technologies are trying to resolve these problems by introducing the concepts of components and services that can be “plugged and played” into distributed platforms. Therefore, a platform supporting the communication and collaboration within distributed teams and organisations should solve the above-mentioned problems and should focus on the following major requirements.

A requirement can be defined as "a condition or capability to which a system must conform". There are a many different kinds of requirements. One way of categorizing them is described as the FURPS+ model [Gra92] [IEEE-610.12-1990]. The acronym FURPS is used to describe the major categories of technical requirements with the following subcategories:

- Functionality,
- Usability,
- Reliability,
- Performance, and
- Supportability

The "+" in FURPS+ adds such requirements as

- Design requirements,
- Implementation requirements,
- Interface requirements, and
- Physical requirements.

Beyond that, security requirements also play an important role within the framework of this thesis.

The following sections do give an overview on the most important requirements, i.e. the requirements that do have the strongest influence on the design and implementation of the Co-operation Platform.

3.3.1 Functionality

Requirement R.1.1

The provision of appropriate communication channels between the team members involved in the work processes within a project is an important aspect. Within teams, bilateral communication (e.g. face-to-face, phone calls, point-to-point multimedia conferencing) and multilateral communication (e.g. face-to-face meetings, audio conferencing, and multipoint multimedia conferencing) take place.

Within geographically dispersed teams, collaborative work processes often span multiple sites. Therefore, the communication support should not only be bilateral but also support multi-party distributed conferencing.

Multiparty communication (i.e. meetings) often includes a certain amount of preparation (agreement of a certain date, time and location, distribution of the necessary documents, setting up the required multimedia conferencing infrastructure, i.e. reserving the necessary rooms and resources).

Among other things, support needs to be provided for:

- conference reservation, management, and control to be performed by end-users rather than by highly skilled operators
- scheduled conferences and ad-hoc conferences
- specified conferences (i.e. all necessary resources and tools are known in advance) as well as unspecified ones (i.e. resources and tools are added when required)
- set-up of point-to-point as well as multi-point conferences
- reservation of the necessary physical resources (i.e. rooms, required devices)
- reservation of the necessary network resources to support high quality audio/visual communication
- reservation of the necessary resources to support cooperative document processing
- set-up of a conference repository for storage and retrieval of relevant documents of different type and media.

As a general requirement, a multimedia conference reservation and management system requires openness in order to be integrated into an application domain, organizational information systems and business processes. For this purpose, it is indispensable that it is based on widespread and open standards.

The existing technical infrastructure found within a team environment needs to be taken into account (i.e. communication support has to be provided for POTS, ISDN and Internet/Intranet networks).

Requirement R.1.2

Work processes within teams are often document-based. Either texts are created and edited as part of the process or the process involves the presentation, discussion, and annotation of documents. Therefore, document-based support is indispensable. This requirement combines several important aspects: Documents need to be integrated directly into the work processes, including the support of existing document bases. In order to work on the documents, cooperative document processing is required. One important aspect of this is the support and integration of legacy applications, since the team members have a certain set of document processing tools which they regularly use (e.g. one of the established office suites). Documents have to be passed to other people involved in a certain task. Thus, document exchange is to be supported. Furthermore, archiving of the documents is necessary. Finally, paper-based documents need to be integrated.

Requirement R.1.3

Present-day physical workspaces often house a larger number of teams and organizations, and must thus be capable of adapting the floor plan, technical facilities, and organization to changing uses and changing tenants on short notice and at no great expenses. Due to increasing economy pressure, the use of office space will become more diverse and intense in the future. This makes it necessary that buildings take over services, which are

presently rendered by traditional facility management systems, information systems, and various specialists. It is also required that these services be offered to users in an integrated and intuitively usable form.

Facility Management (FM) is the practice of coordinating the physical workspace with the people and work of the organization. It integrates the principles of business administration, architecture and the behavioural and engineering sciences. Although the facility management profession has been in existence for many years, only in recent time has it received worldwide recognition. Business entities have come to realize that maintaining a well-managed and highly efficient facility is critical to success.

Within the co-operative workspaces framework physical workspaces (e.g. rooms) and physical objects (e.g. devices) have to be integrated into virtual workspaces. This is motivated by the realisation that modern forms of work and organization require sharing of resources by different users, frequent and dynamic reconfiguration of workspaces and easy adaptation of rooms to varying requirements. Offices need to be equipped with information and communication technology, devices to be located, rooms to be reserved, furnished, set up for a specific meeting, or re-set to a previous state.

The main requirement is to make it possible to monitor, control, and manage both old and new buildings on a unified worldwide platform, irrespective of any particular local technology. In the old days, room control simply meant a couple of switches. Today's definition of room control encompasses a surplus of advanced pieces of environmental and multimedia equipment ranging from e.g. lighting control systems to VHS players, LCD projectors, and others. Of course, simply equipping a room with advanced electronics is not enough. Making a room that is easy to use and versatile enough to keep pace with quickly evolving technology, has to become the ultimate goal.

In a given work context, physical objects such as rooms, equipment, and other resources have to be located, adjusted and administered accordingly. This has to be implemented by a distributed application, which has to provide room services like

- control of light, heating, ventilation, air and climate,
- communication facilities like telephone, fax, etc.
- Internet/Intranet access,
- reservation of rooms and required resources,
- localization of persons and equipment within rooms and buildings,
- organization of maintenance and house keeping, and
- service accounting and billing.

Requirement R.1.4

Co-operative Workplaces have to provide their users, for each of the projects users might be involved in at any one time, with the awareness of:

- the existence of other team members together with their degree of availability,

- the current set of activities of the other team members, i.e. the work contexts they are currently engaged in, and
- the current set of available tools and services and therefore the set of actions which they can perform, i.e. the work contexts they can get engaged in.

The extent to which it is desirable and necessary to be aware of the state of work and of the activities of other team members might vary, this depends very much on whether the actions of the team members need to be synchronised and / or co-ordinated. For example, it might be useful to be able to see whether or not a colleague is currently active on a given project, thus facilitating spontaneous communication, or it might be helpful to know how far a colleague has progressed on a particular subject.

Requirement R.1.5

Information needed for team and context awareness needs to be derived from user's (goal-directed) interaction with the system, instead of asking them for such information (e.g. a user might want to be aware of the availability of his team-mates or the work contexts they are currently involved in). On the other hand, for privacy reasons, the user must be aware of what others can see, and be able to limit the exposure of such information to other users.

Requirement R.1.6

The user's need for information and for being undisturbed has to be balanced. Information given to the user in order to support team awareness over a distance should be presented in a barely perceptible way.

Requirement R.1.7

The creation and configuration of Co-operative Workplaces has to provide the means for mapping and applying infrastructure services and resource components. The creation process includes one or more of the following:

- The team initiator issues invitations, which may specify a role within the team, to potential team members. Such an invitation includes offering an electronic contract. The contract specifies role, responsibilities, and other project-specific terms. A user becomes a team member by signing and sending back the contract.
- The team initiator has default management privileges, i.e. s/he is the first team operator. These privileges include the right to invite supported users to become team members, and to configure the Co-operative Workplace.
- A team operator can create the "work zones" within Co-operative Workplaces, which includes their creation, naming them, providing descriptions, and configuring them. The term "work zone" is defined as a set of resources (services, data), which are grouped together to support a specific work context, i.e. support a particular set of activities.

Requirement R.1.8

Tools for the provisioning of Co-operative Workplaces have to be provided. Team operators can assign services and access rights to such work zones. These tools shall enable the following functions:

- creating a directory of team members,
- defining role templates,
- defining “work zone” templates, each supporting a specific context of work, and
- defining services for each “work zone”.

Requirement R.1.9

Co-operative Workplaces shall provide means to archive their entire state in a compact form, and means to reactivate them from an archive when needed. With the completion of a project, a Co-operative Workplace needs to be transformed into an archive, retaining all the project resources (e.g. documents) as well as the links between them.

3.3.2 Usability

One key aspect with respect to the acceptance of a system by the users is the usability of the supported functionality. To achieve this, a key aspect, which has been considered in the context of this thesis, is the provision of homogeneous and integrated access to probably rich functionality.

Requirement R.2.1

In contrast to a "stop to shop"-solution located in a particular place and requiring users to come to it for using it, a Co-operative Workplace needs to be ubiquitous in the sense that it allows support of collaborative activities anywhere, irrespective of the current location of a user, weaving itself into the fabric of everyday life [SC94]. It may be useful, in this respect, to distinguish this from a "portable" work environment that can be carried along with, e.g. on a laptop computer.

In contrast to this, a Co-operative Workplace has to be ubiquitous in the sense that it is accessible from virtually anywhere, regardless of the terminal or end-user device used to access it - be it a person's mobile phone, a laptop computer or a public Internet terminal. However, particular functions may not be available on every device.

It is worth noting, that the notion of a Co-operative Workplace is different from media spaces which establish a new space that allows for chance encounters as well as focused interaction by means of permanent audio or audio + video connections between two or more physical locations [GMM+91] [DAB+94].

Requirement R.2.2

Co-operative Workplaces will benefit from providing users with a way to personalise

- tools and services to be used together with their grouping in corresponding work contexts,
- the arrangement of work resources (e.g. the location or path to frequently used documents),
- the look and feel of the workspace for a particular team, and
- their personal portal or entrance to them.

Our cognitive capabilities are supported in significant ways by the (physical) environment we find ourselves in and by the way we arrange work resources [Huc95]. In the physical world, for example, when writing a new text, the arrangement of informational items such as other texts and sheets of hand written notes aids the establishment (or scaffolding) of a particular cognitive context that allows us to do a synthesis of prior positions on a given topic. Users of Co-operative Workplaces, therefore, need to have some freedom to arrange informational items. Through Co-operative Workplaces being tailored for particular needs and activities, these can be employed by their users in the scaffolding of specific cognitive contexts.

The relative importance of these different aspects of tailorability and personalization probably depends on personal preferences and organisational culture. Thus, for some workers the tailorability of tools and their arrangement may be more important, for others the tailorability of the look and feel of the workspace. Also, it can be speculated, that features allowing for the personalization of team workspaces will only be appreciated by users once other issues have been successfully dealt with - in particular, the seamless integration of tools and interfaces and the ubiquitous availability of collaboration technology. Again, this may vary with personal preferences and organisational cultures.

Requirement R.2.3

For distributed co-operative work to be successful and efficient, Co-operative Workplaces have to allow for integration with respect to:

- the different tools and services being used,
- the activities pursued within a given work context supported by a corresponding set of tools and services
- the integration of physical and virtual workspaces, and
- the integration of people.

Requirement R.2.3.1

Co-operative Workplaces need to allow the integration of tools that are already well established in an organisation or mastered by participants while at the same time allowing for new applications and features to be added easily to this platform. The various software tools embedded in this platform need to be integrated in such a way, that they can communicate with each other, can dynamically bridge heterogeneity, and therefore

are able to provide up-to-date information in a consistent and coherent view of data, documents, and projects. This means that Co-operative Workplaces have to provide unified services together with a framework for tool integration.

Requirement R.2.3.2

The participation in projects and other activities conducted by an individual is always situated within a work context. People can be involved in several different projects at the same time. While working on a specific project, they can be involved within a certain work context (e.g. a meeting) within that particular project. Normally, the work within such work contexts is being supported by a set of tools and services. Therefore, a Co-operative Workplace needs to support the integration of tools in work contexts, in order to provide context-oriented services to its user rather than a tool-oriented view. Furthermore, a Co-operative Workplaces needs to provide means to allow its users to easily switch between different work contexts.

Requirement R.2.3.3

Co-operative Workplaces need to achieve a symbiosis between virtual and physical workspaces. Even when people are working together in a virtual environment (e.g. having a video conference) they are still living in a physical environment and are accessing this virtual one by using the available devices (e.g. a video conferencing system) present within their physical surrounding. The devices available within the person's physical surrounding strongly determine which services that person can use in the virtual workspace. Therefore, Co-operative Workplaces have to provide means to locate people and the current set of devices in their physical surrounding together with the set of capabilities the individual devices have.

In order to support the idea of a context-oriented rather than a device-oriented view, Co-operative Workplaces have to provide a unified view of the available devices and the services they offer.

Additionally, as rooms and their resources can be shared by several users, sometimes on a non-territorial basis, Co-operative Workplace have to provide means for room and resource management.

Requirement R.2.3.4

A Co-operative Workplace has to support the gathering of geographically dispersed teams in the sense that:

- it assures that all team members are sharing the same common view of the project contexts,
- it provides intuitive and easy-to-use means of communication, and
- it gives the team members access to shared information and resources.

Requirement R.2.4

Co-operative Workplaces need to be configurable. In order to adopt a high degree of flexibility with respect to the needs of different persons and teams, means for the configuration, integration, and customisation of the provided facilities and services are necessary. A set of tools for customising the tools and services within a Co-operative Workplace needs to be provided.

Requirement R.2.5

Co-operative Workplaces need to be created and discarded as projects move from initiation to completion. The workspaces need to be flexibly adapted during the project lifetime, for example by modifying the list of project members, by reorganising the document repository or performing other changes. This implies, among other things, the option to import resources (e.g. documents) from and to export resources to a different workspace.

3.3.3 Reliability

For Co-operative Workplaces to become a useful tool in everyday work, they have to be at least as reliable as the tools and services, which they integrate. The users' primary interest is to get their work done and not playing with technology. However, this thesis is not aiming at any kind of high availability, since this is not the key focus. Therefore, no reliability figures, such as mean time between failures or mean time to repair, are defined here.

Requirement R.3.1

Co-operative Workplaces must guarantee the integrity of all the data that users store in the system. Such data must not change without the explicit or implied consent of the respective user.

Requirement R.3.2

In the event that the underlying platform fails in any way, it needs to fail safely. This means failure of whatever kind must not cause additional damage beyond the system or part of it not being available. A failure within a subsystem must leave intact all other parts of it not directly affected by the failure or dependent on this subsystem.

Requirement R.3.4

The underlying platform should be able to recover from failure, i.e. let the users continue their work with the parts of the system still functional, and let them try again later.

3.3.4 Performance

Within the scope of this thesis, performance requirements that do not strongly influence usability, i.e. reasonable response times, are of limited interest.

3.3.5 Supportability

Requirement R.5.1

Co-operative Workplaces must be expandable. A systematic way of integrating current and future information systems, tools and services into a so-called Co-operation Platform need to be provided. Therefore, support for additional and/or new components has to be provided that conforms to the architectural model. The architectural model needs to be open to be accessible and deployable by application developers and end-users. Therefore, a high degree of interoperability between different multi-vendor products and applications has to be ensured. “Plug & Play” mechanisms will allow the integration of additional functional components even during runtime.

Requirement R.5.2

Co-operative Workplaces need to be based on component technology. The application of component technology preserves autonomy, which means that different “functional” components can be developed and deployed independently. Mechanisms for integration and re-use of components have to be provided. The component-based development of services needs to be based on open, interoperable underlying applications and tools. In distributed systems, “Plug & Play” integration of interoperable components has to be possible.

Requirement R.5.3

Co-operative Workplaces need to follow a service-based approach. The support and integration of a set of different tools and technologies represented by services will provide to the end-users an easy to deploy and use environment. Therefore, third party tools, applications, and components can be integrated to the platform by using an appropriate service oriented middleware (i.e. service announcement and discovery protocols). A flexible and easy management of services is needed in order to adapt to the different organisational and business needs. Such a service management framework must enable consistent and secure access to the services provided.

Requirement R.5.4

Co-operative Workplaces have to support multiple national languages and character sets. All software developed within the context of this thesis needs to support Unicode and all textual features of user interfaces shall be configurable for display in multiple languages.

3.3.6 Design

Requirement R.6.1

There is a world outside. Wherever possible, Co-operative Workplaces should be an enhancement to this world, not a replacement for it. Therefore, interfaces for the import/export of external information as well as for interaction with the outside world have to be provided.

Requirement R.6.2

Co-operative Workplaces must support virtual organisations. Co-operative Workplaces are aiming at the support of communication and collaboration across technical and organisational boundaries.

Requirement R.6.3

Co-operative Workplaces need to be scalable. They must be scalable due to the high degree of distribution of future organisations.

3.3.7 Implementation

Requirement R.7.1

Co-operative Workplaces need to be modular. This will result in a very flexible system allowing customisation according to the individual needs within a project, available infrastructure, and target group size (scalability). Co-operative Workplaces need to provide means for integration of existing legacy applications, as well as, new ones in an easy and dynamic way. The integration of legacy systems, frameworks, and concepts requires the provision of appropriate wrapper mechanisms.

Requirement R.7.2

Co-operative Workplaces need to be platform independent, i.e. they will have to be deployed in different environments and on a variety of platforms. This means that universal access for a wide range of client types and terminals (e.g. PCs, PDAs, mobile phones) has to be provided in order to allow end-users to access the provided functionality from virtually anywhere and to easily integrate within their work environments.

Requirement R.7.3

Co-operative Workplaces should mainly be based on Web-Technologies. Different terminal technologies ranging from thin clients to mobile devices, like for example PCs, Network Computer, PDAs, and Mobile Phones have to be supported. Access to the provided functionality must be platform independent. Therefore, the user (client) side should make use of web-technologies, applets, and plug-ins executing within a browser environment.

3.3.8 User Interface

Requirement R.8.1

The usability of Co-operative Workplaces and the functionality they provide is strongly influenced by a homogeneous, easy-to-use, and intuitive user interface with a familiar look and feel, which gives easy access to the provided functionality.

Requirement R.8.2

The user interfaces of Co-operative Workplaces have to be group-aware, e. g. they not only need to provide their users with a view of contents of e.g. repositories or access to services, but also to display information about which users are currently working on the same project or collaboratively viewing/editing which documents. Integrating individual tools under a common interface offers the possibility of reducing annoying information by suppressing unnecessary functionality and combining various 'low-level' features and tools into higher-level, easier-to-use services. This can help to reduce application complexity, as it relieves users from the compulsion to follow a tool-driven "suite" of applications running side-by-side.

Requirement R.8.3

The user interfaces have to be context-aware, i.e. they have to give clear indication about a users own work context together with information about the work contexts his team-mates are currently in. Additionally, they must provide a means for a user to easily switch between different work contexts.

Requirement R.8.4

User interfaces for Co-operative Workplaces have to be information-rich, i.e. they have to provide their users with all relevant information needed in a certain work context.

Requirement R.8.5

The user interface needs to react on user requests instantly, except for those situations where a delay is unavoidable. Reactions within less than 0.1 second can be considered as instant [Nie93], reactions within less than 1 second are acceptable. If a longer delay is unavoidable, the system should give immediate feedback to the user showing the request has been received and is being processed. In some situations, it might be useful to give information with respect to progress and expected duration.

Requirement R.8.6

User interfaces should be modeless and device independent. Users might want to access some of the provided functions from different end-user devices, e.g. desktop computers, laptops, phones, mobile phones, PDAs.

3.3.9 Security

Requirement R.9.1

Co-operative Workplaces need to provide a secure environment for the management and usage of the provided services and resources. Therefore:

- access to services and resources has to be governed by pre-definable security contexts,

- means for the secure identification and authentication of its users by means of e.g. Smartcards and public key infrastructures have to be provided, and
- it must be possible to equip any Co-operative Workplace with specific security policies.

Such policies will determine who is allowed, at what time and under which circumstances (e.g. roles, work contexts) to get access to the resources and services provided.

Requirement R.9.2

Security measures in Co-operative Workplaces need to be inherent, invisible, and usable. Users, who feel bothered by security measures or do not care about security, are likely to bypass and subvert security measures that they think disturb their work.

Requirement R.9.3

Co-operative workplaces need to provide means for the individual user to control who is going to see what information, which relates to himself and his activities. Therefore, they have to give their users clear indications about what other users may see and at the same time provide means to restrict access of other users to personal information.

Chapter 4

Related Work

Pertinent aspects of future workspaces have been investigated and realized in various places of the world in recent years. In view of this, the proceedings of an international workshop on Cooperative Buildings give an outstanding overview of the state-of-the-art, relevant work in the different disciplines, and an up-to-date list of references to relevant work world wide [SKB98]. The workshop, held at GMD in Darmstadt, was considered a unique gathering which brought together people from a wide range of disciplines such as: computer science, electrical engineering, human factors and ergonomics, psychology, sociology, education, industrial design, architecture, facility management, real estate, office and work planning as well as consultants in the respective fields.

The purpose of the workshop was to discuss what it takes to design and build Cooperative Buildings as models and examples of future work environments blending real and virtual worlds. Since this endeavour requires a highly interdisciplinary approach, the focus has especially been put on the relationships between organizational, architectural, and technological factors. Relevant research, prototypes, experiments, pioneering efforts have been reported and discussed in order to learn from experience. Conclusions have been drawn for an integrated development in the future and new aspects and ideas have been brought up.

The following sections contains a survey regarding the state of the art of tools, systems or services which are of major relevance within the context of this thesis.

4.1 Virtual Workspaces

4.1.1 Multimedia Conferencing

Multimedia conferencing is an available technology, which enables people to collaborate in ways never before possible. However, the user interfaces are generally far from self-evident, and administrating the necessary infrastructure, setting up and scheduling

conferences as well as controlling them can be very difficult, time consuming, and costly [Sha97]. PlaceWare [PlaceWare] and Presentation.Net [PreNet] are (commercial) examples of systems supporting Web-based conferencing. More visionary, the MIT project Oxygen [Oxygen] brings computation and communication to users through natural spoken and visual interfaces, making it easy for them to collaborate, access knowledge, and automate repetitive tasks.

4.1.2 Instant Messaging

Instant Messaging has become an important communication tool within distributed team. It requires special client software that has to be installed on the user's machine and uses the Internet to allow for instant exchange of messages between two or more users. The messages exchanged are very much like e-mails, but instead of the user having to 'pull' them from a server, they are being 'pushed', i.e. they pop-up almost instantaneously at the recipient's site after being sent out by the sender. However, instant messaging is not a synchronous communication like a text chat where all parties can receive and send simultaneously. The messages can pop-up as alerts or can be acoustically signalled. Participants have a distinct identifier registered with the service provider. Each user can set-up his personal buddy-list of people he wants contact by instant messaging. While logged on to the service the system indicates who of the specified buddies is currently available i.e. registered with the same server and is able to receive an instant message. Likewise, the system indicates the availability of a registered user to others, who have this person on their so-called Buddy-Lists. However, the four most prominent services today, namely Yahoo! Messenger, AOL Instant Messenger, MSN Messenger, and ICQ, offer also additional features like e.g. voice-chat, reminder of birthdays, or computer games.

Yahoo! has expanded its pager service, now offering one of the most comfortable instant messaging solutions, the Yahoo!-Messenger [YaMes]. A perquisite is, however, to register an e-mail account with Yahoo!. It offers the management of so-called Buddy-Lists, instant messaging, multi-party chat, instant signalling of e-mails arrival, voice-chat, and file-transfer. There is Java-based version available, so that this tool can be used on all platforms for which a Java Virtual Machine (JVM) exists. The Yahoo!-Messenger seamlessly integrates with other Yahoo! services like online calendar, the Yahoo! search engine, and others.

AOL was among the first to develop instant messaging services. The required software has been integrated with the recent versions of Netscape's Communicator, thus no separate installation of the required client software is necessary [AOLMes]. The AOL Instant Messenger provides the management of Buddy-Lists, instant messaging, file-transfer, and a search engine. It has a big user community, and provides an alert function in order to prevent not appreciated disturbances.

The Microsoft MSN Messenger requires a registered e-mail account at the MS Hotmail service [MSMes]. It provides nearly the same kind of functionality as the AOL Instant Messenger, is only available for MS Windows based platforms, and integrates with MS NetMeeting, Microsoft's solution for multimedia conferencing.

ICQ developed by the Israel-based company Mirabilis is still the number one in instant messaging [ICQ]. It provides the management of Buddy-Lists, instant messaging (online and offline), multi-party IRC chat, voice-chat, IP-telephony. It has a search engine, and allows file-transfer. Plug-ins are available for online computer games.

4.1.3 Support for Document-based Collaboration

A number of research and development projects have taken on the task of supporting document-based collaboration between multiple distributed sites.

The BSCW shared workspace system [AB96] is a Web-based groupware application, which follows a collaboration paradigm similar to the VPO: users can create and structure repositories into which they can import documents, which are then made available to all users who currently have access to the repository. The documents can be retrieved by the users, processed and at a later stage put back into the BSCW system for the other users. As a purely Web-based application, it differs from the VPO in a number of points, which were of importance in the presented application context. Firstly, BSCW is an asynchronous system, which does not provide direct feedback and information about other user's actions (synchronous group awareness). In order to observe changes within the repository, the Web page has to be manually reloaded by the user. In addition, BSCW provides no integration of synchronous multi-party A/V communication.

CoopWWW [CoopWWW] is a project that aims at extending the BSCW system through the integration of A/V conferencing between the system's users. It currently does this by integrating a software A/V communications module, CUSeeMe [CUSeeMe]. The resulting system does not offer the high-quality A/V conferencing supplied by the VPO with the integrated ISDN conferencing facilities. Since CUSeeMe is a pure software solution and employs the Internet for data transport, the available bandwidths and thus the achievable frame rates and picture quality are lower than the ones achieved by integrating ISDN-based multipoint conferencing as has been done in the VPO. An evaluation of user requirements has shown that audio and video quality is of utmost importance for the acceptance and wide-spread use of the resulting technical solution.

Lotus Realtime Notes [RTNotes] is a combination of Intel ProShare and Lotus Notes. Lotus Notes, as a replicated distributed document database can be used to store documents, which are accessible to a group of users. Intel ProShare provides real-time communication and collaboration facilities on these documents by use of the application-sharing component. What this combination does not achieve, is the seamless integration of multipoint communication and multi-party distributed settings. The VPO, developed within the context of this thesis, goes one step further towards a more complete solution by integrating scheduling, control and reservation of multiparty conferences. Furthermore, Realtime Notes does not provide synchronous group awareness to the system's users.

The successor of Lotus Realtime Notes is called Lotus Sametime. Sametime [Sametime] is a so-called real-time collaboration tool and as such, it offers features like:

- awareness (i.e. team members are aware when their team-mates are logged on to Sametime)
- synchronous communication (i.e. it supports audio, video, and chat communication), and
- synchronous sharing of objects (i.e. objects such as e.g. documents can be jointly viewed and edited).

Neither Realtime Notes, nor Sametime, nor BSCW/CoopWWW is open and adaptable to different standards-based collaboration and communication tools in the way in which the VPO is.

The TeamRooms system [RG96] aims at supporting collaboration of distributed teams by providing persistent repositories for the collaboration documents. It provides people with awareness of others who may be available for real-time interactions, and automatically establishes connections as users enter a common room. But TeamRooms does not integrate synchronous multimedia conferencing and uses specialized "applets" for cooperative document creation, has no provision for legacy applications as well as their seamless integration into the collaborative work processes within a geographically dispersed team.

4.1.4 Object and Application Sharing

During the past few years, a lot of applications and frameworks have been built to support collaborative work on shared artefacts and objects, such as documents or drawings. In general, shared objects include any form of data that is distributed for multipoint communication in collaborative sessions between two or more users. The most common examples are:

- shared whiteboards, where users can jointly draw sketches,
- shared viewers, allowing to have a joint view on the same data, e.g. a presentation or a Web-page,
- shared editors, where users can jointly edit text-based documents, and
- application sharing, allowing users to share applications, which originally have not been designed for usage in collaborative sessions and therefore are collaboration unaware.

Those tools are mostly based on frameworks providing support for shared objects in distributed applications. Beside others, NCSA Habanero is a very prominent example for such a framework [CGJP98]. It supports the sharing of Java objects between multiple applications. Such objects can be distributed and their consistency is ensured through data and event sharing mechanisms provided by the framework. Users interact collaboratively with each other by using tools based on the framework; such tools are called Hablets for Habanero Applets.

Rather than developing new application supporting collaborative sessions between dislocated users, one idea is to make use of the vast variety of existing applets on the World Wide Web. JASMINE (Java Application Sharing in Multiuser INteractive Environments) is a collaboration system, which facilitates the creation of collaborative sessions and allows users to share Java applets, which were previously collaboration unaware. The JASMINE architecture makes use of the Java Event Delegation Model in a way that it extends the capabilities of Java applications so that Java applets can be used collaboratively [ESGS00].

DyCE (Dynamic Collaboration Environment) is an object-oriented framework, which allows the development of so-called Groupware Components, for example shared whiteboard or shared editors [Tie01]. As such, DyCE provides a central programming abstraction together with an advanced architecture supporting the development of applications that give their users shared access to artefacts and data objects in collaborative sessions]. Multiple such Groupware Components can simultaneously access shared objects and the DyCE framework ensures that the objects are kept in a consistent state. Additionally, components can notify other components based on object-based event propagation, which allows the coordination between them. This approach gives developers great flexibility in designing Groupware Components.

The ITU-T T.120 series standard defines protocols for real-time, multipoint data communications [ITU-T.120]. Established by the ITU, T.120 is a family of open standards that was defined by leading data communication practitioners in the industry. The key Benefits of T.120 are as follows:

- Interoperability: T.120-compliant products from different vendors can work together at either an application or data transport level.
- Reliable data delivery: Error-corrected data delivery ensures that all endpoints will receive each data transmission.
- Network transparency: Applications are completely shielded from the underlying data transport mechanism being used. Whether the transport is a high-speed LAN or a simple dial-up modem, the application developer is only concerned with a single, consistent set of application services.
- Platform independence: Because the T.120 standard is completely free from any platform dependencies, it will readily take advantage of frequent advances in computing technology.
- Network independence: The T.120 standard supports a broad range of transport options, including the public switched telephone networks (PSTN), integrated switched digital networks (ISDN), and TCP/IP. Furthermore, these vastly different network transports, operating at different speeds, can easily co-exist in the same multipoint conference.
- Scalability: T.120 is defined to be easily scalable from simple PC-based architectures to complex multi-processor environments characterized by their high performance. Resources for T.120 applications are plentiful, with practical limits imposed only by the confines of the specific platform running the software.

As this thesis develops an open architecture, it is in principle not restricted to a specific tool or technology, on the contrary, any of the above solutions can be integrated. The main reason why this thesis focussed on the integration of solutions that are based on the ITU-T T.120 series standards, such as e.g. MS Netmeeting and Lotus Sametime, is that they are very widespread, based on an open standard, and thus ensure interoperability. This allows addressing a large user community, even though the available tools might not always be the bestmost solution from a technical perspective.

4.1.5 Virtual Collaborative Environments

DIVE (Distributed Interactive Virtual Environment) is an early Internet-based multi-user Virtual Reality system where participants navigate in 3D space and see, meet and interact with other users [FS98]. Based on DIVE, the COVEN project (Collaborative Virtual Environments) developed a computational service for cooperative tele-working and virtual presence, based on a platform with avatars in a 3D virtual world [NBB+98] [COVEN].

The exploration and development of the concept of an electronic landscape as a virtual environment that provides interconnections to other virtual environments have been investigated in the eSCAPE project [BSC+97] [eSCAPE].

A groupware system that takes advantage of a team room metaphor is TeamRooms by the University of Calgary [RG96]. It provides people with awareness of others who may be available for real-time interactions, and automatically establishes connections as users enter a common room. TeamWave Workplace [TeamWave] is a commercial outgrowth of TeamRooms. It provides creation and manipulation of workspaces and a mechanism for storing the contents of each room in the state people left them, ready for the next working session.

Another commercial product is eRoom, a Web application made to support collaboration of distributed project teams [eRoom]. Among other things, it puts a focus on managing project documents, including access rights, version tracking and repositories, and capturing the project history.

At a first glance, virtual collaborative environments using a room metaphor, providing immersive 3D virtual environments, and allowing for new ways of interaction between users, seems to be an interesting solution. Based on this impression, a 3D user interface for the VPO has been built based on VRML, HTML and Java technologies (see Chapter 7). However, a series of user trials has shown that a majority of users finds it difficult to navigate and interact in such environment. Another problem is the fact, that at the client side powerful machines are required to do the necessary online rendering of such environments and that there are a lot of compatibility problems between Web-browsers and VRML Plug-ins.

4.1.6 Intelligent Personal Assistants

Intelligent Personal Assistants have been one of the first applications that the scientific community has used to better illustrate the benefits of intelligent agents, such as the early work by Maes [Mae94] or Etzioni and Weld [EW94] on personal assistants.

Nowadays, we are still on the early stages with respect to the development of truly intelligent personal assistants. However, there have been some significant achievements, both on standards and on agent development environments. Examples for agent development environments based on Java include:

- JAFMAS - Java-Based Framework for Multi-Agent Systems [Cha97]
  Department of Electrical & Computer Engineering and Computer Science,
  University of Cincinnati (http://www.ececs.uc.edu/~abaker/JAFMAS/),
- JADE - Java Agent Development Framework [BPR99]
  CSELT, Turino, Italy
  (http://sharon.cselt.it/projects/jade/),
- JATLite - Java Agent Template Lite
  Stanford University
  (http://java.stanford.edu/)
- AMETAS – Asynchronous Message Transfer Agent System [HZ00]
  Verteilte Systeme und Betriebssysteme, Johann Wolfgang Goethe-Universität,
  Frankfurt am Main
  (http://www.ametas.de)

Currently there a few Intelligent Personal Assistants available, the most prominent examples are:

- Narval,
  from Logilab, which is released under the GNU public license
  (http://www.logilab.org/narval/).
- Prody Parrot,
  commercial software from MindMaker (http://www.prodyparrot.com/), or the
- Intelligent Personal Assistant,
  from British Telecom
  (http://www.labs.bt.com/projects/agents.htm).

All of them advertise to help the user by delegating some tasks to their personal assistant, and include functionalities such as email filtering, calendar management, scheduling of meetings, contact finding, voice communication, and others.

4.1.7 Ongoing Research Projects

This section lists some currently ongoing research projects, which are of relevance within the context of this thesis.

The LEAP project (Lightweight Extensible Agent Platform) [LEAP] tries to address the need for open infrastructures and services that support dynamic, mobile enterprises. Within LEAP agent-based services are being developed, supporting three major requirements of a mobile enterprise workforce:

- Knowledge management in order to anticipate knowledge requirements of an individual;
- Decentralised work co-ordination in order to empower individuals, by the co-ordination and trading of jobs;
- Travel management in order to plan and co-ordinate the need for travel.

The OSMOS (Open System for Inter-enterprise Information Management in Dynamic Virtual Environments) [OSMOS] project tries to enhance the capabilities of construction enterprises to act and collaborate effectively on projects by setting up and promoting value-added Internet-based flexible services that support team work. They will provide:

- Internet-based services for construction applications,
- information sharing services (e.g. documents), and
- semantic cross-referencing between information.

This project is specifically concerned with defining the working practices, processes, techniques, tools and technical infrastructure to allow the European construction industry to progress from its current position towards a large scale, computer integrated approach.

The provision of enabling technologies for collaborative engineering is the also the objective of the E-COLLEG (Advanced Infrastructure for Pan-European Collaborative Engineering) [eColleg] project. It tries to

- develop generic collaborative services by augmenting existing integration technologies by scaleable, platform-independent services allowing dynamic configuration of the infrastructure;
- handle security and access management for distributed teams, services and data;
- provide a solution for multi-site and multi-platform tool integration by developing a technique for dynamic tool integration based on multi agent technology;

Transparent data access will be provided by self-describing data formats based on widespread Internet standards such as XML.

The objective of the TEAM-HOS project (Methodology and Tools for World-best Teamwork in Hospitals) [teamHos] is to develop and verify innovative methodologies and a set of tools for the analysis of the hospital's requirements to specify, design, select, and implement adequate information and communication technology (ICT) solutions that will support the development and introduction of new types of ICT supported teamwork organisations in hospitals.

The project tries to develop technologies and systems for:

- resource sharing,
- real-time team co-ordination,
- innovative workplace design,
- knowledge sharing within teams, and
- team inter-communication.

BIDSAVER (Business Integrator Dynamic Support Agents for Virtual Enterprise) [BIDSAVER] is a project that aims at the assessment and development of a framework for the constitution and operation of Virtual Enterprises through web based information agents. It will deliver:

- Methodologies, reference processes, data models, and relevant infrastructure tools, to allow for standardised cooperative project conduction and for dynamic search of project partners through the Web, which best suit with the contingent needs of a virtual enterprise, suitable to support both the fast constitution and the dynamic upgrade and the operations of it;
- A legal framework for the management of relationships among the partners within a virtual enterprises and between the market and the virtual enterprise, suitable to support both the fast constitution and the dynamic upgrade and the operations of the virtual enterprise;

CREA NET (Creative's rights European agency network) [CREANET] aims at creating a collaborative and secure environment using new technologies for authors and independent content producers by the provision of a common rule based trusted environment. The security and trust services will be managed by a so-called Transaction Manager Agent, which according to each transaction's context and to a set of operational rules will invoke the relevant services and thus will guarantee the rights of all actors involved in the transaction.

DYNOCAS (A System to Realise Dynamic Networked Organisations on Heterogeneous Networks in the Consultancy/Agency Sector) [DYNOCA] tries to support distributed teams in planning and executing media projects in the consulting/agency sector. It will support the collaboration between customers, main contractors, sub-contractors, and suppliers by the development of organizational and software models together with the implementation of a prototypical system based on emerging standards, such as XML, digital signatures, SSL secure transmission, Java Beans, and Jini.

The design, development and validation of a multi-service business architecture, supporting teamwork within distributed enterprises, for both, cooperative product design as well as for business management, is the objective of the MOTION (Mobile Teamwork Infrastructure for Organisations Networking) project [MOTION]. The project will:

- define distributed communication services infrastructure,
- identify and develop a core set of services based on such an infrastructure, and
- use a core set of services to build up specific business applications.

Support for mobile working will be provided by the integration of hand-held devices (PDAs, mobile phones). Furthermore, it will allow for configuring and operating a consistent set of so-called Team Work Assistants.

TOWER (Theatre of Work Enabling Relationships) [TOWER] is a research project that tries to support the co-operation within virtual teams by providing group awareness and chance encounters through a 3D virtual environment, which is at the heart of the so-called Theatre of Work. Users and their current actions will be represented by avatars and their symbolic actions. Avatars of users who work in a similar context will appear spatially close in the 3D virtual environment.

The SocialWeb [Socialweb] research program aims at getting a thorough understanding of the principles and techniques for linking people in virtual environments. It is developing a suite of systems, which provide new social media that:

- offer a persistent shared environment,
- can be inhabited by people,
- allow people a wide range of expression,
- portray history, presence, and activities,
- support "populated" knowledge, and
- extend into the real world by the usage of smart appliances.

It will do an assessment of the developed principles and systems in a number of real-world applications. As as a component of the SocialWeb program, NESSIE is the enhancement of social and task-oriented awareness between people who cooperate or inhabit the same context through the provision of a configurable awareNESS envIronmEnt that enables situative event and action notifications [NESSIE] [Prin99] [PPB99].

4.2 Physical Workspaces

Throughout the past few years, various projects tried to more seamlessly integrate the devices available within the vicinity of a user and the services they are offering, together with the users' activities.

4.2.1 Intelligent Rooms

The Intelligent Room project at MIT [Coe98] is a prominent example for the development of a research platform to explore the design of intelligent physical workspaces. As such, it was created to experiment with different forms of natural human-computer interaction during what is traditionally considered non-computational activity. It is equipped with numerous computer vision, speech and gesture recognition systems. These allow the room to watch where people are moving and under certain circumstances where and how they are pointing, and to listen to a wide variety of spoken language utterances.

Empirical studies, including interviews of creative teams in different companies, have shown that creative teams have specific requirements for appropriate working environments. Interactive Landscapes (i-LAND) serve as a collaborative work environment facilitating especially creativity and innovation processes in teams. The i-LAND project [iLand] integrates several so-called Roomware components into a combination of real, physical as well as virtual, digital work environments to support the work of creative teams [SGH98]. Its focus is on the development of experimental work environments, which allow dynamic re-configurations for various situations of creative group work. This results in interactive rooms, where computer modules and wireless computer networks are integrated into furniture and room elements to form so-called Roomware components.

Roomware means computer-augmented things resulting from the integration of room elements (e.g., walls, doors, furniture like tables, and chairs) with computer-based information devices. They provide support for the creation, editing and presentation of information and are the basis for diverse forms of interaction (e.g. with pens and manual gestures) with information objects spread over the room. The initial set of Roomware components to be developed within the i-LAND project consists of an interactive electronic wall (DynaWall), an interactive table (InteracTable), and mobile and networked chairs with integrated slate computers (CommChairs).

Complementary developments for creativity teams, such as natural interaction and Roomware, can be found in [Luc98] and [SGH98] respectively. The Information Building [Soo97] presents an interesting way of looking at a building as a combination of real and virtual rooms.

At the Vienna University of Technology, a project is in progress [Jini01], which proposes to integrate the EIB field bus system and the Jini network technology to allow a better and more flexible management of devices at the physical workspace. Its objective is to combine both technologies in a way that devices can seamlessly interact with each other regardless of which system the devices belong to. The system design is based on a proxy architecture. The so-called Djinn Communication Module (DC) is responsible for the communication with the Jini environment. It does service look-ups for remote services and announces the presence of local services to the Jini network. The EIB communication module (EC) is responsible for the communication to and from the EIB bus. A Jini Control Module (JC) mediates between the DC and the EC.

The Linux Home Automation Project [LHOP] defines a layered architecture. Daemons are implemented to access individual devices within the physical location of a user. Such daemons can be accessed via network sockets to send commands to devices and receive their answers. Applications that send specific commands to the devices are built on top of the daemons. The top layers include facility management applications, and offers graphical user interfaces. Currently, the system architecture is not completely specified. It lacks a universal device access language in order to control devices, in particular devices from different vendors.

At the Georgia Institute of Technology, a research project is carried out towards aware homes. The projects within the Aware Home Research Initiative [AHRI] do cover different aspects of Smart Homes and Ubiquitous computing environments. So far, the

author is not aware of any published information with respect to a common networking infrastructure that uniformly integrates different devices.

4.2.2 Location Awareness

In the context of this thesis, localisation of devices together with location management technologies are being used in order to be aware of the available devices and their capabilities within a users' vicinity. This allows determining, which services can be offered to a user within the context of his location.

The usage of tags based on Radio Frequency Identification (RFID) as an enabling technology for location awareness and ubiquitous computing has already been investigated in some other research projects. For example, Barrett et al. have been working on creating so-called "virtual floppies" – small disk-like objects that can be associated with files and folders on a computer [BM98]. Minar et al. are using RFID tags to create a jukebox, where poker chips can be dropped on a table, which then triggers the playback of an associated song [MGR+00]. The coupling of real and virtual objects is also be considered by Ljungstrand et al. in their WebStickers project [LRH00]. They use barcode labels to associate a Web page with a physical object. Want et al. are describing a number of scenarios where RFID tags are used to connect physical objects with virtual representations or computational functionality [WFG+99]. This project is somewhat related to this thesis. The first three projects mentioned do have one thing in common, that the relationship is bi-unique: a virtual object is always assigned to exactly one real object. Only Want et al. describe the possibility of associating more than one URL with one Tag.

The approach of this thesis goes one step further. Real objects are not associated with only one virtual object, but with a whole set of virtual counterparts, objects and/or applications. Any unambiguous assignment is solved at "activation" time, at which the current relationship is determined by the present context information. In HP's Cooltown, a Web page can be assigned to arbitrary things [Cooltown]. The project uses – like this thesis - the Web infrastructure as an infrastructure for the ubiquitous presence of things.

Context information is often used in order to facilitate the communication and interaction with a user. The Cyberguide Project [AAH+97] uses location information and direction information to adapt the displayed information to the actual situation. The Conference Assistant [DSA98] uses target position information, conference schedule and user preferences. Within this thesis, context information is used to select an information subset out of a given set of possibilities. Much closer to this work is the approach being followed at Georgia Tech [SDA99] that uses an infrastructure to support context-aware applications.

Location-based services often use technologies like GPS [BMW00] [GPS00] [NR96] to locate objects and persons outdoors. In the indoor area Active Badges [WHF+92] [WJH97] are most frequently used. However, within this thesis the low-cost and unobtrusive RFID technology is being preferred. On the one hand, large numbers of everyday objects can be easily tagged and used in multiple locations, using a simple and inexpensive infrastructure. On the other hand this is based on the same kind of principles, which are being used for the Office Identification Card in order to identify and locate persons.

4.3 Conclusion

In conclusion, one can say that in the last few years a large number of tools and systems have been proposed or built, both in research and in industry, in order to support collaboration in distributed teams and organisations. These developments have been inspired by the fields of computer-supported co-operative work (CSCW) as well as by developments in multimedia communication, virtual reality technologies, work on intelligent agents, and others. Nevertheless, only few of them have succeeded on the market. Many prototypes and products have not been well accepted by the users; even more have not reached the market in the first place.

There are several reasons for this. One such reason has to do with requirements analysis, another with the costs and benefits involved in using such applications. With respect to the first, specific problems related to requirements analysis became apparent with the turn towards the development of systems supporting the collaboration in groups. In part, these problems turned on the role and intuitions of the developers involved in these projects. As long as the aim was to provide systems that support individual users, the developers' intuitions about user needs and requirements proved to be sufficient for their job. When developing systems designed to support collaborative work in groups, however, this proved to be the case to a much lesser extent [Gru88]. Additionally, it became apparent that established methods of requirements analysis employed (e.g., structured analysis and design [MH97]) did not provide a sufficiently deep understanding of the users' activities, competencies and needs in the concrete situations of work they face during their workday [And94]. In consequence, there have been efforts to involve users in the design of systems ('participative design') and to embrace new methods for user requirements analysis, such as ethnographically informed approaches [Mei01].

Another problem is that often there is an imbalance between the costs and benefits involved in using such systems. Systems that impose additional work on someone while providing perceptible benefits primarily to others will not be accepted and successful. A case in point are many solutions for project management, which pose additional burdens on work group members, while benefiting primarily project managers [Gru88]. Other examples are workflow management systems or inventory management systems.

Most of the existing tools that support distributed collaborative work do not allow for easy switching between different modes of collaboration and communication within teams, which creates another key problem. In that they are designed to support a particular interaction mode (e.g., dyadic interaction versus group collaboration or application-based

collaboration versus face-to-face communication), they in fact create barriers to such a switching between modes and tasks [SC94].

Those tools and systems that have succeeded generally cover just a limited range of the needs of distributed teams. For example, most of them do not provide any information about the availability of co-workers or awareness about their current involvement in project-related work. In addition to this, they usually can be integrated neither easily with other third-party tools and products nor seamlessly into work processes. For example, to engage in direct communication with other team members or to engage in a joint document editing session, users normally have to invoke and manage different applications. Furthermore, while these applications sometimes provide easy, web based access to an increasing range of functions formerly available only with groupware products such as, e.g., Lotus Notes, they are inflexible when it comes to adapting and configuring them to the needs of particular users or groups. Usually, developers' kits or customising opportunities are not available.

Many of the existing Computer Supported Collaborative Work (CSCW) and team-ware systems deal essentially with virtual worlds, while physical workspaces are traditionally managed by Facility Management systems. This thesis takes an integrated approach to develop a framework for Co-operative Workplaces, seamlessly integrating real and virtual workspaces, which will be needed for collaborative work in the future. It is located at the intersection of information technology, work organization and architecture [RBB+98].

Optimising resources of people, office spaces, devices, rooms and people, are not yet integrated with current systems and procedures, but mostly managed separately. The ability for devices to announce themselves and register automatically in a global resource manager is still non-existing in real systems. There may be some initial experiments in some labs, but those are deployed in a very small scale. One of the main obstacles that were not yet addressed is related not only to the technical solutions but also to their degree of scalability.

There are multiple approaches, solutions, and commercial products available, each providing technology that addresses some of the requirements of a Co-operative Workplaces. However, there is no single solution or product, which addresses all the needs of virtual teams and that integrates all technologies together into a single cohesive and unified view. Therefore, this thesis strives for a global framework, which allows customers to customise all aspects of the solution they need, by way of selecting from and integrating available and legacy components. It aims at addressing the shortcomings of existing solutions in an effort to provide a more viable and user-friendly collaboration platform for the support of collaborative work in distributed teams.

The innovative aspect of this thesis in comparison to existing approaches is its way of composing component services to make customized value-added collaborative work environments, and to offer them to users within an environment intended to support evolving collaboration throughout the life of a project. Another innovative aspect of this thesis is its integration approach, which makes it possible to tie in existing outside software packages as well as the physical workspaces from where people tune into the Co-operative Workplace, and in a new context-oriented collaboration paradigm and user interface of Co-operative Workplaces.

Nevertheless, the existing solutions and products are not always competitors, but instead represent potential components that can be integrated within the Co-operation Platform, because of its advantage of being open openness and capable to integrate existing tools. Thus, one may somehow select the best-of breed and create an optimised platform tailored to address the requirements within e.g. a specific application domain.

As already being said, many technology pieces exist, but there is no integrated solution for the Co-operative Workplace. However, there is a lack of a suitable framework, which allows to seamlessly integrating all these pieces. Therefore, there is a need to identify what the missing pieces and technologies are that will enable the creation of a comprehensive complete solution for the boundary-less workspace.

Chapter 5

Approach

This thesis takes an integrated and interdisciplinary approach to develop future work environments. It brings together innovative work environments by making use of recent advances in information and communication technology, new concepts in work organization, and a comprehensive perspective on office building management. Attention is focussed on ways and means to enable and support flexible forms of communication and collaboration for a variety of groups ranging from small local teams to large distributed organisations, utilising a seamless integration of physical and virtual objects, of real workspaces and virtual information spaces embedded in a users work environments and managed by his work contexts.

This concept is called the Co-operative Workplace. It suggests a setting made to foster human collaboration and communication. While it serves the purpose of collaboration, it is at the same time "co-operative" towards its inhabitants. Co-operative Workplaces are an environment, which offer the people who live and work in and around it, special and unprecedented facilities for communication and cooperation purposes. They adapt dynamically as working conditions or technological conditions change. Co-operative Workplaces do provide cooperative workspaces and will improve the way people and workgroups will communicate and collaborate with each other. The symbiosis between physical and virtual work environments into one single comprehensive framework provides a more flexible and powerful way to support future work processes.

Within Co-operative Workplaces, information technology and physical work environments are highly integrated and can dynamically be re-configured in response to alternating use cases and different work situations. The basic question, which will be answered with Co-operative Workplaces, is how can future cooperative environments be configured for given work contexts, being dynamic constructions of distributed real physical objects and virtual information objects,.

Co-operative Workplaces do result from fitting rooms with given work contexts. In the simplest case, workspaces are being called co-operative if they offer an ideal setting for human collaboration and communication or if its inhabitants can interact or collaborate with the workspace. In addition, the thesis particularly studies workspaces

(real and virtual), which can interact, or "co-operate" with each other and which adapt dynamically as working conditions or technological conditions change.

Cooperative Workplaces provide workspaces, which are meant to improve the way people, and workgroups communicate and collaborate with each other over greater distances. The integration of physical and virtual workspaces into a single and coherent platform together with a unified user interface provides a more flexible and powerful way of supporting future work processes.

The overall goal encompassing this thesis is to develop a framework of a so-called Co-operation Platform, which can provide the common basis for any application case within a Co-operative Workplace. It offers configurable basic building blocks and services with which cooperative landscapes can be flexibly built and customized to user needs. Therefore, the approach is not one to develop a completely new monolithic system, rather one to provide a middleware platform, which makes it possible to incorporate already available component systems and techniques of different kinds. The building blocks are being unified representations of physical and virtual information entities, suitably encapsulated in order to exhibit homogeneous interfaces.

The framework presented in this thesis provides means for dealing with the elements of a given work environment such as its physical objects (e.g. tools, devices, furniture, installation), more virtual objects (e.g. data, files, software, means for audio-visual perception of remote people, state of a project) as well as people. The means comprise facilities to define and configure a work environment or sub-environments on one hand, and to enter, use and leave an environment on the other hand. Moreover, a platform for integrating diversified components is provided in such a way that within an application, the users are given a single window through which they can uniformly monitor, control, and work with the elements needed, and means are provided to free the users from much of the details of the used components.

There are a number of issues, which are critical for running teams of dispersed members successfully. Some of these are: fast and straightforward communications, the best possible co-ordination, reliable information, and knowledge sharing, guaranteed confidentiality and security. None of these issues are new or unique in the sense that solutions had not yet been devised for them. However, when all of this need to come together there are virtually no harmonized solutions that allow one to interlace individual services into a customized service fabric. This is the central issue addressed by this thesis and its Co-operation Platform.

Another, equally important overall goal is the provision of human centred, natural, and intuitive user interfaces for the people working in any of the envisioned new work environments. The user interfaces for different application cases needs to have as much commonality as possible and thus will be constructed from basic building blocks.

5.1 Virtual Work Environments

Virtual teams and organizations do have a strong need for openness in communication both internally and with external partners in order to make their business more effective. Communication and collaboration are among the most critical factors of success for virtual organizations. Therefore, this thesis specifically addresses the easy-to-use provision of communication channels between the people involved in the collaborative work processes, at any time and at any place. So called human centred communication services are being provided to the users, which means that they may use different networks and protocols but still join in the same communicative session. Hybrid communication does support the ability to establish connections between end-points that use different communication protocols as well as blend different media streams. Since collaborative processes often span multiple sites, multi-party distributed communication is a must.

5.1.1 The Virtual Project Office

It is commonly recognized, that computer-based group support systems can improve satisfaction and efficiency of idea generation and decision-making meetings [NBM95]. Fact is also that the physical environment plays a mediating role in this relationship and that the design of the physical environment is important to group work. What was hardly examined up to now is the relationship between physical and virtual work environments. Because of the fact that the physical environment mediates satisfaction, it seems reasonable to apply the same paradigms also to computer supported group work.

Like others, it is being believed that the adoption of physical environments allows people to work together more naturally [GR98]. More precisely, a room-based metaphor can ease people’s transitions across the gaps of today's groupware systems that hinder or block natural social interaction or that does not let people move easily between different styles of work.

The key objective of the Virtual Project Office (VPO) is to provide software that supports tele-collaboration among members of a team, communication between teams and their outside world, as well as dynamic formation of inter-organizational teams. The envisaged area of tele-collaboration is the one, which is typically found in projects, and tasks, which have to be completed in a given period of time, which are based on the division of production and cannot be precisely pre-planned but require continual coordination.

A VPO supports teams spread out and operating over different locations, being arranged on demand and re-arranged dynamically; for example inter-organizational project teams or construction projects, tele-working scenarios, workplace learning activities, virtual companies, etc. Required resources and services are being made available to the team members in the context of their work, e.g. in order to carry out topical face-to-face communications or distributed group sessions, to share project documents holding the team's results.

The Co-operation Platform offers a set of Virtual Project Offices, each of them supporting a different team, which are representations of virtual work environments, meant to give a person the impression as if s/he were in a room, where s/he can find all her/his partners as well as all necessary resources, results, files, contacts etc.

In fact, a VPO is not a real, physical office, but a view of an imaginative work environment, which is generated and accessible by modern information technology. Its purpose is to give persons working on (or with) a team the impression of moving in a room where they will find everybody currently involved in the project as well as all resources, support, results, files etc. The virtual project office is simply being used as a metaphor to refer to familiar ways of organizing work and contacting people.

The design of the VPO is centred on the requirements of users in their context of work. By a proper combination of document archives, communication and security services, it allows flexible information presentation and scalable communication. It is aimed to give users "universal" access to their (shared) work context anywhere and anytime.

The VPO software allows the user to configure and customize basic team services provided by the underlying cooperation platform rather than supporting a set of separate tools. Views can be customized according to the requirements of the team’s work contexts. The tools are being integrated in an easy to use user interface.

Besides team awareness, the VPO presents to its users a context oriented view of project contents and project members. Usually (in absence of a VPO) a project would use a variety of individual, multipurpose tools, which are not or not very much adapted to the specifics of a given project. People in the project do adaptation. Typical such tools are for example, phone, fax, e-mail, conference systems, databases, cost control software, security equipment, etc. In a VPO, the necessary tools, services, and resources are imported, related to each other where meaningful, embedded in the project context, equipped with a context oriented user interface, and thus integrated into value-added project services. Consequently, for example phone and fax calls in a VPO would be only those related to this project, conference sessions are put in the context of project data and cost, any collaboration input and output is continually matched against the project database, cost file, security rules etc. Thus, the VPO works like a well-organized and well-informed project secretariat.

A flexible spectrum of security services makes it possible to tailor a Virtual Project Office so that it can assure confidentiality for commercial reasons and guarantee individual privacy, as needed. For example, security policies for Virtual Project Offices will specify for example, their appearance to the outside world, access to them, and rules for signatures and confidential documents. Much of this is currently based on existing security tools, such as Smartcards and Public Key Cryptography. Security functions are made available to provide confidentiality, authentication, and security profiles, to define responsibilities, rights, and roles within a project and in the long run to allow contracting of experts for a team over the network.

It is possible to dynamically set up, modify, and close VPOs, as projects are set up, modified, and closed. This process is called virtual office configuration. It is based on a reservoir of generic services, e.g. for communication, databases, security functions etc.

Defining relevant work zones and associating service tools and resources with them generate an intuitive and task specific user interface. Depending on the domain of work, this reservoir and this configuration may be augmented with domain specific tools and services. For example, business projects would require the integration of business processes, e.g. by utilizing available tools that allow developers to assemble business process components. The ability to dynamically create Virtual Project Offices is an important factor in creating virtual organizations rapidly and facilitating their operation in cyberspace. A Virtual Project Office is actually the archetype of a virtual organization.

The architectural approach of the Co-operation platform makes it possible to incorporate already available tools and services of different kinds into a VPO. The central approach of this thesis is not one to develop a completely new monolithic system. It is one of integration, enabling one to pick up available and forth-coming component technologies as much as possible, in particular for reasons of economy and installed bases. Suitable solutions are assembled like parts from a construction kit to form innovative integrated solutions, which match a given work situation and possess a reduced complexity of use. The building blocks are unified representations of information entities, suitably encapsulated in order to exhibit homogeneous interfaces.

At the user interface level, the VPO is organized as a Business Club that is an environment, which offers different working zones for different types of work; for example zones for individual work and for collaborative work, zones for conference sessions and for informal information exchange. There are non-territorial workspaces. Zones are areas with specific services and facilities. Team members in the VPO would at any one time turn to the zone, which best supports, their current activities.

5.1.2 Human Centred Communication Services

In the Virtual Project Office, real-time interaction is one of the key factors to running it successfully. Real-time interaction must be carried out at any place and any time by using any kind of media so that both visual and audio communication will be able to enhance and to some degree replace real office interactions. Therefore, this thesis provides so called human centred communication services, which are independent of specific protocols, networks, media types or languages, so that the Virtual Project Office becomes flexible and works over various disciplines, uses any available communication channel and enables media rich real-time interaction. By providing a unified way of communication for all types of real-time interaction, it is possible to develop applications for the VPO with the ability to focus on the application domain rather than on the communication protocols and infrastructure.

In order to communicate with other team members or to co-operatively process a document together, ideally all a user needs to indicate to the VPO is with whom he wants to get in touch with and where appropriate, which document he wishes to process together with his teammates. Since the Co-operation Platform is provided with all the necessary information about the users' set of communication and collaboration capabilities and the underlying infrastructure (e.g. such as the phone numbers of the videoconferencing system available to the user, the supported bandwidths, etc.), it is able to schedule and establish a collaboration session without further user input – apart from the obvious

interaction steps of confirming the communication partners' readiness to enter the collaboration session now or to postpone it to a later point in time. Since the documents, which form the basis for the collaborative activity, are already present in the VPO, distribution of these documents also becomes an easy task.

5.1.3 Multimedia Conference Reservation System

A multimedia conference reservation system is provided within the Co-operation platform, which generates all technical entries required by a Multipoint Control Unit (MCU) and its internal database in conjunction with that, it manages reservations of the required facilities and team rooms. Only a small part of the data needs to be input by the user (names of the participants, the date, time and duration of the conference etc.), while a major part of the information is stored in and retrieved from the database of the cooperation platform (e.g. telephone numbers for the A/V connection, type of A/V hardware). Thus, an operator-less operation of MCUs is possible. Should no MCU resources be available for the chosen time slot, the user is being immediately informed that the conference is not possible at the chosen time. In addition to the MCU resources, any team rooms that are involved are checked for availability and are then reserved for the new conference. The newly created conference automatically appears in the conference list on the desktops of all conference participants, thus informing them about the new conference.

When the time of the conference arrives, the MCU either automatically dials out to all participants or the VPO notifies the user about a conference being started. As soon as the user has confirmed his readiness, the appropriate connections are established (again without requiring the users to remember and dial any numbers etc.). The initiation of ad-hoc conferences is achieved in the same way, simply by scheduling the start of the conference with the current date and time.

Multimedia conferencing as basic means for synchronous communication is being provided based on H.320/H.323 compliant systems. Multimedia conferencing systems based on the ITU-T H.320 standard [H320] [Ste99] are the most widespread systems today. Due to the growth of bandwidth within Intranets as well as the Internet, multimedia conferencing systems based on the ITU-T H.323 standard [H323] are also becoming more popular. In addition, in the given network infrastructure, only those systems could provide the audio and video quality the users asked for.

The compliance of the used multimedia conferencing systems with the international ITU-T standards H.320/H323 and T.120 ensures interoperability with multimedia conferencing systems from different vendors. This was an important design decision, allowing the developed system to be deployed in heterogeneous environments.

To provide multipoint communication, multimedia conferencing systems based on either the H.320 or the H.323 standard require the usage of so-called MCUs (Multipoint Control Units). Additionally, conferences between H.320 and H.323 based systems require an H.320/H.323 conferencing gateway.

Within the user community an MCU has been installed, able to handle a large number (a maximum of 24) of simultaneous conferences, each between several sites (a maximum

of 48 within a single conference), allowing users to view four, seven, 10, 13, or even 16 participants simultaneously on the same screen during a conference. Early usage experiences indicated the end-users' difficulty to successfully use multimedia conferences within their daily work. Especially the lack of direct access to MCU scheduling and control severely restricted the usefulness of the system when faced with multi-party collaboration.

Today's MCUs as well as H.320/H.323 conferencing gateways lack an (standardized) API (Application Programmers Interface), but instead require a human operator for conference reservation, set-up, and control. Another problem is that it is almost impossible for users to set-up ad-hoc conferences. When a user wants to have a multi-point conference with others, he often has to ask an MCU operator to schedule it – often well in advance. This can be very time-consuming (it may take minutes to hours depending on the availability of the operator). In addition, because of the way MCU-controlled conferences are typically set up, a multipoint multimedia conference cannot be resized dynamically in terms of time and number of participants, i.e. once a conference is running, further participants cannot attend unless their participation was anticipated at reservation time.

These experiences led to the decision to provide the users with an easy to use multimedia conference reservation system. The developed Multimedia Conference Reservation System (MCRS) provides the following functionality:

- Advanced conference reservation as well as on-demand reservation and set up
- Conference administration (add, remove, change and display conferences)
- Resource reservation, including:
  - Reservation of MCU resources (on H.320 as well as H.323 based MCUs)
  - Reservation of H.320/H.323 gateways (if required)
  - Reservation of multipoint application sharing servers (e.g. for joint viewing and editing of documents, if required)
  - Detection of conflicts (e. g. unavailability of required resources)
- Creation of conference repositories
- Confirmation for the convener and invitation of the selected participants

For spontaneous multimedia conferences, all the user typically needs to do is to select the other user(s) with whom he wants to collaborate. The MCRS uses its knowledge of the users and their available infrastructure to initiate the (possibly multipoint) multimedia communication. Pre-planned conferences are set up in the MCRS by selecting a set of users (or user groups), entering a conference topic, selecting a conference time and date.

Due to the interactive collaborative nature of the MCRS, the scheduled conference automatically shows up on all users' desktops in order to inform them about this conference. When the time of the conference has arrived, the users are being reminded of this in advance and asked to indicate when they are ready to enter the conference. Upon confirmation, MCRS can automatically start the necessary A/V conferencing tools and

establish the connection to the other user (by using the provided MCUs, in the case of a multipoint conference, and/or conferencing gateways).

5.1.4 Application Integration

Using the provided Co-operation Platform, the users are able to initiate and conduct their collaboration tasks in a manner, which is appropriate to their current work context and the problems they actually need to tackle. For the system to start the necessary tools, it is sufficient to simply indicate the collaboration partners and the corresponding work context. The users are relieved from the effort of having to control (and learn) a number of separate applications and can concentrate on their actual work processes. The provision of a framework for the integration of existing, off-the-shelf tools, and applications and the information distribution aids integration of the technology into existing work processes.

For example opening a document within a collaborative session indicates the user's desire to present this document to the entire group of connected users or to edit it cooperatively with them. The VPO supports this activity by automatically launching not only the required application but also the application-sharing component – if necessary, on all connected machines – and bringing the document's application into the application sharing session ready for collaboration. This support relieves the users of a large part of the burden involved in setting up the conference connections, the required applications, and the application sharing system. Should a document be editable or presentable by truly collaborative software such as the DOLPHIN [SGHH97] system, which does not require the additional application sharing support, this integration can also be made available. Again, since the collaboration platform has access to all the necessary information about the collaboration situation, the external collaborative application can be invoked on all sites, receiving the required parameters for connection and set-up from the collaboration platform.

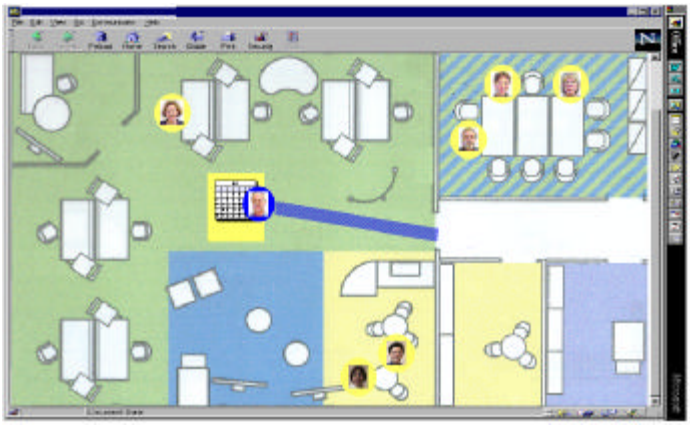


Figure 2 Sample VPO user interface

5.1.5 User Interface

The user interface provided refers to the room as a place where people meet and collaborate on a common subject. More precisely, a room metaphor can ease people's transitions across the gaps of today's groupware systems that hinder or block natural social interaction or that does not let people move easily between different styles of work.

Within the Co-operation Platform, geographically distributed members of a virtual organization are being projected into a Virtual Project Office as if they were all physically present in a single office. That is, the Virtual Project Office is used as a metaphor referring to familiar ways of organizing work and contacting people (Figure 2).

5.2 Physical Work Environments

The effect of new forms of work and work organization on offices are that rooms and physical workspaces, which used to be dedicated to individuals are more and more transformed into rooms for teamwork and various forms of communication. These new office types are being equipped with information and communication technology and arranged as a Business Club, an environment that provides specific workspaces for each type of activity, e.g. multimedia conferencing rooms and electronic meeting rooms for teamwork.

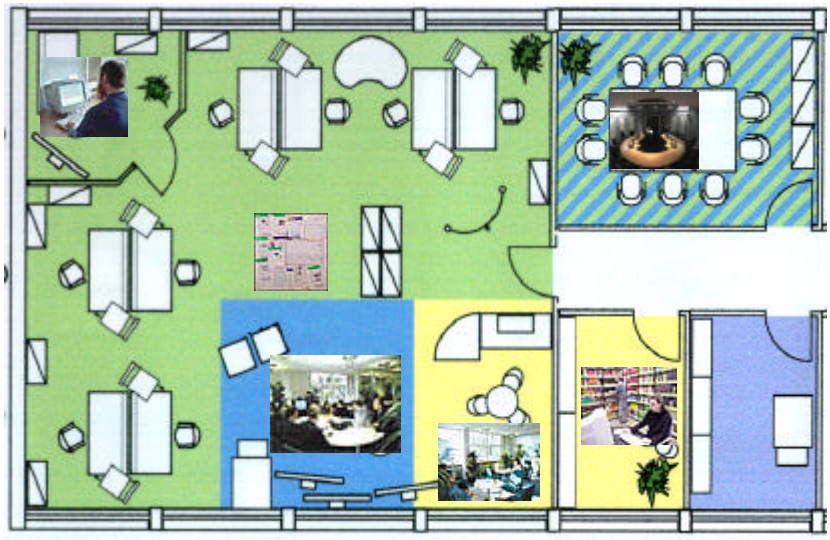


Figure 3 Sample SCLUB office layout

The Science Club (SCLUB) is such an office landscape, which has been set-up for in-company usage in combination with alternating tele-working. It provides a set of different non-territorial physical workspaces tailored to different kinds of activities. In particular, the Science Club offers:

- zones for individual concentrated work,
- zones for joint work in teams,

- zones for presentations, seminars and lectures,
- zones for the work with literature, and
- zones for informal communication.

The Science Club represents an advanced form of the Business Club, particularly designed for research and development work. On the one hand, it is designed to increase the productivity of highly skilled and qualified specialists, which are organized in project teams. It provides dynamically configurable work areas, which can be tailored to specific user needs and between which users can change depending on their work context and actual needs. On the other hand, it brings considerable cost savings through an optimal use of space and resources.

Simultaneously, the SCLUB serves as proving ground wherein results developed within the framework of this thesis have been tested, can be used in practice, and feedback be generated for possible improvements. In this sense, the SCLUB is an environment, which can permanently advance to meet progressive demands on (tele-)cooperation, mobility, flexibility, and dynamic reconfiguration.

5.2.1 CoMeet - The Cooperative Meeting Room

Meetings play an important role in teamwork. It has been shown in the past that the provision of meeting support systems and electronic meeting rooms achieved better and faster results in group problem solving than without [SGHH94]. Therefore, an innovative meeting room, the so-called CoMeet-Room, equipped with advanced digital information and communication technology has been provided within the SCLUB. In particular, the room supports communication and cooperation of individuals and groups during meetings and conferences. The application scenarios range from meetings of local participants with a maximum of information technology support to conferences with remote participants connected via modern communication technology.

Figure 4 CoMeet room

The CoMeet room takes into account that meetings, in particular face-to-face meetings, play an important role in teamwork, and thus creates a physical work environment for synchronous joint work of teams, including remote (virtual) partners.

CoMeet combines innovative information and communication technologies and considerably contributes to the co-ordination of the technical equipment available at the individual workspace and the equipment of the meeting room to avoid media breaks. It is designed for daily use within the ScienceClub and for far-reaching and trend setting R&D activities in the field of telecooperation [RBB98].

Despite its high-standard technical equipment, CoMeet has a pleasant ambience and room climate, as technology does not dominate. Experience from previous developments has shown that users do not accept studio-like rooms with a predominant technology. Furthermore, operating high technology equipment is too complex for the average user so that an operator is required. Therefore, the technical equipment of CoMeet is kept in the background, monitored, and controlled by a central RoomComputer. This still requires some research and development work in the field of ergonomic conferencing control technology with intuitive user interfaces.

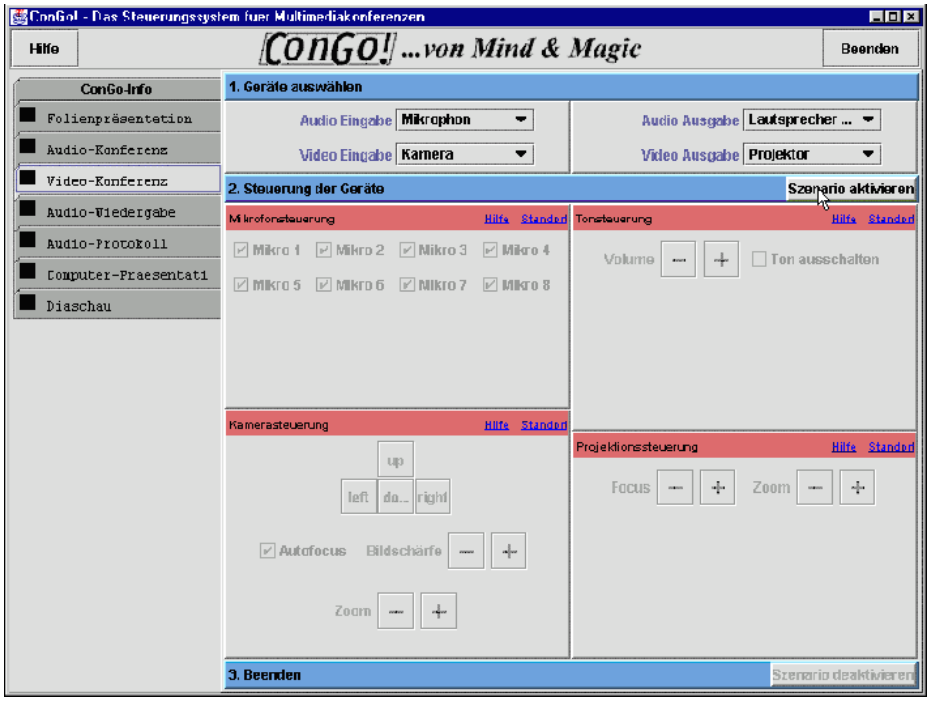


Figure 5 CoMeet room media control system

CoMeet reflects the state of the art in teleconferencing technology and digital audio and video. Audiovisual communication with remote partners and groups is done via audio and videoconferencing systems based on Internet and ISDN technology. Suitable codecs are used as interfaces to the remote video conferencing systems. MCUs (Multipoint Control Units) as well as IP multicast are used for teleconferencing with more than two participants. Those participants who lack a suitable video conferencing system can

participate by using a regular phone-set. Conferences are being set-up and managed using the MCRS system (see above).

During teleconferences, large flat screens are used for the visualization of remote groups or individuals and for the presentation of video and other electronic information. A so-called SmartBoard is used like a conventional blackboard for presentation and simultaneous input.

Each participant is provided with a laptop connection facility over a wireless LAN. The laptop can be used, for example, as a private notebook or for the presentation of meeting documents. Laptops with penpads or Handheld PC's can be used for pen-based input. Instead of the SmartBoard, any local system can be used for input and reproduction on a local presentation medium. Non-electronic meeting documents, such as transparencies, slides, photos or paper documents can be digitised by means of suitable equipment like a document camera, a digital slide projector, a scanner, and a digital flipchart.

The digitised information can either be processed, presented and distributed directly or can be stored in a multimedia document repository. For importing and exporting meeting documents, a multimedia PC is provided. Remotely controllable cameras are used for visualizing individuals or groups as well as other objects. Digital audio and video recorders work as additional sources for audiovisual information. Audio and video crossbar distributors are being used for mixing and/or changing between different audio/video sources.

A central Web-based control system has been developed to monitor and control all the media devices and technical components being used in the meeting room, both locally and remotely. Its user interface is being shown in Figure 5 . The CoMeet control system seamlessly integrates with:

- the VPO, i.e. it is aware of the users work context and scheduling a meeting from within the VPO might result in a reservation of the CoMeet room together with an appropriate configuration of the media devices within the CoMeet room
- the RoomComputer, i.e. media devices are being controlled via the RoomComputer
- the MCRS system, i.e. tele-conferences are being set-up and managed via MCRS

Control functions are automated to make the manual operation of sometimes complex technical equipment and infrastructure more transparent, reduce it to a minimum, and provide a work context oriented rather than a device oriented view to its users.

5.2.2 Location Awareness – The Sm@rtLibrary

Location based services and therefore location management is playing an increasing role. This is valid also for the Co-operation Platform, and in particular for e.g. the reachability and communication services provided, i.e. in order to e.g. establish a communication link between two users, it has to be known where they are located physically and what type of communication devices are available within their surrounding or vicinity.

The Sm@rtLibrary addresses the need for locating people and devices together with the need for privacy protection [MRH01]. Within the SCLUB it represents the library section and serves as a test and demonstration environment for location-aware and ubiquitous technologies and their application. In this area, this thesis has been concentrating on localisation and location management technologies, context-dependent services, platform, and device independent access to the services and information provided in such environments.

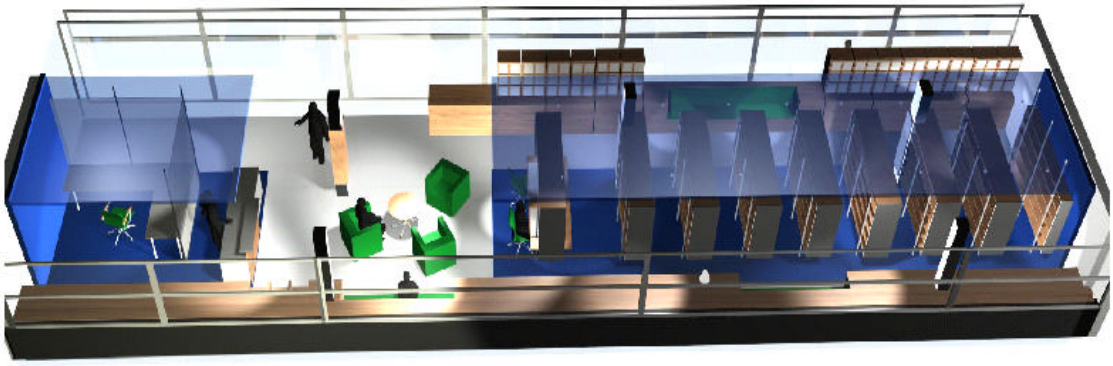


Figure 6 Sm@rtLibrary

To be able to deliver arbitrary information on physical objects, such as e.g. books or devices, it is necessary to identify them. To achieve a secure identification every physical object has to be equipped with a Sm@rtLabel, which can be identified using radio frequency based identification device (RFID) readers placed in physical workspaces like the Sm@rtLibrary.

Figure 7 Sm@rtAssistant user interfaces

Before people can enter the Sm@rtLibrary, they have to identify them using their digital Office Identification Card (OIC) and holding it in front to a contactless smartcard reader attached to the RoomComputer. The library user can now take one of the freely available Sm@rtAssistants, which are flexible PDAs with an integrated RFID-reader and have wireless access to the local network (by either using wireless LAN or Bluetooth). By approaching the Sm@rtAssistant with his digital office ID card, the user interface on the PDA is being personalised automatically to the needs, preferences, and roles of the user and his possibilities based on his current spatial location.

To adapt the user interface of the Sm@rtAssistant and the information shown on its screen according to its current location, the location of the device has to be accessible by the applications. This information is made available using the RoomComputer infrastructure.

5.2.3 World Wide Facility Management

Facility management is the practice of co-ordinating the physical workspace with the people and work of the organisation. It integrates the principles of business administration, architecture and the behavioural and engineering sciences. Although the facility management profession has been in existence since the evolution of the office, only in recent years has it received worldwide recognition. Business entities have come to realise that maintaining a well managed and highly efficient facility is critical to success. New technologies, environmental consciousness, and health concerns also have had a major impact on the importance of and need for facility professionals in organisations.

The term facility management is often used to refer to a variety of activities ranging from control of facilities within in individual rooms up to the management of buildings, business plants, real estate etc. Here, the management of facilities in buildings and rooms is being addressed mainly, which is traditionally backed by (more or less centralized) systems and databases. Within this thesis, a different approach will be followed, namely an approach from the side of data communication and data networks, and such has Internet/Intranet. It is based on the availability of ubiquitous universal networks (such as Intranet/Internet) and inexpensive computing power (e.g. provided by embedded systems).

Facility automation systems can make physical workspaces more comfortable, safe and efficient by integrating systems such as heating, air conditioning, lighting, security and telecommunications into one centrally controlled, automated system. In the old days, facility automation simply meant a couple of switches. Today's definition of room control encompasses a surplus of advanced pieces of environmental and multimedia equipment ranging from e.g. lighting control systems to VHS players, LCD projectors, and others. Of course, simply equipping a room with advanced electronics is not enough. Making a room that is easy to use and versatile enough to keep pace with quickly evolving technology is the goal within this thesis.

In modern facility automation, many devices are providing services and interacting with each other. It is necessary that these devices can be simply networked with each other in order to be able to provide the desired services in a flexible manner to a potential

user. By connecting these devices to the local Ethernet and integrating them with software applications, buildings, and therefore physical workspaces, can become active structures that receive signals from diverse controlling and monitoring devices and then send them to the appropriate appliances and systems in the building. The result is a facility automation system that allows users to do everything from setting lighting, temperature, controlling energy usage, and adjusting the physical workspaces according to their current work context via the Intra-/Internet by using e.g. a standard web browser.

Unlike the computer industry there has not been a generic foundation or framework for software developers to build on in the facility automation industry. Most of the software applications come from the supplier of the automation hardware. One reason for this is that, the facility automation industry is permeated with many different field bus systems and communication protocols. There are only a few standard field bus network solutions or software object standards. Most of the manufacturers see their proprietary solutions as their competitive advantage. Today the proprietary nature of these systems has become a problem for everyone.

Because of the fact, that devices made by different manufacturers are often unable to communicate with each other, integrated building automation systems are difficult to create. One trend in the building control industry has been to attempt to solve the integration problem using limited open or “standard” communication protocols, which would allow developers and users to integrate systems regardless of their manufacturer.

At present, the movement to this completely open environment for building automation still has hurdles to overcome. Several initiatives to create standard communication protocols in building control devices (such as e.g. EIB, LonWorks and BACnet) have created opportunities for constructing open system environments. However, these protocols, which are focused at the embedded device level, only provide interoperability when all of the devices (e.g., controllers for air conditioning units or security systems) are based on the specific communication protocol. So while interoperability is achievable, it is segmented by the technology of the underlying hardware – this is not a solution for the end-user.

These emerging open system protocols have seen some success as an option for newly constructed buildings as long as system designers insure that all installed systems come from manufacturers who have embedded the chosen protocol in their hardware. New buildings, however, account for about one to two percent of the facility automation market. Older buildings, which account for a majority of the market, come with the previously mentioned legacy issues, which are very complex. Consequently, building owners choose to maintain their legacy systems rather than replacing them, and therefore are forced to work with the limited individual applications provided with these legacy systems.

In order to break this cycle, a standard framework and software architecture for building automation systems is needed which:

- allows software developers to develop applications that can be used across a wide variety of systems, including competing systems from different manufacturers and different types of similar systems;
- embraces the emerging standard field bus protocols (BACnet, LonWorks, EIB, etc.) while at the same time giving full support to the existing legacy systems;
- is built on open Internet protocols and standards.

Today, the movement to software based on widespread open standards and protocols has profoundly affected the way applications are being created. The Internet has spawned a wealth of reusable software components and consequently a multitude of standard frameworks, which are being used to leverage these components. Most notably, Java-based frameworks are beginning to provide suites of application components, or APIs, for the development of Web-based business applications. Just as the standards movement of the Internet made it uneconomical for software developers to build their own protocol stacks, Java frameworks are making it uneconomical for developers to create all of the software components necessary to tie business logic into transactional databases and Web servers.

With the development of the RoomComputer, this thesis is trying to overcome the existing problems by creating a framework, which provides communications with different field bus systems and protocols, both emerging standards and legacy protocols. This framework creates an abstraction layer that takes all of the data from the diverse systems found in buildings, and “morphs” them into a standard software object model that supports different types of devices and provides standard Java-based API’s for interacting with them. This gives software developers full access to all of the information and commands available in the target devices, enabling them to develop software applications that work for all of the systems that have been brought into the framework.

With the RoomComputer technology, a building can be seen as a system of physical workspaces, connected via Intra-/Internet technologies. These spaces can be loaded with different kinds of services:

- Facility automation
- Facilities management
- Telecommunication
- Tele-cooperation
- E-commerce

Users of physical workspaces identify themselves against such spaces, i.e. by using their digital Office Identification Card. Dependant on their profiles and work context, physical workspaces can then be configured automatically towards the needs of their users. Based on the RoomComputer Technology, the World Wide Facility Management (WWFM) system integrates physical workspaces and physical objects into the Co-operation Platform. That makes it possible to monitor, control, and manage both old and

new buildings on a unified worldwide platform, irrespective of any particular local technology.

Via the WWFM system, several of the distributed RoomComputers can be networked, interconnected on the Co-operation Platform, and thus embedded in a Virtual Project Office for individual room control. It is also possible to administer a set of rooms, or buildings, and to cluster rooms into a virtual building under a unified application view.

On the one hand, the Virtual Project Office integrates a collection of physical rooms within a single user interface. On the other hand, physical rooms can be configured via the WWFM system and the RoomComputers from within the Virtual Project Office, regarding to the specific environmental conditions, resources and the context of work required by the members of a team.

5.2.4 Managing Physical Workspaces – The RoomComputer

The approach being followed in this thesis to address the need to integrate and configure the physical workspace within the co-operation platform, is to develop a smart device which can be installed in any room and gives access to a full set of services for that room: the so called RoomComputer. It is an embedded system and as such offers unprecedented chances within the Co-operation Platform to administer distributed physical work environments of people from virtual ones and vice versa. The openness and expandability of the RoomComputers allows technological advances to be incorporated more quickly, and everyone is able to master it.

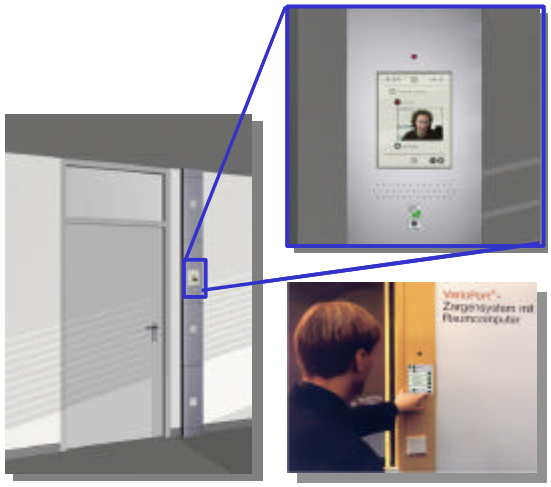


Figure 8 RoomComputer

The RoomComputer is an autonomous installation unit (typically located in the doorframe of a room), which is connected to the Intranet/Internet. It is based on an embedded PC with a touch sensitive LCD display as local interface and has integrated loud speakers and a microphone. A Smartcard reader and devices are being attached to it.

RoomComputers create the preconditions for locating, adjusting and administering physical objects such as rooms, equipment, and other resources in a given work context. They implement a distributed application, which provides room services like:

- control of light, heating, ventilation, air and climate,
- communication facilities like telephone, fax, etc.
- Internet/Intranet access,
- reservation of rooms and required resources,
- localization of persons and equipment within rooms and buildings,
- organization of maintenance and house keeping, and
- charging and billing.

With RoomComputers, a facility management system simply becomes a network of rooms, linked together via the Intranet or Internet to form a facility management system. An expansion to include more rooms and resources can easily and inexpensively be done by adding RoomComputers and connecting them to the Intranet or Internet. The deployment of RoomComputer networks together with smart controllable room devices enables the development of compelling new facility management applications, which moves closer to the vision of the Co-operative Buildings that enhances users' comfort, convenience, and ease of use.

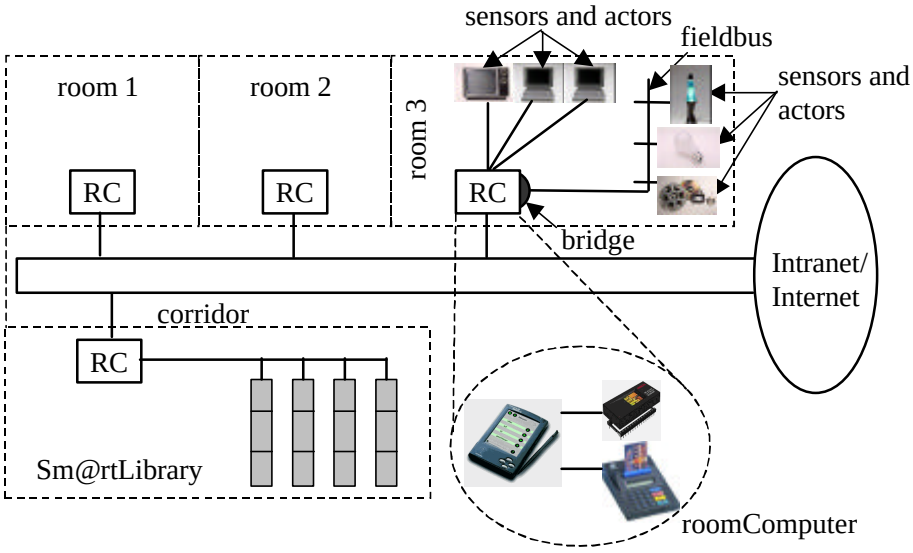


Figure 9 Sample RoomComputer installation

Since each individual room has a RoomComputer of its own assigned to it, this makes it possible to autonomously control them from the Co-operation Platform and therefore all the devices they contain, in an efficient way. Several of these RoomComputers can be networked via the Intra-/Internet, which makes it possible to administer a set of rooms, or buildings, and to cluster them into a virtual building under a unified application view. All the steps required to set up the appropriate work environments and to establish the necessary network links between distributed users are then automated in a way that communication and collaboration in virtual teams is just a mouse-click away.

5.3 Security

People collaborating in virtual workspaces are not always physically in contact with each other. Thus, authenticity becomes highly important. Actually, authenticity is being taken, as the base for further security services, like role-based access control and auditing. Access control manages the capabilities of users (who can read document X?, who is allowed to enter room Y?, etc.). Auditing is responsible for logging user activities.

Within most organisations, security is generally not very restrictive so secure collaboration is not difficult to achieve. One internal group is often responsible for dictating security policies, administering security, and deploying a Public Key Infrastructure (PKI), which, although not trivial, is not difficult within a single organisation. Firewalls are usually deployed at the perimeter of the network and will (mostly) affect neither internal collaboration nor the use of mobile code (e.g. Java applets, software agents).

In contrast, securing a Co-operative Workplace is more challenging when a group of organisations agree to collaborate. Many questions arise when attempting to secure collaboration across extranets. Who will administer the root CA? How are firewall requirements addressed? Every organisation has different policies for security, Public Key Infrastructures, firewalls, and mobile code that impacts joint collaboration.

A Co-operative Workplace that is secure, presents new and unique challenges. Good security is based on the concept of least privilege, where people are given only the privileges needed to perform their assigned tasks—and no more. Collaboration enables anyone to freely exchange information with anyone else. This dichotomy of sharing information freely and restricting its access to certain groups of individuals is what is challenging in the virtual environment.

In an ideal environment, secure collaboration enables people to authenticate the identity of the person(s) with whom they are collaborating. Secure collaboration also prevents eavesdropping by securing the communications channel and protects the integrity of the communication. Finally, the environment should make it easy for users to establish and enforce need-to-know. The most significant challenge to overcome is user presence and authentication. User presence and authentication is less a problem in one organisation, where the policies are controlled in-house. However, problems arise when different organisations, each using different tools, try to integrate them. Given the

opportunity, most users want to collaborate with others outside of their organisation. Without standards-based tools that support presence and authentication each organisation will continue to operate as an island.

An innovative aspect on security within the context of this thesis is that it is applied to teams, as opposed to individuals. Team members of these teams usually are dispersed around the world, working for different companies with different security policies. Building a virtual team has to integrate available security techniques and infrastructures to fulfil these different needs. Any virtual project office is provided with its own particular security functions controlling access to zones and to documents.

Throughout the Co-operation Platform, security services are being realized as system inherent services, running in the background whenever possible so that users can concentrate on their work rather than on security measures [FP97] [Gri97a]. A single sign on mechanism is being used, i.e. a user must prove his identity only once. The platform verifies his identity and is enabled to prove the identity to the VPO, whenever needed.

All security measures are based on cryptographic techniques; passwords are never being transmitted in clear. A PSE (Personal Security Environment) can be used to store private keys and sensitive public keys of a user. An encrypted file can implement a PSE in the local file system of a user, referred to as software PSE, or as hardware PSE by a smartcard, like the digital Office Identity Card. A secure storage of the PSE is only possible only on a smartcard. Standards (e.g. X.509 certificates, authentication protocols, encryption algorithms, etc.) are used wherever possible.

5.4 Semantic Model

This section describes the Semantic Model of the Co-operation Platform, a formal model on which the design and architecture of the platform are being based on. It defines the basic actions a user of the platform will be able to perform, and thus also provide a basis for the design of its user interfaces.

As a basis for the Co-operation Platform a common semantic model for the platform itself and its user interfaces in particular has been developed. It captures the basic functionality of the Co-operation Platform in a widely device and tool independent manner. Based on this semantic model, concrete user interfaces have been designed for a set of end-user device.

The semantic model underlying the Co-operation Platform and its user interfaces is concerned with *what* functions are available and visible in a specific situation to the user, rather than *how* they are presented. It captures static and dynamic relations between persons, information objects, and services with regard to presence, past and - as part of further plans - future.

The purpose of the semantic model is to bring out and clearly define the elementary logical constituents and rules for consistently putting them together to make a variety of more or less complex and customized team environments.

5.4.1 The Notion of Work Context

The basis for approaching the semantic model is the notion of *work context* being composed of virtual and physical objects related to projects, persons, tools, and rooms and integrated under a common user interface.

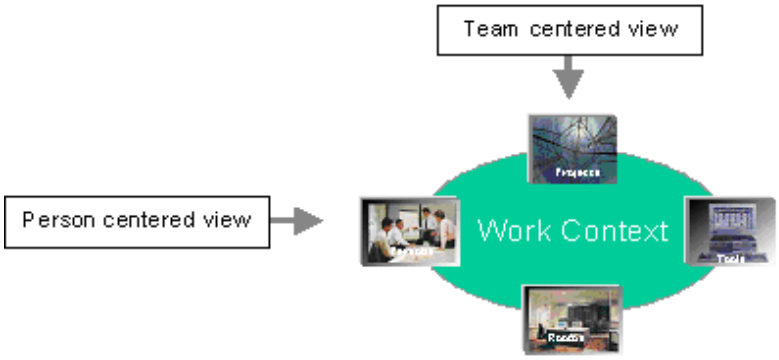


Figure 10 Co-operation Platform work context components

Two sets of components or work context types are being differentiated: *personal work contexts* and *teamwork contexts*. These are two ways of looking at work contexts:

1. In the first case, an individual person is put in the centre of interest and is regarded as working on zero or more projects using tools and rooms. A personal work context is uniquely associated with an individual.

2. In the second case, a project is put in the centre of interest and regarded as being worked on by one or more persons (i.e. a team) using tools and rooms. A teamwork context is uniquely associated with a team, which is a group of individuals working together on a project, in order to reach an agreed upon common objective. Within the framework of the Semantic Model, a *project* is an undertaking by a team and we use the terms teamwork and project work synonymously.

The relation between project and team is unique in that for any project there is exactly one team working on it. Other cases are for further study, such as for example the case that the same team works for two or more projects, which can be modelled by two virtual teams, each working for one project. The case that two teams work on the same project can be modelled by either virtually merging the teams or substructuring the project into subprojects.

Personal work contexts and teamwork contexts partly overlap, when individuals become members of a team.

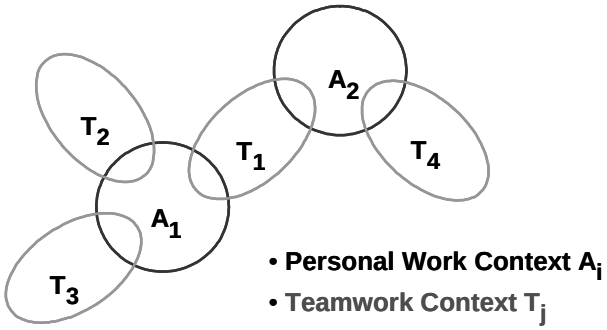


Figure 11 Overlapping of personal work contexts and teamwork contexts

Figure 11 shows two personal work contexts, A_1 and A_2, and four teamwork contexts, T_1 to T_4. The personal work context A_1, which is associated with P_1, overlaps with the teamwork contexts T_1, T_2 and T_3, which means that the individual P_1 is a member of the corresponding teams. The personal work context A_2, which is associated with P_2, overlaps with the teamwork contexts T_1 and T_4, which means that the individual P_2 is a member of the corresponding teams.

When personal work contexts and teamwork contexts overlap, new work context areas emerge (see Figure 12 below), which are defined as follows:

B_i: Personal work context A_i AND NOT any team work context;
it is owned by individual P_i

$C_{j,i}$:Teamwork context T_j AND personal work context A_i;
it is owned by individual P_i

D_j: Teamwork context T_j AND NOT any personal work context;
D stands for a set of collaboration scenarios which are owned by the entire team T_j

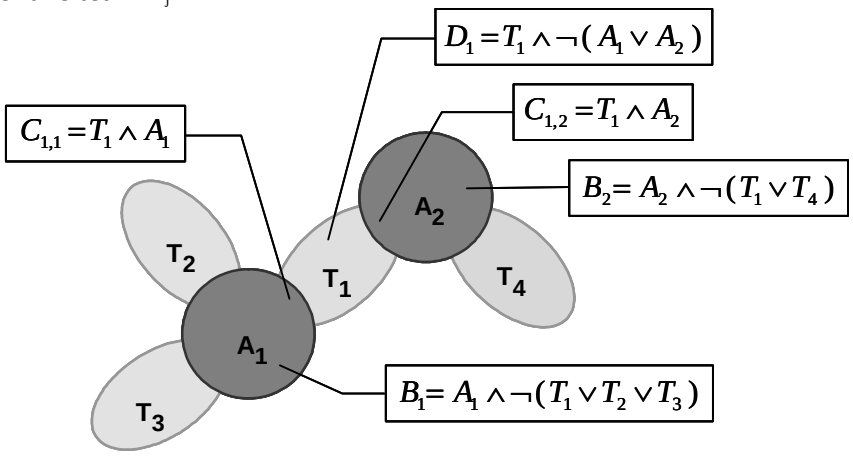


Figure 12 Work context areas

The meaning of these work context areas is depicted in Figure 13 below and further discussed in the following sections.

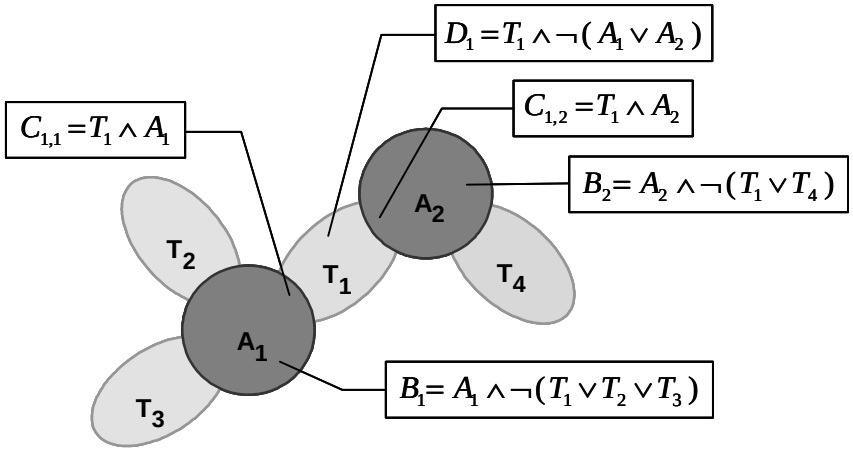


Figure 13 Meaning of work context areas

5.4.2 Working within Work Contexts

The basic idea and motivation for distinguishing personal work contexts and teamwork contexts and structuring them further into work context areas as described above is to enhance the efficiency of IT-supported work, and in particular, of IT-supported teamwork, by grouping under unified interfaces all the virtual and physical resources which are needed to produce work results.

The model distinguishes users outside the platform from users inside. It assumes that a user (i.e. a real person) is represented inside the Co-operation Platform by his *Personal Agent*, which is a system component that acts on behalf and under control of the associated person. A prerequisite for a person to work in any of the work context areas is that he has been assigned a Personal Agent (so the Co-operation Platform can recognize him) and a personal work context to go with it (the Personal Agent carries that information).

A person P_i, who has been assigned a personal work context A_i, can at any single point in time, work in either of the following work context areas in different capacities, namely:

- in B_i as its owner, or
- in $C_{j,i}$ as its owner and member of team T_j,or
- in $C_{j,k}$ as member of team T_j with permission of its owner person P_k, or
- in D_j as member of team T_j.

Generally, within a single teamwork context users are not differentiated with respect to roles and rights. Conceptually, if there were groups of users with different rights within a single teamwork context, they should be modelled as teams in separate teamwork

contexts inside a nested project structure; but this is beyond the scope of this thesis. However, the Co-operation Platform provides for temporary "ownership", which is to avoid conflicts of access to non-sharable pieces of information. Similar to a semaphore technique, the owner of information has to release it before somebody else can get at it.

5.4.2.1 Working in Personal Work Context Areas

Since the Co-operation Platform focuses on team environments, personal work areas are only modelled to the extent necessary to establish a "home" for entering and leaving team environments. Any work within a personal work context area B_i is invisible to any teamwork area. The main feature of B_i is that it offers a number of edges of teamwork contexts, to which a person may transfer (see sections 5.4.3 and 5.4.4). However, a person in work context B_i can neither observe what is going on in these teamwork contexts, nor can he access anything from them without leaving B_i. This is of importance with respect to access control.

Moreover, the personal work context area B_i may be equipped with a set of services and tools, which are normally found in teamwork context areas (see below). But it is important to note that within A_i and B_i, they can only be applied to private, not to project related information resources. Of particular interest are services and tools, which support B_i's owner in organising his personal work. e.g.:

- a private in-box for messages which are addressed to him personally, outside of any teamwork context;
- a private repository for documents;
- a private address book;
- a private calendar.

5.4.2.2 Working in Teamwork Context Areas

A teamwork context T_j is equipped with a set of services and tools, which allow team members to collaborate with each other and to work on project related information resources, but not on private ones.

A teamwork context T_j is structured into areas $C_{j,i}$ and D_j. There are as many C's as there are members and one or more D's, depending on a particular project. There are team resources, which are only accessible in certain areas, e.g. video conferencing might only be available in a work area D_j which provides the necessary facilities. Additionally, there are resources, which are accessible from anywhere inside T_j, for example the project document repository.

Within a teamwork context T_j, each team member P_i owns something like a *personal desk* (work context area $C_{j,i}$). This is the person's "base", with functional attributes such as the following:

- Other team members can expect to find here information on its owner, e.g. when he is absent, or to leave a message / information for him.
- This is the place where person P_i does individual project work (e.g. access and process documents, send mails, etc). Access to team resources within T_j is not restricted when here.

- The owner may leave here documents to indicate he is currently working on them or wants to resume work later.
- Here is the owner's personal project in-box for all messages, which are addressed to him as an individual inside the current teamwork context.
- The owner, when in, may be contacted here by another team member (e.g. someone is calling to ask a question). He can accept or refuse. Conceptually, when he accepts, a communication with another person is started in a collaboration area D_j. (Switching back and forth between areas should happen implicitly.)

Personal desks represent those parts of the teamwork context that are under control of individual team members. Ownership means that other team members may need to have the owner's permission to read or change information placed here. Owners are free to decide who else may interact with them when they are here and about what. The extent to which things are put under owner control is configurable and project dependent. The basic platform will not support owners' control.

Besides personal desks for each team member, a teamwork context T_j offers collaboration resources (work context area D_j) which are shared among all team members and which support them in organising and performing their team work, in particular:

- a project in-box for all messages which are addressed to the team as a whole inside the given teamwork context;
- a project out-box storing all messages which were sent from the given team work context;
- a project repository for documents;
- a project address book;
- a project calendar and
- a predefined set of collaboration areas, which each team member (with the required access device) may enter freely (meaning, that there is no distinction of roles and access rights, at least for the time being). Each collaboration area is composed of basic communication services and supports a specific kind of interactions among team members. Sometimes, access to a collaboration area may be restricted, not because of user rights, but because of a lack of resources (e.g. limited capacity, lacking facility at a user site, etc.)

5.4.3 Switching Work Contexts

This section and the following one discuss the rules for switching between work context areas. This (more formal) section covers the roadmap and possible routes between areas, while the next (more semantic) one takes up user authentication and information transfer.

A person P_i can switch between work context areas according to the following automaton, where O stands for all work contexts outside the Co-operation Platform (see Figure 14). What the graph shows, are the elementary steps; what a user sees, may well be a higher-level macro, built from these elementary steps.

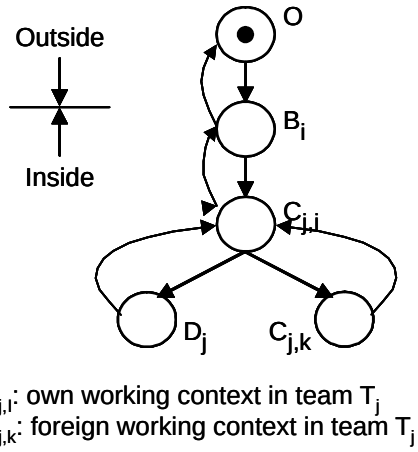


Figure 14 Automaton describing work context switching for one user

Such an automaton, which describes a person's current state at any point in time, can be assigned to each individual.

To express dynamic relationships between persons, e.g. to indicate who is working with whom in which teamwork context, the states and state transitions of different persons have to be related to each other. For this purpose, Product Nets are being used [BOP89]. Product Nets are labelled Petri Nets with individual tokens and inhibitor and erase arcs, which were designed specifically for formally describing co-operating systems. A Product Net consists of

- a preface, in which the sets and functions, subsequently used are defined;
- a net N = (S, T, F, I, E) with places S, transitions T, flow relations F, inhibitor arcs I and erase arcs E;
- mandatory labels
  - at places, serving as unique identifiers for the assignments of token structures
  - at arcs, which need to be consistent with the token structure of the places to which they are connected;
- optional inscriptions of transitions;
- an initial marking.

The following Product Net (see Figure 15) describes globally all possible states and transitions for all users of the Co-operation Platform. It can roughly be regarded as superposition of individual automata as shown in Figure 14 above.

The model distinguishes users outside the Co-operation Platform from users inside. A global state is represented in a Product Net as markings of its places, i.e. as a distribution of an arbitrary number of tokens onto places.

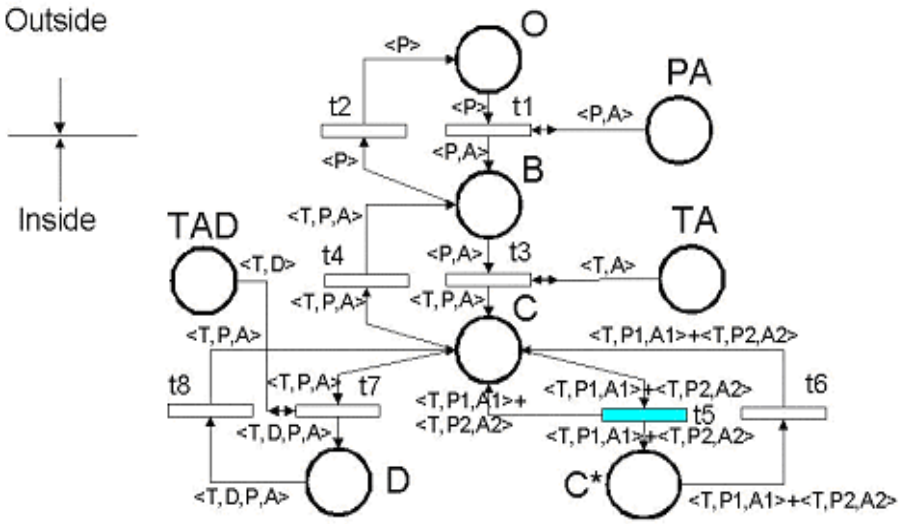


Figure 15 Product Net describing the work context switching for all users

The above net contains the places O, B, C, C*, and D, which correspond to the states in the automaton above:

- O is the place putting up tokens <P> representing persons P_i, who are currently working outside the Co-operation Platform;
- B is the place putting up token <P, A> representing persons P_i, who have been assigned personal work contexts A_i and are currently working in work context area B_i;
- C is the place putting up tokens <T, P, A> representing persons P_i, who have been assigned personal work contexts A_i and are currently working in work context area $C_{j,i}$ of teamwork context T_j;
- C* is the place putting up pairs of tokens <T, P, A> representing persons P_i, who have been assigned personal work contexts A_i and are currently working in work context area $C_{j,k}$ of teamwork context T_j which is controlled by persons P_k, who have been assigned personal work context A_k;
- D is the place putting up tokens <T, D, P, A> representing persons P_i, who have been assigned personal work contexts A_i and are working in work context area D_j of teamwork context T_j; D may comprise a set of collaboration scenarios as defined by tokens <T, D> in place TAD.

The net contains further the places PA, TA and TAD, the markings of which reflect (as part of the initial markings) the configuration of the Co-operation Platform:

- PA is the place putting up tokens <P, A> representing the assignment of personal work contexts A_i to persons P_i;
- TA is the place putting up tokens <T, A> representing the assignment of personal work contexts A_i to teamwork contexts T_j; (Note: In general,

only part of a person's personal work context will be assigned to a team, as illustrated in Figure 11 .)

- TAD is the place putting up tokens <T, D>, each representing a distinct collaboration scenario D_n supported within teamwork context T_j.

The occurrence of an "enabled transition" transforms the marking of the places connected to it, from a predecessor marking into a successor marking. A transition is enabled under a marking if each of its input places contains tokens which fulfil - with regard to number and values - the labels of the corresponding input arcs and if none of its inhibitor places contains tokens which belong to the inhibitor sets defined by the labels of the corresponding inhibitor arcs. It is important to note that an enabled transition may, but not necessarily needs to occur.

The above net comprises the following transitions

- t1, t2: their occurrences model that a person P_i enters /leaves work context area B_i within its personal work context A_i; note, that only persons who have been assigned a personal work context, can enter the Co-operation Platform;
- t3, t4: their occurrences model that a person P_i, who has been assigned a personal work context A_i, enters /leaves work context area $C_{j,i}$ within teamwork context T_j;
- t5, t6: their occurrences model that a person P_i , who has been assigned a personal work context A_i, enters /leaves work context area $C_{j,k}$ within teamwork context T_j which is controlled by person P_k , who has been assigned a personal work context A_k. t5 should be regarded as short hand notation for persons P_i and P_k negotiating permission or denial to enter $C_{j,k}$;
- t7, t8: their occurrences model that a person P_i , who has been assigned a personal work context A_i, enters /leaves work context area D_n within teamwork context T_j.

5.4.4 Transfer between personal and teamwork contexts

Conceptually, personal work contexts A_i and teamwork contexts T_j are distinct from each other. Entering one work context area means leaving the other one. It will not be possible to see and access the insides of one T_j from another one; nor will it be possible to look into any A_i from any T_j and vice versa. In general, different Security Policies will be defined for different teamwork contexts (i.e. projects).

When a user enters the Co-operation Platform, that is when he migrates from O to B in Figure 14 , he has to authenticate himself. Successful authentication results in an assignment of a Personal Agent and a personal work context to him (this is the meaning of transition t1 and place PA in Figure 15).

When the user enters a teamwork context T_j, he always has to go through $C_{j,i}$ (his personal desk). Upon entrance, he is identified and authenticated by the "Team Agent", which is the system component that acts in behalf and under control of its associated team

(i.e. project), and his assigned to the teamwork context T_j (this is the meaning of transition t3 and place TA in Figure 15).

When entering a project from his personal work context area, the user is accompanied by his personal context (context overlap in $C_{j,i}$, information transfer only through $C_{j,i}$). Actually, only part of the entire personal work context A_i is relevant for the project. Thus, the transferred part mirrors a portion of the state of the personal work context.

The only mechanisms to relate personal work contexts and teamwork contexts are

- import / export functions, e.g. for transferring documents between work context areas B_i and $C_{j,i}$, and
- notification functions; e.g. a user may find in the private in-box of his personal work context a notification stemming from a teamwork context T_j to make him aware of project related messages waiting for him in his personal project in-box.

5.4.5 Interfaces to personal and teamwork contexts

Work contexts have been defined as being composed of virtual and physical objects related to projects, persons, tools and rooms (see Figure 10) and integrated under a common user interface. Further, it has been differentiated between personal work contexts uniquely associated with an individual person and teamwork contexts uniquely associated with a group of individual persons, who committed themselves to reach a common objective and to observe agreed common rules for sharing resources and for interacting with each other.

In the following, global relationships will be related to interfaces, i.e. to local views as seen by individual persons.

According to the two types of work contexts, two types of interfaces are being distinguished, one for working within the personal work context – this is the *personal office*, and one for working within the teamwork context – this is the *team (project) office* (both are *Virtual Project Offices*).

Personal offices are isolated local views of personal work contexts, as seen by their individual owners; they are not correlated to each other. Team offices, which belong to the same teamwork context, are local views of the teamwork context, as seen by its team members, and are intercorrelated (because of the same teamwork context). However, team offices, which belong to different teamwork contexts, are not correlated to each other.

The following figures (see Figure 16 – Figure 18) relate selected global states (in the middle, depicted by markings of work context areas) to the local views of the persons P1 (green circle) and P2 (black square). Grey background indicates "invisible".

Global state S_0 shows person P_1 working in working area B_1 and person P_2 working in working area B_2. The local views of P_1 and P_2 show that only B_1 or B_2, respectively, is visible to them, together with an indication which teamwork contexts are accessible from B_1/ B_2.

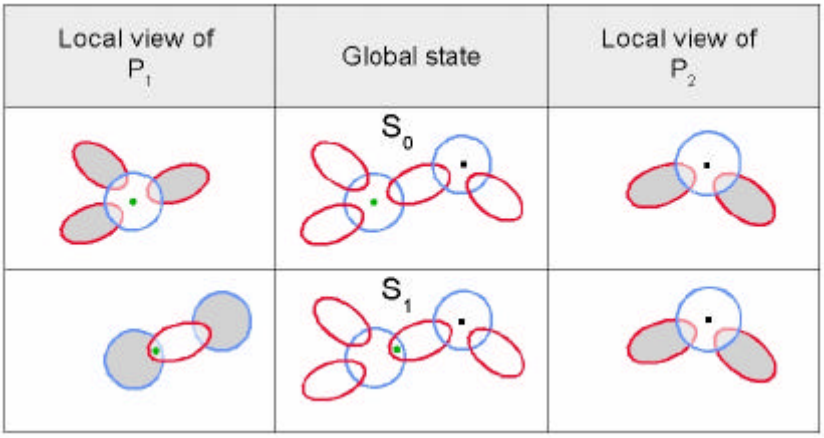


Figure 16 Local visibility of global states S_0, S_1

The global states S_1 to S_5 and related local views show sucessively, that

1. P_1 enters the teamwork context,
2. P_2 enters the teamwork context,
3. they meet at P_2's personal desk,
4. P_1 moves to a collaboration area,
5. P_2 joins P_1 in the collaboration area.

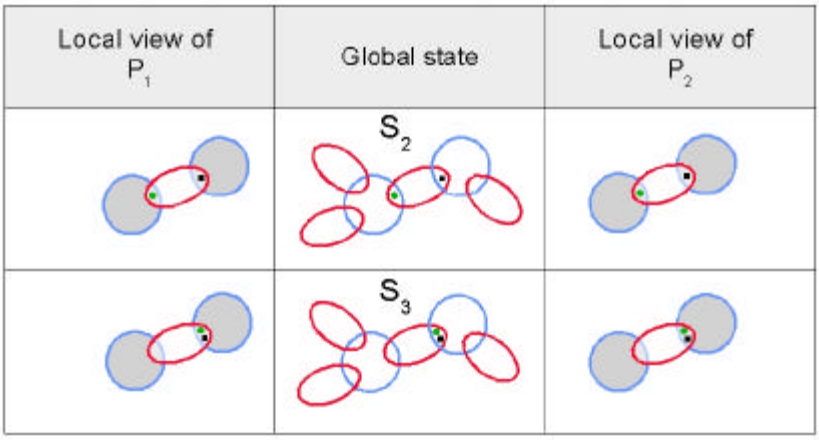


Figure 17 Local visibility of global states S_2, S_3

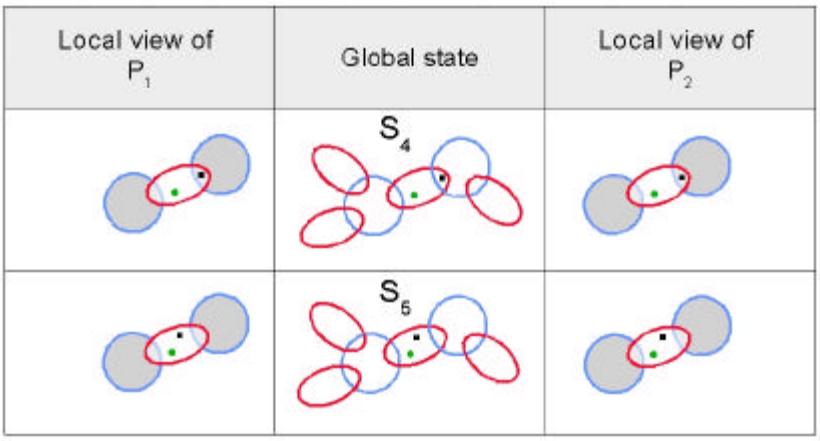


Figure 18 Local visibility of global states S_4, S_5

A user of the Co-operation platform enters his personal office, which is uniquely assigned to him, by passing a login procedure from outside the platform.

His personal office provides him with

- a set of services and tools (see under personal work context above), which support him in organising his work outside of each teamwork context;
- a working area interfacing the currently activated tools,
- a set of switches to team offices, one for each team office he is a member of, and
- a logout procedure to leave it.

Co-operation platform users always enter a team (project) office from their personal office, which leads them there to their personal desk.

A team office offers to a team member besides his own personal desk information about the teamwork context, in particular:

- who are the members of the team;
- which members are currently working in it;
- which are the tools and collaboration scenarios supported;
- which members are currently working / collaborating in which work areas;
- which work area can he currently enter (may depend on his current device);
- a working area interfacing the currently activated tools;
- a switch to his personal office, and
- a personal desk for every other team member.

Chapter 6

Architecture

6.1 Architectural Goals

Within this thesis, an integration architecture and a platform have been developed from which customised Co-operative Workplaces can be generated. It offers a set of configurable basic building blocks and services with which Co-operative Workplaces can be flexibly built and tailored to user needs. Such a platform is more effective than a new monolithic system, as it makes it possible to incorporate established tools and services from an installed base.

At the technology end, the approach includes the integration of various previously heterogeneous software tools and basic services by wrapping them in a homogeneous way and providing them with unified interconnection interfaces. Such interfaces are used to publish a component's behaviour and data. The integration approach starts by defining interfaces to address most objects used within a workspace, such as e.g. a calendar. It publishes their methods and data members, much like in the Java Beans technology within the Java world. Implementations of the CORBA standard, or Microsoft's COM data objects, would need to publish their preferred players (viewers and editors) and renderer although they may inherit default players/renderer. By defining the architecture of the Co-operation Platform, it is possible to plug into the platform every tool that can comply with the provided interfaces and deploys the required functionality.

At the user end, context oriented integrated services are being offered, as requested by the users. These can be constructed from an extensible base of individual services. Users may import such services, gaining location-transparent access to services, and multi-modal user interfaces do provide team members with access from different devices. Projects can compose their own Virtual Project Office that will provide specific services for a particular project. For example, one can think of applications like browsers for document repositories and editors for documents, which can be linked to abstract services of specific types, but with the user's ability to import and incorporate other applications to

work with those same services, and to interoperate with other existing services. Moreover, it is possible to create domain-specific customised core services that can be used by teams working in those domains and simplify their tasks. Web Application Servers such as Apache or Jakarta Open Source are being used to create project portals and domain specific services.

The management of physical workspaces and resources, in addition to virtual workspaces, is also incorporated in the Co-operation Platform. The presence of the users in the Co-operative Workplace and his information processing and communications capabilities in terms of software and hardware, (e.g. PCs, mobile phones, other appliances, and applications) at the locations where the users currently are, is being recognised and managed. This is also true for physical rooms and the equipment they are hosting. Therefore, discovery and awareness technologies have been applied. Multi-agent technologies are being used to represent resources and to negotiate among them. RoomComputer together with appropriate sensor technologies are being used to sense and control equipment as well as services offered in physical rooms.

6.2 Architectural Approach

In order to resolve the existing technical problems mentioned earlier and to offer new possibilities to virtual teams and virtual organisations, a new architectural framework and a service-oriented Co-operation Platform have been developed. This platform provides the following:

- Virtual workspaces providing a virtual working infrastructure with a rich set of tools, devices, and services in order to promote and enhance the collaboration process within teams.
- Computer-supported communication and collaboration services supporting co-operative work within geographical dispersed teams on joint projects.
- Integration methodologies thoroughly supporting the integration of different applications and services.
- Interoperable components and a service-based infrastructure enabling, supporting, and managing the integration of third party applications. This distributed platform provides a set of basic services that can be used in order configure a Co-operation Platform and the set of services it offers more rapidly towards the needs of its users.
- Service management services providing the appropriate mechanisms for the secure access to services, dynamic service execution, and provisioning.
- Mobile and intelligent agent systems supporting specific activities and contexts of work.
- Universal device access supporting network and device-independent connectivity and access to information, resources, and services by different types of end-user devices.

- Graphical user interfaces supporting new human-computer interaction models and providing methods for the description of device independent and abstract user interfaces together with a set of renderer, which translate those user interfaces description to a target end-user device, e.g. a PC or a PDA with a Web-browser. These interfaces enable non-IT-familiar users to collaborate among each other more effectively and allow them to participate in virtual workspaces and using the offered services more efficiently. Such interfaces provide virtual workspaces that enable and support complex collaboration processes and activities between members of a team. Through them, a virtual working infrastructure with its rich set of tools is accessible.

Several key innovative technologies have been developed and integrated into the Co-operation Platform:

- Discovery of devices, people, and physical locations (rooms): Discovery technologies are currently emerging but require real implementations to validate their scalability and develop new technologies to enable them to scale. Active directories and notification servers are being used to notify when a user registers or becomes active so that it is possible to send events on any change to the state of a user in the directory.
- Awareness technologies: This emerging technology enables awareness of people and their "presence" in virtual and physical locations. Integrating products that provide location awareness provides logical and physical location management. Such enhanced services provide global physical location awareness, and the RoomComputer provides local physical awareness. To give a straightforward example, one can think of being notified when a team-mate enters her/his office using the RoomComputer sensors and communication capabilities.
- Hybrid Communication and Collaboration Technology: It is necessary to present a common unified communication infrastructure regardless of the protocols and the media used. The communication infrastructure is a key element to the success of virtual societies in general and virtual workspaces in particular. Another key technology to enable remote workers to work together is the ability to collaborate seamlessly on multiple channels using several senses at the same time.
- Agent technology: This technology is well suited for modelling the higher abstraction layers of the Co-operation Platform. Agents represent the physical and logical participants in the Co-operation Platform. They are being used to model the representatives of persons (Personal Agents), teams (Team Agents), and rooms (Room Agents) within the Co-operation Platform. Message based communication is being used for negotiation between Team Agents, Personal Agents, and Room Agents. Based on this together with a set of pre-definable rules, suitable communication ways between the participants can be established.

- XML technology: XML is being extensively used within this thesis to define people information, activities device characteristics, and to define abstract and device independent user interfaces. XML is a new direction in the metadata direction that augments data in any form with information about the data itself.

Due to the Co-operation Platform, the collaborating people in Co-operative Workplaces get a context-oriented view at the user interface instead of a set of different views on individual tools, which are neither integrated nor related to the work context. Without the Co-operation Platform, people would just see individual tools and services. They would have to co-ordinate everything by themselves. With the Co-operation Platform, the collaborating people get a different view, namely a context oriented view together with a corresponding interface. They can interact with the people and objects of their work context through this interface and all required tools are handled by the service oriented platform middleware.

The Co-operation Platform provides means for dealing with the elements of a given work environment such as:

- its physical objects (e.g. tools, devices, furniture, installation),
- more virtual objects (e.g. data, files, software, means for audio-visual perception of remote people, state of a project) as well as
- people, who want to collaborate with each other for a common purpose, using tools, resources, and rooms which are needed to carry out individual tasks.

The means comprise facilities to define and configure Co-operative Workplaces on one hand, and to enter, use and leave them on the other hand. Moreover, the platform provides for integrating diversified components in such a way that within an application, the users be given a single view through which they can uniformly monitor, control and work with the elements needed, and it provide means to free the users from much of the details of the used components.

Basically, the general architecture of the Co-operation Platform is made up of three main layers of abstraction. The base is the real (or physical) world with real objects, categorised in people, resources, and rooms. People are mobile sources and targets of actions, resources (e.g. communication devices, computers, databases, printers, and furniture), fixed or movable, are temporarily or permanently assigned to people, and rooms are physical locations of people and resources. In general, these categories cannot communicate with each other.

On top of this rests an IT-model of the real world with permanent agents as representatives of the real world objects. The agents carry uniform interfaces and addresses, which allows combining objects, to address them and have them communicate with each other. This middle layer is also called the (generic) collaboration platform.

The third layer is the work context layer, dealing with the specifics of applications. It has temporary agents, each of which stands for, and interworks with an application specific assemblage of agents of the middle layer and represents their actual cooperation state. An example would be a team agent, which reflects a collaborative team together with their work environment, resources and locations.

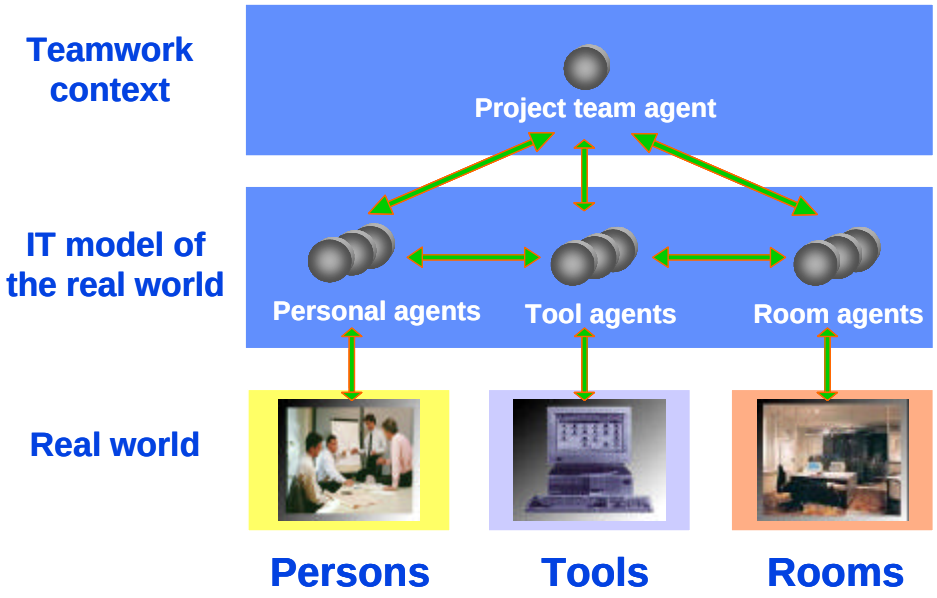


Figure 19 Co-operation Platform – layered architecture

Thus, the lower layer defines that part of the existing world that is of current interest. In the middle layer, various heterogeneous objects of the real world are encapsulated in homogeneous envelops to make them manageable and integrable into a common collaboration platform. Virtual rooms and objects are created here. Finally, the upper layer is the place to establish and maintain working relations among objects of the middle layer for processing specific tasks. Thereby, the real and virtual objects are embedded in the context of a task to be solved.

6.3 Integration Approach

Integrating already existing tools and services is essential to the Co-operation Platform. The approach taken is not the development of a completely new monolithic system. Instead, an integrating approach has been taken, enabling to pick up available and forth-coming component technologies as much as possible, in particular for reasons of economy and installed bases. This makes it possible to incorporate already available component systems and techniques of different kinds into the Co-operation Platform. Suitable component solutions can be assembled like parts from a construction kit to form innovative integrated solutions, which matches a given work situation and possess a reduced complexity of use. The building blocks are unified representations of information entities, suitably encapsulated in order to exhibit homogeneous interfaces.

To integrate various applications and business systems across the Internet, there are several phenomena driving the development of the Co-operation Platform. For example, users demand richer, deeper, more personalised and more proactive services that add more value to their work - services that work on their behalf while saving them time, money, and hassles. Nevertheless, integrating different services and combining different devices remains difficult, because tools and common conventions for interconnection are still lacking.

What is needed is a common way to tie all the different services required by the users together into a single platform. For services to work together seamlessly, one must be able to easily co-ordinate interaction between services over the Internet and to readily create new and customised services. Developers should be able to integrate more than just services. They need to assemble solutions from any combination of software components, applications, business processes, devices, or any other intellectual property. Thus, this thesis is driving for a common perspective that lets developers harness together pieces from anywhere on the Internet - a new application model that provides core programming services for distributed components in order to connect distributed services and components in a quick, reliable manner.

Traditionally, creating this kind of rich application-to-application communication has been solved by employing an object model such as Microsoft's DCOM or the Common Object Request Broker Architecture (CORBA). However, these technologies have limitations when it comes to creating Web-based services. In particular, DCOM and IIOP/CORBA are rich environments, which means that implementations and applications with them tend to be complex and symmetrical. Typically, a distributed application using them needs the same distributed object model running at either end of the connection. However, the Internet does not guarantee a specific kind of client or server software running at the other end of a connection. Moreover, it is often politically or technically impractical to get everyone to run either IIOP or DCOM.

In response to this challenge within this thesis, it has been decided to use Java Beans and Enterprise Java Beans in combination with the Internet standards HTTP and XML.

As has already been said earlier, there is a strong need to integrate existing pieces of software or whole applications developed by others. For the integration of already existing applications and pieces of software, there exist mainly three different approaches:

- a data-oriented approach,
- a message-oriented approach, and
- a component-oriented approach.

The data-oriented approach is appropriate when the focus is on data consistency. The problem of data consistency is that, for example in large organisations, there are multiple data sources that contain information about a single entity (e.g. a customer or a product).

In such a case, those different representations of the same entity have to be kept consistent.

The message-oriented approach is useful when there are business processes that span multiple application systems and the problem is about providing notification of events from one step of the process to the other. For example, in an order fulfilment process, the first step is to accept the order. As soon as the order is accepted, a notification is sent to begin the second step, checking the availability of the product in the warehouse. After that, the next step then is fulfilling the order. Thus, between the various business processes, events have to be communicated between the different processes in order to trigger them, notify them about the completion, and exchange business data.

Component-oriented integration instead provides real-time request/reply integration. More than that, it can be combined with a data or message oriented approach. Component-oriented approaches and therefore component technologies can speed up the development process for an application as they provide a structured and well-defined way to integrate parts of an application, which may have been developed by third parties. Additionally, component technology can serve, for example, for the integration of already existing (and sometimes legacy) applications into a platform without the need to modify them.

Object-oriented programming techniques have existed for some time; however, in recent years, constructing applications through the assembly of re-usable software components has emerged as a highly productive way to develop custom applications. Components range from small graphical user interface widgets, such as e.g. buttons, to more complex components, such as e.g. a stock ticker display, to full-size applications, such as e.g. word processors and spreadsheets. Software components may or may not be visible to users.

Efforts such as Microsoft's Visual Basic have been used in the deployment of applications on Windows-based desktops. Products such as Borland's Delphi have further advanced this methodology by adding more powerful data access components and extension capabilities. Sun's Java Beans extends the component-based software paradigm into platform-neutral, network-aware computing. Java Beans are re-usable in a variety of components, assembly tools, and containers such as e.g. Hot Java, Netscape Navigator, Borland's Delphi, Microsoft's Internet Explorer, Visual Basic, Word, and Claris Works.

It was thus decided to choose the component-oriented approach for realising the Co-operation Platform. The following sections elaborate on it in more detail.

6.3.1 Component technologies

A software component is a modular piece of software that encapsulates data and exposes a well-defined interface. When speaking about components, one should make a distinction between technical components and business components. Technical components are entities such as an Enterprise Java Bean (EJB) or a Component Object Model (COM) component.

A software component is a piece of software, which has at least three characteristics:

- it encapsulates data,
- it isolates data, and
- it exposes the business logic inside it through a well-defined interface.

Essentially, a technical component is a deployment unit. It is the minimal entity recognised at deployment time. On the other hand, a business component is recognised at runtime. A component differs from an object in the way that an object is essentially a design-time unit. When objects are packaged together into components, the notion of objects disappears and what one gets then is a technical component.

Within this thesis, a component-based integration of existing applications, and a service-oriented design are used for employing business components and technical components, to make up the Co-operation Platform and to provide the necessary context-oriented services to its users.

One of the big advantages of component technologies is that they offer a greater scope of reusability than the simple reuse of code, because components are self-contained—they are literally plugged in and work. Second, the widespread use of common platforms such as the Windows 9x/NT operating system and the Java Virtual Machine provide a market that is large enough for third-party vendors to create and sell components to at reasonable low cost. The main disadvantage to component reuse is that a component library is being needed in larger software systems because components are typically small and encapsulate only one concept.

Component-based software models and their APIs must provide five basic types of services:

- interface publishing and discovery,
- event handling,
- persistence,
- layout control, and
- application builder support.

In component-based environments, components are typically assembled at run-time. When a component is placed in a container, it needs to identify itself and the interfaces it supports. The new component registers or publishes its interfaces with the framework. Consequently, other components learn of its existence, and how to interact with it. Because components do not depend on a-priori knowledge of other components, they may interact with one another using published interfaces, even though the components might not come from the same software build.

In object-oriented programming paradigms, objects communicate with each other through message-based communication. An object raises or broadcasts a message or event, and the framework is responsible for delivering the message to the appropriate object. Messages may be generated by the system itself - a mouse click, for example - or they may be generated by other objects - such as when a database record is changed.

Components may need to store non-volatile state or other information. Persistence mechanisms are provided by component-based frameworks to allow components to store information persistently. The simplest mechanism might be to store persistent information in a flat file.

There are two types of layout control supported by component-based frameworks. Most commonly, they provide a way for a particular component to control its visual appearance within its own space. Component-based models also provide mechanisms and services to determine the layout of components inside a container. Programmers typically specify the appearance of a component at build time; however, layout control also provides mechanisms for components to modify the layout based on their own state.

Application builder support interfaces enable components to expose their properties and behaviours to development tools. Using these interfaces, tools can determine the properties and behaviours of arbitrary components. The tools can provide mechanisms such as tool palettes, inspectors, editors, and debuggers that developers can use to quickly assemble components into an application. Through the support of application builder interfaces, developers can modify the state and appearance of components and establish relationships between them.

Two widespread technologies that fall into a “competing” category are Java Beans and ActiveX — component technologies from Sun and Microsoft, respectively. From a technical standpoint, they have similar goals and try to address the same issues in software development, e.g. object-orientation and reusability.

6.3.1.1 COM and DCOM

The Component Object Model (COM) refers to both a specification and implementation developed by Microsoft Corporation, providing a framework for integrating software components [COM95]. This framework supports interoperability and reusability of distributed objects by allowing developers to build systems by assembling reusable components from different vendors.

COM defines an Application Programming Interface (API) to allow the creation of components for use in integrating custom applications or to allow diverse components to interact. However, in order to interact, components must adhere to a binary structure specified by Microsoft. As long as components adhere to this binary structure, components written in different languages can interoperate.

The Distributed Component Object Model (DCOM), also specified by Microsoft, extends the Component Object Model to support communication among objects on different computers via a local area network, a wide area network, or even the Internet [CK98]. With DCOM, a single application can be distributed throughout a network. Because DCOM is an evolution of COM, one can take advantage of already existing COM-based applications, components, and tools, to move into the world of distributed computing.

DCOM defines a protocol that enables software components to communicate directly over a network in a reliable and efficient manner. Previously called "Network Object Linking and Embedding (Network-OLE)", DCOM is designed for use across multiple

network transports, including Internet protocols such as HTTP. DCOM is based on the Open Software Foundation's DCE-RPC [Sri95, DCE98] specification and works with both Java applets and ActiveX components through its use of the Component Object Model (COM).

DCOM supports remote objects by running on a protocol called the Object Remote Procedure Call (ORPC). This ORPC layer is built on top of DCE's RPC and interacts with COM's run-time services. A DCOM server is a body of code that is capable of provide serving objects of a particular type at runtime. Each DCOM server object can support multiple interfaces, each representing a different behaviour of the object. A DCOM client calls into the exposed methods of a DCOM server by acquiring a pointer to one of the server object's interfaces. The client object then starts calling the exposed methods of the server object through the acquired interface pointer as if the server object resides in the client's address space. As specified by COM, a server object's memory layout conforms to the C++ vtable layout. Since the COM specification is at the binary level, it allows DCOM server components to be written in diverse programming languages like C++, Java, Object Pascal (Delphi), Visual Basic or even Cobol. As long as a platform supports COM services, DCOM can be used on that platform. DCOM is heavily used on MS Windows-based systems.

Because COM and DCOM are based on a native binary format, components written to these specifications are not platform-independent. Thus, they must be recompiled for a specific platform, or an interpreter for the binary format must be available. Because COM/DCOM components have access to the Microsoft Windows API, "bad actors" can potentially damage the user's computing environment. In order to address this problem, Microsoft employs "Authenticode" which uses public key encryption to digitally sign components [MS96]. Independent certification authorities such as VeriSign also issue digital certificates to verify the identity of the source of the component. However, the certification as such does not protect the user’s environment from accidental (or even malicious) security compromise.

6.3.1.2 ActiveX

ActiveX is an established component technology based on Microsoft’s COM. It is a subset of COM for reusable and lightweight components to be deployed on the Web. While COM and DCOM represent "low-level" technology that allows components to interact, ActiveX represents a higher-level application service that is built on top of COM and DCOM [DSP00].

Microsoft’s ActiveX is a set of technologies that lets one building applications for the Internet and Intranets that can run across multiple platforms. Developers can write applications using the programming languages C, C++, and Java and run these applications across platforms such as the Microsoft Windows family, Mac OS, and UNIX.

The key advantages of ActiveX are that it is based on mature technology, with thousands of components currently available. It can integrate object code from different programming languages, and can provide access to rich desktop functionality and strong desktop integration. ActiveX currently has a large development and marketing focus from Microsoft.

Nevertheless, there are many potential drawbacks of ActiveX. ActiveX can only be used in environments to which Microsoft has ported the framework and the components. Promises of ports to other platforms are unproven, and will still only address a limited number of environments. ActiveX components are built upon Microsoft's eight-year-old OLE controls. These controls could not profit from advances in software technology and design, they lack exception handling, and they bring along other engineering deficiencies into a new environment for which they were never intended. Just to run an ActiveX component, a heavyweight application must be invoked to provide a container for the component. The industry has standardised on CORBA as a means to provide object-oriented network connectivity for enterprise-wide applications. Users of ActiveX must rely on DCOM, which does not provide a means for linking to CORBA-based applications, as well as being a closed and single-source technology.

ActiveX security depends on digital signatures on components. By verifying a digital signature, end users can ensure that the code came from a known source and is not counterfeit or affected by tampering. Although code signatures verify the source of a component, they place no limitations on the functionality accessible by that element. As with DCOM components, the mere fact of the component being signed does not provide protection against security compromises. Viruses or Trojan horses inadvertently included in software from a third-party source will have free reign over the host platform, probably long before the attack is noticed.

6.3.1.3 Common Object Request Broker Architecture (CORBA)

CORBA is a distributed object framework proposed by a consortium called the Object Management Group (OMG) [OH98]. The core of the CORBA architecture is the Object Request Broker (ORB) that acts as the central object bus over which objects transparently interact with other objects located locally or remotely. CORBA relies on a protocol called the Internet Inter-ORB Protocol (IIOP) for accessing objects from remote. A CORBA object is represented to the outside world by an interface with a set of methods. A particular instance of an object is identified by an object reference.

The ORB is responsible for finding the implementation of a CORBA object, preparing it to receive requests, communicate requests to it, and carry the reply back to the client. A CORBA object interacts with the ORB either through the ORB interface or through an Object Adapter like the Portable Object Adapter (POA). Since CORBA is just a specification, it can be used on diverse operating system platforms from mainframes to UNIX boxes to Windows machines to handheld devices as long as there is an ORB implementation for that platform.

Like DCOM, CORBA provides client-server type of communication. To request a service, a client invokes a method implemented by a remote object acting as the server in the client-server model. The client of a CORBA object acquires its object reference and uses it as a handle to make method calls, as if the object were located in the client's address space. The ORB is then responsible for all the mechanisms required to find the object's implementation, prepare it to receive the request, communicate the request to it, and carry the reply (if any) back to the client. The object implementation interacts with the ORB either through an Object Adapter (OA) or through the ORB interface.

Each CORBA server object has an interface and exposes a set of methods. To request a service, a CORBA client acquires an object reference to a CORBA server object. The client can now make method calls on the object reference as if the CORBA server object resided in the client's address space. The service provided by the server is encapsulated as an object and the interface of an object is described in an Interface Definition Language (IDL). The interfaces defined in an IDL file serve as a contract between a server and its clients.

Clients interact with a server by invoking methods described in the IDL. The actual object implementation is hidden from the client. Some object-oriented programming features are present at the IDL level, such as data encapsulation, polymorphism, and single inheritance. CORBA also supports multiple inheritances at the IDL level, but DCOM does not. Instead, the notion of an object having multiple interfaces is used to achieve a similar purpose in DCOM. CORBA IDL can also specify exceptions.

Like with DCOM, the interactions between a client process and an object server in CORBA are implemented as object-oriented RPC-style communications. CORBA components use the CORBA Naming Service to locate other components, and use CORBA/IIOP to invoke methods on those components. To invoke a remote function, the client makes a call to the client stub. The stub then packs the call parameters into a request message, and invokes a protocol to ship the message to the server. At the server side, the message is delivered to the server stub, which then unpacks the request message and calls the actual function on the object. In DCOM, the client stub is referred to as the proxy and the server stub is referred to as the stub. In contrast, the client stub in CORBA is called the stub and the server stub is called the skeleton. Sometimes, the term "proxy" is also used to refer to a running instance of the stub in CORBA.

6.3.1.4 Java Beans

Java Beans is a component technology for reusable Java programs for the Java network-computing platform [Eng97]. As with ActiveX, components can be deployed over the Network or as part of a Java application. Java Beans get their name from the way in which they provide an environment where lightweight, secure, and platform-neutral applets can be directly accessed in a component-based framework. Rather than encapsulating existing components in a new platform-specific software layer, Java Beans allows the components - the Beans - to be directly accessed by applications.

Java Beans is an architecture- and platform-neutral component-based software model for building and using dynamic Java components. Java Beans blends the advantages of component-based software development with the power and flexibility of the Java network-computing platform. Java Beans provides mechanisms for interface publishing and discovery, event handling, persistence, layout control, and application builder support. In fact, many of these facilities directly build on Java's inherent strengths - for example, Java's serialisation mechanisms are used to provide object persistence, and Java's self-disclosing class structure is used to support component interface publishing and discovery mechanisms.

A Java Bean is a Java class that adheres to specific coding conventions so that the class's properties and methods can be discovered by other software components. A Bean allows itself to be customised in appearance and behaviour. It supports events that communicate its actions, and it can support a persistence mechanism to save its state for later restoration. However, in its simplest form, a Java Bean need be nothing more than a Java class that is made available to other components through a special packaging mechanism. A Bean-aware software development tool or environment can identify Beans and make them available for placement into an application. Thus, one can add Java Beans to an application in the same way that ActiveX controls are inserted into an application, by selecting them from a list and "dragging" them into the application.

For applications intended to run not only on MS Windows platforms, needing access to enterprise-wide resources and data, being security-conscious, and needing to operate in a heterogeneous environment, Java Beans is the preferred component-based framework. Java Beans components are truly platform-independent and allow true "write-once, run-everywhere" programming. The benefit of not tying applications to platform-specific features is a true saving of application porting cost. The component registration and discovery mechanisms of Java Beans are cleanly and simply integrated within the existing Java network-computing platform. These mechanisms are integral, not complex add-ons. By not attempting to encapsulate other vendors' components, Java Beans interoperates with other architectures, including ActiveX itself. In contrast to ActiveX, where heavyweight applications must be invoked in order to provide a container, Java Beans require no more than the Java interpreter or an applet-viewer. The lightweight design of Java Beans makes it suited for a wide range of target systems, from mainframes and supercomputers to low-administration Thin Clients. Network devices that will probably never support a port of ActiveX - such as cellular phones - can support Java Beans.

The security mechanisms inherent in the Java network-computing platform allow un-trusted applets to be incorporated into trusted applications by providing strict rules on what resources the component can manipulate. The Java Beans framework extends this model to component-based applications. In contrast to ActiveX components, which, once loaded, have free reign over the computing platform, Java Beans allows developers and users to determine allowable program behaviours down to a very fine degree of control. Java Beans supports signed components, which can be used to authenticate the source of a component. By combining these two security features, Java Beans offers the accountability of digital signatures with the control and confidence of the existing Java security mechanisms.

6.3.1.5 Enterprise Java Beans (EJB)

Enterprise Java Beans (EJB) is an extension of the Java Beans architecture to the now-standard three-tier architecture [RMH00]. Such an architecture consists of three layers or so-called tiers:

- presentation tier,
- business tier, and
- data tier

The presentation tier contains components dealing with user interfaces and user interaction, such as rich graphical user interface code or flavours of HTML code. The business tier contains components that work together to solve business problems, such as logic to bill credit cards or process orders. The data tier contains any databases, existing enterprise information systems, and legacy systems. The advantages of three-tier architectures are:

- scalability of number of clients and servers,
- increased perceived client performance,
- integration with existing web architectures,
- increased ease of deployment to clients and servers,
- increased server reliability, and
- increased adaptability of servers to varying clients.

Specifically, EJB defines a set of services and APIs for server-side Java components. EJB fills the need for definition of run-time and design-time environments for the server-side of three-tier architectures. The most significant facet of the EJB specification is the container model. A container provides mechanisms for a Java bean to define how the container is to treat it, in effect the services that the bean requires from the container. The services that the container provides for the bean are manifold: naming, life cycle, persistence, transactions, messaging, security, and packaging.

EJB has two primary extensions to Java: a container model, and a complex specification of services. The container model is a standard design pattern, used in many frameworks including Microsoft's COM. The power of EJB is in the specification of services and the implementation of the services in the containers. Objects can stay simple, because they gain service by co-operating with the container for the services. The previous models, CORBA and COM, required that objects grow to add behaviour. On the server, this led to very large objects, completely bound to services. The objects are difficult to program and almost impossible to widely re-use.

The EJB server provides the home for all the EJB containers. It allocates system resources to the various containers. The EJB container is responsible for the services that the Bean requires - persistence, transactions, naming, security, and life cycle management. There is one container for every enterprise bean class. The vendor’s tools, using the enterprise bean specification as the input, typically create the container.

The container wraps/intercepts all method calls to an enterprise bean. It makes use of any services need before and after the method invocation. A typical example of this is the persistence service. When a client invokes a method on a server object, the framework:

- instantiates the object in memory - perhaps resurrecting from a relational database
- invokes the method on the object, and
- optionally stores the object to the database after the invocation is complete.

Enterprise Java Beans are named using the Java Naming and Directory (JNDI) API. The naming hierarchy has the EJBs at the top level, layered on the application name, then the container name. A primary key within the container identifies the bean. There are no naming convention requirements for the primary keys. In addition, the CORBA Naming standard is supported. An EJB server supporting this naming standard will automatically publish a corresponding version of the EJB name.

The Container provides the life-cycle services of an object. All containers have a Home interface that provides Factory methods for the creation of objects. Entity beans also have a set of Finder methods for retrieving objects based upon a primary key. Home interfaces have destroy methods for deleting objects.

EJB provides an infrastructure for distributed transactions using two-phase commit protocols. It is based upon the OMG Transaction Service (OTS), specifically the Java Transaction Service (JTS) implementation of OTS. JTS supports multiple transaction managers propagating transactions to multiple databases. One of the exciting aspects of EJB is that it brings the strength of database transactions to objects. Distributed objects can participate in the begin/prepare/commit sequence typical of transaction protocols.

The problem with the EJB specification is that it is focused on the traditional distributed object protocols. It does not say anything about the distributed environment such as the World-Wide Web. EJB can be useful within the context of web servers, but there are no announced plans to integrate EJB Servers with Web Servers. For now, Java Servlets can be used to map HTTP requests to requests on EJB objects. The downside is that the Servlet must use Java RMI to communicate with the EJB Server. It is possible to run the JavaSoft Java WebServer inside the EJB Server's Java Virtual Machine. A very important next step is to provide tighter integration between a Web Server's Java Virtual Machine and the EJB Java Virtual Machine. A Pure Java implementation of an EJB Server accepting local method calls to containers would integrate very well with a Web Server and would provide a natural path for web developers to use.

6.3.1.6 Conclusion

Java Beans together with Enterprise Java Beans are the most advanced component technologies. Since the Co-operation Platform has mainly be developed in Java they seemed to be the most appropriate techniques to apply integrating existing applications, tools and components into the Co-operation Platform.

6.3.2 Component interaction

Interaction support between software components is necessary in large-scale systems, such as the Co-operation Platform, enabling loose coupling between various components, resulting in enhanced scalability and flexibility of the platform. This section describes the most widespread existing methods on which component interaction within the context of this thesis can be based upon.

6.3.2.1 Common Object Request Broker Architecture (CORBA)

CORBA can be used to interact with code written in a variety of languages, such as C, C++, or even COBOL. A potential disadvantage to this approach is that CORBA may cause communication protocol overhead. Fortunately, optimised CORBA products (such as Inprise's Application Server) perform code delegation as an Inter-Process Call rather than a Remote Procedure Call, eliminating network bottlenecks.

Another disadvantage is that CORBA programming requires knowledge of the CORBA Interface Definition Language (IDL), which requires a thorough understanding of IDL to perform serious CORBA programming.

6.3.2.2 Java Remote Method Invocation (Java RMI)

Java components use the Java Naming and Directory Interface (JNDI) to locate other components, and RMI-IIOP to invoke methods on those components [LS00]. The RMI-IIOP interfaces enable method invocations across Java virtual machines [SHM+00]. If the IIOP protocol is used, Java components can integrate with CORBA compliant components.

Java RMI relies on a protocol called the Java Remote Method Protocol (JRMP). Java heavily relies on Java Object Serialisation, which allows objects to be marshalled (or transmitted) as a stream. Since Java Object Serialisation is specific to Java, both the Java RMI server object and the client object have to be written in Java. Each Java RMI Server object defines an interface, which can be used to access the server object outside of the current Java Virtual Machine (JVM) and on another machine's JVM.

The interface exposes a set of methods, which are indicative of the services offered by the server object. For a client to locate a server object for the first time, Java RMI depends on a naming mechanism called the RMIRegistry that runs on the server machine and holds information about available server objects. A Java RMI client acquires an object reference to a Java RMI server object by doing a lookup for a server object reference and invokes methods on the server object as if the Java RMI server object resided in the client's address space. Java RMI server objects are named using URLs, and a client to acquire a server object reference should specify the URL of the server object as you would with the URL to a HTML page. Since Java RMI relies on Java, it can be used on diverse operating system platforms from mainframes to UNIX boxes to Windows machines to handheld devices as long as there is a Java Virtual Machine (JVM) implementation available for that platform.

6.3.2.3 Simple Object Access Protocol (SOAP)

Existing RPC-style protocols such as DCOM and IIOP have not proven to be adaptable to the Internet. Both of these protocols require a non-trivial amount of dedicated runtime support in order to implement the complete set of services that both protocols have to offer. Additionally, the closed nature of these runtimes combined with the relative complexity of the underlying protocols has made it difficult for developers to "drop down to the wire" when more flexibility is needed. Finally, the existing Internet security infrastructure has embraced HTTP to the point that trying to communicate across

organisations using anything other than HTTP requires an excessive amount of organisational and engineering resources.

The Simple Object Access Protocol (SOAP) improves Internet interoperability with an XML-based, platform-agnostic approach to programming the Web [SOAP00] [KS00]. SOAP is a protocol specification for invoking methods on servers, services, components, and objects; as such, it defines an asynchronous message exchange mechanism to pass commands and parameters between HTTP clients and servers.

SOAP codifies the existing practice of using XML and HTTP as a method invocation mechanism. The SOAP specification mandates a small number of HTTP headers that facilitate firewall/proxy filtering. The SOAP specification also mandates an XML vocabulary that is used for representing method parameters, return values, and exceptions. SOAP does not care what operating system, programming language, or object model is used on either the server side or the client side: it is agnostic, except that it needs HTTP as a transport.

SOAP is comparably simple. The client sends a request to a server to invoke an object, and the server sends back the results. The messages themselves are formatted in XML and encapsulated in HTTP protocol messages. SOAP works with existing infrastructures. One does not need to make any special accommodations on any router, firewall, or proxy servers for SOAP to work.

SOAP does not define all aspects of a distributed object system. That means, there is no specification of distributed garbage collection, type safety or versioning, bi-directional HTTP communications, message box carrying or pipeline processing, object-by-reference passing, and object activation. SOAP is meant to be simple—something that could be implemented by a developer in a couple of days using any programming language on any operating system. While these are not goals for SOAP itself, they could be implemented within a SOAP environment—just not with SOAP.

6.3.2.4 Conclusion

For component interaction, the Co-operation Platform makes use of the SOAP protocol, because it appears to be the most advanced and most promising technique.

With interaction support based on the asynchronous messaging system offered by the AMETAS platform (see below), a client can send a message (such as the submission of a request for an appointment by a person’s Personal Agent) and resume its task immediately, rather than having to wait for the appointment to be processed. The appointment persists in a temporary message queue and will eventually be processed when a server is available, perhaps during non-peak time.

Furthermore, the client does not have to wait if the server is currently unavailable (perhaps due to a crash). This enhances the overall scalability of the Co-operation Platform because more requests can be processed over time. The two most popular messaging models supported by AMETAS are point-to-point (single message source, single message sink) and publish/subscribe (multiple message sources, multiple message sinks).

6.4 Service-oriented Middleware

In the future, there will be an increase in the number and variety of both networked devices and mobile systems with radio-networked links, such as IEEE 802.11 Wireless LAN or Bluetooth. This creates the problem that in such environments of multiple networked devices and large numbers of mobile systems, the traditional approach of installing and maintaining device drivers, software, and services on each user system for every device that a user may wish to use, breaks down. This applies also for the Co-operation Platform.

The approach of mandating, across an organisation, a single operating system, down to the version, for all user systems, and a single maker and model for each type of device, becomes increasingly untenable. Not only does it create very inflexible systems, with high concentrated costs when devices, operating systems, and/or the software have to be upgraded. In this context, 'plug-and-play' operation, with the equivalent of software being dynamically downloaded to a particular device on demand is needed for the Co-operation Platform. Additionally, mobile users may want to discover what services are available in the physical workspace they just have moved in.

Therefore, the Co-operation Platform uses the paradigm of service-oriented programming (SOP), i.e. devices and applications can be dynamically plugged into the Co-operation Platform and configure themselves. They then set-up and offer their services within the platform, as for example setting up an audio connection between two or more parties. In order to be able to co-operate with other devices and applications, a device or application is offering its services within the platform. This means that it must be possible to find the required services, in particular during run-time.

To better appreciate the paradigm of service-oriented programming, consider the following scenario: A presentation is to be given in a meeting room. The speaker has prepared slides on his notebook. The meeting room has a beamer. The speaker now puts his notebook on the table and informs the meeting room via RoomComputer that a lecture is to be given. By means of the network infrastructure within the room, the RoomComputer then communicates with actuators for the lights as well as with the beamer. By addressing the services offered by lights and beamer, the RoomComputer can configure them according to the lecture scenario, i.e. the lights and the beamer are adjusted. The lecturer can now begin with his lecture without any need for further configuration.

To understand service-oriented programming from a more technical perspective, one needs to understand paradigms like object-oriented programming (OOP), client-server computing, and component models. OOP is built on the premise that programming problems can be modelled in terms of the objects in the problem domain. OOP has specific characteristics: inheritance, encapsulation, and polymorphism. Service-oriented programming builds on top of OOP, adding the premise that problems can be modelled in terms of the services that an object provides or uses.

Component models prescribe that programming problems can be seen as independently deployable black boxes that communicate through interfaces. The traditional client-server model often lacks well-defined interfaces and protocols that are

independent of the client or server implementation. This has made the client-server model brittle. In service-oriented programming, components publish and use services in a peer-to-peer manner, i.e. a client is not tied to a particular server. Instead, service providers are interchangeable. So, one can say that service-oriented programming is a paradigm for distributed computing that supplements OOP. Whereas OOP focuses on what things are and how they are constructed, service-oriented programming focuses on what things can do.

The question is how service-oriented programming can be achieved. It constitutes three different layers within the communication hierarchy.

- *Network layer:* This layer is responsible for the communication between the different services. Spontaneous networking requires the automatic assignment of the necessary communication parameters to devices and applications.
- *Infrastructure layer:* On this layer the procurement and exchange of information between a service provider and a service requester takes place. The description of a service and the protocols for registration and request are the central aspects of this layer.
- *Service layer:* This layer is accessed after the procurement of the services that are intended to be used. The use can be independent of the infrastructure layer, but may be influenced by it.

In principle, using a service works as follows: First, a service being offered by a service provider is registered with a so-called service directory or service broker. Second, a service requester can query the service directory/broker for a specific service. It then determines a suitable service and informs the service requester about the result. Now, the service requester can interact directly with the service provider.

In order for the service broker to be able to provide a suitable service to the service requester, every service includes a service description. For such a description of a service, one can distinguish fundamentally two possibilities: The services are already known in advance, that is during the development time of the service requester (Closed World Assumption) or not (Open World Assumption). In the case of the Open World Assumption, a semantic description of a service must be provided so that unknown combinations between service requesters and service providers are allowed in advance. The mediation of services via semantic information is very difficult and is dealt with in the area of artificial intelligence. In the case of the Closed World Assumption, it is sufficient if the services are being described syntactically by means of specified interfaces. In this case, the selection of a matching service by the service broker is simpler because it only has to select a service with a matching interface among the amount of the available interfaces offered by the service providers.

6.4.1 Service-oriented Middleware Solutions

In order to announce, discover, and use services in an open environment, such as the Co-operation Platform, service discovery protocols are being used. This section compares some of the different approaches that exist today and support the announcement and discovery of a service in a networked environment. The following solutions that are more promising have been analysed:

- Universal Plug and Play (UPnP),
- Home Audio Video Interoperability (HAVi),
- Bluetooth Service Discovery Protocol (SDP),
- Salutation,
- Service Location Protocol (SLP),
- Jini, and
- Universal Description, Discovery, and Integration (UDDI).

The objective of a service-oriented architecture is used to develop the Co-operation Platform as a collection of autonomous services, each implementing a well-defined function and interface. Every qualified client application can interact with the service if it knows its location, the interfaces it is offering, and how to interact with it. This looks very similar to a component-oriented approach, but they reside at different levels. Service-orientation is a concept that is applied at design and execution time. Component orientation is a concept being applied at implementation time. The business component is the unifying concept, i.e. technical components are mainly used to implement a service, which itself is realising a business component.

Within this thesis, a component-based integration of existing applications, and a service-oriented design have been used for employing business components and technical components, to make up the Co-operation Platform and to provide the necessary context-oriented services to its users.

The services provide XML-based interfaces that comply with open standards such as the Simple Object Access Protocol (SOAP). Service-oriented technologies have been used to expose them over the Internet.

Comparing service-oriented middleware technologies like Sun’s Jini with other popular middleware technologies like CORBA (Common Object Request Broker Architecture) or even Java RMI (Remote Method Invocation) is a bit like comparing apples with pears. CORBA and Java RMI enable objects to communicate with each other, i.e. enable one object to invoke a method on a remote object.

One of the big advantages of CORBA is that it allows objects to be developed in any programming language such as e.g. Java, C, and C++. With Java RMI, objects are normally coded in Java, although RMI-IIOP (Internet Inter-ORB Protocol) allows the interaction with CORBA clients. With IDL (Interface Definition Language), CORBA provides a definition language to describe objects and their interfaces, independent of a particular programming language. RMI, on the other hand, uses Java's own interface semantics.

Service oriented middleware technologies provide an interaction model and the necessary infrastructure for distributed objects to interact with each other. They can create a functional network of objects that can dynamically be linked together. Within CORBA, the ORB (Object Request Broker) is used to get the reference of a remote object. RMI provides the so-called RMI Registry to locate the reference to an object. Both approaches more or less represent the "classical" approach in distributed computing. However, technologies like Jini provide a different model: a sophisticated infrastructure for objects to find each other, join, disconnect, reconnect, and to recover from network failures.

What service-oriented middleware technologies often do not specify is the communication mechanism between the objects. They may either be based on exchanging messages (e.g. by using the SOAP protocol) or just be based on RMI or CORBA. That means that service-oriented middleware technologies do not compete against classical middleware technologies like CORBA, they can be seen as being complementary and making a perfect team.

As already said, there exist a number of different service-oriented middleware technologies. These support varying degrees of dynamic configuration functionality and conditions of use. There are also two main approaches to service discovery, one using a registry or lookup server, the other using peer-to-peer discovery, with some protocols supporting both. This provides two criteria to help evaluating those technologies: discovery and configuration abilities, and whether discovery is peer-to-peer or server based, or both. A further criterion is whether it is device or service-oriented, or both. The following is a brief description of the most advanced alternatives to be used within the Co-operation Platform with respect to service orientation and features to perform service announcement, discovery, and usage. Although it is wishful to choose the best available technology, it is more likely that no single approach will dominate, at least in the near term.

6.4.1.1 Universal Plug and Play (UPnP)

Universal Plug and Play (UPnP) is an architecture developed by the Universal Plug and Play Forum for pervasive peer-to-peer network connectivity of PCs of all form factors, intelligent appliances, and wireless devices [UPnP00] [UpnP01]. The goals of the Forum are to enable the emergence of easily connected devices and to simplify the implementation of networks in corporate environments and the home. This is achieved by defining and publishing UPnP-conformant device control protocols built upon open, Internet-based communication standards.

UPnP defines a distributed, open networking architecture that leverages TCP/IP and the Web to enable seamless proximity networking in addition to control and data transfer

among networked devices in the home, office, and everywhere in between. Its usage is targeted towards small office or home computer networks, where devices are reachable through a TCP/IP network. It enables peer-to-peer mechanisms for auto-configuration of devices, service discovery, and control of services.

UPnP supports zero-configuration networking and automatic discovery whereby a device can dynamically join a network, obtain an IP address, announce its name, convey its capabilities upon request, and learn about the presence and capabilities of other devices. This means that in UPnP, there is no central service register, such as the Directory Agent in SLP or the lookup table in Jini (see below).

The Simple Service Discovery Protocol (SSDP) is used within UPnP to discover services [SSDP99]. It provides a mechanism whereby network clients, with little or no static configuration, can discover desired network services. SSDP uses HTTP over multicast and unicast UDP and is thus designed for usage in IP networks. DHCP and DNS servers are optional and will be made use of only if available on the network. Furthermore, a device can leave a network smoothly and automatically without leaving any unwanted state behind.

6.4.1.2 Home Audio Video Interoperability (HAVi)

Home Audio Video Interoperability (HAVi) is a specification for home audio-video peripherals [HAVi98]. HAVi is based on the IEEE 1394 standard also called FireWire by Apple or i.Link by Sony [IEEE-1394-1995].

HAVi classifies devices into four categories: two categories of controlling devices (called Controllers) and two categories of controlled devices:

- *Controllers*:
  - Full Audio/Video devices (FAV), being able to execute Java byte code
  - Intermediate Audio/Video devices (IAV), like FAV but unable to run Java byte code
- *Controlling devices*:
  - Base Audio/Video devices (BAV), devices with included Java byte code.
  - Legacy Audio/Video devices (LAV), non-HAVi devices. Although they are not "HAVi-aware", they are part of the HAVi network and may be controlled.

The features offered by HAVi are:

- Plug and play support: HAVi can make use of FireWire to detect new devices.
- Device interoperability: this means that all HAVi devices on the network can make use of functionalities offered by any other HAVi device.
- Platform independence: HAVi is independent of any particular operating system or CPU. This is ensured by the use of Java.

6.4.1.3 Bluetooth Service Discovery Protocol (SDP)

Bluetooth is a new short-range wireless transmission technology [BLUE01]. The Bluetooth protocol stack contains the Service Discovery Protocol (SDP), which is used to locate services provided by or available via a Bluetooth device. SDP is described in the Bluetooth specification part E [SDP01]. It originates from the Piano platform developed by Motorola and has been modified to suit the dynamic nature of ad hoc communications. It addresses service discovery specifically for this environment and thus focuses on discovering services, supporting the following inquiries: search for services by service type; search for services by service attributes; and service browsing without knowledge of the service characteristics.

SDP does not include functionality for accessing services. Once services are discovered with SDP, they can be selected, accessed, and used by mechanisms out of the scope of SDP, for example by other service discovery protocols such as SLP or Salutation (see e.g. [MP99], where a mapping from Salutation to SDP is shown). SDP can co-exist with other service discovery protocols, but it does not require them.

6.4.1.4 Salutation

Salutation is another approach to service discovery. The Salutation architecture [SAL98] is developed by an open industry consortium, called the Salutation Consortium (see http://www.salutation.org). The Salutation architecture consists of Salutation Managers (SLMs) that have the functionality of service brokers. Services register their capabilities with an SLM, and clients query the SLM when they need a service. After discovering a desired service, clients are able to request the utilisation of the service through the SLM.

6.4.1.5 Service Location Protocol (SLP)

The Service Location Protocol is a product of the SVRLOC Working Group of the IETF. It is a protocol for automatic resource discovery on IP networks [EG99] [GDP99] [GPK99].

Applications running on a computer are represented by a user agent that understands the service and resource needs of that particular application. Each network service is represented by a service agent, which makes it available to user agents. Current applications and services, which are not written to utilise the features of Service Location Protocol, can be represented by SLP proxies, which perform the same function as a User or Service Agent. SLP dynamically maintains service attributes, so that a User Agent may obtain current information. Services may be located automatically by applications or presented to the user in a service browser. SLP supports existing applications and may easily allow advertisement and discovery of new services.

SLP aims to be a vendor-independent standard. It is designed for TCP/IP networks and is scalable up to large enterprise networks. The SLP architecture consists of three main components:

- User Agents (UA) perform service discovery on behalf of the client (user or application),

- Service Agents (SA) advertise the location and characteristics of services on behalf of services, and
- Directory Agents (DA) collect service addresses and information received from SAs in their database and respond to service requests from UAs.

When a new service connects to a network, the SA contacts the DA to advertise its existence (Service Registration). When a user needs a certain service, the UA queries the available services in the network from the DA (Service Request). After receiving the address and characteristics of the desired service, the user may finally utilise the service.

Before a client (UA or SA) is able to contact the DA, it must discover the existence of the DA. There are three different methods for DA discovery: static, active, and passive. With static discovery, SLP agents obtain the address of the DA through DHCP (Dynamic Host Configuration Protocol [RD99]). The necessary DHCP options for SLP are defined in [PG99]. DHCP servers distribute the addresses of DAs to hosts that request them. In active discovery, UAs and SAs send service requests to the SLP multicast group address (i.e. 239.255.255.253). A DA listening on this address will receive a service request and respond directly (via unicast) to the requesting agent. In case of passive discovery, DAs periodically send out multicast advertisements for their services. UAs and SAs learn the DA address from the received advertisements and are now able to contact the DA themselves via unicast.

6.4.1.6 Jini

Sun’s Jini technology provides simple mechanisms which enable devices to plug together to form an impromptu community – a community put together without any planning, installation, or human intervention [Edw99]. Each device provides services that other devices in the community may use.

The Java programming language is the key to making Jini technology work. In a network employing Jini technology, services are tied together using Java Remote Method Invocation (RMI). The discovery and join protocols, as well as the lookup service, depend on the ability to move Java objects, including their code, between Java virtual machines. For Jini connection technology to succeed, the underlying protocols and infrastructure must become pervasive. Jini comprises the following key concepts:

- Service
- Discovery
- Lookup
- Service Proxy
- Leasing
- Remote Events
- Transactions
- Groups and 'Djinns' or Jini Communities

Jini unifies all participants, whether software or hardware, under the notion of a service. Thus, the distinction between a device and a service in Jini is blurred and the term device is mainly used for convenience to distinguish smaller I/O devices form larger system level components. In Jini, everything has to be able to represent itself as a software entity with an interface known to any client. Jini can dynamically federate services into distributed systems to meet particular requests. Services can call on other services and so can act as clients. A client in Jini is a service that initiates a call on another service.

Discovery is the process whereby clients and services, when plugged into a network, use multicasting to locate available Jini Lookup services in the local network domain. A service will then register itself with the Lookup Service by providing information about itself, the key part being a proxy to it to be downloaded by prospective clients. Clients also use Discovery to locate Lookup services. The discovery process means that services and clients do not need to have a Lookup services address or location in order to make use of it. There is, however, a unicast discovery method as well which does require an address and then allows access to a Lookup service anywhere on the Internet.

Essentially all a Jini Lookup Service does, is enabling clients to look up services that are available and, when the client selects a service, passes a proxy to the client, which enables the client to communicate directly with the service.

A client, after locating a Lookup Service, will request information about available registered services. The client can then select an appropriate service and make use of it. Having made the introductions, the Jini Lookup Service plays no further part in that interaction, although it might be used by the service to gain access to other services to help carry out the client's task. Through services being clients of other services, Jini enables the dynamic building of distributed systems 'on demand'.

On selecting a service, a client will be shipped a copy of the service's proxy which will handle (and hide) all communications with the service. The client interacts with the proxy through a predefined interface. The proxy then handles all translation to and from the service using whichever on-the-wire protocol the service implements (it does not have to use Java's RMI or any other particular protocol). Beside protocol transparency, the proxy mechanism also provides service location transparency to the client. The mobile service proxy is Jini's unique feature and is the source of much of its capabilities and flexibility.

Leasing is a crucial part of Jini's management of partial failure. When a service registers with a Lookup, it is granted a timed lease for its registration. This has to keep being renewed before the lease runs out while the service is available. The service can cancel the lease if it is closed down, but if it crashes or the network link fails, the lease is not renewed and therefore removed by the Lookup service from its list of available services. On being restarted, or when the network link is restored, the service performs its Discovery procedure and re-registers with the Lookup Service.

Remote Events allow clients to be notified by a Jini Lookup service when a service registers or by JavaSpaces when a specified kind of object is posted in the space. They are also available as general-purpose events that can be used by any other service to provide notifications of a local event to remote clients or third party helpers that have registered

their interests. Remote event registration also makes use of leases which time out if not renewed.

Jini provides clients and services with a lightweight, general-purpose transaction framework. This addresses the problems of maintaining distributed data consistency in the face of partial failure. It enables a set of operations to be grouped, reporting their success to a Transaction Manager. This manager then sends a commit command unless one or more participants fail or not all participants complete within the transaction's lease period; in that case, a rollback command is issued.

A Lookup Server may optionally have one or more group names associated with it. A group is an arbitrary string, which can be used to partition the Jini namespace. It is recommended that the string follow the DNS naming convention to avoid naming collisions. When searching for a Jini Lookup service, the discovery method can specify a group name. Services will be registered under the group name, and clients using a group name will only find services registered under that name. Both services and clients can belong to several groups, and an array of group names can be used for discovery.

A set of devices, resources, and users linked by a Jini group is referred to as a 'Djinn' in the Jini specifications. This is also referred to as a Jini Community. A Jini Lookup Service and associated communities are thought of as operating at the workgroup level of perhaps 10 to 100 people working on the same subnet.

Members of a group can all access each other, and services may also act as clients to other services and these in turn may call on others. This would typically happen when a client requests a function that a service cannot perform on its own. An example might be the conversion of an incoming file into a format that a service can handle or outputting a file in a format, which the service cannot itself generate. Several lightweight format conversion services might be available to a group. This opens the way for dynamic configurations of services to be created in response to user needs. It also suggests a change in the way distributed applications are provided, breaking them into smaller lightweight component services.

Jini Lookup services are themselves Jini services and as such have the capability to register themselves with other Jini Lookup services. This enables Jini to be scaled up into larger groupings called Federating Jini Communities of Federations. For large federations this is best done by pre-configuring the Lookup services with the address of each other to avoid multicasting spreading too far across the network. Federations allow a wider variety of services to be made accessible in a controlled way.

Besides waiting for services to register with it, a Jini Lookup Service multicasts an announcement of its presence when it starts up, so that any potential services can register with it. This is also useful after a network failure to inform disconnected clients that a Lookup service is available to them again.

The Announcement Protocol can be made use of by a client if it fails to discover any Lookup Services: it then pretends to be a Lookup Service itself and multicasts an announcement. Available services respond with their registrations, and the client discards those it is not interested in but extracts the proxy to the one that meets its needs. However, this would need additional software in a client to multicast announcements, send out a

proxy to itself to responding services, and accept and handle their registrations. This mode of operation is actually only suitable on small networks such as found in small business and home networks, since the amount of multicast traffic can become too great. It might be useful though if the multicast is confined to the local subnet (i.e. its time-to-live is set to 0 so that it is not passed on by a router).

Jini builds on the facilities provided by Java RMI, particularly for moving code between systems. This appears to make it a 'Java only' system, as servers need to pass a (Java) Registration including its serialised Java proxy to a Lookup Service, and clients need to download the proxy and run it in order to interact with the service.

However, Java's ability to act as a front-end wrapper to other systems can also be used to enable non-Java services and, with a bit more work, non-Java clients can participate using a Java wrapper.

It should also be noted that RMI is needed to set up and establish a connection between a client and a service, the connection between the service proxy, which resides on the client, and the service itself, does not have to be RMI. It can be any protocol chosen by the provider of a service. Instead, CORBA, DCOM, or any other standard or proprietary protocol that is appropriate to the service could be used. The protocol being used is completely hidden from the client by the proxy. This is one of Jini's strengths and plays an important part in Jini's systems integration role. Even the service itself does not have to be written in Java, but the service proxy that is sent to the client needs to be available in Java. Again, it is possible for a client to be provided with a Java wrapper, which handles the service proxy.

While it is true that Jini enables device plug-and-play in a networked environment, and that initial Jini marketing efforts has focused on this aspect, this is only a subset of the capabilities, which it offers. A complementary consequence is that it also allows dynamic addition, removal, and relocation of clients, enabling mobile clients to make use of local services without the necessity of being pre-configured with an appropriate set of drivers. It also offers enterprise integration, and reliable, flexible, self-healing provision of networked services and systems. In the context of this thesis, this aspect is significant as its network plug-and-work functions can be used.

Jini provides a dynamic discovery ad-hoc connection service. It supplies would-be clients of other services with the proxy supplied by those services when they registered with the look-up service. The protocol between the proxy and its associated service is undefined by Jini. It therefore can be any protocol chosen by the service implementer, be it CORBA, DCOM, RMI or any other open or proprietary protocol. As it defines the connection at the higher, object interface level, the interface can define a generic service type. A client programmed to handle interfaces to this type of service, may use services, which implement different protocols on different sessions, without any change on the client side. Far from competing with other protocols, Jini can be used to simplify the operation of a multi-protocol environment.

Although the actual code needed to implement Jini is quite small, perhaps 50 KB, Jini builds on RMI, specifically the upgraded RMI that comes with Java Platform 2. This means that Java 1.1.x JVMs are not able to run Jini. The minimum requirement to run Jini at present is Java 2 Micro Edition (J2ME). The architecture for J2ME is to provide a small

and efficient core virtual machine, and provide various modules of different sizes that sit on top of it. These modules, or 'Profiles', provide the classes needed for different types of functionality and range from the Personal Java Profile down to the JavaCard at the minimum end.

The Graphical UI support is sometimes not needed in an e.g. embedded device, but using the standard components made available by Sun, the minimum memory needed to support Jini in an embedded device, such as the RoomComputer , is about 1.5 MByte, and depending on what extra memory is required at run-time, 2 to 2.5 MByte of memory is likely to be needed to provide a fully Jini-capable device. Together with an appropriate processor and network interface, this is still a cost overhead that rules out a range of smaller devices.

With respect to security, there are two types to consider with Jini:

- security against badly behaved downloaded code
- authentication and authorisation

Security against badly behaved downloaded code is one of Jini's strengths which it inherits form Java and RMI. This is very important when using downloaded code of any sort, and much of Jini's unique capabilities and flexibility derive from its ability to move a proxy from the device or service to the client. The Java 2 Security Manager provides a flexible way of defining security policies, and for the most part, device drivers can be confined to a strict sandbox and only permitted to interact with the service they represent.

Authentication and authorisation is currently not implemented within core Jini, although this is likely to change. It might be argued that when a user is connected to a service, it is up to the service to provide its own authentication and authorisation with its client in the normal way. While this might be true for services, it is less clear for networked devices, which typically do not provide such security services but which may nevertheless be desired to restrict access. At the level of the Lookup Service, it may also be desirable to protect access to certain groups and limit their distribution at the level of service entries as an option to a restricted list of users.

6.4.1.7 Universal Description, Discovery, and Integration (UDDI)

The Universal Description, Discovery, and Integration (UDDI) specifications define a way to publish and discover information about web-based services. The term “Web-service” describes specific business functionality exposed by a service provider, usually through an Internet connection, for providing a way for another service provider (i.e. a company or software program) to use the service [UDDI00].

UDDI takes an approach that relies upon a distributed registry of services and their service descriptions implemented in a common XML format. Based on widespread open standards by the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF), it defines an interface to query the service registry and discover information about a particular service provider and the services it exposes.

The registry is the core component of UDDI. It is an XML file, used to describe a service provider together with the services it is offering. Conceptually, the information provided in a UDDI registry consists of three components:

- White Pages including address, contact, and known identifiers,
- Yellow Pages including categorisations based on standard taxonomies, and
- Green Pages, the technical information about services that are exposed by the service provider, including references to specifications for services, as well as support for pointers to various file and URL-based discovery mechanisms if required.

The core information model used by the UDDI registry is defined in an XML schema. This schema defines the following core types of information that provide the kinds of information about a service:

- business information,
- service information,
- binding information, and
- information about specifications for services.

Cross-platform features are addressed in UDDI by adopting the SOAP (Simple Object Access Protocol) messaging specifications [BEK+00]. The UDDI Programmers API Specification defines approximately 30 SOAP messages that are used to perform inquiry and publishing functions against any UDDI-compliant service registry.

6.4.1.8 Conclusion

UPnP is designed for TCP/IP networks only. In its current version, it does not allow clients to search for service attributes, as e.g. SLP is able to do. Jini and SLP are better suited for ad hoc networking than UPnP.

Another drawback of UPnP is the lack of security. Like Jini, it is also based on standardisation of interfaces, but the more limited search criteria could possibly be a drawback compared to Jini. Another problem is that if several clients want to use a single service simultaneously, the same state variables would be manipulated. This problem is avoided in Jini, since the proxy object, located at the client side, can host the state of the service, and the Jini service itself can be stateless.

HAVi is based on JDK 1.1, but could possibly be implemented in other languages as well. There are specified Java APIs for controlling home audio-video devices like digital cameras, CD players, TV sets, etc. Since HAVi is more specialised on audio-video, and is supported by major manufacturers, it might be a hard competitor in this area. However, even this speaks for Jini, as HAVi is easily bridged to Jini (both are Java based). Jini has advantages in front of other technologies not especially aimed at the home area. HAVi also supports user interfaces. Just like Jini proxies, a so-called HAVlet can control HAVi devices from remote. A HAVlet is a Java class that enables a GUI to be displayed on a HAVi device that has a display. The disadvantage with HAVi compared with other

technologies is that it is only standardised for IEEE 1394 FireWire networks, which makes it only suited for transmission of high quality audio and video.

Bluetooth provides encryption and authentication between Bluetooth units. Unlike higher-level service discovery technologies, such as e.g. Jini, the SDP of Bluetooth, does not provide a mechanism for using discovered services. Specific actions required to use a service must be provided by a higher-level protocol. However, it defines a standard attribute Protocol Descriptor List that enumerates appropriate protocols for communicating with a service.

Within Salutation, service discovery is defined on a higher layer, and the transport layer is not specified. Thus, Salutation is independent on the network technology and may run over multiple infrastructures, such as over TCP/IP and IrDA. It is not limited to HTTP-over-UDP-over-IP, as UPnP is. Moreover, Salutation is independent of the programming language, i.e. it is not limited to, nor does it have a prerequisite for Java (as e.g. Jini has). Salutation is a rather settled approach, with some commercial implementations, including fax devices and Windows enablers (95/98 and NT), e.g. from IBM and Axis.

The Service Location Protocol is standardised through the IETF, and there exist several reference implementations as well as commercial products. SLP offers a flexible and scalable architecture and the utilisation of service templates make service browsing and human interaction possible. SLP is a low-level protocol compared to Jini and only includes Discovery and a limited security model. The simplicity of SLP is both an advantage and a disadvantage. It requires less device capabilities and a Lookup service is optional, unlike in Jini. Since SLP is able to operate with or without a DA, it is suitable for networks of different sizes, ranging from very small ad hoc connectivity to large enterprise networks. SLP also includes a leasing concept with a lifetime that defines how long a DA will store a service registration. Whereas Jini is dependent on Java, SLP is independent on the programming language.

However, if a network does not contain an SLP Directory Agent, then IP servers must use their own Service Agents to specify the services that are available. If the Service Agent does not support multicasting and if there are any services advertised by that Service Agent that are needed by the User Agent on this machine, a network broadcast must be used. Broadcasting has the disadvantage of being limited to a local LAN segment (as if the multicast radius were set to 1). Additionally, as with UPnP, SLP does not use mobile code, and so clients must know in advance how to talk to any service they will ever use. A bridge could publish SLP services into a Jini technology-enabled network.

Jini distinguishes from the other approaches mainly by the fact that it is based on Java. On the one hand, this concept makes Jini independent of the platform and operating system to run on. Most important, Jini uses Java Remote Method Invocation (Java RMI) protocols to move program code in the network. This introduces the possibility to move device drivers to client applications, which is its main advantage over the non-Java based service discovery concepts. On the other hand, the fact that Jini is tightly tied to the programming language Java makes it dependent on the programming environment. It also requires its devices to run a JVM, which consumes memory and processing power. This can be a hard requirement for large device drivers and might not be fulfilled in embedded

systems. Due to the dynamic nature of ad hoc networks, Jini employs the concept of leasing. Each time a device joins the network and its services become available on the network, it registers itself only for a certain period, called a lease. This is especially useful for very dynamic ad hoc network scenarios.

The concepts and goals of Jini related to emerging UDDI-based Web Services seem to be almost identical. The main concept of Jini is the idea of a service being provided to some other entity that needs to use that service. UDDI is based on the same concept. The ways in which the services are found, accessed, and used are also similar. UDDI is based on the idea of publishing, finding, and binding. Jini uses discovery and join, lookup, and service invocation to perform the same functions.

In more specific terms, UDDI web services are equivalent to Jini's lookup service, which provides a common list of available services. UDDI uses SOAP (Simple Object Access Protocol) whereas Jini uses Java RMI. Jini invokes services using the service object's interface, which would be defined in UDDI by using WSDL (Web Services Description Language).

The difference between UDDI and Jini is that UDDI is a document-based framework (using XML documents) whereas Jini is following an object-oriented approach (using Java interfaces) for distributed computing platforms. The advantage of UDDI is that it is independent of programming languages and operating systems, like CORBA. Until now,

UDDI just defines the discovery protocol, but does not provide the concept of events, transactions, and security, as Jini does. Up to now, Jini is the most powerful and advanced technology with respect to service orientation. Therefore, the service-oriented middleware of the Co-operation Platform will be based on Jini.

6.4.2 Service Interaction

To integrate various applications and systems across the Internet, there are several service-related phenomena driving the development of the Co-operation Platform. For example, users demand for richer, deeper, more personalised and more proactive services that add more value to their work — services that work on their behalf while saving them time, money, and hassles. The problem with that is, that today, integrating with other services and touching different devices remains difficult, because tools and common conventions for their interconnection are lacking.

The ability to describe structured data in an open text-based format and to deliver this data using standard HTTP protocol is significant in enabling a new generation of Co-operation Platforms. XML is the "lingua franca" within the Co-operation Platform for the exchange of information, object and service interaction, and application data.

In order for different services to interact with each other in a proper way, interoperability is needed. To achieve interoperability, two approaches exist. First, one can work out a common Document Type Definition (DTD) that will define a class of XML documents for the Co-operation Platform domain. This requires defining a set of tags and the structure of the used data. The resulting DTD needs to be general-purpose

enough to be able to represent most data. In addition, it has to be reasonably simple to handle and relatively flexible to allow future extensions.

Another approach towards interoperability is to combine the XML metadata standard and ontology mapping. This approach is still under active research and development. The mapping of ontologies also requires involving domain experts to define a set of common vocabularies and their relationships in the domain. Among some examples of such XML metadata formats, the XML Metadata Interchange (XMI) is an object metadata interchange format endorsed by the OMG (Object Management Group) for exchanging objects in distributed environments in general and in distributed development environments in particular [XMI00].

Also needed, with respect to service interaction, is a way to describe a service. Therefore, the Web Service Description Language (WSDL) is being used [WSDL01]. It is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described in an abstract way and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages, regardless of what message formats or network protocols are used to communicate.

6.4.3 Conclusion

The chosen solution for the service-oriented architecture of the Co-operation Platform was to build it on service announcement and discovery using Jini, service description using WSDL, and service interaction using SOAP. The answer which of the two approaches in order to achieve semantic interoperability is better suited for the Co-operation Platform needs further study.

6.5 Agent Orientation

The requirements on the services to be provided by the Co-operation Platform are manifold, ranging from technological constraints to the consideration of the particular interests and needs of end-users, administrators, and service providers. A large number of end-user devices exist with different capabilities concerning user interfaces, communication services, and protocols. As already stated earlier, one of the most important design goals within the context of this thesis is to provide access to services in a device and network-independent way – from anywhere, at any time. Standards and market solutions within the areas of telecommunication, tele-cooperation, e-business, and security have to be considered and have to be integrated.

Users are not going to tolerate the straightjacket of "standard services" in the future: they will want to tailor them to their own needs. Therefore, users, or rather their "personal agents", negotiate with each other to establish an optimum mode of communication and then requesting the underlying communication infrastructure for the appropriate services.

Thus, with respect to the Co-operation Platform, the concept of "network transparency" is of importance. The underlying communication infrastructure has to deliver the requested service (e.g. audio/video communication between two or more users) with the appropriate quality of service characteristics, on demand, at a cost that the user is willing to pay.

The multiway negotiation between users' personal agents and the communication infrastructure is a negotiation with many networks and service providers in order to provide the optimum amount of transparency for a particular instance of an application, at any point in time. The future trend is to increase connectivity with every Co-operative Workplace and people having multiple network connected devices. No single technology will provide a universal solution and so the question remains: How is this very heterogeneous collection of physical and logical networks made working together? New concepts are needed which provide adaptable, scaleable, dependable, ubiquitous network services in order to support collaborative work within geographical dispersed teams.

Today, one might think about the answering machine as a simple example of a machine acting as a user's agent or proxy, representing its owner in the style and choice of salutation given, and then receiving information from a caller for later transmission to its owner. In the near future, this familiar concept is expected to be extended in all sorts of directions as the machine intelligence starts to take on various aspects of the functionality of a so-called Personal Assistant or Personal Agent.

The Co-operation Platform starts at the "nuts and bolts" level by negotiation with another user's terminal to establish the highest level of compatibility possible/sensible (Can we use video? - What quality is available?). It works through things like exchanging calendar information, and in the future may eventually include language translation, speech/text conversion and so on.

The user has control of his terminal's "personality", in respect of how it interfaces both with him, and with various classes of caller or called parties. Another basic function of Personal Agents is to settle for the best and/or cheapest network service available - something the user increasingly has to do with special offers and rates on fixed as well as mobile services. This may be looked at as an extension of the "Least Cost Routing" mechanism, which has been around for some time in the corporate network context, but is becoming increasingly relevant to private users as well. With the possibility of charging for information and other services, the agent's responsibilities will extend beyond network service selection to include choice of information provider and intermediaries and to "look after" the several subscriptions to network and service providers.

Provisioning services in telecommunication networks is a complex process, which usually involves several parties. Mobile agent technology can help to streamline the process. This fact has been recognized by agent-related activities of organizations working towards service provisioning standards, e.g., the Telecommunication Information Networking Architecture Consortium (TINA-C) [MRK96] [KM96].

The Co-operation Platform upon which Co-operative Workplaces are reliant must have the flexibility to adapt to the different needs of its users. This flexibility must be achieved in an environment that is complicated by distributed tools and services. Advances in distributed object computing are helping but something more is required to control such a complex world. Being inherently distributed, flexible, and able to cope with

change, agent technology offers a different approach to building distributed software systems and therefore provides 'something more'. This approach helps to make an uncertain and dynamic environment manageable and at the same time at viable efforts.

Multi-agent technology provides an adequate approach for implementing high quality services within less time compared to other approaches. Furthermore, an agent-based approach brings the Co-operation Platform closer to the ultimate support environment for geographically dispersed teams: a plug-and-play network of team services. Such a network can automatically self-configure to accommodate team and user requirements. Before the agent-based architecture of the Co-operation Platform is being explained in detail, a closer examination is being given on what is to be understood as an agent within the context of this thesis and why agent technology is suitable as a basis for the Co-operation Platform.

6.5.1 Agents

In these days, the buzzword "agent" seems to be inevitable in every discussion about distributed systems. However, each research group provides another definition for agents; the main streams can be found in the Artificial Intelligence research and the Operating Systems research.

The agent paradigm is still discussed controversially. Critics often state that most agent application areas could satisfactorily be supplied with non-agent solutions. The object-oriented paradigm, indeed, experienced similar objections, but obviously, it is widely supported today. However, the agent technology is gaining a considerable role in the future, comparable to the object-oriented technology today.

Some possible agent scenarios include agents collecting data in a computer network, assisting in network management, being employed in mobile clients, and representing buyers and vendors in the electronic market place, i.e. the area of electronic commerce. One of the most salient motivations is the reduction of the network traffic in the currently already very crowded Internet by performing actions locally on the destination host, preferably without further interaction with the originator.

At the time of writing, there is still no universally agreed definition of an agent, but what follows is a list of the characteristics that agents can exhibit. It is important to note that there is no clear boundary between software that is an agent, and software that is not. It is also important to realise that not all of the following characteristics have to be present in a single agent [GK94] [Woo97] [WJ95]:

- independence of action or autonomy (i.e. an agent has its own agenda and desire)
- pro-activeness (i.e. an agent takes initiative, changes its environment and plans ahead)
- reactiveness to changing circumstances (i.e. an agent senses and reacts to the environmental changes)
- learning capability (i.e. an agent is able to learn from previous actions and reactions)

- negotiation and co-operation capabilities (i.e. an agent can negotiate and co-operation with other agents)
- mobility (i.e. an agent can migrate through the network from one place to another)

What can be said in general is that an agent exhibits autonomy in addition to one or more of the above characteristics. The more of these characteristics exhibited and the stronger they are exhibited, the more 'agent-like' that software is.

A software agent with the ability to travel is called a mobile agent. Formally, an (mobile) agent can be defined as a self-contained software entity that encapsulates some state and interacts with its environment in a way that helps the agent to proceed to certain goals as part of some internal plan. This may require it to communicate with other agents or to utilise mechanisms to change its location in a network.

In this short section, it will be pointed out why an agent-based solution is the best suitable candidate as an implementation base for the Co-operation Platform. The underlying models of the Co-operation Platform and mobile agent technology are very similar, thus greatly enhancing the design of an implementation of the Co-operation Platform based on this mobile agent technology.

The software systems upon which the Co-operation Platform is reliant must have the flexibility to adapt to the different needs of its users. This flexibility must be achieved in an environment that may provide distributed tools and services. Advances in distributed object computing, though helpful, are not sufficient to control such a complex world. Being inherently distributed, flexible and able to cope with change, agent technology offers a different approach to building distributed software systems. This approach can help to make an uncertain and dynamic environment manageable at comparably low effort.

Agent-oriented software engineering is often compared to object-oriented technology [GK94] and mobile agent systems are compared to distributed object-oriented systems like DCOM and CORBA [HCK95]. The objects and agents have many characteristics in common. Like objects, agents provide a message-based interface, independent of their internal state, data structure, and algorithms. Agents encapsulate their internal data and algorithms, just like objects. The primary difference lies in the interface [GK94]. Agents use a common universal language with agent independent semantics. Whereas in object-oriented programming, the meaning of messages may vary with language. Another important difference is due to characteristics of agents, like autonomy, pro-activity and mobility. A mobile agent can decide to move from one host to another while completely maintaining its internal state, whereas an object needs external activation to perform migration. Consider the following application, which highlights the need for mobile agents [HCK95].

Agents facilitate operation for disconnected mobile clients. Persistence of connection is not required for communication between the remote clients and servers. The client generates an agent for performing a remote operation, then launches the agent onto the network, during a brief connection session, and disconnects thereafter. When the client connects again, the agent returns back to the client with the outcome of the operation, if

any. This not only reduces the connection time of the remote client, but also might reduce the network traffic. This provides the ability that an agent can act on behalf of its user (Personal Agent) even if the user is not logged on.

The advantages of taking an agent-based approach for the implementation of the Co-operation Platform are:

- *Natural abstraction*: Agents model the real world (a community of entities each with their own goals, communicating and often working together to achieve mutual benefit). In this way, agent-orientation can be considered a natural extension to object-orientation.
- *Unifying technology*: Agent technology consolidates and builds upon a number of important computing technologies (object-orientation, distributed computing, parallel processing, and mobile code) and research results from other disciplines (e.g. artificial intelligence).
- *Power*: The power of agents stems from the underlying technologies. Agents can have built-in domain expertise, they can learn, reason logically, and adapt to new circumstances. Agents can co-operate to achieve more than they could individually.
- *Flexibility*: Agent-based systems are constructed from a loosely coupled set of software components. No single agent is fully reliant on any other. Relationships change dynamically at runtime as agents are removed, modified, or become overloaded. New agents can be introduced at any time.
- *Structure*: The agent paradigm can offer a well-organized structure in which domain, problem knowledge, and reasoning abilities can be encapsulated within the agent. Knowledge is power but it has to be used effectively to create the desired benefits. An agent's efficient behaviour, often viewed as 'intelligence', emerges through the interaction among its inner capabilities under the influence of the environment. The structures inside the agent are also in component form and so agent capabilities are straightforward to update and enhance.

From a user's point of view, the most important reason for using agent-based applications is that it enables tasks to be completed that would otherwise be mundane, very difficult, time consuming, costly, or just impossible.

6.5.2 Mobile Agents for Internet-based System Structures

An agent, which has the ability to travel, is known as a mobile agent. Formally, a mobile agent can be defined as a self-contained, software entity that encapsulates some state, reacts to the environment, acts upon the environment, can independently transport itself over an electronic network, and is able to communicate with other agents and systems via message passing.

Conventional system architectures like remote procedure calls, which were designed nearly 20 years ago with a more static and reliable system structure in mind, are not well suited for large Internet-based applications. Mobile agent technology, when combined

with mechanisms that are more traditional in an appropriate way, enables architectural concepts (such as function shipping, 'call by visit', or ‘code on demand’) that deal much better with these conditions. Furthermore, mobile agents are a higher-level abstraction than messages or procedure calls; their inherently distributed nature often provides a natural view of a distributed system, and they seem to enable structures in a networked environment that fit more naturally the real world.

It should also be noted that software agents, bringing together the two concepts "process" and "object", are interesting building blocks for flexible system architectures, even if they are not always mobile. In fact, stationary agents may be as important as mobile agents may: considering the aforementioned agenthood characteristics, agents are a suitable design pattern for large distributed systems the components of which may be regarded as interacting autonomous entities.

Since the dynamic creation of agents is a basic functionality of typical agent systems, agents are also an ideal mechanism to enable parallel processing. A typical example would be a search agent that sends out child agents to visit multiple machines in parallel. Of course, mechanisms are required to control the high degree of dynamic of such agent-enabled parallel computations.

For mobile agents to be useful, they must be able to communicate with various hosts and agents and move within heterogeneous networks. This requires a standardised framework and methodology for agent operations across the network. It is important to define and differentiate between mobile agent architecture and mobile agent execution environment. The former provides higher level (abstract) view of a mobile agent system.

A mobile agent architecture not only specifies the different types of agents and their internal characteristics but also define how agents interact with each other and with the environment. In essence, it defines the components, relationships, and behaviour of components in a mobile agent system. The latter provides an implementation level (a lower level) view of a mobile agent system. A mobile agent execution environment is a software system that provides a runtime system for agents to execute, a standard interface for interactions, services for creation, locomotion, and termination of mobile agents, supports agent mobility and communication while providing security for both hosts and agents. A Place is an abstraction of an agent execution environment. Place is formally defined in [Gra95].

In order to make use of existing distributed system functionality, it is desirable to have interoperability mechanisms that connect agent platforms to middleware concepts like CORBA or emerging Internet infrastructures like Jini. One suggestion coming from the OMG is called MASIF (Mobile Agent Systems Interoperability Facilities) [MBB+98], a standard that deals with interoperability issues between different agent systems and CORBA services.

6.5.3 Agent Systems

A brief overview on existing candidates for the underlying agent infrastructure is being given in the following section. After that, the decision why AMETAS as a development base for the Co-operation Platform has been chosen is being explained.

6.5.3.1 Open Agent Architecture

The OAA (Open Agent Architecture) is targeted towards creating a network of independent agents communicating by a common message exchange mechanism [SRI00]. The inter-agent communication language is called ICL. It enables agents on different platforms to communicate with each other.

The agent model behind the OAA is an extension of the distributed object model where the components, instead of using low-level method calling, rely on a high-level, declarative language message exchange. This assists users to communicate with agents easily.

The OAA system structure allows for delegated computing. Requesters may specify ways to reach a goal; service providers can provide performance estimation values; so-called facilitators route requests to suitable agent communities and meta-agents provide the facilitators with knowledge about the task solution competence of the overall community. This information may be utilised for task delegation decisions.

As the primary target of the OAA is to ensure the interoperability of agents on different, heterogeneous platforms, the mobility aspect is ignored; OAA agents may be spread over the network but are stationary after their installation. This invalidates any decision to choose OAA as a middleware platform for the Co-operation Platform.

6.5.3.2 IBM Aglets

The IBM research lab in Tokyo developed an agent development environment called "Aglets" [LaOs98], analogous to the Java "Applets". Aglets have lately been released as Open Source. As for the original Aglet release by IBM, Aglets do not run under Java 2. The latest open source release seems to incorporate ways to get them running, but it "is somewhat tricky" as the FAQ states.

The philosophy behind Aglets is similar to most Java-based agent systems. Aglet hosts provide the execution platform (context) for the mobile agents, called Aglets. Aglets migrate weakly, that is, they keep their variable states but lose the execution state. They may migrate actively (initiating the migration by a command) but can also be retracted (forced to return). Communication between Aglets is organised by intermediate proxy objects that allow for synchronous and asynchronous communication. Proxies do not migrate but forward messages to their respective Aglet.

The security model is based on the Java Security Architecture of Java 1.1. The Aglet Workbench does not present any special implementation. There are articles covering a suitable security model [KLO97].

There is no resource abstraction in Aglet contexts, which means, Aglets may access resources like any other Java object, if the necessary privileges are present. They may also open files as well as windows or sockets. It is somewhat unclear whether and how access control between different Aglets takes place; the privileges refer to system resource access and Aglet control (like retraction).

These unclear aspects in the security infrastructure, the missing resource abstraction, missing weight on autonomy, and the unclear state of the availability of Aglets in a stable version for Java 2 environments give reasons for doubts in being a suitable middleware for the Co-operation Platform.

6.5.3.3 MOLE

The Mole project was one of the first mobile agent systems based on Java [BHR+97]. A research group at the University of Stuttgart, Germany developed it. The development of Mole has ended with the latest version 3.0. The authors state that there will be no further support.

6.5.3.4 Grasshopper

Grasshopper merges the traditional client-server-oriented software design with the novel agent paradigm [IKV01]. The so-called agencies care for the execution and communication of the agents, as well as for the transport to another agency. The system design emphasises security precautions, which allow for encryption of travelling agents and access control at agencies. Furthermore, Grasshopper offers persistency mechanisms and management facilities.

Grasshopper comes with two add-on packages: One package provides MASIF compatibility; another package allows the integration of knowledge-based computing by implementing FIPA standards. The latest version of the core system is 2.2, enabling Grasshopper to run on Java 2 platforms.

Although Grasshopper offers a standard-aware implementation, advanced security concepts, and a number of useful features, the applicability for the Co-operation Platform is not guaranteed. The Grasshopper philosophy is heavily oriented towards the standard client-server communication mechanisms using remote method invocation, just adding mobility. The mapping of the Co-operation Platform concepts to components of Grasshopper is not straightforward; measures have to be taken to ensure the agent autonomy, which cannot be ensured in the presence of method calls. Furthermore, Grasshopper does not present a specific user-integration model beside the ability for agents to interface with the user directly.

While Grasshopper provides MASIF compatibility, this seems of minor importance for the realisation of the Co-operation Platform middleware because the adaptation to certain external components does not require generic access methods that limit the agent autonomy in turn.

Grasshopper was developed by Thomas Magedanz from GMD-FOKUS, now being distributed by a spin-off company named IKV++.

6.5.4 Asynchronous MEssage Transfer Agent System (AMETAS)

AMETAS, which stands for Asynchronous MEssage Transfer Agent System, is a multiagent development environment developed at the University of Frankfurt [HZ00] [ZH99]. The term ‘message transfer’ emphasises a central aspect of AMETAS: Agents, users, and services communicate via a mailbox system in an asynchronous way without the necessity to directly contact one another. AMETAS agents are fully autonomous and run on special execution environments called places, which are equipped with services that allow them to fulfil their tasks. With users being modelled as agents, there is no need to distinguish between users and agents, thus facilitating a consistent communication and handling. Agent-oriented programming needs to take into account the strict autonomy of the agents.

AMETAS consists of a set of Java class packages, which are required for the creation of agent places, new agents, and new services. Based on Java technology, AMETAS can be applied on arbitrarily heterogeneous networks (if a Java Virtual Machine is available for all components).

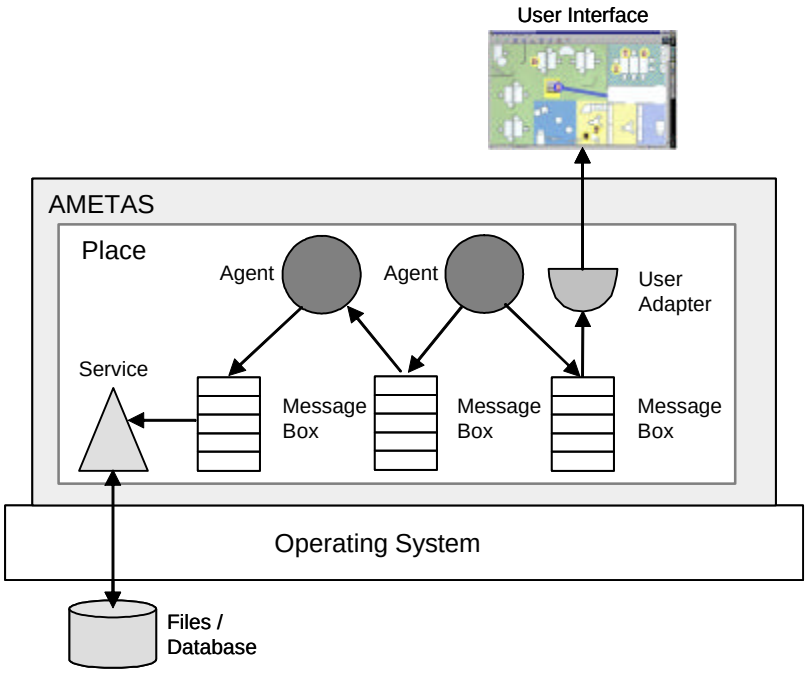


Figure 20 AMETAS platform

The following sections will show why AMETAS has been chosen as the best suitable candidate as an implementation base for agent-based part of the Co-operation Platform. The underlying models of the Co-operation Platform and AMETAS are very similar, thus greatly enhancing the design of an implementation of the Co-operation Platform based on mobile agent technology.

6.5.5 Suitability of AMETAS

The subsequent sections will elaborate on the suitability of AMETAS for the Co-operation Platform. Important design aspects with respect to the agent-based architecture of the Co-operation Platform will be discussed and explanations will be given, why AMETAS is an especially suitable basis is for that.

6.5.5.1 Design

The first and foremost argument for the application of a mobile agent system in the Co-operation Platform is the design aspect. The target of the Co-operation Platform is to integrate the human user into a virtual office world so that he is no longer limited by his physical location. Therefore, many of the objects of the physical world have to be modelled in the software, many of them being mobile by nature.

6.5.5.2 Mobility

Mobility and flexibility are the most important factors in today's work places. Employees are expected to be highly mobile to achieve their goals. The global society is working towards unlimited mobility leading to a total independence of location. The telecommunications market is evolving towards this goal as well as the software market. Since the Co-operation Platform puts a high emphasis on mobility and location independence of employees, the underlying infrastructure must offer enough functionality to support autonomy, mobility, and flexibility without a compromise. A rigid client/server paradigm would have two drawbacks: First, mobility must be implemented as an add-on to an immobile architecture, therefore not being directly supported. This would render the design process much more difficult. Second, the need for mobility could not be met to the extend necessary to achieve the goals of the Co-operation Platform.

6.5.5.3 Autonomy

In addition to mobility, autonomy is a big issue in a system like the Co-operation Platform. The software entities modelling real objects have to be as autonomous as the objects themselves, because otherwise the software would limit people's work environment. The more the design of applications is targeted towards real-world simulations, the more precisely the genuine aspects of real-world objects must be taken into account. These real-world objects are - unlike most of their counterparts in traditional software design - completely autonomous entities, especially humans who must be integrated in the Co-operation Platform model.

The trend in modern work environments and in the architectural design of office buildings is towards autonomy. The worldwide facility management system as part of the Co-operation Platform is an excellent example for this trend. In this project, largely autonomous installation units, called RoomComputer, are used as building blocks for the construction of complex buildings [RBB98]. Each of these installation units possesses actors and sensors to perceive its environment and act according to changes. This requires a certain degree of

- autonomy for being able to make their own decisions,
- intelligence for ensuring that these decisions are adequate, and
- co-operation for integrating with other installation units.

This is very similar to definitions of agenthood given for example by Wooldridge and Jennings in [WJ95]. Consequently, intelligent software agents may be considered the most adequate design paradigm for this project. The same applies to other Co-operation Platform components since they also require their basic building blocks to posses the above-mentioned properties to model real world work environments.

6.5.5.4 Communication

AMETAS allows an easy but very precise modelling of real-world processes by focusing on the autonomy aspect. As people do not offer methods to be invoked by other people, neither do AMETAS agents. Communication between autonomous entities can only be adequately modelled using an asynchronous message transfer scheme. The most prominent examples for asynchronous communication techniques in the real world are mail, e-mail, and fax. The fact that messages need to be passed asynchronously was the main idea behind the design of AMETAS. AMETAS enables agents to act autonomously in much the same way as human beings do. No other entity can take control of an agent if it does not allow it.

Even if a scenario suggests a synchronous message passing strategy, this is easily achievable on the base of asynchronous communication by using handshake message acknowledgement. AMETAS components offer special customisable methods to assist the implementer in creating message synchronisation. A system that models user behaviour, actions and interactions must apply the same communication paradigms that are used by human beings. This simplifies the design process considerably and improves usability.

6.5.5.5 Decentralisation

In systems with a global perspective such as the Co-operation Platform, decentralisation is a major factor. Platform components potentially run on a large number of hosts spread all over the Internet. As client machines are getting more powerful it becomes a logical step to move functionality from the server to the client to make the overall computational load manageable. A central server can become a bottleneck or even a central-point-of-failure while client machines are idle. Mobile agents have a great potential for decentralisation since they can move from one host to another at runtime, making it possible to react to changes in their environment. If it is beneficial for them to communicate locally, they move to their communication partner. Any communication

that is not directly relevant for the end user can be done locally, resulting in less bandwidth usage and potentially faster response times. The resulting distributed system is much more flexible and scalable than classical client/server architectures.

6.5.5.6 Scalability

One big challenge of complex distributed systems is their configuration and maintenance. Major configuration changes have to be propagated to all participants. This can be very time-consuming and often requires a system shutdown to make the necessary changes. This is due to the tight coupling of components, which often makes it impossible to replace one system, and keep the others running. AMETAS applications, on the other hand, are lightweight in the sense that all components of an application are decoupled due to the asynchrony of communication. One benefit of this decoupling is that the installation of new components and the exchange of old ones become very simple since they are only loosely linked. The second big advantage is that mobility can be used as a means for installation. Mobile AMETAS agents can take the necessary software (other agents, adapters, or services) and configuration data with them and install them on several hosts according to a specified itinerary [ZH99]. Thus, installation and configuration changes become quite simple and can be largely automated. Therefore, the time and effort spend on these issues can be reduced to a minimum. The lightweight maintenance process and the resulting reduction in the required time make AMETAS-based applications very flexible.

6.5.5.7 Integration

AMETAS has been designed in a way that permits AMETAS-based applications to interact and integrate with external applications. Therefore, adapters and services can be used to build interfaces to other software entities [ZH99]. Therefore, AMETAS-based applications are in no way limited. Other techniques, which at first glance may seem to be incompatible with software agents, can interoperate with AMETAS. For example, EJB technology can be used to enhance an AMETAS based application with special interfaces.

6.5.5.8 Security

The AMETAS agent system includes a flexible security system especially designed for use in an agent system [ZMG98] [ZH99]. Although the platform only requires Java 1.1.7, the security configuration is at least as flexible as the security architecture known from Java 2. The availability of a strong security system is indispensable for employing agent technology in real-world applications. AMETAS security is based on identities being assigned privileges at each place. Agents wandering in the network are authenticated as being started by some identity and are assigned privileges accordingly. Migration is protected by encryption. Using sender signatures can ensure non-repudiation and integrity. Furthermore, because of the agent system design, agents are isolated against each other, being unable to inspect private data even when they interact at the same place.

6.5.5.9 Drawbacks

AMETAS also has some drawbacks that should be mentioned.

- lack of standards,
- high memory footprint, and
- low performance.

AMETAS currently neither conforms to the MASIF [MBB+98] nor to the FIPA standard for agent systems [FIP00]. However, AMETAS allows adding components that enable interaction between components of AMETAS and the outside world. There are concepts how to integrate e.g. CORBA services using special adapters. Furthermore, AMETAS offers KQML [FiWi91] (but no FIPA ACL [FIP00]) capabilities. These restrictions can easily be dealt with by providing special interface components where necessary instead of utilising some general interoperability solution like MASIF.

AMETAS is an agent system solution based on Java. Although this entails the application of the agent platform in heterogeneous networks, the memory consumption is relatively high. Being run by Java 1.1.8, AMETAS places typically consume at least 5 MB of memory; with user adapters, the consumption rises to about 10 to 12 MB per place. In Java 2, the places require even more. This is due to the large footprint of the Java Virtual Machine, which affects any Java-based agent system.

The flexibility of AMETAS component design allows alleviating this problem by offering remote connections between the user terminal and the agent system, thereby allowing even small appliances like PDAs to be used.

The integrated security system and its cryptographic subsystem - rather than the asynchronous message system - causes much of the performance loss. However, clever application design may prevent too much performance loss, for example by avoiding migrations where possible. Turning off the security system was possible with former versions but has been abandoned because of incalculably high security risks.

6.5.5.10 Conclusion

AMETAS is a highly suitable platform for the implementation of the Co-operation Platform. Considering the goals and challenges of this thesis, the usage of an agent platform like AMETAS adds valuable functionality to conventional middleware platforms like Enterprise Java Beans. The nature of the Co-operation Platform, the objects to be modelled and the co-operation paradigm to be realised favour an agent-based design since mobile software agents possess all properties required. It would be very hard if not impossible to model all aspects of autonomy, mobility, decentralisation, communication, and flexibility using a conventional client/server platform. AMETAS is especially suitable for this thesis because it is built around the concept of autonomy.

The strict decoupling of application components enables software engineers to build a very flexible, maintainable, and configurable application. The asynchronous communication paradigm responsible for this decoupling meets the requirements of communication between autonomous entities much better than other synchronous techniques. The decision to build the Co-operation Platform on top of AMETAS opens

new possibilities and is in no way a limitation. Other application components based on different platforms can be integrated using adapters and services. Thus, a decision for AMETAS is not a decision against other paradigms that can be useful to this application area.

6.6 Transparent Data Access

The persistent data, on which the various components of the Co-operation Platform rely, are far more than just a dataset stored in some relational or object oriented database. They typically include data in databases, files, information available in directory services, documents available on a hard drive or on the network, and the wealth of data available from the Internet. Each of these data types needs to be taken into account when developing the Co-operation Platform, yet each is in a different format, stored in a variety of containers, and probably accessed in fundamentally different ways (Figure 21).

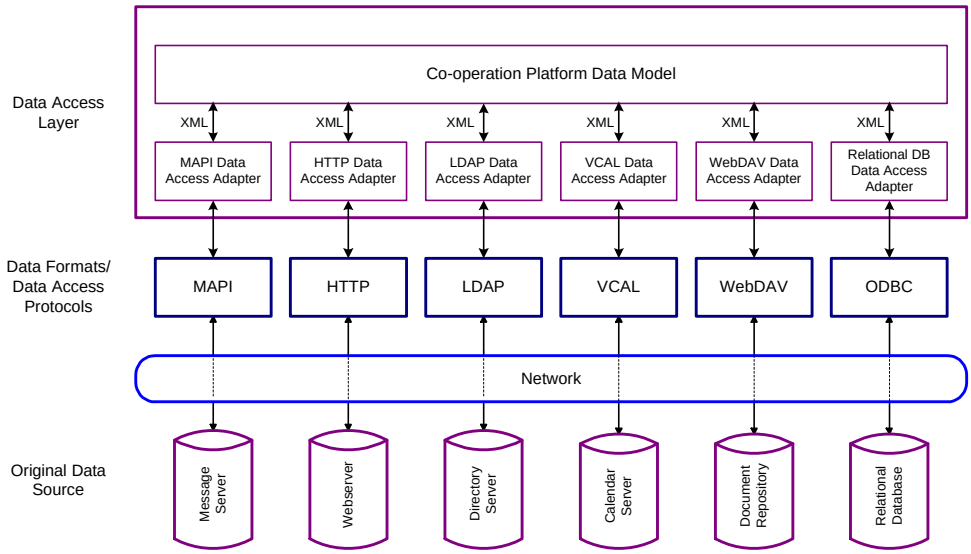


Figure 21 Data Access Layer (DAL)

This requires a developer to use separate APIs to access the different types of data, i.e. he has to become an expert in a multitude of data access methods. One solution would be to incorporate all the data from the different data sources into one single database. This approach solves the problem of having a single API that also gives transactional control - but it has a number of compromises.

First, it requires moving (sometimes huge amounts) of data. Second, one typically does not own much of the data being needed, such as the wealth of information on the Internet. The next problem with moving the data into a single database is that tools today do not look for or store data into a single relational data store. Messaging applications such as Email, directories, and other tools look into their own data stores for their data.

One would have to rewrite the tools to access the data in the single database or duplicate the data to the single data store from where it naturally lives - then one will run into problems with synchronising the data.

Another problem is that one size does not fit all. The way that databases deal with data is not the best way to deal with all types of data. Data (structured, semi-structured, and unstructured) is stored in different data sources based on how the data is used, viewed, managed, and accessed, i.e. data storage often depends on the underlying data model being used.

Rather than transferring all data needed from different data sources into a single database, the Co-operation Platform implements a Data Access Layer that provides transparent access to various data sources through a common data model, and a common interface, regardless of where the original data resides. The objective is to obtain a virtually transparent access to all types of persistent data needed in the Co-operation Platform through a single data access model.

Each component within the platform speaks to a common interface that generalises the concept of persistent data. The Data Access Layer is based on XML technology. This allows an individual database to easily expose its native functionality and allows generic, reusable components to augment the functionality of simpler data providers when needed. All of the intelligence and code for accessing and manipulating the data exists on the Internet server. Centralising the data access logic on the Data Access Layer and reducing the components downloaded to the client are ideal for many data access scenarios.

6.7 Extensible Mark-up Language (XML)

The ability to describe structured data in an open text-based format and to deliver this data using standard HTTP protocol is significant in enabling a new generation of Co-operation Platforms. The Extensible Mark-up Language (XML) is the "lingua franca" within the Co-operation Platform for the exchange of information, object and service interaction, and application data.

XML is a data format for structured information interchange. Developed under auspices of the World Wide Web Consortium (W3C), XML is the industry consensus for next generation Web architectures [W3C-XML]. XML promises to increase the benefits that can be derived from the wealth of information found today on Internet Protocol (IP) networks around the world. This is because XML provides a uniform method for describing and exchanging structured data. The ability to describe structured data in an open text-based format and to deliver this data using standard HTTP protocol is significant in enabling a new generation of collaborative applications.

XML employs the concept of generic mark-ups: tags that are inserted into a document, structuring it into nested elements. HTML is the most popular format using this technique. While HTML has a fixed set of tags, XML allows authors to freely choose vocabulary from their field of application to name tags.

Standard vocabularies are summarised in Document Type Definitions (DTD), formal grammars that declare tags and their structural relations. DTDs are already available for many domains, including electronic commerce (e.g. XML-EDI), science (e.g. MathML), synchronised multimedia (e.g., SMIL), software documentation (e.g. DocBook), or agent technology (e.g. WIDL). The XML data format is complemented with standardised concepts for document display, hyperlinks, multimedia, and metadata, that is, XML can be considered as a "family of standards" [W3C-XML].

In particular, with respect to the user interface of the Co-operation Platform, the power and beauty of XML is that it maintains the separation of the user interface from the structured data. In contrast, the Hypertext Mark-up Language (HTML) specifies how to display data in a browser. For example, in HTML one uses tags to tell the browser to display data as bold or italic; in XML one only uses tags to describe data, such as company name, order number, and order items. In XML, one uses style sheets such as the Extensible Style Language (XSL) and Cascading Style Sheets (CSS) to present the data in a browser by translating or rendering it into an HTML representation. XML separates the data from the presentation and the process, enabling to display and process the data by applying different style sheets and applications.

This separation of data from presentation enables the seamless integration of data from diverse sources. Customer information, purchase orders, research results, bill payments, medical records, catalogue data, and other information can be converted to XML on the middle tier, allowing data to be exchanged online as easily as HTML pages display data today. Data encoded in XML can then be delivered over the Internet to the desktop. No retrofitting is necessary for legacy information stored in mainframe databases or documents, and because HTTP is used to deliver XML over the wire, no changes are required for this function.

6.8 Platform Architecture

The high-level architecture of the Co-operation Platform comprises the following layers and blocks of abstraction as illustrated by Figure 22 :

- **Real World**: At the bottom one finds the real world with tools, resources, – physical as well as informational ones, and real rooms. Rooms are physical locations of people and physical resources.
- **Integration Layer**: The Integration Layer provides an IT-model of the real world with multiple representatives of real world entities, i.e. rooms, resources, and tools. Those representatives carry uniform interfaces and addresses, which allow managing entities, to combine and address them, and have them communicate with each other. Wrapping them in a homogeneous way and providing them with unified interconnection interfaces does the integration of various previously heterogeneous entities.
- **Middleware**: The middleware layer is the place where relations between the various entities within the co-operation platform are established and maintained for realising specific applications. Thus, real and virtual entities are

put into a specific work context. Abstract services, provided by the underlying Integration Layer, can be configured, customised, and put in a context. It is possible to e.g. dynamically set up, modify, and close Virtual Project Offices, as projects are set up, modified, and closed. This process is called VPO configuration. It is based on a reservoir of generic services, e.g. for communication, databases, security functions etc. Defining relevant work zones and associating services and resources with them generates an intuitive and task specific user interface. Depending on the domain of work, this reservoir and this configuration can be augmented with domain specific tools and services. For example, business projects require the integration of business processes, e.g. by utilising available tools that allow developers to assemble business process components. The ability to dynamically create VPOs is an important factor in creating virtual organisations rapidly and facilitating their operation in cyberspace. A flexible spectrum of security services makes it possible to tailor the platform so that it can assure confidentiality for commercial reasons and guarantee individual privacy, as needed. Security policies for VPOs specify for example, their appearance to the outside world, access to them, and rules for signatures and confidential documents. Much of this is based on existing security tools, such as smartcard and Public Key Cryptography. Security functions are being made available to provide confidentiality, authentication, and security profiles, to define responsibilities, rights, and roles within a project and in the end to allow the contracting of experts for a team over the network.

- **Work Contexts**: This layer is the place where working relations among persons are established and maintained for supporting specific work contexts. Together with the middleware layer, this layer forms the core of the Co-operation Platform. The platform presents its users a context-oriented view of its contents and members. This is achieved by importing the necessary tools, services, and resources into the co-operation platform, relating them to each other where meaningful, embedding them in a specific context, surrounding them with a context oriented user interface and thus creating value-added project services.
- **User Interface**: On top of the Co-operation Platform architecture is the user interface layer. Users can access the features and services of the Co-operation Platform through it. The access is mediated by concrete end-user devices (e.g. PCs, PDAs, and mobile phones), the characteristics, and capabilities of which strongly influence design and appearance of concrete user interfaces. By separating interface functionality and appearance, it is possible to add new end-user devices to the user interface layer without major problems to other layers. The Co-operation Platform addresses the development of a new paradigm for presenting Co-operative Workplaces. The paradigm is reflected by the user interface that is application specific, context oriented and thus intuitive to use. Views on the user interface can be customised according to the requirements of the work contexts of its users. At the user interface level, a VPO may look like a real team office. This preserves some of the essential advantages of a local shared office, such as transparency, ease of interaction,

availability of project data etc. Geographically distributed members of a team are projected into VPOs (e.g. being represented by avatars) as if they were all physically present in a single office. By including a project's input and output, its resources, tools, resource and services, a VPO captures the context of the project.

- **Persistent Data Storage**: The Persistent Data Storage layer provides persistence of data objects as well as propagation of real-time update notifications. Through the Persistent Data Storage layer, all information required by the co-operation platform is stored persistently, e.g. user data, the group modelling data, the infrastructure, and the document management metadata as well as the actual document contents.
- **Persons**: At the top, one finds the persons, who can interact with the Co-operation Platform in different roles.

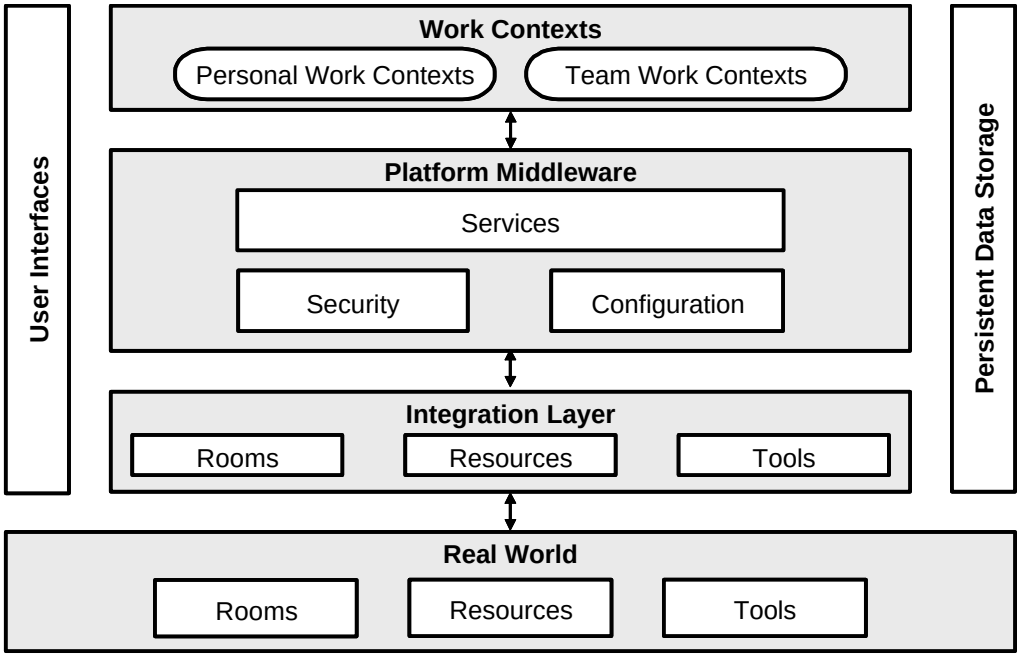


Figure 22 High-level architecture of the Co-operation Platform

6.8.1 General Platform Functions

The Co-operation Platform is the key system component which provides the facilities for devices, components, and networks to fully interact, despite their possible original inherent heterogeneity, and which takes care that a uniform and ubiquitous view is presented to all team members regardless of their physical location. The platform itself addresses:

- integrated flexible support to deal with (changing) co-operative work contexts,
- means to make the environment (physical and virtual) aware of existing situations and adaptable to new ones,
- the integration of existing tools, resources, services and sources of information, and
- the provision of different modes of co-operation, dedicated to specific work contexts.

The platform provides means for dealing with and integrating the various elements of a given work environment such as its physical objects (e.g. rooms, devices, resources, installation), virtual objects (e.g. data, files, tools, means for audio-visual perception of remote people, state of a project) as well as people. Those means comprise facilities to define and configure a Co-operative Workplace.

From the platform, customisable Co-operative Workplaces can be generated, as the result of the symbiosis between physical and virtual workspaces, the latter being represented by so-called Virtual Project Offices. Therefore, the platform offers a set of configurable basic building blocks and services with which such Co-operative Workplaces can be flexibly built and tailored to user needs reflecting the existing underlying technical infrastructure. Such a platform is more effective than a new monolithic system, as it makes it possible to incorporate established tools and services from an installed base.

Obviously, it would be unacceptable to expect from a user or an organisation to adapt their infrastructure to that of the Co-operation Platform. So, mechanisms are being included that bridge a number of different device capabilities, so that available and preferred media channels, bandwidth, compression algorithms or other device dependent capabilities can be used automatically, without direct user involvement.

Moreover, the Co-operation Platform has been designed to be network aware (i.e. capable of adjusting dynamically to changing network situations) and device/machine aware, i.e. capable recognising and identifying devices by their addresses and capabilities using a common registry and the specific capabilities of each device.

The Co-operation Platform addresses awareness. During sign-on, every user of the platform is registered with a central registry so that the VPOs do show at any given time only the members who are currently available and active. This technology uses registration and keep-alive schemes so that it is always possible to know in real-time the user that are present in a specific Virtual Project Office.

The Persistent Data Storage layer of the platform provides persistence of data objects as well as propagation of real-time update notifications. A database is being provided using a commercially available Database Management System (DBMS) accessed via the Open Database Connectivity (ODBC) database access layer [ODBC97]. The use of a standard DBMS and ODBC is chosen in order to meet the adaptability requirement, as it does not limit the system to a specific DBMS and thus ensures applicability in a wide range of environments. Through the Persistent Data Storage layer, all information required by the Co-operation Platform is being stored.

People collaborating in Co-operative Workplaces are not necessarily physically in contact with each other. Thus, authenticity becomes highly important. Actually, the Co-operation Platform takes authenticity as the base for further security services, like role-based access control and tracing. Access control will manage the authorisations of users. For further details on security, see chapter 8.

Finally yet importantly, the platform provides the necessary means, infrastructure, and runtime environment for the different building blocks. As such, it also supports error recovery mechanisms. Therefore, it guarantees information access from any place at any time. It provides human centred and intuitive user interfaces for the people working in a virtual organisation. It is possible to construct these user interfaces from basic building blocks so that they may differ for different application cases yet have as much commonality as possible.

Based on an extensible set of services, the platform offers context oriented integrated services as required by a specific work context. Previously heterogeneous software tools and basic services can be integrated by wrapping them in a homogeneous way and providing them with unified interconnection interfaces based on the paradigm of service-oriented programming [BC01].

The specific services offered by rooms, resources and tools are being integrated into the co-operation platform via the integration layer using the approach being described earlier. Dedicated manager's control access to the available services; e.g., so-called Room Agents control the access to all services offered by the corresponding room. Users use the services offered through Virtual Project Offices. A Virtual Project Office may only use a subset of the services available. Together with a Service Provisioning Environment (SPE), a configurer can be used for a VPO to determine, which services are being provided within a particular VPO. In order to be flexible, the concept of abstract services is introduced, representing a service class, offering a specific functionality. Abstract services serve as interfaces or proxies to specific services, which implement the functionality.

The services being offered can be classified as follows:

- Service Provider Services (SPS)
- Network Provider Service (NPS)
- Network Management System (NMS)
- Location Information Services (LIS)

The SPS represent the interests of service providers and support the provisioning of e.g. telecommunication services to users. The SPS try to find the optimal solution in terms of quality of service and cost for providing the service to the end-user through their Personal Agent (PA). They adopt two distinct roles:

- Client of network services offered by Network Provider Services (NPSs)
- Provider of a variety of services to users represented by their Personal Agents.

An SPS may also support management activities such as user profile management and billing of the users for the services used. The key functions performed by the SPS are as follows:

- Negotiation with the NPS for the service connections agreed between the SPS and the PA.
- Requesting connections from the NPS.
- Informing the PAs upon activation of the agreed service according to the service scheduler.
- Receiving service management requests (configuration, termination) from the PAs, validating them and to realise such management operations by operation requests to the NPS.
- Informing the initiating PA about the service management results.

The Network Provider Services (NPS) represent a network domain. Their major responsibility is the provisioning of network connectivity upon requests from an SPS. For this purpose, the NPS has to interact with the SPS representing the user, the NMS representing the local network domain and with other NPSs representing other network domains in the global environment. The key functions performed by the NPS during service provisioning are as follows:

- respond to call for proposals for connections,
- set-up and teardown connections on the NMS,
- negotiate with the SPS for connections,
- provide connections for PAs, and
- maximise utilisation of the network via the NMS.

The Network Management System (NMS) is a service that is used by the NPS to control the set-up and teardown of connections.

Location Information Services (LIS) allow the localisation of e.g. users or devices, which are being provided by a yellow page service, based on LDAP technology.

6.8.2 Agent-based architecture

This section maps the high-level architecture of the Co-operation Platform with the agents developed based on AMETAS. Referring to the middle layer of the high-level view of the UNITE architecture the UNITE Middleware one of the most important challenges is the integration of physical resources, i.e. users, rooms (hardware), and tools (software). The virtual world has to be linked with the real world in order to achieve the best support for team collaboration. In order to achieve this goal we propose the multi agent approach.

6.8.2.1 Users

Probably the most intuitive mapping is a mapping of users to agents, where agents in an AMETAS point of view are autonomous, mobile, and to some extent intelligent. On this basis, the Co-operation Platform introduces the concept of the Personal Agent (PA), an agent that is associated to exactly one user. Once the user has authenticated him to his

Personal Agent, the PA acts on his behalf following goals and requests the user has defined before or adapted and configured at runtime.

The Personal Agent is the representative of the corresponding user at any time. Even when the user is offline, the PA continues to aspire to its goals. Typical goals are keeping track of calendar entries, e.g. sending notifications via SMS when a scheduled meeting is supposed to start, searching for information in general, answering less complex requests of other users and/or their Personal Agents, e.g. requests for appointments.

Furthermore, the PA has access to the users' persistent data storage as well as to particular services available on a specific place depending on its privileges. The privileges depend on the user's role in a particular project, i.e. the project manager might have general access rights on documents. The service, which offers access to, documents checks these privileges and grants or denies access. Another example is the instantiation and configuration of the project environment and its work zones.

The PA represents the user in its dealings with setting up communication links to other users. Therefore, it communicates with other Personal Agents (e.g. for the purpose of user localization, capability exchange, session initiation etc.).

The key functions performed by the PA during service provisioning are as follows:

- elicit and validate local service resources such as local telecommunication resources, e.g. connection points, and terminals;
- elicit and validate service partner's destination and service constraints, e.g. starting time, and duration;
- negotiate with Service Provider Services (SPS's) the best fit for service provisioning in terms of technical constraints and/or preferences defined by the user;
- provide an interface to allow co-operative tools such as MS Netmeeting, to be started up by the SPS chosen.

6.8.2.2 Teams

In addition to Personal Agents, the Co-operation Platform introduces the concept of Team Agents (TA). A TA is an agent, which is associated with a team of users collaborating in a joint project. The Team Agent represents the team as a whole to persons outside the team or to other teams. A Team Agent, which represents the entire team of a project, also represents the corresponding project as a whole to the outside world.

The most outstanding purpose of the Team Agent is access control. It has access to their whole project database, particularly, the user directory to reliably identify each agent asking to get access. After a user e.g. entering a VPO the corresponding Team Agent exchanges some information with the personal agent to get informed about what features and services on the client side are currently available and vice versa. This information is for example needed when a connection for communication purposes between two or more users should be established. Thus, on server-side, the availability of each party can be decided and the appropriate services on client side can be invoked.

Another task for the Team Agent is providing a Personal Agent with appropriate access rights to the projects database depending on the user's role in this project, e.g. project manager, team-member, administrator, etc. A detailed and differentiated role-based rights management offers adequate means to provide e.g. any document with different access rights. Access rights particularly include the right to read, write, print, or copy parts or even the whole document on the server-side as well as on client-side.

6.8.2.3 Rooms and Devices

Physical workspaces such as rooms can contain different devices such as computers, printers, phone-sets, audio-/video-conferencing equipment, etc. The RoomComputer offers a consistent interface to these facilities, which then can be wrapped by services to integrate these room facilities into the Co-operation Platform in order to make them available to users, agents or other services. Within the Co-operation Platform, rooms are being represented by Room Agents.

6.8.2.4 Tools

The term tool contains all sort of software that supports team collaboration in the sense of the Co-operation Platform paradigm. Tools are being wrapped and integrated by means of services.

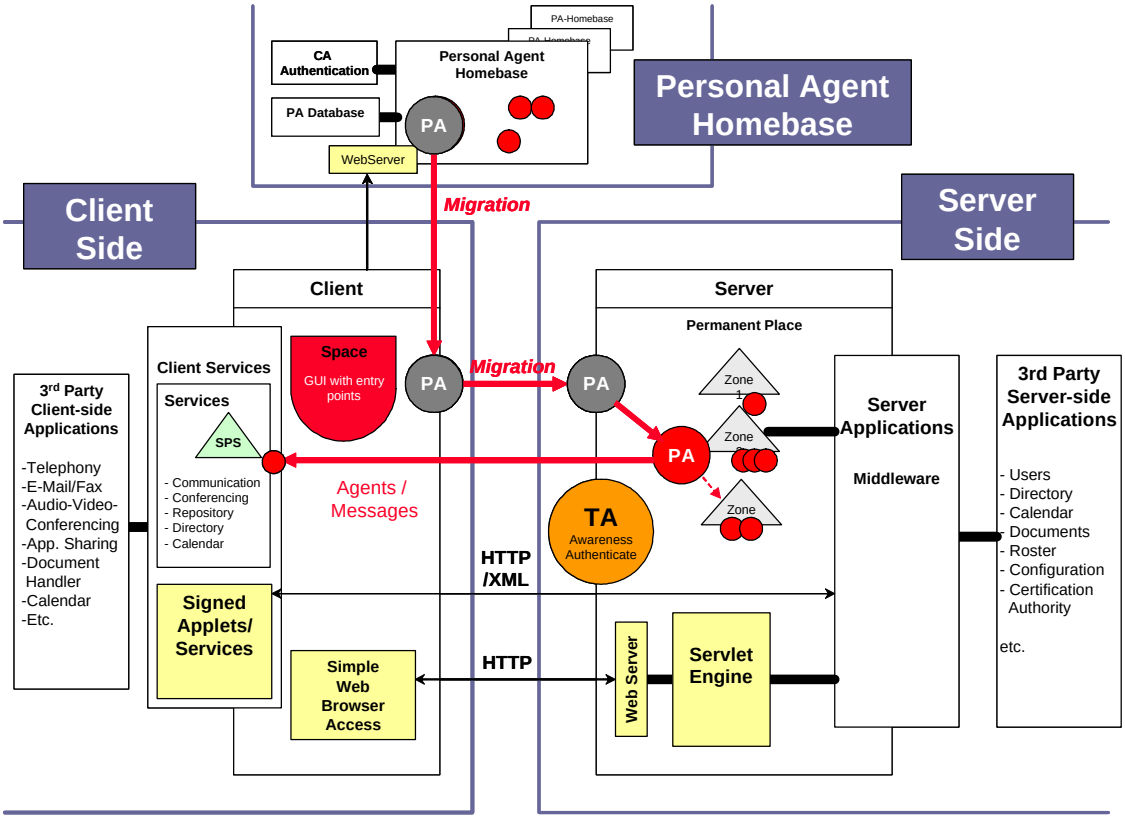


Figure 23 Agent-based architecture of the Co-operation Platform

Figure 23 shows a high-level adaptation of the Co-operation Platform architecture from an implementation point of view. In terms of AMETAS, on the client side a so-called Temporary Place runs. Temporary places can be switched off easily and are being used at the user's end. In contrast to that, so-called Permanent Places are building the core infrastructure of our multi agent system. They are used to implement the server side.

Personal Agents live at so-called Homebases, which are Permanent Places. Each time a user gets offline, the corresponding Personal Agent migrates back to his Homebase. The Homebase allows the Personal Agent to stay alive and to reach his goals defined by the user like collecting information about the projects. The Homebase serves as a place for the agent, where it can store information and messages it has collected or even makes itself persistent.

Personal Agents do migrate to a Temporary Place on the client-side, whenever a user has logged on. The Personal Agent can be configured and maintained there. The user can redefine goals or add new ones either for online or offline phases. The client-side services are wrapping all third party applications that reside on the client-side. As already mentioned earlier, rooms, their facilities, and software tools, which support team collaboration in distributed project environments, are being provided by services concept. Personal Agents can access and invoke the client-side services in order to, e.g. establish a videoconference between the corresponding user and his teammates.

On the server-side, a permanent place provides access to e.g. a VPO. The same mechanisms as on the client side wrap specific services, which are being provided at the server side. Examples are services, which give access to the project's document repository, the project's roster, or the team calendar. Personal Agents migrate to the server side in order to use the offered services there. The Team Agent acts like a guard, asks for authentication, and grants certain privileges depending on the user's role.

Two graphics are discussed in the following in detail. First, according to the high level view of the platform architecture Figure 24 details the architecture of the client side, whereas Figure 25 details the high-level view for the server side. On both sides, one can find the same general building blocks adapted to the requirements of the client side and the server side respectively.

On the client side, the real world containing users, rooms, and tools is represented through Personal Agents (PA), Room Agents (RA), Resource Managers (RM), and Tool Managers (TM) respectively. The Personal Agent is characterised at least by three main building blocks: the user session management service, user work context management service, and the user capability management service. Each of these building blocks interacts with services of the so-called Personal Office to provide a specific view of the user.

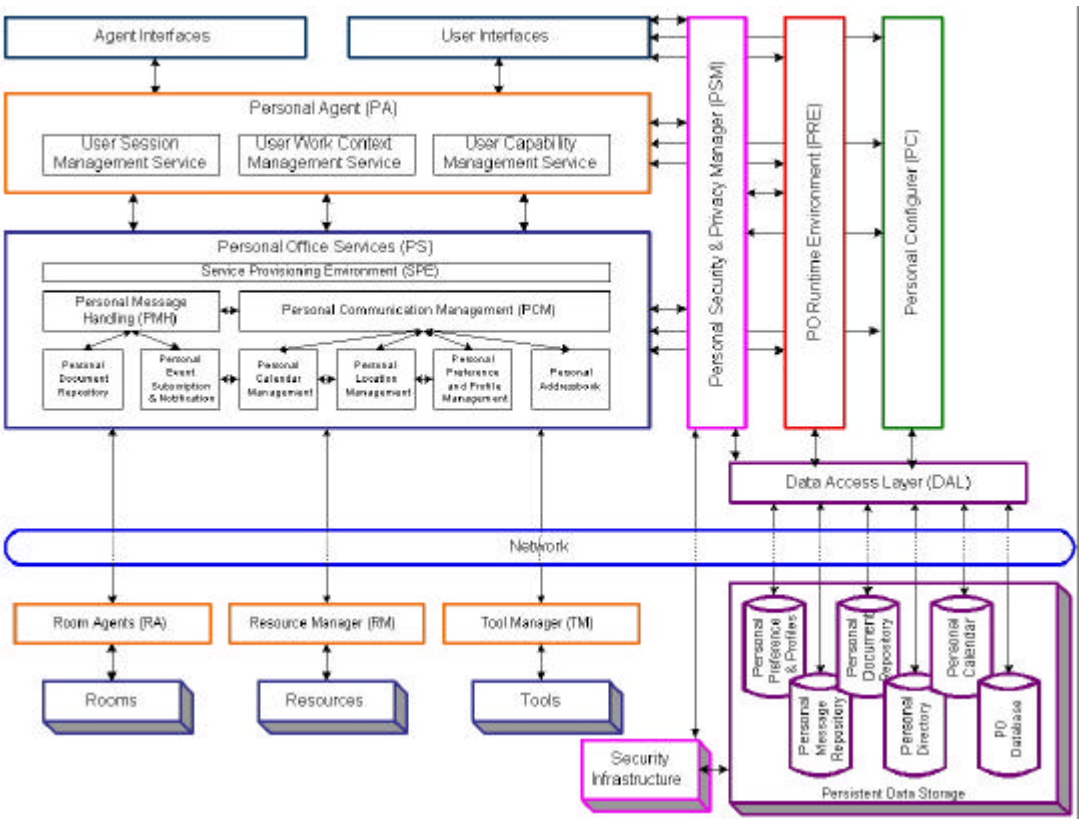


Figure 24 Client-side architecture

The user session management takes care of the current state of the user. It provides information about whether the user is online and in a particular project, whether he is available or reachable. Furthermore, the work context management keeps track of the specific work context the user is currently working in. Work contexts can be individual work, group work or working in a conference for example. In all these cases one wants to know if the user is currently available and if he can establish e.g. a video communication. Finally, the user capability management knows about users current set of devices and tools present at the users workspace, and therefore the set of services available to him.

On the client-side, several client specific services called personal office services are offered. Mainly, the user's Personal Agent uses them, because they provide a personal message handling and a personal communication management. Supported by services like the personal document repository, the personal calendar and location management, and a personal address book, these high-level services are handling requests by the Personal Agent. For example, based on integrated calendar tools provided by the Resource Manager, the Personal Agent can decide whether a requested date is still available for an appointment or not. Last but not least, the Personal Configurer can be used to redefine the goals of the Personal Agent, enhance the services in the personal office, or even add new ones.

Finally, all these services have access to several personal databases monitored by the security infrastructure. Through a uniform Data Access Layer (DAL) Persistent Data Storage provides and maintains for example the users message box, document repository, a directory server, or a calendar server.

The following figure, Figure 25 , describes the architecture and mechanisms for the server-side. Again, the users, the rooms, and the tools are represented by their corresponding agents as stated in the high-level view of the platform architecture. A data access layer similar to the client-side implementation provides access to a persistent data storage containing the team policies and profiles, the team document repository, or a calendar server – just to mention a few. The Security Manager takes care that only granted access is possible and secures the transmission of any data.

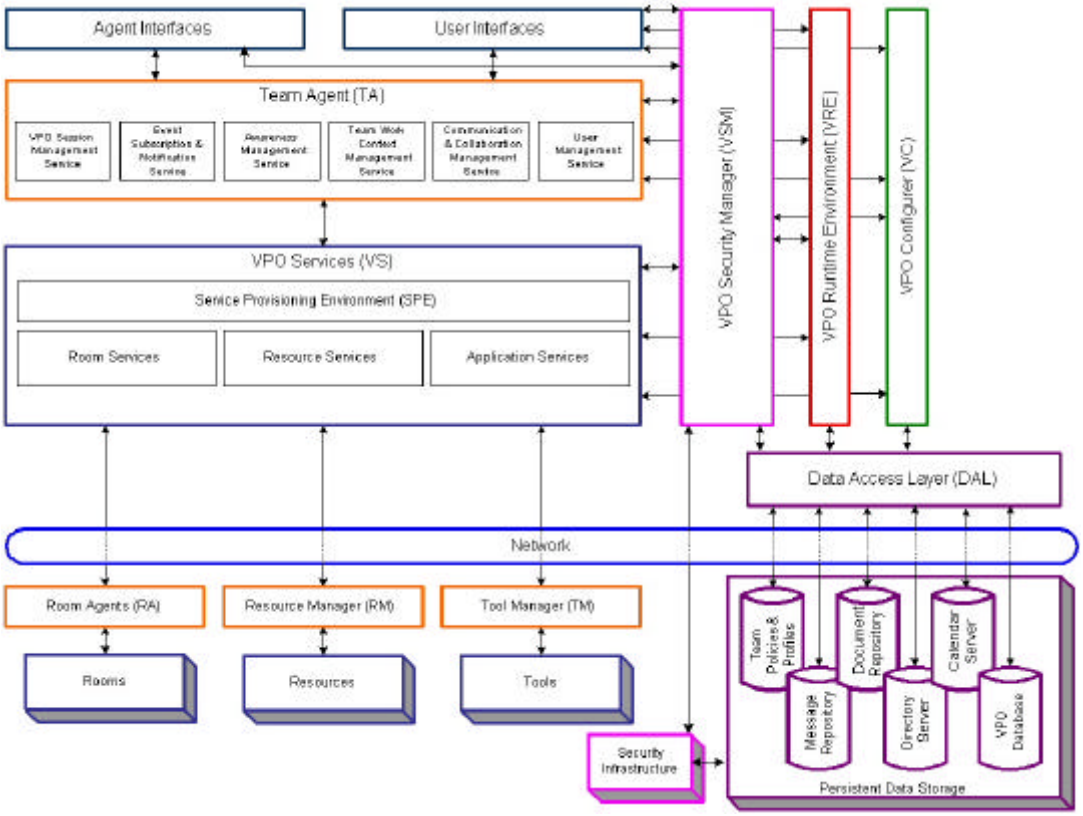


Figure 25 Server-side architecture

Unlike the client-side, there is a so-called Team Agent on the server side, which represents the whole team to individual team members as well as to the out-side world. The Team Agent comprises the session management, event subscription and notification, and an awareness manager. Moreover, the different work zones, each supporting a specific work context are managed and maintained by the Team Agent. Last but not least, this agent, if necessary, will negotiate communication and collaboration between the team members.

As mentioned above, on the client side the Personal Agents rely on the personal office services. On the server-side, the Team Agent depends on the VPO services, which include user services, room services, and application services. For example, access to several third party applications that support team collaboration like a document repository is provided by the application services. Room services integrate a proxy service to get access to physical rooms on the client side without any need to change from the VPO application to some building management tool. Physical rooms are integrated and accessible in the virtual world. Finally, the user services support a consistent team calendar and a message repository, for example.

6.9 Virtual Project Office

The overall architecture of VPO deals with people who want to collaborate with each other for a common purpose, tools and services which are needed to carry out individual tasks, and the Co-operation Platform which embodies the principles and policies for a coordinated effort. Due to the Co-operation Platform, the collaborating people get a team or context oriented view at the user interface instead of a set of different views on individual tools, which are neither integrated nor related to the work context.

The VPO is a distributed collaborative system, providing the users with a cooperatively accessible document storage as well as immediate information about other users' actions. It is realised as a client-server application, which allows the dynamic creation of teams together with configuration, and customisation of their work environments and which supports teamwork in such a collaboration environment.

Its architecture is centred on a database server, which provides persistence of documents and data objects as well as propagation of real-time update notifications. This central database is provided using a commercially available relational database management system (RDBMS) accessed via the Open Database Connectivity Layer (ODBC) [ODBC97] database access layer. The use of a standard RDBMS and ODBC as an access interface was chosen in order to meet the adaptability requirement, as it does not limit the system to a specific DBMS and thus ensures applicability in a wide range of user communities.

The central database stores all information required by the system: the document management metadata, the infrastructure and group modelling data as well as the actual document contents. For VPOs a central data model was developed, based on previous work on the model of a cooperative document repository for the public administration performed within the POLIWORK project [TBR98]. The original object model was revised to address the specific requirements introduced by the collaborative settings to be supported by the project’s application goals, especially the stronger need for group and access right modelling. The VPO object model contains all relevant information about not only the corresponding project, but also about the system environment, the users accessing it and the communication and collaboration capabilities at the different user sites.

The client applications of the VPO were developed in Java, using the JDBC [HCF97] interface as a means to access the database server. Java was chosen as an implementation language in order to make use of the existing and future infrastructure within designated user communities. The use of the Java programming language and the JDBC database access mechanisms ensured operability of the solution over Intra-/Internets. The use of the standard database access mechanism JDBC provides a large degree of flexibility in the selection of the database system used within the system as well as the actual driver library used for database access. The database access module is configurable in order to allow easy exchange of database drivers, which also aids the evaluation of the performance characteristics of the different access libraries.

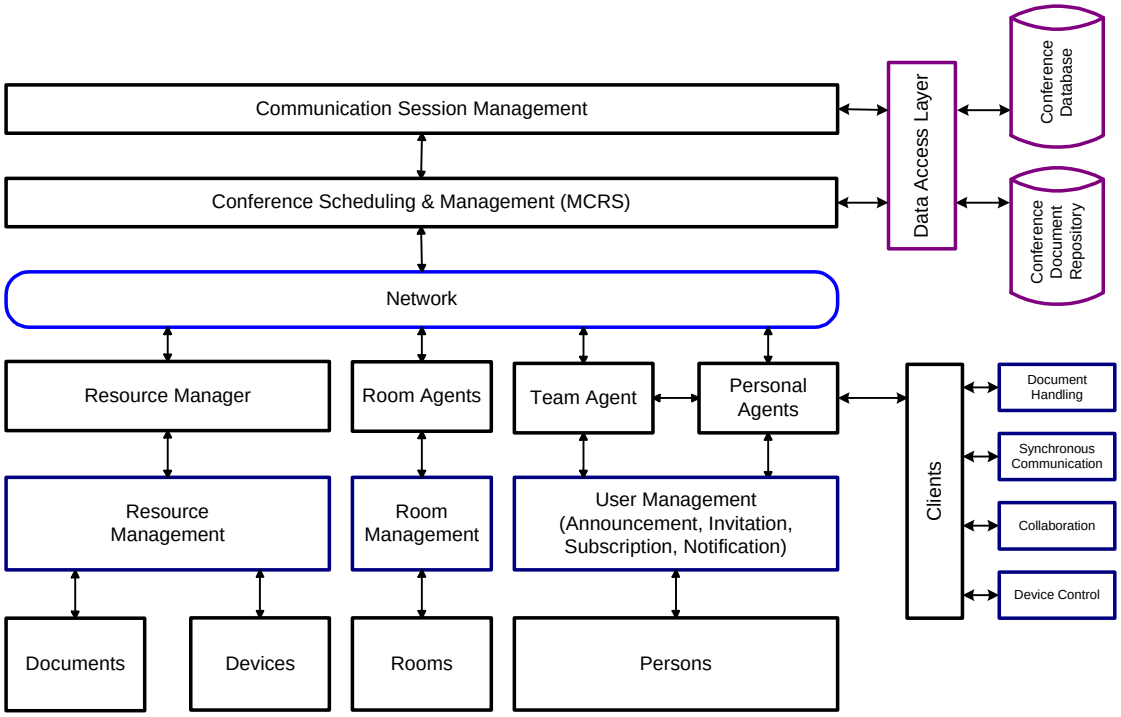


Figure 26 VPO communication and conferencing subsystem

The truly co-operative use of the VPO in the context of a synchronous audio/video conference necessitates the synchronous coupling between the users' client applications. The client applications contain interfaces to external components such as A/V conferencing, and application sharing. Many of the off-the-shelf components used in the project, such as the application sharing system or the videoconferencing system, merely provide C++ programming interfaces, which had to be made available to Java applications in a consistent manner. In order to ease flexibility and exchange of components at a later stage, generic Java interface modules were specified for the individual functional modules which abstracted from the actual programming API and presented the module's functionality in an abstract and reusable manner. These interface modules were then mapped onto the interfaces available in the application modules currently used. This abstraction layer fulfils the adaptability requirement, since it effectively hides the set of third-party cooperation and communication tools actually used and facilitates easy exchange of system components.

In order to fulfil the requirement for supporting existing document processing applications and enable the cooperative work on documents created and maintained with non-cooperative applications such as the familiar office suites, the VPO provides integration with external applications (e.g. MS Word) as well as with third-party multimedia conferencing and application sharing packages (e.g. MS Netmeeting). This is achieved through an abstract application integration layer, which can be tailored to support a wide range of applications for different document types as well as a number of application sharing packages, accessed through their API functionality. The integration with the application-sharing package allows automatic launching of the appropriate application along with the application-sharing module within a conference. Due to its knowledge of the current interaction situation and the set of connected users, the VPO can launch the relevant tools and applications in a mode, which is required for the current communication setting.

6.10 Multimedia Conference Reservation System

Multipoint conferences based on the ITU-T H.320 and H.323 protocols demand a so-called Multipoint Control Unit (MCU) together with a reservation of required resources (e.g. number of audio/video channels, bandwidth) on such an MCU [ITU-H320] [ITU-H323]. The functionality for making reservations on MCUs is provided by the Multimedia Conference Reservation System (MCRS) developed within the framework of the Co-operation Platform.

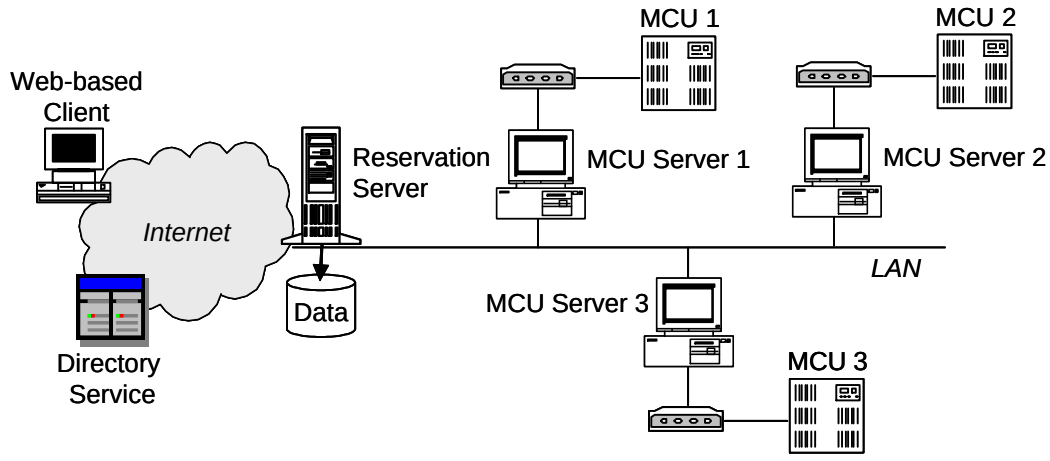


Figure 27 MCRS architecture

The control and scheduling of MCUs is performed via their own reservation database, since most MCUs do neither offer a directly nor a standardised usable API. For this task, a specialised service was developed which mediates between an MCU's reservation database (resident on a dedicated server machine) and the MCRS conference database. Any relevant changes performed in the Co-operation Platform database (e.g. the

scheduling of a new conference) is immediately fetched and transferred to an MCU reservation database. This service interfaces to the Co-operation Platform just like the other services, by way of database based distributed shared memory which is automatically kept in a consistent state when changes occur, thereby informing the daemon that changed data needs to be propagated to the MCU reservation database.

The system architecture of the MCRS sub-system consists of several components, which can be categorised in the following functional areas: a Web-based user interface client, a directory service, a reservation server, and MCU dependent servers (Figure 28). This modularisation of MCRS facilitates a very flexible design, better scalability, and reusability of the implementation.

6.10.1 Web-based Client

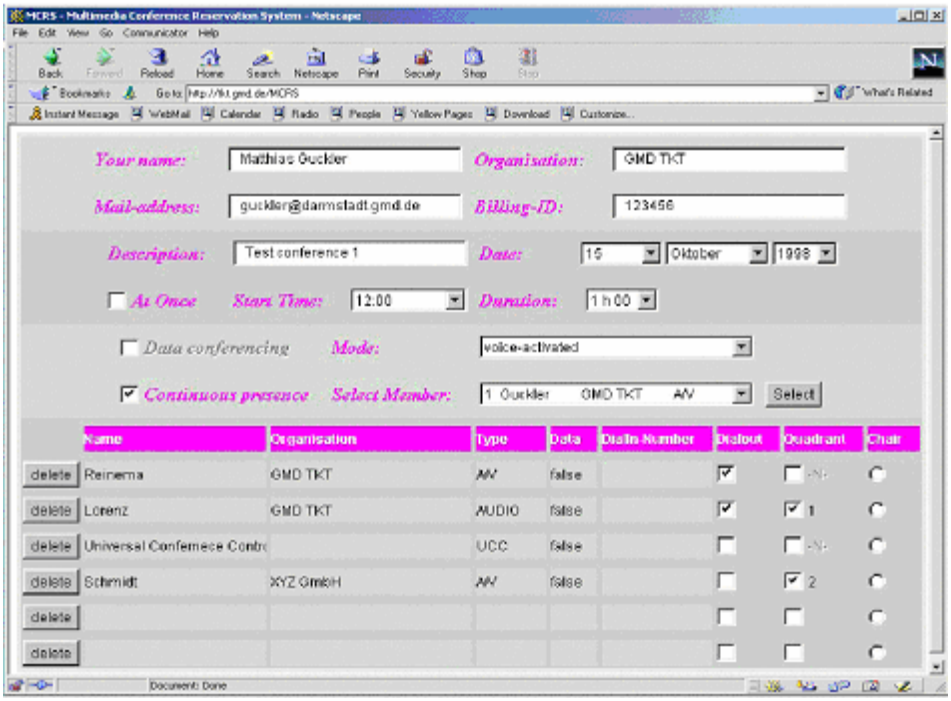


Figure 28 MCRS user interface

A major user requirement was an easy to use user interface for setting up a conference with other partners. In order to make the functionality of MCRS easily accessible, a Web-based user interface was developed. Its usage can best be presented in the form of short usage scenarios.

Conference reservations conducted using MCRS can be roughly subdivided into:

- spontaneous ad-hoc conferences (e.g. triggered by a user's need for direct communication with others) and
- pre-planned scheduled conferences (e.g. a multi-party conference at a certain point in time and with a specific topic).

Both types of conferences can be initiated and administered by using the Web-based user interface client.

For spontaneous multimedia conferences, e.g. on a certain document, all the user typically needs to do is to indicate the document on which to collaborate and to select the other user(s) with whom the document is to be shared from the list of available users. The Reservation-Server then uses its knowledge about the users and their available infrastructure (by looking up a directory or a connected database) to initiate the (possibly multipoint) audio/video communication, start the document processing tools required and begin the collaborative editing/presentation session.

Pre-planned conferences are characterised by a specific point in time at which the conference takes place, a conference topic, and a set of documents, which form the basis for the conference (e.g. the conference's agenda and documents that are to be presented in the conference). These conferences are set up by selecting a set of users (or user groups), entering a conference topic, selecting a conference time and date, and creating a document repository, which contains the conference's documents. Due to the interactive collaborative nature of MCRS, the scheduled conference automatically shows up on all users' Web-Browsers in order to inform them about this conference. When the time of the conference has arrived, the users are being reminded of this in advance and asked to indicate whether they are ready to enter the conference. Upon confirmation, MCRS automatically starts the A/V conferencing module, establishes the connection to the other users (or to the MCU, in the case of a multipoint conference), and opens the associated conference repository. The users can now conduct their conference, cooperatively present and edit documents placed in the conference repository.

6.10.2 Reservation Server

The Reservation-Server forms the heart of the MCRS system. It is the central process, which automates the reservation and administration of conferences without the need for an operator. It receives reservation requests from the Web-client and transmits them to one of the connected MCUs, depending on the type of a conference (e.g. audio graphics, multimedia) and the required resources.

The software architecture of the Reservation-Server is shown in Figure 29 . The chosen software architecture makes it possible that conferences can be reserved and administrated by several users at the same time. This assures a higher degree of availability of the reservation system and shorter response times.

As already mentioned earlier, the conference reservation system is based on the Session Initiation Protocol (SIP), a protocol, which enables the invitation of users to participate in multipoint multimedia conferencing sessions [HSS97]. This protocol has been designed initially for multipoint conferences over the Internet using IP-Multicast. Within the developed system, it is has been enhanced for H.320/H323 based conferences.

However, SIP is not suitable for all administrative purposes, which a multimedia conference reservation system should fulfil. For example, SIP does not offer direct support for listing conferences, which have already been reserved. Additionally changing existing conference reservations is also not supported by SIP.

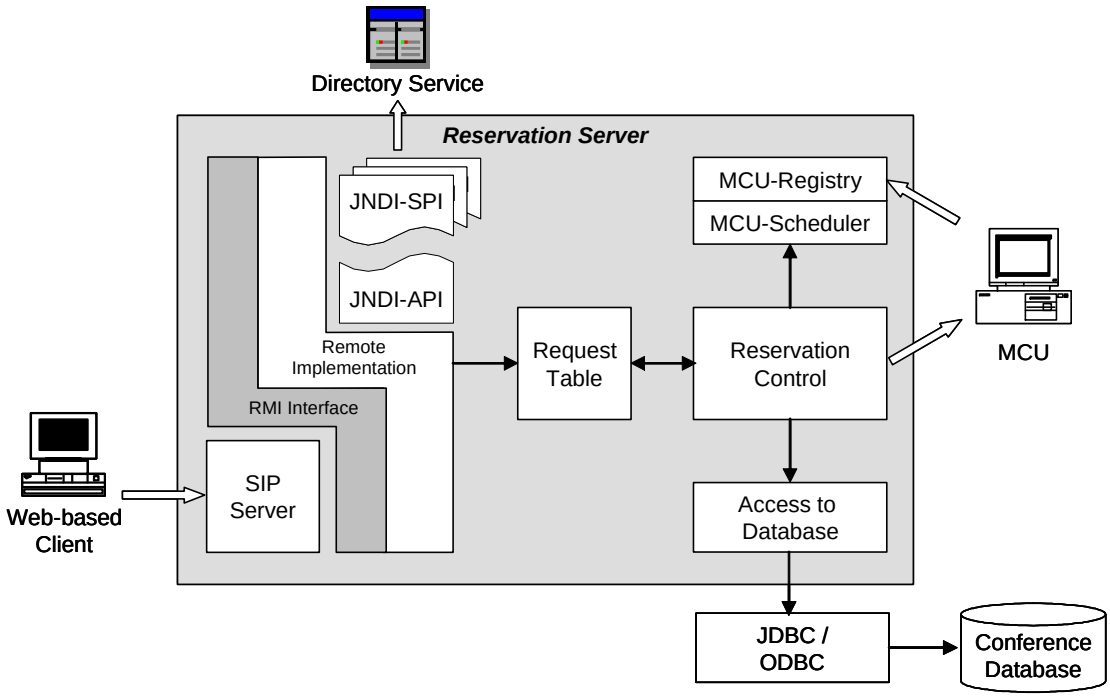


Figure 29 Reservation server

The SIP-Server provides the interface to the Web-client. After it has received a reservation request, it stores them in an internal database via the RMI-Interface of the Reservation-Server. In principle, the SIP-Server would be able to carry out the reservation of the required resources by itself. However, the RMI interface offers the advantage that a dedicated SIP-Server can be assigned to each connected Web-client, which provides a better de-coupling of the overall system and therefore shorter response times.

The MCU-Registry is used for the registration of available MCUs. Such a registration of an MCU is automatically performed whenever the corresponding MCU-Server is started. During this registration process, the Reservation-Server gets information about the capabilities as well as the available resources of the respective MCU.

The MCU-Scheduler performs the selection of a suitable MCU for a specific conference. For example, if the conference is going to be a pure H.320 conference, the MCU-Scheduler reserves the required resources on a corresponding H.320 MCU. In the case of a mixed H.320/H.323 conference, it reserves the necessary resources both on a corresponding H.320 and an H.323 MCU as well as on the required gateway between the both. Before a reservation request is going to be transmitted to the respective MCU-Server, the MCU scheduler checks, whether the required resources are available on the corresponding MCU.

6.10.3 Directory Service

At present most of the current available conference reservation systems provide only a static assignment between users and related conference sites. Nowadays it is getting more and more important to support people in a desk-sharing situation, mobile users and the usage of non-predetermined rooms for team meetings. This means that whenever one user wants to call another user, he currently needs to know the corresponding site. Therefore, directory-based user location services have been integrated. Within MCRS user data can either be stored and retrieved from so called directory servers (in this thesis the Netscape Directory Server has been used [NDSDK98]) by using LDAP (Lightweight Directory Access Protocol) or stored and retrieved from conventional relational databases by using ODBC (Open Database Connectivity). This provides a greater flexibility and allows the integration of already existing user directories.

Directory services are integrated via the JNDI-API (JAVA Native Directory Interface - Applications Programmers Interface) and the JNDI-SPI (JAVA Native Directory Interface - Service Provider Interface). The JNDI-SPI transforms queries from the JNDI-API into protocol specific queries for the corresponding directory service. Such a service provider interface is used for each corresponding directory service. A widespread directory access protocol is the LDAP protocol (Lightweight Directory Access Protocol) [WHK97].

6.10.4 MCU-Server

The MCU-Server is a server process, which carries out a conference reservation on a particular MCU. In order to be able to reserve conferences on a MCU the MCU server must register at start-up at the Reservation-Server. During this registration, the resources (e.g. number of the free slots, supported bandwidths, supported coding and compression techniques) available on the respective MCU are being transmitted to the Reservation-Server. The MCU Server maps the received conference reservation requests from the Reservation Server onto the MCU specific interfaces. The used MCU does provide an API, although it is not very convenient, which allows to control its operation. This API can be accessed by using asynchronous dialup connections, a native command set, and the so-called Operations Support Systems Interface (OSSI) protocol [Luc97], running over a modem.

6.11 The RoomComputer

The RoomComputer is an autonomous installation unit (typically located at the doorframe of the room) which is connected to the Intranet/Internet. It is based on a tiny embedded PC with a touch sensitive LCD display as local interface and has integrated loud speakers and a microphone. A smartcard reader and so-called Single-Chip-PC’s are being attached to it.

The RoomComputer provides a flexible distributed framework architecture implemented by means of JAVA and Web technology which can easily be tailored to the needs of its

users. Compelling new facility management applications are enabled by the development and deployment of RoomComputer networks together with smart controllable room devices, that moves closer to the vision of “Cooperative Buildings”, which enhances users’ comfort, convenience, and ease of use.

However, one problem in developing such applications is that there are numerous competing building automation protocols (e.g. EIB, LON, BACnet, X.10) and different application programming interfaces (APIs) for accessing room devices. Since building automation systems are likely to remain heterogeneous, with multiple incompatible protocols used to discover and control different types of devices, application programmers are faced with the prospect of using multiple protocol-specific APIs with no common underlying programming model.

In addition, electrical installations in rooms and buildings are getting more and more complex these days. In the past it was often sufficient, to switch on and off devices. Today developers of facility management applications have to deal too much with the protocol-specific details of the room devices and networks that application uses. Furthermore, when new room devices are being installed, developers have to modify their application and/or install the appropriate drivers to use them, i.e. there is no standard software infrastructure to shield applications from these low-level details.

To control the different devices in a room (such as lights, blinds or heating) there are at present many conventional island solutions which require great effort of wiring (with a great number of cables) and configuration. Moreover, these systems can often not being installed, configured and managed independently from each other. Changes in the usage of rooms and buildings often require rewiring and reconfiguration of existing technical components, which results in great efforts and expenses. In addition, in modern facility management systems it is nowadays unavoidable to also provide functions for operating, controlling and managing as well as for reporting, billing, and accounting.

The RoomComputer provides a flexible distributed framework architecture implemented by means of Java, Internet and Web-technologies, which can easily be tailored to the needs of its users. This section describes the layered architecture of the RoomComputer framework.

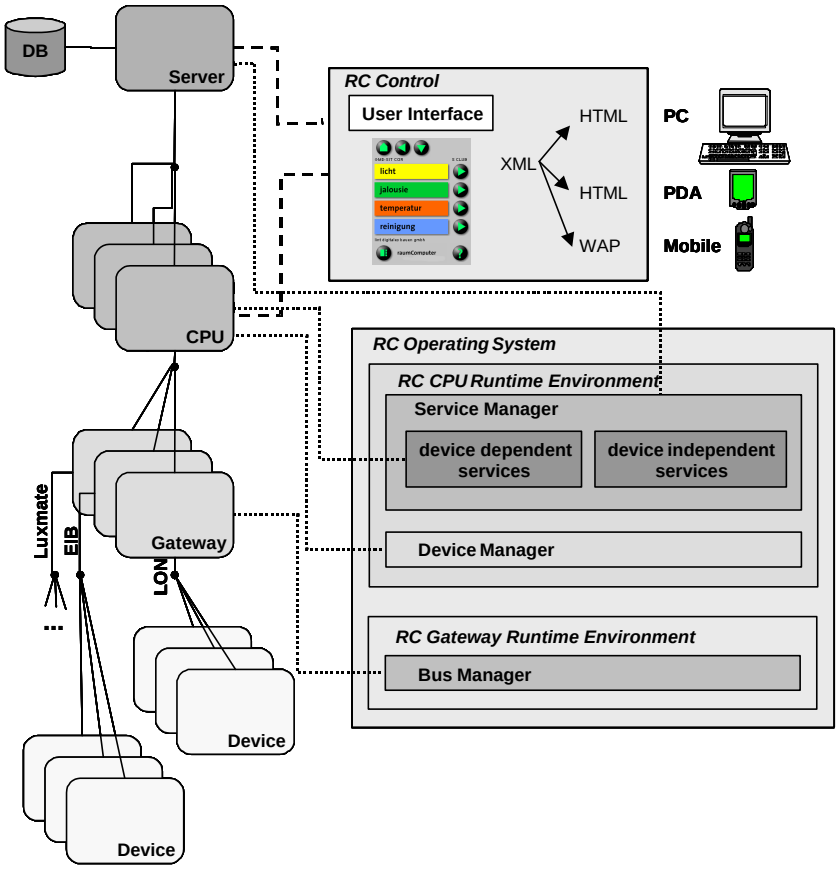


Figure 30 RoomComputer architecture

Similar to the architecture of the Co-operation Platform the RoomComputer architecture is also following a layered architecture. On the lowest layer, one finds the devices (e.g. sensors, actuators) with their different functions and interfaces. Devices are either being connected via a gateway or via an Ethernet connection to the RoomComputer. The RoomComputer itself is an embedded system based on an ARM processor and Embedded Linux as its operating system.

6.11.1 Bus Manager

A bus manager provides for, that devices can be connected to the RoomComputer independent of the field bus systems they are using (e.g. EIB, LON). In this case the bus manager provides an abstraction of the protocol being used by the specific device. Instead, the RoomComputer can interact with the devices connected to it by means of XML of formatted messages, i.e. the bus manager supports an interface, which is independent of the protocol of the connected devices and is designed to allow application programmers to access, control and query the status a device by means of XML formatted messages. For

example, one can query the current temperature inside a room by sending a message to the temperature-sensing device and then control the heating device accordingly.

The bus manager manages all devices connected via the respective gateway and provides plug and play functionality, i.e. it recognises if a new device is attached to the field bus, it is monitoring if a previously attached device has been removed and/or is not working anymore.

6.11.2 Device Manager

Device integration is integral to the RoomComputer's support for devices. Device integration provides the glue between the physical device and the software object model of the device. The services offered by a particular device allow the Co-operation Platform to communicate with the actual devices using their native protocols and their respective networks. The software objects use their integration services to read real-time data, send commands to the device, and provide support for reconfiguration and reprogramming.

A Device Manager manages all the devices connected to a RoomComputer and provides abstract interfaces to them, i.e. it provides the necessary device drivers and allows physical devices to be integrated in a consistent manner, independent of the communication protocol they are using. An Event Handler monitors all the events transmitted by the Bus Manager (e.g. attaching or detaching a device). It also monitors the state changes and error conditions of the connected devices. Such events are being described in XML formatted structures.

The advantage to the developer is that the way an application accesses a room device is the same, regardless of the underlying protocol that is used to talk to the device. By shielding applications from the underlying heterogeneity of building automation components and providing XML based interfaces for controlling room devices, the RoomComputer help getting applications that use room devices developed quickly and easily.

The Device Control module is used for the actual communication with the connected devices. This module provides an abstract interface to the respective device. In the DeviceRegistry the Device Manager can store the data of the connected devices, what will lead to shortened restart times in the case of an error.

6.11.3 Service Manager

A RoomComputer does not only offer simple facility automation services (e.g. switching a light), but it also offers the possibility to implement complete scenarios in order to support a certain work context. Beside the pure provisioning of facility automation services, the architecture of the RoomComputer is flexible enough also to integrate other services like telecommunications or e-commerce services (e.g. setting up a phone connection or establishing a videoconferencing link, allowing the user to order coffee for a meeting or even a taxi just by simply clicking on a button). Therefore, the RoomComputer architecture provides a Service Manager, which manages all available services and provides access to them. Service providers for underlying devices and

networks translate the high-level operations on the service into corresponding commands sent to a device. Two different kinds of services can be distinguished: Basic Services and Complex Services. Complex Services can flexibly combine various Basic Services in order to support scenarios and specific contexts of work.

According to the kind of the provided services, different kinds of Service Managers are provided. The Service Manager Facility Automation manages all device-dependent services, whereas the ServiceManager Telecommunication and the ServiceManager E-Commerce do manage all device-independent services.

6.11.4 RoomAgents

RoomAgents (RA) offer a set of services together with a corresponding set of user interfaces that allow other agents within the Co-operation Platform to configure physical workspaces for a specific work context; i.e. to discover and control room devices such as lights and climate control systems, blinds and curtains, LCD projectors, VCRs, and telecommunication devices (e.g. in electronic meeting rooms). The primary goal of developing such RoomAgents is to simplify and reduce the cost of developing software applications that enhance users' comfort and convenience through the intelligent use of controllable room devices via the RoomComputer. Other agents within the Co-operation Platform, such as the Personal Agents, interacting with the RoomAgents, are being shielded from differences in the underlying networks and protocols used to communicate with room devices.

6.11.5 Infrastructure

On the lowest layer of the RoomComputer architecture one finds the devices. RoomComputer devices are inexpensive terminal equipment. They can configure themselves over the network, can dynamically be added during runtime and/or replaced by other devices. Different from traditional facility automation systems, RoomComputer devices make use of the Ethernet as a powerful and high-performant field bus. This allows connecting different sensors, actuators, and closed-loop control systems to the RoomComputer as well as cameras, finger print sensors and access control systems.

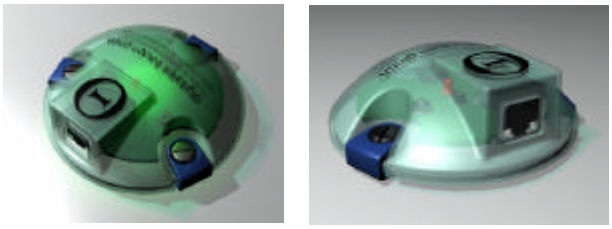


Figure 31 RoomComputer devices

6.11.6 Network Layer

The emergence of digital devices, such as digital TVs, beamers, camcorders and other (consumer) electronic devices controllable across high-speed A/V networks (such as e.g. IEEE 1394), as well as “no new wire” data networks using powerline and RF technology, has the potential to create a revolution in the capabilities of appliances and facility automation. Wireless networks, such as e.g. IEEE 802.11 Wireless LAN or Bluetooth, enable multiple PCs, Internet appliances and other computing devices to exchange data and share Internet connections. In addition, appliances like lighting, security systems, heating, and cooling systems can increasingly be integrated into co-operating systems via open networks, such as the Internet.

For example, if one wants his physical workspace being set-up for a specific work context when he enters it (e.g. the lights being turned down and a videoconference being set-up within a meeting room), he/she may swipe his/her office identification card past the access reader at the entrance to that particular workspace. Then the card reader communicates with the lights and the videoconferencing system. Although these devices may not have been designed to talk to each other, they can be modelled enabling communication and interoperability. This modelling of devices as software objects provides a more powerful architecture for constructing applications that can interact with devices and among each other.

This approach allows the Co-operation Platform to seamlessly integrate physical workspaces together with virtual ones and vice versa. Furthermore, the deployment of building networks and smart controllable room devices enables the development of compelling new facility management applications that move closer to the vision of so-called Co-operative Buildings. They will enhance users’ comfort, and convenience, ease of use, and provide better support with respect to the contexts of work of their users.

However, one problem in developing such applications is that there are numerous competing networking protocols (e.g. LON, EIB, BACnet) and no standard application programming interfaces (APIs) for accessing the various devices within a single room or physical workspace. Developers of facility automation applications have to deal with the protocol-specific details of the room devices and networks that application uses. Furthermore, when new devices are being installed, developers may have to modify their application and/or install the appropriate drivers to use the new devices.

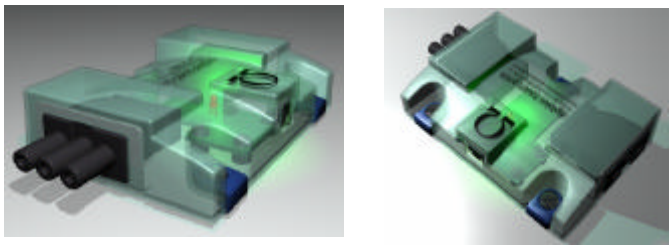


Figure 32 RoomComputer gateway

Since facility automation networks are likely to remain heterogeneous, with multiple incompatible products and protocols used to discover and control different types of devices, application programmers are faced with the prospect of using multiple protocol-specific APIs with no common underlying programming model. Often, there is no standard infrastructure or API to shield applications and their programmers from the low-level details of how to interact with a certain device by using sometimes-proprietary protocols.

In addition, electrical installations in rooms and buildings are getting more and more complex these days. In the past it was often sufficient, to switch on and off devices. Today developers of facility management applications have to deal too much with the protocol-specific details of the room devices and networks that application uses. Furthermore, when new room devices are being installed, developers have to modify their application and/or install the appropriate drivers to use them, i.e. there is no standard software infrastructure to shield applications from these low-level details.

The RoomComputer approach differs from other initiatives in that it does not define a protocol. Instead, it defines a protocol-neutral API and a common programming model for all devices based on XML technologies within a physical workspace. It also defines an extensible service oriented architecture that enables device dependent services to interact with device independent services (e.g. room reservation) over open networks, in order to support the contexts of work of their users in an appropriate manner.

6.11.7 Service Layer

As the RoomComputer integrates several different devices based on different hardware platforms into one unified system, an adaptor platform is needed, which offers adaptors for the different hardware interfaces on one side, and a uniform way of accessing them on the other side. Therefore, the RoomComputer architecture uses gateway components for adaptation on the hardware and on the network layer. This frees the RoomComputer CPU from having different hardware interfaces embedded and adds a great degree of flexibility with respect to expansion. On the software layer, the RoomComputer logically integrates the different hardware interfaces, based on a software framework, which has the ability to flexibly plug-in or remove components during runtime. Such components are called drivers and provide the flexibility to plug-in or plug-out devices during runtime. Such drivers have device dependent interfaces at their bottom layer and device independent ones towards the RoomComputer.

The major design goal of the RoomComputer is to provide an end-to-end service based architecture, which

- on the one hand side provides an open architecture and gateway interfaces in order to connect devices (e.g. lights, blinds, and heating) and the services they are offering to the Internet, and
- on the other hand side brings non-device based services, provided by third-party service providers, into buildings and rooms.

These design goals are being addressed by the Open Services Gateway Initiative (OSGi), which was founded in March 1999. Its mission is to create open specifications, giving access to devices across networks based on a common service oriented API [OSGI]. The central component of this specification is the so-called Open Service Gateway (OSG). The implementation of the RoomComputer software architecture has followed the OSGi specification and provided extensions where necessary, in particular with respect to:

- remote configuration of OSGs,
- co-ordination of concurrent access, based on a lease-concept and priorities,
- interoperation between two or more OSGs on the basis of Jini, and
- security

By concentrating on the device implementation aspects of the RoomComputer, its OSG component supports a variety of device access technologies, such as the European Installation Bus (EIB). This enables users to control devices from different places by using diverse end-user devices (in conjunction with the AUI-ML user interface description language, see chapter 7). The OSGi standard defines the discovery process for devices within a network, how to communicate with them, and how events from devices can be received.

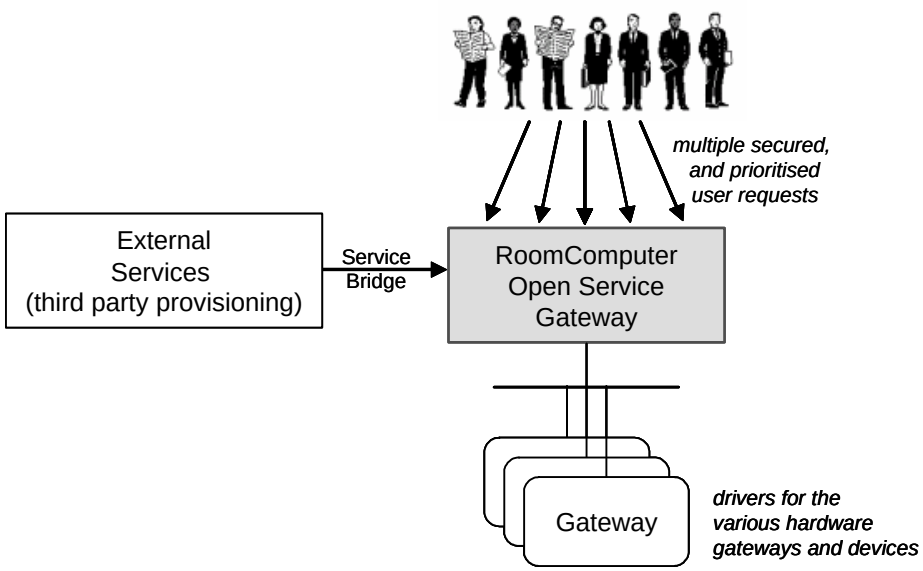


Figure 33 RoomComputer as an Open Service Gateway

OSGi is complementary to service-oriented technologies such as Jini. They address different problems and therefore co-exist in the RoomComputer in the sense that the OSG component provides device-based services whereas device independent services are based on Jini. The usage of Jini (JAVA Intelligent Network Infrastructure) is intended to provide a set of services that makes it easier to develop applications on top of the RoomComputer, enabling them to dynamically discover and interact with devices connected to RoomComputers. With Jini [Jini99] [Edw99], the development of a distributed system (like the Co-operation Platform) and therefore the usage of

interconnected devices and services becomes more simplified. Jini brings to the network the facilities of distributed computing, network-based services, seamless expansion, reliable smart devices, and ease of administration. Connecting a Jini-based service to the network, allows direct access to this service from virtually everywhere in the Intra-/Internet.

With OSGi and Jini the RoomComputer appears as a set of distributed service. Its interfaces are always present, simply and uniformly, regardless of how they are implemented or where they are physically located. Thus, adding new devices and services to the RoomComputer simply means plugging them in. Jini not only simplifies connecting the devices to the RoomComputer, but also the interconnection of the RoomComputer themselves (e.g. to manage a set of rooms or buildings). Jini makes it possible to create a powerful infrastructure of interconnected distributed devices and services, which is very robust, flexible, and scalable.

On top of a relatively small number of software drivers for the hardware interfaces supported by the different RoomComputer gateways, OSGi conformant drivers have been implemented that support the various devices installed in a room or building and offer a uniform programming interfaces for the different device categories (e.g. lighting, heating, and alarm management) to higher application layers.

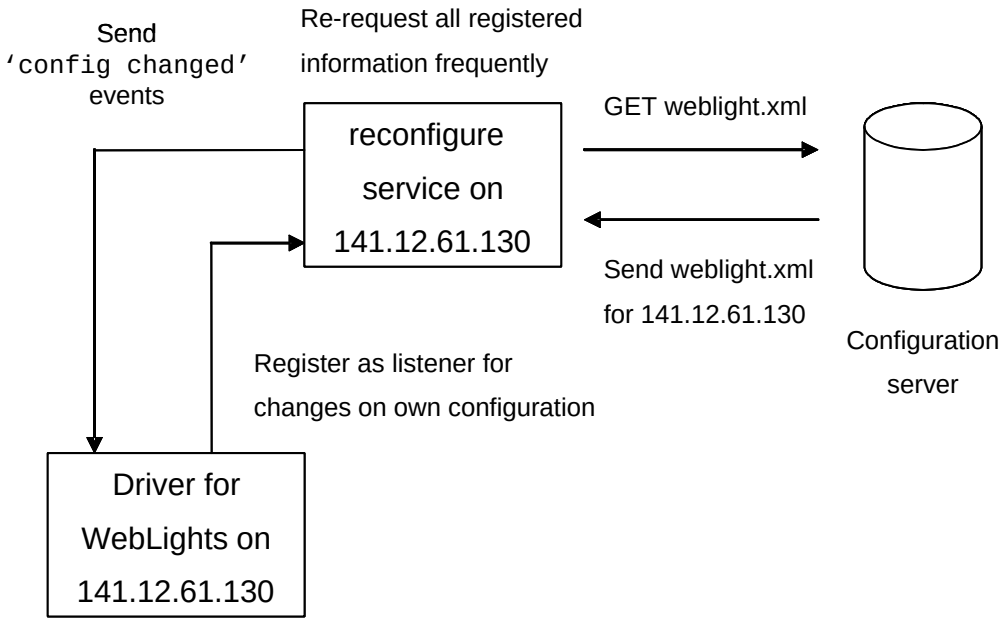


Figure 34 Remote configuration

To be able to integrate new devices into the RoomComputer CPU it is necessary to provide means to download the appropriate driver software for device category of a device plugged into a RoomComputer and for which the RoomComputer does not have the appropriate driver. Therefore, a driver locator is being provided, that can be registered with the OSG. Using the HTTP protocol, it can connect to a driver repository, download

the appropriate driver, and instantiate it in the OSG. In order to do so, devices do have a so-called device descriptor based on XML, identifying the device class together with a URL, which points to a location, where the appropriate driver can be found, if necessary.

As already mentioned, RoomComputers have to be flexible with respect to dynamically changing office situations, mixed used and different tenants, i.e. is has to be possible to dynamically reconfigure a RoomComputer installation. Ideally, such configuration data would come from a CAD-based planning system. With conventional systems based on field busses, such a reconfiguration has to be done more or less manually. Instead, the RoomComputer provide means to flexibly reconfigure it from remote. Therefore, the OSG on the RoomComputer offers a remote configuration service. Device drivers can register with the remote configuration service at installation time. At runtime the configuration is frequently checked by the remote configuration service and the OSG as well as the drivers are notified of changes in the configuration.

With respect to the OSGi specification, a reference implementation for such a service is being provided within the context of this thesis. An HTTP server is being used, which serves as a repository for configuration data. If the configuration service gets notice of changes in the configuration, it connects to the HTTP server and fetches the new configuration. Configurations are being specified using XML. In the future, this will also allow direct integration of planning systems in to a RoomComputer network, as more and more CAD systems are going to support XML for data interchange.

As the OSG has to be able to process multiple requests from different users in parallel, it has to take care of conflicting user requests, which requires appropriate concurrency control mechanisms. For the RoomComputer the OSGi specification has been extended in the sense that it provides the concept of device and service leases. By this, exclusive access to a device or its services can be given for a certain period. Additionally, priorities have been introduced so that services can be released, it other users (e.g. administration vs. regular usage) or specific events (e.g. alarms) can have higher priorities.

One important enhancement to the OSGi specification is the introduction of security features based on standard cryptographic techniques and Java 2 security [PRG+99]. Communication between devices and the OSG is encrypted, devices and OSGs can be authenticated, user authentication and authorisation is based on the Office Identification Card (OIC, see also chapter 8).

6.12Location Awareness

In the context of this thesis, location management splits up into two fields, location management for fixed and mobile devices (e.g. locating a specific device within a building) and location management of persons (e.g. locate team-mates in order to communicate with them).

The information about the devices available within a room is accessible through the corresponding Room Agents. This information can either be more or less static, i.e. somebody is managing a database containing the information about the available devices and their physical location, or it can by dynamic, i.e. the RoomComputer actively senses its environment (in this case by using Radio-frequency based identification techniques and applying so-called Radio-tags to devices) and make this information available to the corresponding Room Agent.

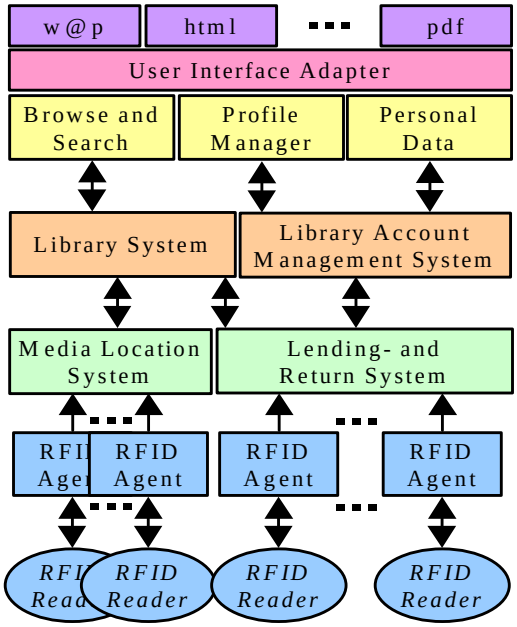


Figure 35 Sm@rtLibrary components

The Sm@rtLibrary consists of different components (see Figure 35), one of that is a traditional library application with a Web-Browser interface for the management of the stock of books. The borrowing and return processes are managed by a Library Account Management System. All physical objects (such as books, folder, or devices) have a Sm@rtLabel, which contains a reference number to support the identification of the corresponding object in addition to a traditional signature on paper-basis. The bookshelves within the Sm@rtLibrary are equipped with RFID-antenna and reader comprehensively so that it is possible for the Media Location System to locate books in the shelves. Books can both be borrowed and returned at specific terminals (Lending and Return System) or by using the Sm@rt-Assistant. They comprise a PDA and an RFID reader, which can read both the Sm@rtLabels and the digital office identity card (for details on the OIC see chapter 8), by which the user is identified and authenticated.

Sm@rtLabels are based on I-Code tags from Philips with a 512 bit read/write memory operating at 13.56 MHz [I-Code]. The maximum operating distance from the antenna to the smart label is 1.2m. The I-Code label itself is 80x50mm2 in size and self-adhesive. This makes attaching the label to a book easy and secure. The labels do not need a battery, as they are powered by the electromagnetic field of the antenna [Fin00].

Figure 36 Sm@rtLabel, RFID reader, and Sm@rtshelf (from left to right)

The Sm@rtLibrary project uses three types of readers for the Sm@rtLabels (Figure 36): one very small unit that is attached to the Sm@rtAssistent, an antenna and reader especially designed and developed to fit into a bookshelf, and an antenna and reader that fits into a door frame capable of identifying persons and objects entering or leaving the corresponding room.

The reader attached to the Sm@rtAssistant (Figure 36) is low in power consumption and small in size. The reading range of this reader is approx. 0.1m and does not provide collision detection. The RFID readers integrated into the shelf do support fast collision detection and have a reading range of approx. 0.5m to cover the size of one shelf compartment. To reduce the cost of a Sm@rtShelf (Figure 36 shows a compartment of the Sm@rtShelf with integrated antenna) a group of several antennas is being attached to a single reader via a multiplexing unit. The antenna and reader integrated into the doorframe have a reading range of approx. 1.0m to cover the width of a door. Collision detection as well as the support of both protocols, Mifare Pro and I-Code, is integrated.

The main objective of for the development of the Sm@rtLibrary was to implement a flexible and scalable software architecture which allows for the easy integration of different types of location-aware information and would be able to represent that information in a manner, which is independent of the particular end-user device (e.g. PC, Handheld, PDA, mobile phone) being used. For this reason, a layered event model has been implemented (Figure 37). Independence from end-user devices has been achieved by use of JAVA and XML technology.

A Sm@rtAssistant - a client, which can run on a PDA - consists of an RFID agent that collects and processes the sensor data and transmits it to the User Agent. The User Agent controls the Presentation Agent, which consists of a standard component, for example a Web-browser. Via the User Agent, the Presentation Agent requests information

about the identified object from the Information Server and sup-plies the necessary information about the particular end-user device being used.

Based on the present context the Information Server determines information, which was requested and returns this to the requesting client in a format suitable for the particular end-user device. The requested information can originate from different databases, the Location Services or from any external information sources.

Every RFID reader has its own agent that collects the sensor data, processes it and transmits the extracted information to the next higher layer. Dependent on the kind of sensor, the sensed data can either be reported continuously to the agents or requested by the agent when needed.

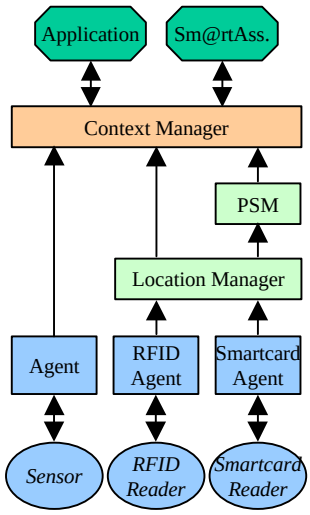


Figure 37 Sm@rtLibrary architecture

The next higher layer is the Context Manager. It collects the data from the different sensors that are transmitted either by the sensor agents or by processing layers in between through an event mechanism. The different data are processed, combined, and provided to the context-dependent applications by a standardized interface. In this way, they are freed of the necessity for compiling and aggregating detailed context information.

With respect to location management a specific security problem is that of allowing the subjects (e.g. the persons) of such a system to retain control over the distribution of the information about their location, e.g. in order to prevent others from location tracking. Therefore, in order to protect privacy of users, the location data of them are not being transmitted directly but stored in a protected manner and only be passed to authorized applications. For example, if a person enters the library, his/her presence will be registered by the RFID-reader built-in in the doorframe, which recognizes the OIC of that person. The corresponding agent passes this information, together with the information that objects carrying Sm@rtLabels are “entering” or “leaving” a room to the Location Manager. By the identification of the corresponding ID’s and depending on whether it is a person or an object, the Location Manager performs an action. Target position information of subjects is reported to the Context Manager.

The Location Manager stores location information about persons in a secure way, i.e. protected by means of cryptography. This encryption means that the information about the location of a person can only be decoded under specific circumstances, e.g. theft, fraud suspicion, according to the 4-eye-principle. However, in order to be able to provide the information about the location of a person to specific applications, this information is passed to a person's Personal Agent via its Personal Security Manager (PSM). The PSM is an application, which runs in a protected environment and is exclusively under control of the respective person. A person can configure their PSM and decide which sensitive data will be passed under which circumstances to which other persons or applications. A request about the current location of a person (e.g. through the Personal Agent of another person) within a physical workspace can be handled only via its Personal Agent and the respective PSM.

This layered architecture for context information has the advantage that the collection, processing, and aggregation can be separated from each other. This allows a high degree of flexibility to be achieved. The addition of further sensor data or processing steps is always possible without problems, not affecting existing parts. Applications, which use context data, do not have to care about where context data originates. They can simply get the necessary context information from the Context Manager. Using event-mechanisms the Context Manager can subscribe to relevant context-data and will not be flooded by data not needed.

Chapter 7

User Interface

Because of their heterogeneous nature, Co-operative Workplaces have to hide the underlying complexity and present a unified and easy-to-use user interface. The user should be able to access the provided functionality from virtually any device (e.g. desktop computers, laptops, PDAs, mobile phones), even if it may not be possible to access all functions from every device. Furthermore, beside graphical user interfaces also other types of user interfaces should be supported in the future, e.g. voice based.

In order to support different types of end-user devices and therefore types of user interfaces (e.g. HTML-based, WML-based, Voice-based), it is necessary to specify the user interface elements in a more abstract way and provide the ability to translate or render such abstract user interfaces into specific representations (e.g. HTML, cHTML, WML). This also supports the collaboration between a user interface designer and a developer, because the developer can focus on the functionality and the designer can focus on the design. In addition, based on such an approach, it is easily possible to generate different user interface designs (e.g. for different organizations, reflecting their corporate design), without changing its functionality.

The implementation of the user interfaces for Co-operative Workplaces is based on two components developed within the framework of this thesis: the Abstract User Interface Mark-up Language (AUI-ML) and the AUI-ML Renderer.

AUI-ML is an abstract user interface description language, realised as a textual mark-up language based on XML. One design goal for this language was that it should be in a human readable format; the other design aspect was that it should be easy to translate abstract user interface descriptions into different user interface representations (e.g. based on HTML for presentation on desktop computers and laptops, or based on WML for usage on mobile phones).

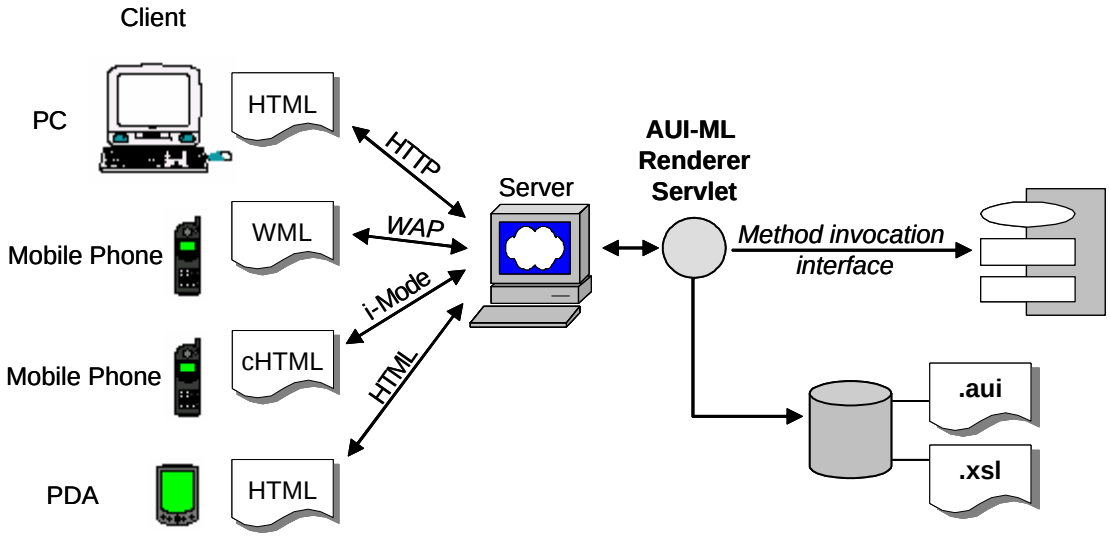


Figure 38 AUI-ML based user interface rendering

The AUI-ML Renderer is a server side component, which renders AUI-ML based user interface descriptions into a target user interface representations, handles the incoming user request, and processes them. It has been implemented as a Java Servlet. It composes user interfaces based on the type of device from which a user accesses the provided functionality and renders (i.e. translates) it to the required output format.

7.1 Abstract User Interface Description Language

In this section, the advantages and disadvantages of existing user interface description languages are being discussed first. Because of the above requirements for the user interface language four candidate approaches have been taken into closer consideration:

- the User Interface Mark-up Language (UIML) [Pha00],
- the user interface toolkit from Java AWT/Swing [SHM+00],
- Java Beans [Eng97], and
- the Formatting Objects (FO) part from the XSL recommendation [XSLT].

Other description languages like HTML, WML or cHTML do not need to be discussed, because they are restricted to specific environments and browsers. Thus translating a user interface description represented in one of these languages to another

one is too difficult or nearly impossible, in particular with respect to WML and HTML/cHTML. The same reason applies for other user interface toolkits like Motif [Fla94] or Microsoft Foundation Classes (MFC) [Bla97]. They are too closely bound to specific operating systems.

7.1.1 User Interface Mark-up Language (UIML)

The User Interface Mark-up Language is a declarative language based on XML [Pha00]. It divides the user interface into three layers:

- the logical layer,
- the presentation layer, and
- the interface layer.

The logical layer contains descriptions about the communication between the user interface and the underlying application. Whereas the presentation layer contains a description of the user interface, independent of the target user interface representation or toolkit. The interface layer describes the interaction between the user interface and the application.

The major advantage of UIML is that that the target user interface representation is not limited to graphical output. Instead, there is even a renderer, which translates from UIML to VoiceXML, a mark-up language based on XML for voice controlled user interfaces [VXML].

As of today, several commercial renderer based on UIML are available. However, it seems that none of these, in particular the freely available Java-based renderer, is complete with respect to the UIML specification. Performance problems are another issue. It seems that performance has not been a major design goal. Additionally, because of its complexity,

- it is very difficult for a developer to learn,
- renderer do only support subsets of the UIML, and
- user interface renderer do consume too many system resources, i.e. memory, storage space and processing time.

Despite that, the underlying ideas of the UIML are addressing the needs of Co-operative Workplaces: an abstract user interface description language, being independent on any specific features of a specific user interface representation or toolkit.

But because of its complexity, the poor support of existing renderer, the high resource consummation, and its slowly ness, UIML is currently not well suited for user interface deployment within the context of Co-operative Workplaces. What is needed is a simpler, but still powerful enough, approach to be independent of specific user interface representations.

7.1.2 AWT/Swing

The AWT/Swing user interface toolkit for Java is the standard for programming user interfaces for Java-based applications [SHM+00]. The benefit of AWT/Swing is that nearly all classes are compiled in Java Bytecode and because of that, they are independent from the underlying operating system and processor. In order to couple the AWT/Swing elements to the user interface toolkit of the underlying target environment, so-called peer classes are needed. A peer class is an abstract widget. Based on such peers Java AWT/Swing-based user interface are being translated to the target environment, but such peers are not compiled into Bytecode. They have to be natively coded for a specific target environment, i.e. processor and operating system.

A possible approach would be to use the AWT/Swing toolkit in order to generate an HTML or WML representation of the user interface. The basic idea of this approach is to replace the peer classes, so that they do not map the peers to the operating system user interface toolkit, but instead express the peers in HTML or WML. Of course, they have to encapsulate the requests and responses by using the HTTP or WAP protocol. From the perspective of an application programmer, this makes no big difference. It is like coding any other user interface in Java.

The most important advantage of this concept would be that the programmer could continue using the usual Java API for coding user interfaces. Another advantage is the powerfulness of this approach. It is possible to express nearly everything as a user interface; even the metaphor of drag and drop can be used. Theoretically, this seems to be possible by using JavaScript at the client side.

Nevertheless, one of the disadvantages is that the AWT/Swing and sometimes Java Virtual Machines are not always available for target end-user devices. This is particularly true for small devices, like e.g. PDAs and mobile phones, because they do consume many resources (memory, processing power). Therefore, it does not seem reasonable to follow this approach in order to get a slim and fast system. What is needed is something not too costly with respect to consummation of system resources.

7.1.3 Java Beans

Another approach to generate different user interfaces for different target environments by using Java would be to employ the Java Beans specification [Eng97]. In order to do this an application has to be developed as a Java Bean.

Java Beans is the Component Object Model for Java developed by SUN Microsystems. Every Java Bean is actually just a Java class. However, either the fields and methods of the class have to follow a special pattern in naming or the class has a co-class, which describes the properties of the corresponding class. The properties of a Java Bean can be queried. The idea is that one can build applications by just connecting their properties, mainly in a graphical way. A good example for this is the Visual Age for Java development environment from IBM and in particular its graphical user interface builder [VAJ].

The basic idea of using Java Beans for the generation of user interface for different target environments is, that a renderer gets a Bean and maps its properties e.g. to HTML fragments. The properties of a Bean do hold enough information about themselves in order to do this, for example, for a read only property only its value needs to be presented, a read/write property of type String, could be presented as a text input field with the value of the property as a default value.

The advantage of this approach is its low overhead. The programmer does not have to do any additional coding to get a user interface for the provided functionality, except the implementation of the optional co-class for the Java Beans properties description.

The disadvantage is that the user interface might look ugly. That is because of the fact that no information is available how the different properties have to be displayed in relation to each other. Another aspect is that every public property of a Java Bean will be displayed, if necessary/intended or not. There is no easy solution to influence this. Possible this could be done by the help of the co-class. Even then, the problem with the arrangement and layout of the user interface still exists. Because the user interface for Co-operative Workplaces should be compelling to their users, this approach is not being followed.

7.1.4 XSL Formatting Objects

The XSL recommendation of the World Wide Web Consortium contains two subparts [XSLT]:

- the transformation part, which is called XSLT (eXtensible Style Language Transformation)
- XSL Formatting Objects (XSL FO).

XSLT is being used by most of the user interface renderer, because it easily allows the transformation from one mark-up language (XML) into another one (e.g. HTML) That is because XSLT treats user interface representations based on mark-up languages as trees and developers only have to define rules, stating how the nodes of these trees are being transformed.

XSL FO is a declarative description language. Its objective is to provide a user interface description language, which can be easily transformed into different representations, such as e.g. HTML, and therefore it could satisfy the needs of user interface development within the context of Co-operative Workplaces. A disadvantage is that the XSL FO recommendation does not provide means to describe the interaction between the user interface and the underlying application or the communication between the two. This thesis follows the same approach; however, it tries to provide a slimmer, but still powerful solution.

7.2 Abstract User Interface Mark-up Language (AUI-ML)

This section describes the Abstract User Interface Mark-up Language (AUI-ML) to be used for developing user interfaces within the context of Co-operative Workplaces. The basic idea of this approach is that the language is not bound to any specific target environment and user interface toolkit. User interfaces represented in AUI-ML can easily and automatically translated into different representations, in particular HTML, WML, cHTML.

This approach has the advantage that services within the Co-operation Platform using AUI-ML can come along with their own user interface, which can easily be transformed into a specific representation to be displayed on a target device (e.g. a desktop computer or mobile phone). From a developers perspective the advantage is that he does not have to take care about the specifics of different target devices. The only he has to do is to describe user interface on an abstract level using AUI-ML. When it comes to support different target devices, which have not been taken into consideration at development time, the only thing to be done is to write a new renderer, which translates the abstract user interface language to the user interface toolkit for the new target device.

The AUI-ML is a simple user interface description language and as such has a restricted set of user interface elements. The provided element-set is powerful enough to express complex user interfaces. It is independent from a specific user interface toolkit, which ensures easy transformation from AUI-ML to other user interface representations like e.g. HTML.

Within AUI-ML, user interface elements are being categorized into two classes. The first class, called widgets, contains all elements necessary to describe the elements and layout of a user interface. It contains elements for displaying texts, images, widgets and describing their layout to each other. The second class of elements is for binding user interface elements to method calls. There are two directions of binding:

- from the device to the user interface, e.g. default values for widgets coming as a result from a method call to the device.
- from the user interface to the device, i.e. submitting user input from widgets back to the device or linking to another user interface from another device or website.

The user interface elements of the AUI-ML mark-up language are called widgets and can be categorised as follows:

- layout elements
  - `container`
- abstract widgets
  - `textinput`
  - `slider`
  - `setselection`
  - `output`

- and binding elements
  - `activator`
  - `call`

In addition, AUI-ML contains user interface elements supporting the linkage of AUI-ML elements with regular URLs (`link`) as well as for conditional rendering (`choose` and `case`). In the following, each AUI-ML element is described in detail. Figure 39 shows all AUI-ML elements. The specification of the AUI-ML grammar is based on XML Version 1.0 by using a simple Document Type Definition (DTD). For the complete specification, refer to the AUI-ML DTD in Appendix C.

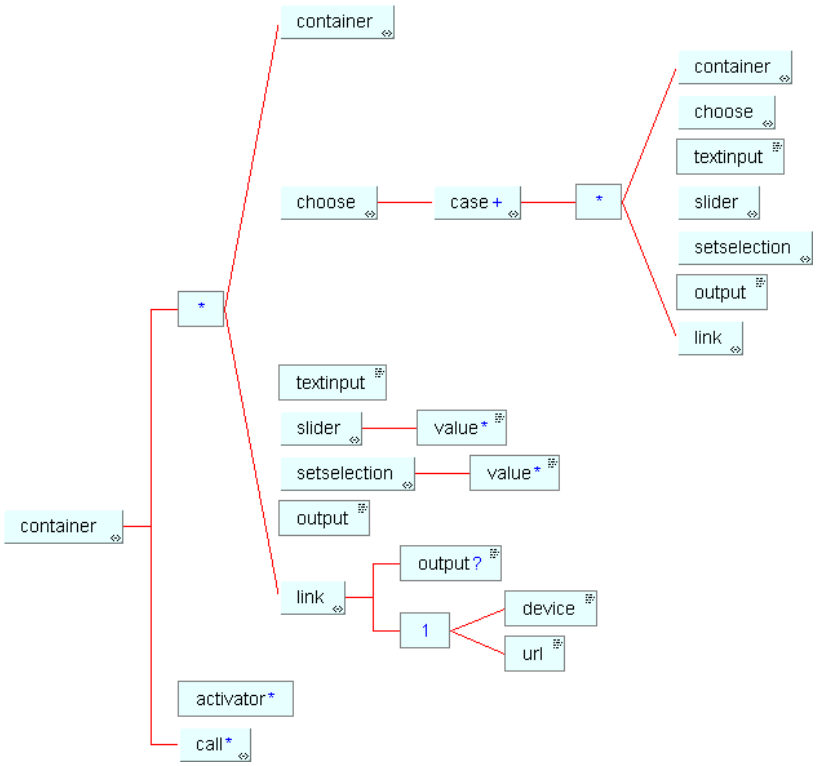
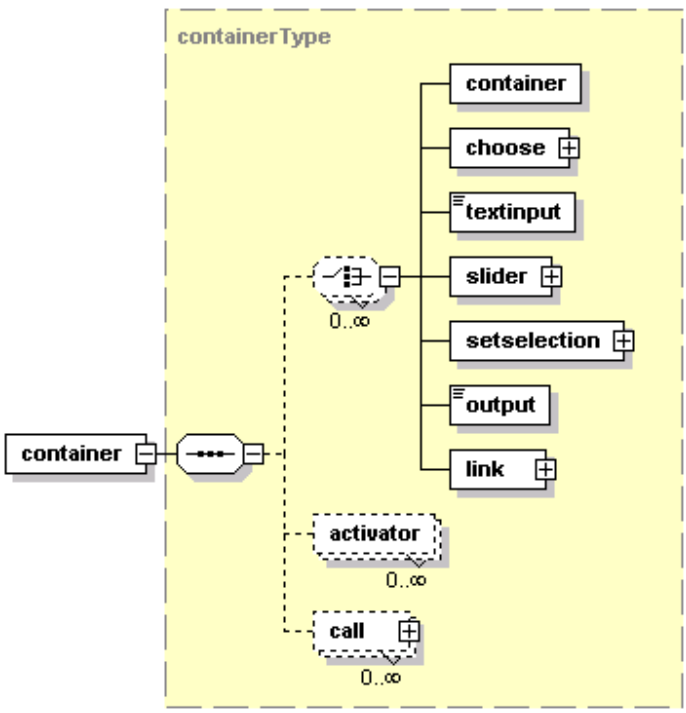


Figure 39 Overview of the AUI-ML DTD elements

7.2.1 Layout Elements

The only layout element within AUI-ML is the `container` element. A `container` is always the root element within an AUI-ML document, and as such, it can contain widgets, `activator` definitions, and `calls`. Container elements can be used to group user widgets, including the `container` element itself. This allows the description of hierarchical structured user interfaces and layouts.

Diagram



Children

Attributes

Name	Type	Usage	Default Value
layout	{horizontal, vertical}	required	
stretch	{yes, no}		yes
size	number		
%BorderStyle			

Table 1 Container element

The `layout` attribute specifies how the widgets within a `container` will be placed, either in horizontal or in vertical order. The `stretch` attribute determines whether a `container` should fill the whole space, in which it resides or not. With the `size` attribute, the elements within a `container` can be ordered in rows or columns. For example, having 7 elements within a `container`, setting its `size` to 4 and `layout` to horizontal will result in the first 4 elements being placed on the first row and the last 3 elements on the second row. If on the other hand `size` is set to 3 and `layout` to vertical, the first 3 elements will be placed in the first column, 3 in the second column and the last element in the last columns (Figure 40)

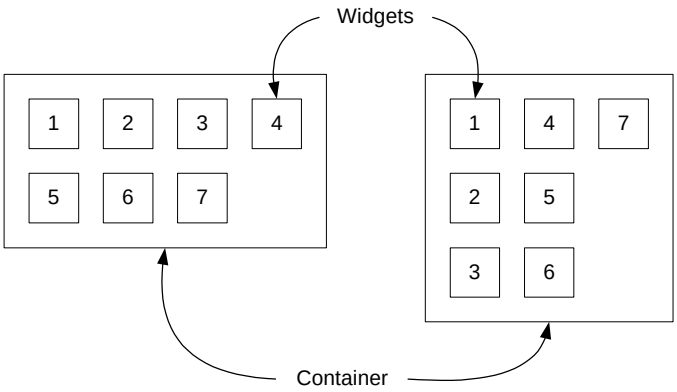


Figure 40 Sample layouts

7.2.2 Abstract Widgets

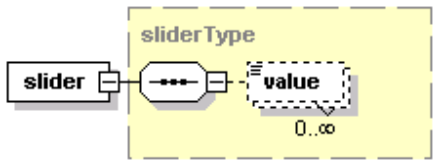
The `textinput` widget represents a user interface element where a user can enter text. With the `pattern` attribute, it is possible to restrict the textual input to specific values using a regular expression, i.e. only such textual input will be accepted, which follows the given pattern. With the `length` attribute, the maximum input length of the text can be restricted to a specific number of characters. The `call` attribute allows to retrieve a default text by referencing a `call` element (see below), which can call a Java method and return the result as a string. The result will be checked against the given constraints (pattern, length) and if valid, being displayed otherwise not. With the value attribute, a default input value can be specified. Again, if it follows the given constraints, it will be displayed, otherwise not.

Diagram	textinput			
Attributes	Name	Type	Usage	Default Value
	pattern	pattern		
	length	number		
	call	#IDREF		
	value	string		
	%NameId			
	%BorderStyle			

Table 2 Textinput element

The `slider` widget is a user interface element, which allows the selection of data out of a given range (specified by the `value` element, see below) in a continuous fashion.

Diagram

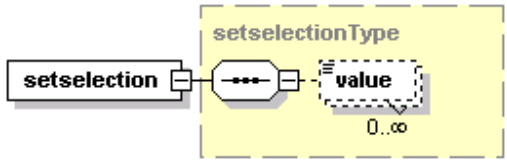


Attributes	Name	Type	Usage	Default Value
	%NameID			
	%BorderStyle			

Table 3 Slider element

The `setselection` element describes an abstract widget for the selection of data, out of a given set (specified be the `value` element, see below) in a discrete fashion. The user has the possibility to select either a single value or multiple ones, which is determined by the attribute `multiple`.

Diagram

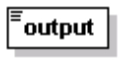


Attributes	Name	Type	Usage	Default Value
	multiple	{yes, no}		no
	%NameId			
	%BorderStyle			

Table 4 Setselection element

With the `output` element, either text or an image can be presented on a user interface. If the attribute `text_or_image` represents a valid URL and the format of the image is being supported by the destination toolkit and environment (e.g. a Web-browser), then the corresponding image will be presented on the user interface, otherwise the URL will be treated as text.

Diagram



Attributes	Name	Type	Usage	Default Value
	text_or_image	string		
	span	number		1
	%NameId			
	%BorderStyle			
	%TextStyle			

Table 5 Output element

7.2.3 Common Entities and Elements

The `BorderStyle` entity defines attributes, which are common to all layout and abstract elements and can influence their appearance on the user interface. With the `border` attribute, the designer of the user interface can specify whether an element either has

- no border (`none`),
- a line border (`line`),
- a raised 3D border (`bevelup`), or
- a lowered 3D border (`beveldown`).

For 3D borders, the `highcolor` and `lowcolor` attributes determine the colours being used by the renderer for the higher (default is white) and the lower (default is black) borderlines. Colours are always being specified using the RGB colour model. They are defined as a hexadecimal strings with a preceding '#', the first two characters do represent the red portion, the second two the green and the last two the blue portion. For the background of an element either a colour can be specified or a background image. The `span` property specifies the number of cells an element will span within its parent container.

Diagram **BorderStyle**

Attributes	Name	Type	Usage	Default Value
	border	{none | line | bevelup | beveldown}		none
	highcolor	{#rrggbb}		#FFFFFF
	lowcolcor	{#rrggbb}		#000000
	background	{#rrggbb | image-url}		#C0C0C0
	span	number		1

Table 6 Borderstyle

The following image shows an example, where a container contains 3 elements. The layout attribute of the container has been set to horizontal, and size has been set to the value of 2. The container has a 3-D raised border, i.e. border is set to `bevelup`, `hicolor` is set to white and `lowcolor` to black and `background` is set to white. For element 3 the `span` value has been set to the value of 2 (Figure 41).

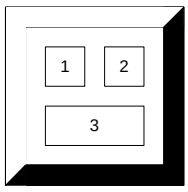


Figure 41 Container example

The `TextStyle` entity defines attributes, which specify the rendering of text on a user interface. `Fontface`, `fontsize` and `textcolor` can be specified, whereas the attribute `textalign` specifies the alignment of the text, i.e. either left justified, right justified, or centred.

Attributes	Name	Type	Usage	Default Value
	textalign	{left, right, center}		center
	textcolor	{#rrggbb}		#000000
	fontface	string		
	fontsize	number		

Table 7 Textstyle

With the `NameId` entity, widgets can be given a unique name, which can be used to bind the widget to a Java method.

Attributes	Name	Type	Usage	Default Value
	name	ID		

Table 8 NameId

The `value` element has three attributes and describes a value being used by a parent element like e.g. a `setselection` or `slider`. This element can be used to specify the `start` (default is set to 0), `end` (default is set to 100) and `default` (default is set to (`end – start`) / 2) values for a `slider`. It can also be used to provide a list of values (numbers or strings), including a default value, for a selection set. The elements of this set can be either numbers or strings.

Attributes	Name	Type	Usage	Default Value
	type	{start, end, default}		
	%JavaTypes		required	
	display	string		

Table 9 Value element

With the `%Javatype` attribute the type of the value is being declared using the primitive data types from Java. The `display` attribute contains the value going to be displayed on a user interface, i.e. if the `value` is of type `bool` and has the value `true`, one might want to display the string `"yes"` instead on the user interface, which can be done using the `display` attribute.

Diagram JavaTypes

Attributes	Name	Type	Usage	Default Value
	javatypes	{void, bool, string, char, byte, short, int, long, float, double}		

Table 10 Javatypes

The `JavaTypes` entity contains all supported primitive data types of Java and is being used to declare the type of the return value from a call to a Java method (see below).

7.2.4 Binding User Interface Elements to Method Calls

Binding is being used to bind user interface elements to method calls and vice versa. Currently AUI-ML only supports Java method calls. As already said, there are two directions, one from the user interface to a method (e.g. when an element on the user interface has been clicked, an action needs to be performed) and the other one from a method to the user interface (e.g. when a value has been calculated by a method, it needs to be displayed on the user interface).

The `activator` element can be used to call a Java method, where the result of the call will be ignored, e.g. the activator can be used to generate clickable user interface elements and clicking on them triggers an action. Such user interface elements are sometimes also referred to as so-called hot spots. By its attribute, `output` an output element can be referenced by using the value of the `NameId` attribute of the corresponding output element. The Java method to be called is specified using the `call` attribute. Activator elements can be marked with a border (`border`) and marked in a specific color (`markcolor`, or `highcolor` and `lowcolor` in case of 3D borders).

Diagram activator

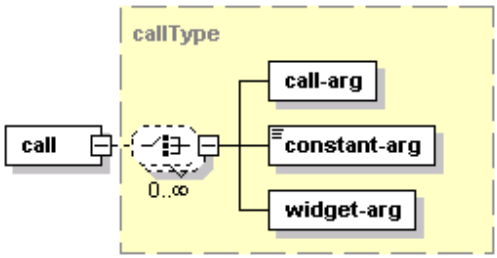
Attributes	Name	Type	Usage	Default Value
	output	#IDREF	required	
	call	#IDREF	required	
	border	{none, line, bevelup, beveldown}		none
	markcolor	{#rrggbb}		#0000FF
	highcolor	{#rrggbb}		#FFFFFF
	lowcolcor	{#rrggbb}		#000000

Table 11 Activator element

With the `call` element, Java methods can be called and their results being displayed on the user interface. Its attribute `method` refers to the ID of the Java method being called, whereas the `%JavaTypes` denotes the primitive Java data type of the return

value. With `constant-arg`, `widget-arg`, and `call-arg`, parameters can be passed to the corresponding Java method.

Diagram



Attributes	Name	Type	Usage	Default Value
	method	#IDREF	required	
	%JavaTypes		required	

Table 12 Call element

In order to pass a constant value to a Java method `constant-arg` can be used.

Diagram

Attributes	Name	Type	Usage	Default Value
	%JavaTypes		required	

Table 13 Constant arguments

The `widget-arg` references that the argument for a method call should come from a widget, e.g. text being entered by a user in a `textinput` element or a value selected by a user using a `slider`.

Diagram

Attributes	Name	Type	Usage	Default Value
	widget	#IDREF	required	
	%JavaTypes		required	

Table 14 Widget arguments

Another method to pass arguments to a method is to call another method and send its return value as input to the method being called. Therefore, one can use the `call-arg` element.

Diagram

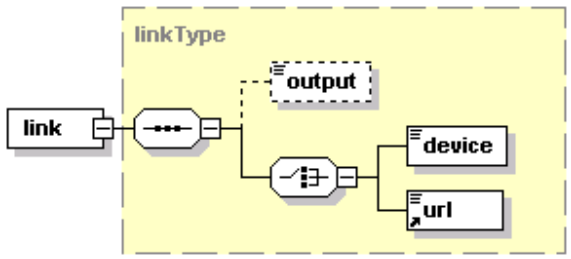
Attributes	Name	Type	Usage	Default Value
	call	#IDREF	required	
	%JavaTypes		required	

Table 15 Call arguments

7.2.5 Links

Within AUI-ML, there is the concept of Hypertext references or links, which are represented using URLs. Either links can be displayed to the user using the `output` element, or they can be replaced by what they are referencing using the `url` element. For example, a link to an HTML document within an AUI-ML based user interface can be replaced by the HTML document itself rather than just pointing to it.

Diagram



Attributes	Name	Type	Usage	Default Value
	type	{device, url}		device

Table 16 Link element

Links cannot only reference HTML documents; they might also reference other AUI-ML user interface specifications. This allows user interface implementers to design parts or to make their entire UI reusable as a component in another user interface, i.e. it allows the development of templates for user interfaces, which can be reused.

Diagram

Attributes	Name	Type	Usage	Default Value
	type	{device, url}		device

Table 17 URL element

With respect to the RoomComputer, another powerful concept has been introduced in AUI-ML. As the RoomComputer is aiming at supporting ‘Plug & Play’ not all the devices (i.e. sensors and actors) to be connected to a RoomComputer are known at design time. Therefore, it must be possible to add devices at later stages. This can create problems with respect to the user interfaces, as it requires a, more or less, manual modification of the existing user interface, in order to integrate the new device.

Diagram

Attributes	Name	Type	Usage	Default Value
	class	NMTOKEN		

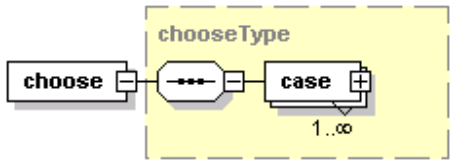
Table 18 Device element

The key idea that is carried with the RoomComputer is that each device comes with an abstract representation of its user interface, using the AUI-ML mark-up language. The 'Plug & Play' integration of new user interfaces can be done using the `link` element together with the `device` element. When a new device is plugged into a RoomComputer, it provides its abstract user interface to the RoomComputer, which from now on can transform it to an appropriate representation (e.g. an HTML page displayed on a PC or PDA). If a RoomComputer device does not come with an AUI-ML-based user interface, the RoomComputer can select a generic one, which can be selected using the `class` of the device, e.g. a lamp, a blind etc.

7.2.6 Conditional Rendering

AUI-ML has a concept for conditional rendering. The `choose` element allows the conditional rendering of AUI-ML elements based on the result of a Java method referenced by the `call` attribute.

Diagram

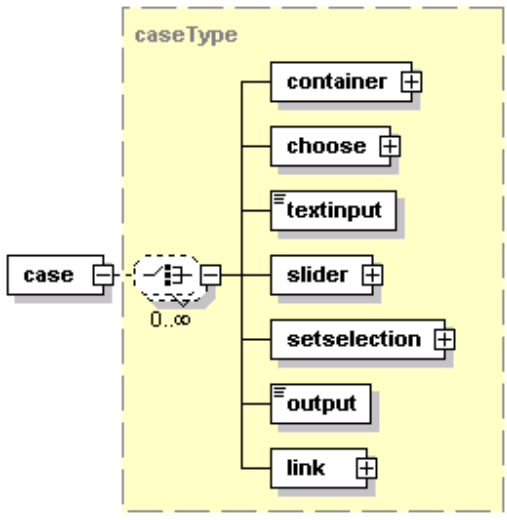


Attributes	Name	Type	Usage	Default Value
	call	#IDREF	required	

Table 19 Chose element

The choose element comes along with the case element is used to specify the case conditions, i.e. if the return value of the method call matches the attribute value, the corresponding AUI-ML element will be rendered.

Diagram



Attributes	Name	Type	Usage	Default Value
	value	string		

Table 20 Case element

7.2.7 AUI-ML Renderer

The rendering of AUI-ML user interfaces, i.e. the transformation of an AUI-ML document containing the abstract description of a user interface into e.g. an HTML representation, is being done by a Servlet, residing on a Web-server. Basically, the Servlet takes an AUI-ML document and renders it using a corresponding XSL stylesheet. Such a stylesheet describes the transformation of AUI-ML user interface elements into user interface elements of the target environment. Currently it supports the rendering of AUI-ML into HTML, cHTML and WML. However, beside the transformation, the Servlet also handles the call to Java methods, the handling of embedded links, and the conditional rendering of AUI-ML elements.

If a client sends an HTTP request to the Web-Server requesting a URL with the suffix `".aui"`, the Web-server passes this request to the AUI-ML Servlet. As a first step, the Servlet tries to find out the requested output format. This can be done evaluating the `User-Agent` property defined by the HTTP protocol, e.g. for the Mozilla HTML-Browser the value is:

```
User-Agent: Mozilla/5.0 (Windows; U; WinNT4.0; en-US; rv:0.9) Gecko/20010505
```

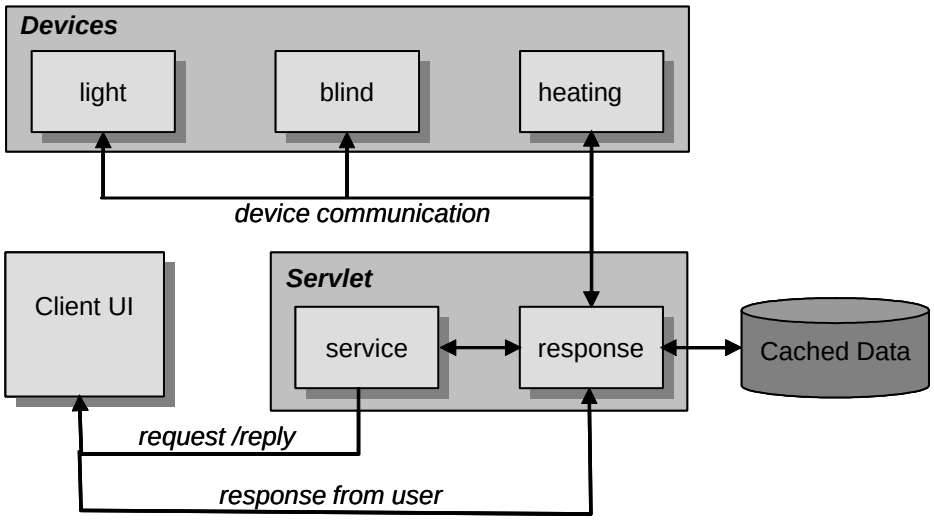


Figure 42 AUI-ML Servlet

In the second step, the Servlet does a complete parsing of the AUI-ML document. After that, it handles all method calls, their responses, and resolves links before it renders the AUI-ML document into the requested output format, using the corresponding XSL file. Using XSL technology for the transformation has the advantage that the input (i.e. the AUI-ML document) and the output (e.g. an HTML document) are handled as trees. The XSL document, which corresponds to an AUI-ML document defines rules, how input nodes are transformed into output nodes. The Servlets uses the XP XML parser [XP] and

XT [XT] as the corresponding XSL engine. Compared to other solutions, XP and XT are very fast and have a small memory footprint.

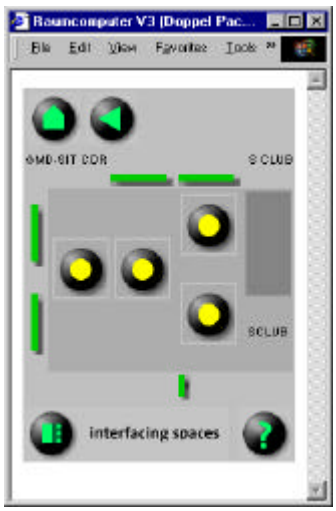


Figure 43 AUI-ML-based user interface

7.2.8 AUI-ML Example

After all elements of AUI-ML have been defined, Figure 43 shows an example user interface being specified using AUI-ML and being rendered into an HTML representation. This user interface has been developed for the SCLUB. Via this user interface, users can switch on and of the lights or close and open the blinds via the RoomComputer, using a standard Web-browser. For the corresponding AUI-ML document, refer to Appendix D.

7.3 The RoomComputer

The RoomComputer provides simple and easy to use graphical user interfaces based on the Abstract User Interface Mark-up language (AUI-ML). Figure 44 shows an example where an AUI-ML based user interface description has been rendered into an HTML representation for usage on a PDA.

Therefore, through e.g. a standard Web-browser any RoomComputer can be accessed remotely in the same way as locally. Each of the user interface elements on it leads to one of the software modules that make up the system. When such a user interface element is selected (e.g. by clicking on an icon), either an action is immediately initiated, or a control panel opens to allow the selection of options.

Figure 44 RoomComputer user interface

Users can control room devices like lights, blinds, heating, or can call for services like the cleaning of a meeting room. A Servlet, called user interface adapter, converts device-independent representations of the user interfaces into device-specific ones (e.g. PC, PDA, mobile phone). By using the AUI-ML together with the XML/XSL technologies, it is also possible to adapt the user interfaces to different users and user groups or to provide customised interface designs (i.e. representing a specific corporate design).

7.4 Virtual Project Office

As already mentioned, the Virtual Project Office is a representation of a virtual work environment, meant to give its users the impression as if they were in a room, where they can find all her team-mates as well as all necessary resources, results, files, contacts etc. Such resources and services are being made available to the team members in the context of their work, e.g. performing peer-to-peer communication, distributed group sessions, jointly view and share project documents and others.

It is commonly recognized, that computer-based group support systems can improve satisfaction and efficiency of idea generation and decision-making meetings [NBM95]. It is also the fact that the physical environment plays a mediating role in this relationship and that the design of the physical environment is important to group work. What was hardly examined up to now is the relationship between physical and virtual work environments. Because of the fact that the physical environment mediates satisfaction, it seems reasonable to apply the same paradigms also to computer supported group work.

Like others, it is believed also that the adoption of physical environments will allow people to work together more naturally [GR98]. More precisely, a room metaphor can ease people’s transitions across the gaps of today's groupware systems that hinder or block natural social interaction or that do not let people move easily between different styles of work. Therefore, a Virtual Project Office tries to give it users a view of an imaginative work environment, which and a team the impression of moving in a room where they will find everybody currently involved in the project.

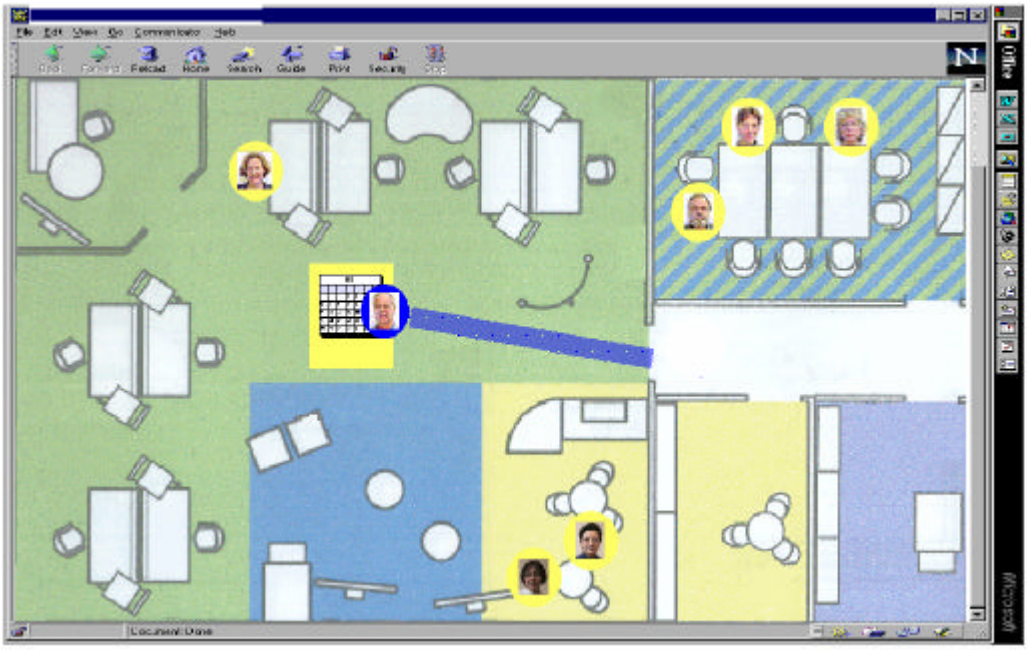


Figure 45 VPO user interface - 2D version

Currently, the user interfaces of the VPO are not based on AUI-ML, because at the time the VPO was developed, AUI-ML was not available. Currently the VPO is offering two types of user interfaces, both running in a Web-browser. The first one is a 2D user interface looking like a floor plan and being built on HTML technology (see Figure 45). The second one is a 3D user interface built on VRML and HTML technology (see Figure 46). In the future, AUI-ML based user interfaces for the VPO will be available, allowing to access a VPO for example from a WAP enabled mobile phone most likely the same way as from a Web-browser on a PC.

At the user interface level, VPOs look like a real team office, which preserves some of the essential advantages of a local shared office, such as transparency, ease of interaction, availability of project data etc. Geographically distributed members of a team are projected into the Virtual Project Office and represented as avatars, as if they were all physically present in a single office. By including a project's input and output, its resources, tools and services, the Virtual Project Office captures the context of the project.

A Virtual Project Office shows different work zones, supporting different types of work and work context. The main function of such work zones is to provide an integrated collection of tools and resources needed to support the corresponding work context. For example, a VPO can offer zones for individual work and zones for collaborative work such as conference sessions or informal information exchange. Anyone "in office" will be shown in a zone depending on what he currently does. Whenever a team member enters a VPO, he immediately becomes aware of who is in and what is going on, like in a real project office.

Figure 46 VPO user interface – 3D version

With the concept of work zones, it is easy for a user to switch back and forth between different work contexts. However, besides offering a collection of integrated services, work zones do support group awareness. At a glance, team members can see who else is available an in what work contexts are their teammates. Additionally, the concept of work zones can also being used to define access control conditions

Team members can move about in the room by moving their avatars and observe others moving. They can thus approach a colleague, enter the conference area or go to the project files; simply by going there, and without having to select or invoke the proper phone, conference or work space tool, they can communicate with the colleague, participate in the conference or inspect and update files, respectively. Another example would be a user moving his avatar to the calendar, by which he will be allowed to inspect the team calendar and to make entries, without having to explicitly open and configure a calendar tool. They can also move their avatar into a conference zone, which will result in automatically becoming a member of that conference session. Such conference sessions are being created and managed using the MCRS system. The necessary audio, video, and

data communication tools are being implicitly started when the avatar enters and stopped when it leaves.

In the current version, the VPO is offering the following work zones:

- personal desk zone,
- individual work zone,
- teamwork zone
- conferencing zone,
- chat zone, and
- library zone.

Being in the personal desk zone indicates that the corresponding user is personally reachable for other team members and open for requests. Thus, moving to somebody working at his personal desk invokes communication software, like a chat tool, a voice connection using either a telephone or even voice over IP (VoIP).

In contrast to the personal desk zone where direct requests and communication are possible, the individual work zone offers a protected virtual space. Working in this zone indicates that one is working on the project, but does not want to be disturbed by other teammembers for a certain amount of time.

A conferencing zone generally offers access to multimedia conferencing services in order to commence a meeting with one or more participants. Thus, the services involved have to take care about different kinds of audio and video hardware and software available on client side and be aware of the related protocols. Therefore, the Personal Agents together with the Team Agent negotiate appropriate compatible software tools and send messages to their clients in order to invoke such tools. In addition, the Team Agents reserves the necessary resources by interaction with the MCRS system. If necessary, this might also include the reservation of a gateway bridging between different standards and protocols, such as the ITU-T H.320/H.323 protocol series.

The teamwork zone supports the drawing of sketches on a whiteboard, joint viewing of documents, and sharing of applications together with audio and video conferencing facilities.

The chat zone supports spontaneous and informal chat sessions. The library provides access to the project's document repository, which stores all project related documents (e.g. contracts, agreements, agenda, milestones, project plan, meeting minutes). It can also contain online tutorials and links to relevant material.

Chapter 8

Security Aspects

Besides the simple intuitive handling of the user interfaces, security is decisive for the success of the Co-operation Platform. In virtual teams, because of the Internet-based approach, the entire spectrum of security-relevant considerations is brought to bear. Cooperation in teams whose members usually belong to different organizations requires unilateral, bilateral, and multilateral security technologies in order to guarantee the trustworthy authentication of their members, their authorization, secure communication among one another as well as the integrity and confidentiality of all project documents. The different organisation-dependent security policies, which must be matched and/or considered, constitutes another problem.

As the objective of the Co-operation Platform is to provide a platform, which allows the integration of different resources, cooperation and communication services in a heterogeneous environment into an Internet-based system, this constitutes a special challenge for a security architecture with regard to its scalability, flexibility and modularity, since both on the client and on the server side most different technologies have to be supported.

Security objectives, which have been addressed by this thesis in connection with the services offered by the Co-operation Platform, are:

- *authenticity,* i.e. confirmation of the identity of an entity (e.g. verification of the identity of a person)
- *confidentiality,* i.e. keeping information secret from all but those who are authorised to get it to know
- *privacy protection,* i.e. to protect the personal sphere and therefore the ability of a person to give others access to its personal data in a controlled manner
- *integrity,* i.e. ensuring information has not been altered by unauthorized or unknown means.

- *non-repudiation and accountability,* i.e. preventing the denial of previous commitments or actions (e.g. actions are unambiguously accountable to the executing persons, who can not deny this afterwards)

In order to meet the above objectives, the Co-operation Platform provides the following security mechanisms:

- identification, authentication, and certification to confirm the identities of the communicating partners and in particular the identity of platform users
- authorisation and access control to prevent unauthorised access to e.g. platform services, documents, or devices
- data confidentiality to protect data against eavesdropping, i.e. to provide end-to-end security
- data integrity to ensure the correctness and consistency of data against loss and spoofing in the sense that data can neither be modified without authorisation nor without being noticed
- non-repudiation and auditing to provide proof of origin and delivery of information.

In principle, relevant security functions to implement these mechanisms are as follows:

- secret key encryption (most commonly used methods: DES and IDEA, both operating on blocks of 64 bits length),
- public key encryption (e.g. RSA),
- hybrid encryption (combination of the above two methods),
- key exchange mechanisms (e.g. Diffie-Hellmann),
- hash functions (e.g. MD5, SHA-1),
- message authentication (e.g. MD5-MAC),
- digital signatures (e.g. DSA, ElGamal, RSA),
- passwords and challenge-response techniques,
- key management, and
- logging

A general survey of security objectives, requirements, mechanisms, and cryptographic methods can be found in [Sch96] [Eck01]. In the following, the measures taken within this thesis, with respect to the above security objectives and mechanisms, are being explained. The security architecture of the Co-operation Platform consists of the following main components:

- a public key infrastructure (PKI);
- an Office Identity Card (OIC) for the secure identification and authentication of people and as a carrier for the security tokens, authorisations, and personal profiles;

- secure end-to-end communication services (including mutual authentication, integrity and confidentiality);
- inherent multi-lateral security services provide different security functions, such as authentication, digital signatures together with an access control system, that check every access to resources and services for authorisation and include integrated non-repudiation and auditing and functionality;
- a security policy manager, which provides functions for the management of security policies;
- Personal Security and Privacy Managers supporting the protection of privacy related data, such as information about a person's location.

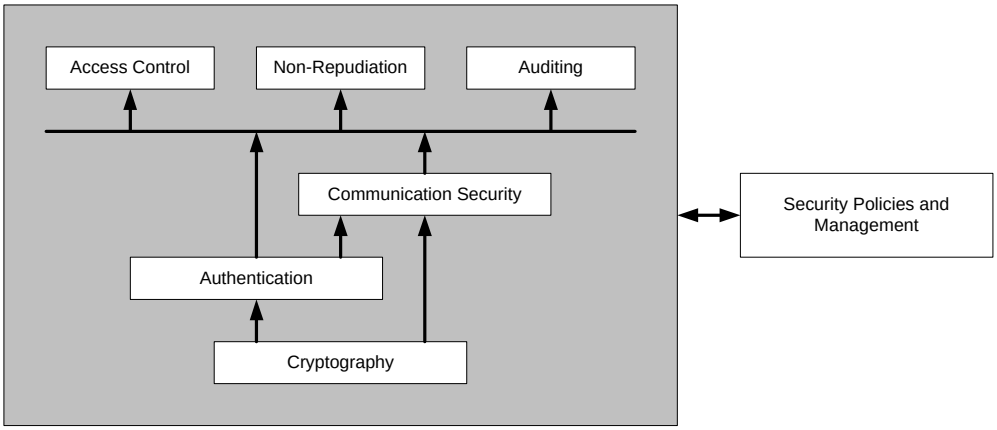


Figure 47 Security functionality and dependencies

The security services provided by the Co-operation Platform are being implemented as inherent services [FP97] [Gri97] running in the background so that users are not being confronted directly and constantly with specific security measures (e.g. identification and authentication). The following sections will give a more detailed view on the above components; special focus is being put on security measures within the Co-operation Platform and for the RoomComputer.

8.1 Public Key Infrastructure

Beyond a user merely being present in a Co-operative Workspace, a challenge is user identification. Many of the tools being integrated into the Co-operation Platform (such as e.g. MS NetMeeting) allow the user to control the user identification information without system validation. Users are free to use and change whatever name, title, and address they please, creating the risk of spoofing, or the deceitful assumption of another person's identity. Once this information is registered with a directory server (e.g. the MS Internet Locator Server, ILS), the user may compromise security by choosing whatever identity suits him.

To overcome these issues a robust Public Key Infrastructure (PKI) has to be established with the support of all participating organisations. Therefore, asymmetric cryptographic procedures with private and public keys and an associated public key infrastructure are a prerequisite for the protection of communication links in order to guarantee confidentiality, integrity, authentication, and non-repudiation.

Although it is not a panacea, a public key infrastructure (PKI) is the foundation upon which user identification is built. Each user's set of secret keys becomes their so-called virtual identification card so that they can more easily enforce need-to-know. A collaboration environment that supports PKI makes it easier to validate the security credentials of people. Administrators can exercise control to the platform services by usage of a role-based access control model.

A PKI is a combination of hardware and software products, guidelines, and procedures. It mainly consists of the following components:

- Crypto-token or Personal Security Environment (PSE): the crypto-token/PSE is used for the secure storage of private keys. A PSE can be implemented either by an encrypted file stored in the local file system (so-called software PSE) or by a smartcard (so-called hardware PSE). The developed Office Identity Card (OIC) is one possibility to store such sensitive information (see section 8.2).
- Certification authority (CA): in the CA, the certificates are being created by linking user identities to their respective public key, their period of validity and rights are linked and managed. Within this thesis, the generation of the asymmetric key pair is also done in the CA for performance and manageability reasons.
- Registration authority (RA): the RA constitutes the interface to the user. It guarantees the identity of the user with respect to the CA. If no CA is locally available, the RA is used as a mediator between the user and the CA.
- Directory services for certificates (DIR): in a directory service, the public keys are made freely accessible to all users so that they can verify, e.g. digital signatures made by other users with respect to their correctness and validity.
- Certificate revocation lists (CRL): certificate revocation lists are data structures within the directory service, which contain information such as the date together with the reason for revoked certificates whose validity period has not expired so far.
- Time stamp service (TS): The time stamp service certifies the presence of digital data at a specific date and time. This service is the basis for the non-repudiation service. It guarantees that a specific action was done at a specific time.
- Security guidelines (SG): the SG defines a uniform security standard and guarantee for the utilisation and the management of the PKI.

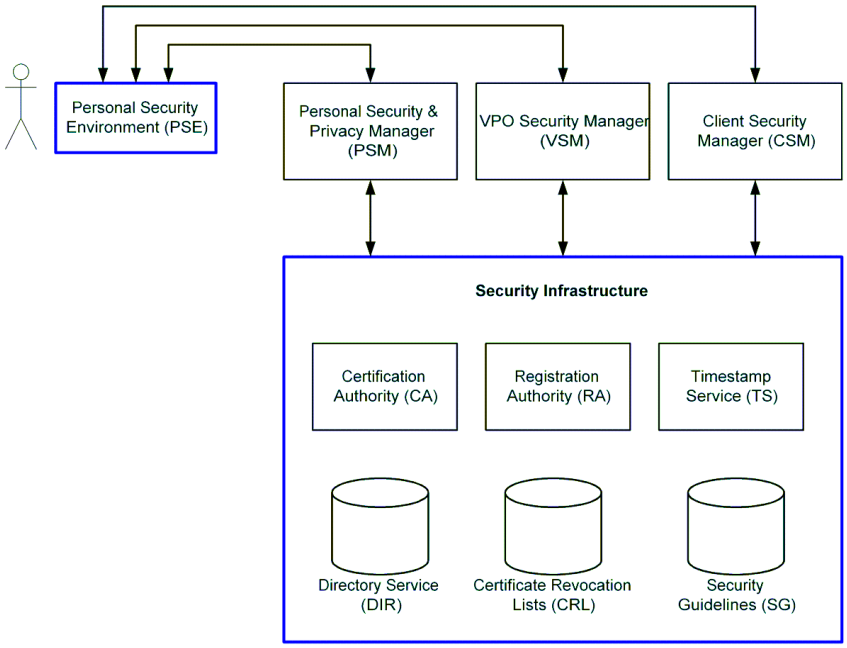


Figure 48 Security infrastructure of the Co-operation Platform

Since it is to be expected that in the future every enterprise will have its own PKI, this component is to be considered as an external platform component. Although a thorough discussion of PKI is beyond the scope of this thesis, everyone must agree on a root Certificate Authority (CA, in Germany this would be e.g. RegTP) that provides standards-compliant, open solution utilising well-known security standards such as X.509v3, PKCS #12, and SSL. For a supra-organizational use, interoperability between the different products available on the market is necessary. Currently, this is still a problem to some extent.

The security infrastructure for the Co-operation Platform has been build upon SecuDE [ABF+96] [Secude]. SecuDE is a toolkit, which offers a library of various security functions. It provides basic cryptographic functions (like RSA or DES), digital signatures, support for public key infrastructures (PKI) like X.509 key certification, operation of certification authorities, secure access to public X.500 directories for the storage and retrieval of certificates, cross-certification and management of certificate revocation lists. With SecuDE certificates according to the international standard X 509 (in the versions X.509v1 or X.509v3) can be generated and revoked, certificate revocation lists can be managed, together with the maintenance of a logbook.

8.2 Office Identity Card

A secure storage of the crypto-token, sometimes also referred to as PSE (personal security environment), generated by a CA is possible only in a smartcard. Neither does the storage on a computer correspond to today's security requirements and standards, nor is it practical in the era of mobility and modern work forms with different places of work.

As a design decision such a smartcard should not only be used within the context of the Co-operation Platform for the storage of the certificates, for encryption, and digital signatures, but also as a digital office identity card allowing supplementary applications for in-house functions such as admission control, single-sign-on, electronic cash or also as a library identity card. For this reason, the concept of the Office Identity Card (OIC) seemed suitable.

The OIC developed within this thesis represents a prototype of an identity (ID) card for members of German ministries and public service in order to replace traditional paper/plastic-based ID cards [MRH00]. As such, it is based on the specification of identity cards for the public sector and government departments [TTT00]. As a technological basis, dual-interface smartcards (offering two interfaces, one with contacts, the other one contactless) with an integrated crypto-controller have been selected. They provide means for the secure identification and authentication of people, allow the secure storage of important data (e.g. private signature keys) and the secure execution of crypto-graphical functions (e.g. data encryption).

The OIC has the same functions as a paper-based identity card. For example, it contains information about its cardholder, such as his name, date of birth, citizenship, department information, a unique card number, and period of validity. Furthermore, it carries digitised pictures of the cardholder and of his signature. Such information can be read visually as well as electronically.

In addition to the pure identity card functions and the authorization information, the developed OIC provides signature functionality according to the DIN specification for digital signatures [DIN174]. Therefore, assuming an appropriately security infrastructure, it can be used to sign documents electronically in accordance to the German/European signature law. For example, this allows a cardholder to access and retrieve electronic documents from a document repository and sign them by using his digital signature.

An important feature of the OIC is that it implements security-related functions like authentication and encryption. Additionally, authorisation data is being stored directly on the card. Such authorisation data consist of the access rights and roles of the cardholder together with the implicitly assigned authorisations to such roles. These authorisations can be granted either permanently, temporarily, or for n-times usage.

The OIC can serve as a basis for different authentication procedures such as TLS, Kerberos with the extension PKINIT and CV-based terminal authentication [Eck01] [RE99]. It can support key management in the sense that signature and authentication keys are securely stored in the card.

Supplementary applications are possible; however, they were not part of the original specification in [TTT00]. For the use of the OIC in the context of the Co-operation Platform, it was equipped with the following supplementary applications:

- role- and task based access control system for data and services,
- single sign-on (access control),
- storage of user profiles, and
- privacy-protected time recording.

Further applications developed for the OIC are admission control to buildings and rooms, library function (e.g. book lending, storage of user profiles) as well as context based payment functionality.

Since the OIC provides standardized interfaces such as PKCS#11 and a crypto API, integration into other security applications is possible as well. In such a way, the OIC can be used for example for authentication in the case of secured communication via the SSL protocol. The encryption of a session between a client and a server can then be performed without further access to the smartcard.

In order to be able to describe the rights of an OIC cardholder, a combination of authorizations and roles has been developed within this thesis. In the literature, similar approaches are known as Role Based Access Control (RBAC) [San96] [FK92] [BG98]. Authorisations determine whether the cardholder has the permission, to do or to activate something with the presentation of the card. Examples of that are:

- the opening of doors,
- the access to computers,
- the access to files.

Such authorizations can be assigned either to an individual person or to a group of persons. The set of all possible authorizations is called authorization space A. An example of such an authorization space could look as follows:

A = {„open doors building Rh75“, open doors building Do15“, „access computer A“, „access computer B“, „sign contract“, „hire and fire“}

On the one hand, a role allows a grouping of persons. On the other hand, it also allows assigning authorisations to such a group in an implicit manner. Examples of such roles are:

- department manager
- member of the software development team

For the definition of a specific role the set of all possible roles (i.e. the role space) has to be defined, which at the same time describes the most extensive role.

Departments	R&D	Sales	IT
Positions	manager	tech staff	adm staff
Projects	smartcards	networks	

Table 21 Example of a role space

The example from table 2 can be written also as a set of 2-tuples:

R = {(Department, R&D), (Department, Sales), (Department, IT), (Position, manager), (Position, tech staff), (Position, adm staff), (Project, smartcards), (Project, networks)}.

In the following, the term "role" is being used in an a little varied form: every role, which is defined from now on, consists of a partial set of the role space and a partial set of the authorization space. By that, a set of authorizations can be assigned to a role. The advantage results in the fact, that to all persons, who are assigned to a specific role, the same authorizations can be assigned in an implicit manner.

For example, for Bob, who manages a smartcard development project, a role could be as follows:

R_{Bob} = {(Department, R&D), (Position, manager), (Position, tech staff), (Project, smartcards)}.

A_{Bob} = {„open doors building Rh75“, „sign contract“, „hire and fire“}

The entire authorisation of Bob consists of R_{Bob} and A_{Bob} and can be used for other project managers as well. It thus describes the group of all project managers who are responsible for smartcard projects.

An extension to this authorization model becomes necessary if one considers the following situation. Project manager Bob, participates in a company-intern training. This lasts a week and takes place at another company site, to which Bob usually has no admission. Some additional attributes, which do not fit into the above authorization model, can be derived from this example. For example, for the duration of the course Bob should get the following additional authorizations:

- admission to the building where the training takes place
- temporary access to computers used within the training

From that, the following requirements on an extended authorization model can be derived: It has to be possible to limit authorizations for one or n-time usage, limit them within a given period, or a combination of both. Such extended authorizations can be represented as 5-tuples in the form

A = {function, subfunction, counter, validity period {valid from, valid until} }

The function designates the actual permission to be granted, for example opening a door. The counter can be used to assign authorizations that are supposed to be used only n times. Additionally a temporary validity period can be assigned as well. By this for example Bob can be granted 5-times permission to access a certain computer, but only during a certain period.

After developing a model for roles and authorisations, their relationship to a cardholder of an OIC will be described. By the extended role term, it is obvious, that roles, and therefore a set of authorizations, can be assigned to a group of cardholders. In

addition, individual sets of authorizations can be assigned to every individual OIC cardholder. Figure 49 illustrates the relationships between persons, authorisations, and roles by means of an example. The three represented persons are cardholders of an OIC. They carry, (from to the left to the right) the individual authorization sets {A, B}, {B} and {A}. All three persons hold role Role1 and therefore, together with this role the elements {(X, Y), (X, Z), (V, R)} are being assigned from the role space. Additionally, authorization C has been assigned to Role1 and therefore to all three persons.

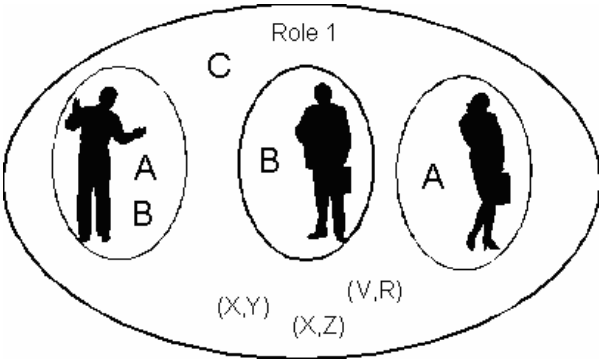


Figure 49 Persons, roles, and authorisations

A trustworthy authority, so-called Trust Centre (TC), which issues and manages the identity cards, is an important basis for security infrastructures. One can distinguish between two types of Trust Centres: Commercial Trust centres (CTC) and Private Trust Centres (PTC). According to the security requirements within a company or governmental authority, which uses the OIC, the security requirements on a PTC can be less high than on a CTC. Furthermore, the operation of CTCs is often controlled by law (for example in Germany by [SigG97] and [SigV97]).

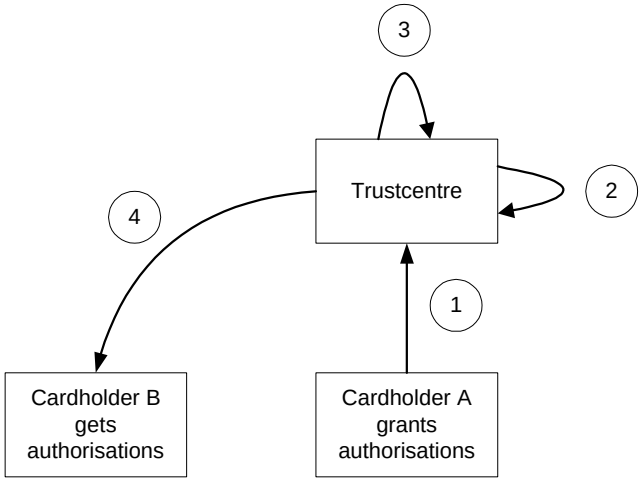


Figure 50 Delegating authorisations

Within this thesis, means have been implemented, which allow authorisations not only be directly assigned from the TC to a cardholder, but also being delegated from one cardholder, having the appropriate authorisation, to another. As illustrated by Figure 50 cardholder A wants to delegate a sub-set of his authorisations to cardholder B. A signs the authorisations to be delegated and sends it to the trustcentre (1), which verifies the authorisations, the signature of A and whether or not A is allowed to delegate these authorisations based on the data being stored at the TC (2). If everything is okay, the TC generates a new file named EF.OIC.CH3, which holds the new authorisations of cardholder B, and is being signed by the TC (3). After that, this file is being stored on the OIC of cardholder B (4).

An advantage of this approach is, that for the verification of an authorization only the signature certificate of the TC is needed, and not the complete delegation way has to be known. The TC is logging the delegation of authorisations, because sometimes lost or defective OICs have to be reconstructed. By this, it is also verifiable who has delegated which authorisation to whom.

In the central directory structure at the TC, which stores the personal data, assigned authorizations, certificates and roles, lists of revoked authorizations are being stored, so-called Certificate Revocation Lists (CRL). Inconsistencies may occur in the sense that it might happen, that an authorisation has been revoked, but the authorisation file on the corresponding OIC still does not reflect that. Because of this, the lists of revoked authorizations are regularly being distributed to the card terminals. As soon as e.g., an OIC is being presented to a card terminal, revoked authorisations can be removed, i.e. the card terminal requests the new authorisation file from the TC and stores it on the card. This has the advantage that a TC does not have to ask a cardholder to show up in order to get a new authorisation file or even to mark an OIC as invalid.

The implementation of the OIC consists of the OICs together with a central OIC management system for the issuing authority. This thesis has put big emphasis on the OIC being usable in a great number of applications and a broad application environment, by an architecture and implementation being based on widespread open standards.

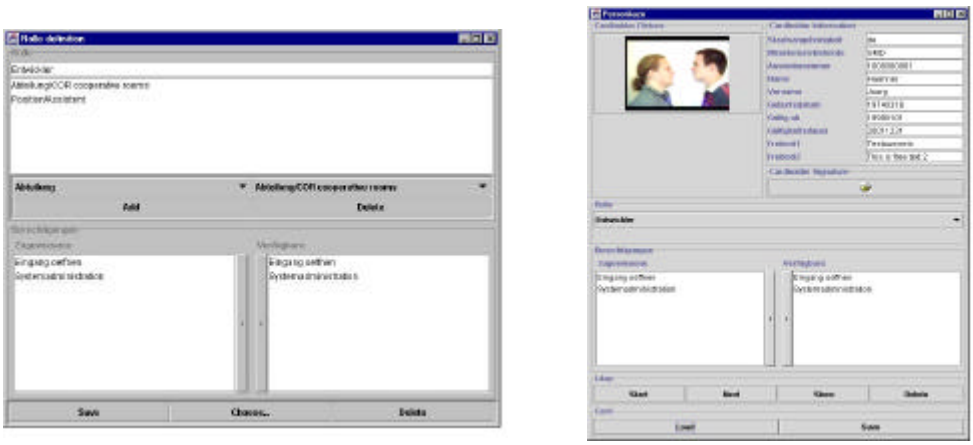


Figure 51 User interfaces of the OIC management system

All smartcards with a file-based smartcard operating system compliant to ISO 7816-4 [ISO7816-4] can be used as a basis for the OIC. Currently two different types of such smartcards have been used: smartcards from Giesecke & Devrient with the file-based smartcard operating system STARCOS SPK, and Mifare Pro smartcards from Philips, which have two interfaces: contact and contactless. This dual interface technology enables secure operations via the contact interface, such as digital signatures, and comfortable operations like identification and micro payment via the contactless interface.

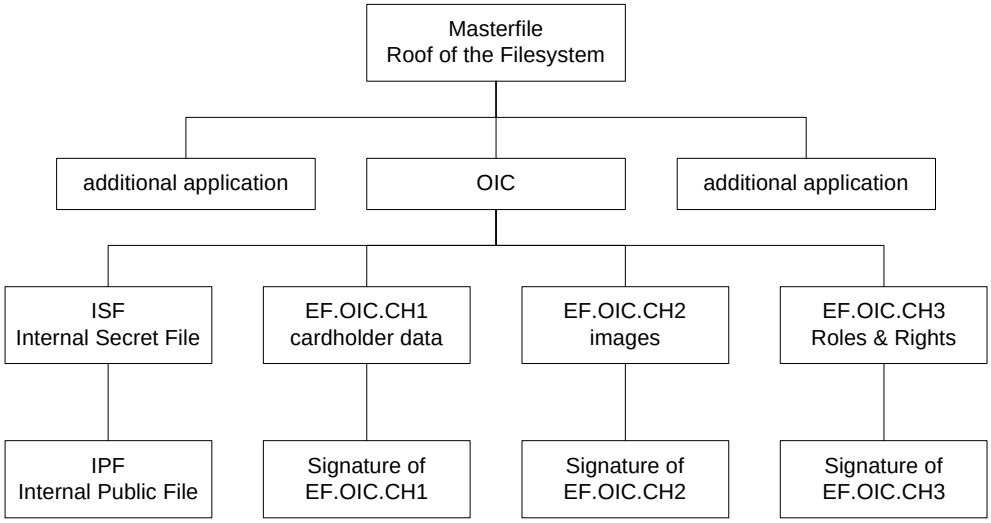


Figure 52 Structure of the OIC file system

The file system on the OIC (Figure 52) is based on the description of the identity card data in [TTT00]. All invariable cardholder data, except the pictures, is stored in the file EF.DIA.CH1. The pictures (passport photo, digitised signature) are stored in the file EF.DIA.CH2 and the roles and authorizations in the file EF.DIA.CH3. All cardholder data are being signed using the DSA algorithm. By this, the signatures of the files EF.DIA.SIG.CH1, EF.DIA.SIG.CH2, and EF.DIA.SIG.CH3 can be coded in binary form in the X.509 format and therefore directly by verified within the Java Cryptographic Architecture (JCA, see below) without further need for conversion.

The Internal Secret File (ISF) is used to store secret keys as for example a Personal Identification Number (PIN). Additionally, the verification keys of the so-called Root TC (in Germany for example RegTP), the signature, as well as the authentication certificate of the issuing authority are being stored on the card. Additional public keys and certificates can be stored in extra files.

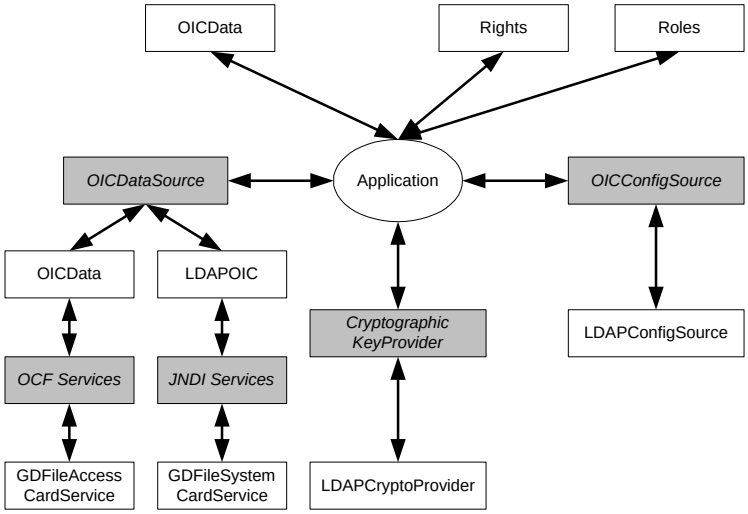


Figure 53 Architecture of the OIC system

For the implementation of the directory service a directory server, which is compatible to the Lightweight Directory Access Protocol (LDAP, which is a simplification of the X.500 protocol) has been chosen [OLDAP] [WY00]. The Java Naming and Directory Interface (JNDI) is being used for the integration of the directory service in order to be able to access the hierarchical structured data on the directory server in a system and manufacturer independent manner [LS00]. Thus, at any time the used directory server can be replaced by another (JNDI compatible) directory server, without this affecting the OIC implementation.

Furthermore, the OIC implementation is compliant to the Open Card Framework (OCF) [OCF99], i.e. it provided a Java API for the integration of the OIC into Java-based applications. This allows other software developers a simple and easy integration of the functions of the OIC as well as it simplifies access to the data stored on them.

Additionally, the OIC is compliant to the Java Cryptographic Architecture (JCA, [Knu98] [JCA97]) and serves as a so-called Cryptographic Key Provider. The JCA provides a standardised API for Java programs to use cryptographic functions and digital signatures. However, the JCA does not include any implementation of cryptographic algorithms. They are being integrated into the JCA via so-called Service Provider Interfaces (SPI), which can be developed by an application developer himself or bought from third party providers. The Java Cryptographic Extensions (JCE, [JCE00]) provides a collection of implemented cryptographic algorithms, which were integrated into the JCA. For the OIC the IAIK JCE [IAIK] has been used for the generation and verification of digital signatures and for secure messaging.

8.3 Confidentiality and Integrity

A typical work process in the Co-operation Platform is the establishment of contact to other team members of that particular project, which, if they are available, is initiated, for example, by means of suitable multimedia communication software (e.g. video conferencing). In the case that group-members are not reachable in person, a message deposit for example via email, voice-mail, or messaging system (e.g. SMS) is provided.

In the implementation of confidentiality and integrity of communication data one has to distinguish between measures in the application layer and below, in accordance with the layers of the OSI model. The application layer is particularly suitable for efforts to use the Co-operation Platform through dependability systems in a trustworthy environment for virtual teams. Here, the Office Identity Card and cryptographic procedures in Web browsers already support numerous mechanisms for encrypting, signing, and authenticating. Consequently, the configurable, flexible use of these procedures makes confidential communication and co-operation possible for all team members and secures the integrity of the connection depending on the respective team security policy.

A prerequisite is that the application programs support the procedures to be used, which often requires an adaptation of the used software. Therefore, not every application is equally suitable for the integrated use in the Co-operation Platform. In addition, information, which is relevant in deeper layers (e.g. sender and recipient addresses), cannot be encrypted, but can in turn be exploited for traffic flow analyses. Therefore, confidentiality and integrity of the transmitted data below the application layer is guaranteed by the use of encryption protocols such as SSL [FKK96], IPSEC [DH00] and IPv6 [Min96] [Ste99].

Within this thesis, a special focus is on securing multimedia conferencing sessions, in particular video conferencing, between two or more participants. Therefore, a scalable secure conferencing gateway has been developed, using a special partial encryption method for video data streams [BRK97] [KRSB97].

Multimedia conferencing supports synchronous real-time communication between two or more distributed participants. Characteristic for these systems is the combination of live media like real-time audio and video between different participants and the possibility of sharing documents and applications. In multimedia conferencing systems, the need for confidentiality and privacy has gained in importance, particularly in open networks like the Internet.

Multimedia conferencing systems have the following security requirements: access control, authentication, data confidentiality, data integrity, and non-repudiation. The basic building blocks meeting those requirements are encryption, authentication, certification, and integrity preservation [Schn96]. For real-time video transmissions, there is a special need for selective and scalable encryption of the transmitted data.

In principal, secure communication for multimedia conferencing systems can be built up on two different layers [FRR00]:

- Security in the transmission or networking layer, i.e. security is already provided by the networking protocol used, e.g. through the Secure Socket Layer (SSL) [FKK96]. An additional data manipulation by security applications is not necessary.
- Security in the data layer, i.e. before data is transmitted from a sender to a receiver it will be manipulated by the appropriate security functions in the application. Either the security functionality can be applied to the application, or the application itself is designed to gain security for other programs, e.g. the Secure Shell (SSH) [Ylo95].

One of the drawbacks of network layer security mechanisms is the need for secure underlying transport protocols as provided by version 6 of the Internet Protocol (IPv6). The advantage of data layer security is that the transmitted data can be subdivided into parts with sensitive and insensitive data with respect to the human perception. In comparison to providing security on the data layer, all transmitted data are protected in the network layer. This tends to raise problems when transmitting huge amounts of data, which is typical for example in multimedia conferencing systems. The network layer is not capable of subdividing the data stream in parts with a higher need for protection and parts with a lower or no need for protection. Implementing security functions in the data layer has the advantage that only some parts of the data need to be protected, and so the amount of time spent on protecting them can be extensively reduced.

As cryptographic functions for multimedia conferencing systems must cover different aspects of security, an elegant way to combine these is the provision of a scalable and secure conferencing gateway, offering different security functions, adaptive to the requirements of specific applications and the properties of special forms of multimedia data. Scalability for such encryption methods can be achieved by partial encryption of multimedia data. Therefore, a special scalable partial encryption method has been developed, which allows a security level of nearly every granularity [KR97a] [KR97b]. It is applicable to the secure transmission of video data streams and provides various ports for the different applicable video compression modes used in video conferencing: M-JPEG, MPEG-1/MPEG-2 and H.261/ H.263 [Ste99]. For further details on the used partial encryption method, see [K98].

The secure conferencing gateway supports the secure end-to-end transmission of video data streams over open internetworks between two or more distributed sites and as such provides the following advantages:

- security can be achieved even if the used conferencing applications do not support it
- video conferencing systems can be implemented independently of the used encryption methods

The gateway consists of two parts, an Encryptor and a Decryptor using the scalable partial encryption method. Between two or more such gateways, there are two channels established, a data channel that carries the encrypted video data, and a control channel.

The control channel is used for authentication, exchange of session keys, the chosen security level, and synchronisation during the session (e.g., changing the session key or the security level during a conferencing session). This architecture is characterized by its modularity and scalability in order to reflect different application scenarios and different levels of security needed in various conferencing situations.

Unlike in [BHS94] where all session keys are transmitted to the participants in advance by means of secure e-mail, the session keys are chosen by random during the conference. The initial session key is chosen by the first site entering a conference. When the next site enters the conference, it recognizes that there is already another participant, identifies and authenticates itself, and then requests the current session key. The session key is then encrypted using the public key of the requester. The initial security level is set to a default value, which is suitable for typical conferencing situations.

The session key and the security level can be changed by any site at any time, which requires synchronization through the control channel. First, the change of the session key or security level will be announced. After the acknowledgment of all sites, changes take place at the pre-scheduled time.

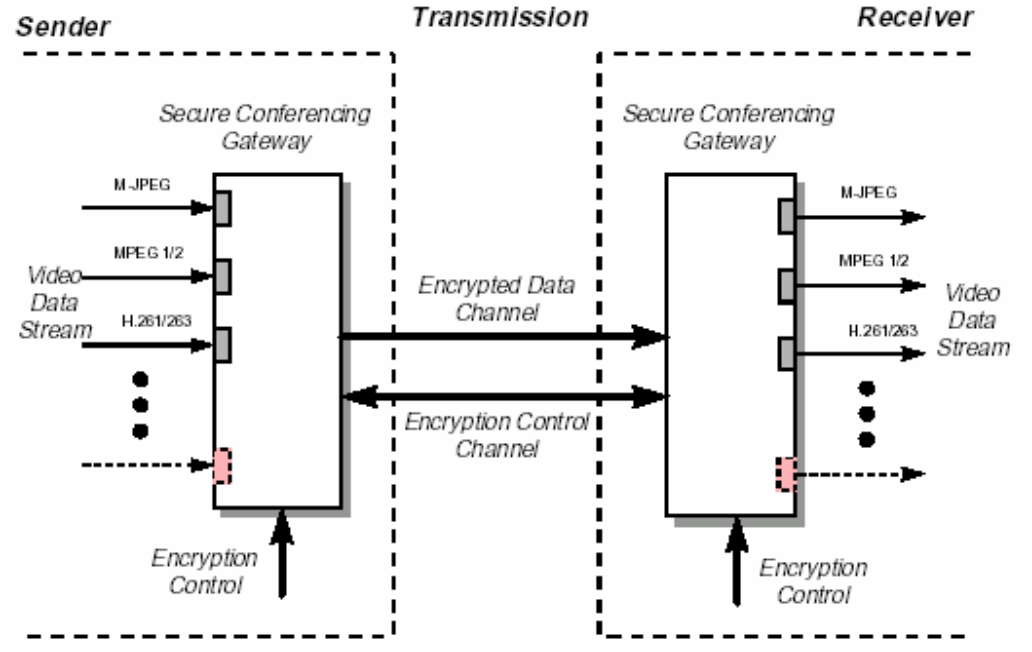


Figure 54 Secure conferencing gateway

The implementation of the secure conferencing gateway is not just restricted to point-to-point communications between two distributed participants. It can also be used for point-to-multipoint transmissions between two or more distributed sites. To achieve this goal, IP-Multicast is being used [Eri94].

In some application scenarios, like seminars and discussions, people often enter and leave a conference at their convenience. This is reflected by the fact that there is always one master or server gateway that supervises the change of session keys, security levels and ensures the consistency between all involved gateways. The change of the master role

would generate more communication overhead than the distributed negotiation for changing the session keys and security levels using IP-Multicast.

The implementation of the gateway is based on the Berkeley MPEG encoder/decoder for the video parsing routines [PSR93]. For encryption, decryption and certification of public keys the security toolkit SecuDE is being used (see also section 8.1, [ABF+96] [Secude]).

8.4 Non-repudiation and Auditing

In the Co-operation Platform, the users should be held responsible for their actions. Therefore, the support of co-operative, co-ordinated work on e.g. digital documents in teams distributed geographically makes it indispensable to implement individual project processes as reconstructable and finally non-repudiable processes. For this purpose, in addition to the authorisation check, the access to services and documents provided by the Co-operation Platform is being logged. In particular, the integrity and attributability of documents is secured by signing the documents and/or the executed modifications. In addition to the document name, the user name together with the date and time are stored in a document history. As an additional benefit, such a recording forms the basis for accounting and billing of the used services. By this, the costs for services provided by third party providers can be assigned to individual project members or projects.

Non-repudiation requires auditing. The use of encryption to secure communication does not obviate or otherwise affect the need for auditing. Auditing of e.g. multimedia conferences, whether scheduled or ad-hoc, is essential for tracking user interaction in a secure environment.

Auditing should always take place in a centralised manner — usually on a server. User information, including login and exit, should be logged whether or not encryption is enabled. Although the server does not need to decrypt the content of the communications — e.g. the video data stream — to pass the data from user to user, the capability to decrypt the audio and video if necessary for auditing purposes may be considered.

Audit information must be selectively kept and protected so that actions affecting security can be tracked to the responsible party. This is another reason, why the Co-operation Platform has to identify and authenticate its users. Within the platform an audit trail of security relevant events is being kept. If a security violation has occurred, information from the audit trail may help to identify the perpetrator and the steps that were taken to compromise the system.

Currently, the Co-operation Platform keeps track of user login/logout events, access to documents within the document repository, and multimedia conferencing attendance. Auditing is being dictated by security policies. Consent to auditing is being displayed to inform users of such action. In addition to warn users of the consequences of abuse, they serve as legal protection in the event of such abuse.

8.5 Management of Security Policies

Within the framework of selectable security policies, the Co-operation Platform makes it possible to identify persons uniquely, to address them in a location-transparent manner, to guarantee the authenticity and confidentiality of communicated messages and stored data. Team-members can openly communicate within such a team and resources, such as files, tools and services, can be shared. However, the team is cut off from the outside world in defined manner.

According to the Trusted Computer System Evaluation Criteria (also known under the name Orange Book), a security policy (SP) or security policy is a “collection of laws rules and practices which define the handling, the control, and the dissemination of sensitive information in an organization” [TTT00]. Security policies are part of the security architecture of an IT system [Eck01]. For their enforcement, they require both tangible security mechanisms for the fulfilment of security requirements, such as authentication, access control, confidentiality, integrity, non-repudiation, delegation, and auditing together with an actor – a so-called reference monitor - that implements the compliance, checks and guarantees it.

Security policies are mostly described informally for the better understanding and legibility. A precise formulation in the form of temporal calculi or predicate calculi is called a formal security model and is the basis for formal verification and certification procedures. Well-known security models are, for example, Bell/La Padula [BL73], Biba [Bib77], Clark/Wilson [CW87] or Chinese Wall [BN89], which however have in common that they are only tailored to the corresponding security policy. The balance model from Grimm is based on a balance principle between states of obligation among the partners and proofs of these states [Gri93].

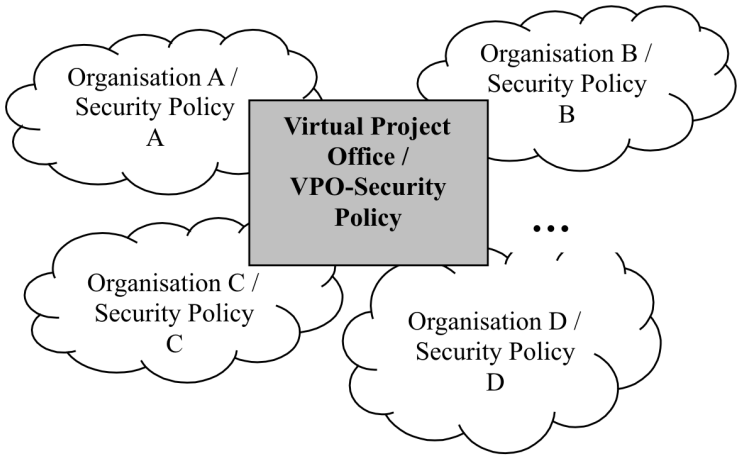


Figure 55 Different security policies

Since distributed teams using the Co-operation Platform may result both within a larger organization (for example between different departments) and as supra-organizational teams, it is not to be expected that their Co-operative Workplace can be based on a single, uniform security policy, which also considers the security requirements of the Co-operation Platform. On the contrary, since an essential application scenario is the support of business-to-business processes, it is to be assumed that different SPs will coexist, which define the most different security requirements with most different levels of granularity. Therefore, it is a central and highly complex security problem of the Co-operation Platform to arrive at a simple, flexible and as automated as possible definition of security policies for a team together with its mapping onto the SPs of the participating organizations.

In the literature, there are mainly two approaches to unite different SPs: interoperation and combination. With interoperation, the superior security policy must not violate the sub-SPs and must guarantee their autonomy [GQ96]. In the case of a combination, the superior SP can include inconsistencies with respect to the sub-SPs, but must guarantee the security in the present context [Bel94]. Since not only two organizations or departments usually participate in virtual teams, within this thesis a variation of the second alternative (i.e. combination) is being pursued.

When e.g. a VPO is being set up for a virtual team, an independent security policy can be defined for this particular project and team. The consideration of the subordinate SPs of the participating organizations is done by searching a mapping function between the team SP and the relevant sub-SPs via automated negotiating strategies. However, this is only possible if a universal specification language, which is not tailored to a uniform security model, is available. Therefore, this thesis pursues an approach based on XML [W3C-XML]. XML is platform-independent, easily integrable into the Internet world and offers the possibility to bridge the gap between policy specification and implementation and to allow automated negotiating strategies.

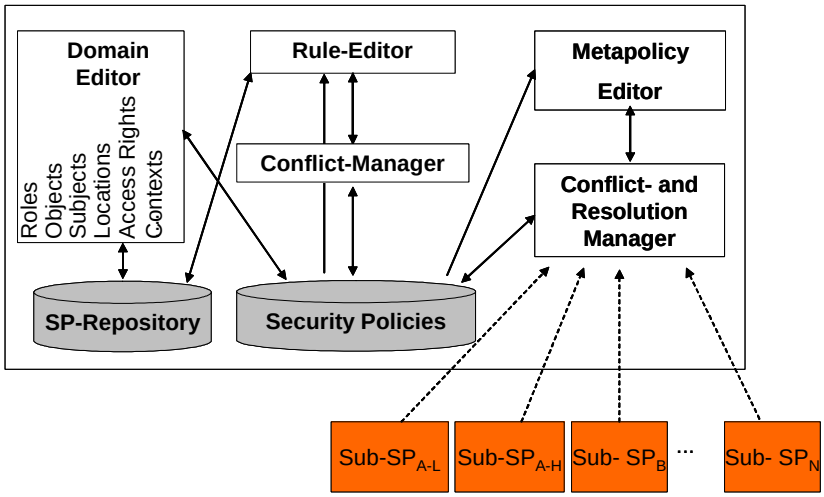


Figure 56 Security domains and meta-policies

As a first approach, an access control manager for the VPO has been designed. For the definition of an SP for a newly established team, the access to its documents and services, a prototypical tool is being provided. This prototypical tool is based on a security policy repository via which essential rules can be defined and selected. The users interface allows the definition of the roles within the team, subjects, objects, access and other rights, as well as their relationships via a rule editor. Additionally, a conflict manager checks for conflicts and helps to ensure consistency.

8.6 Location Management

Location management plays an important role in this thesis. In the context of this thesis it splits up into two fields, location management for fixed and mobile devices (e.g. locating a specific device within a building) and location management of persons (e.g. locate team-mates in order to communicate with them). A specific security problem of location management is that of allowing the subjects of such a system to retain control over the distribution of the information about their location, e.g. in order to prevent others from location tracking, i.e. to protect their privacy. Within this thesis a concept for controlling the reachability of an individual has been developed, which at the same time allows to maintain a high degree of privacy and data protection.

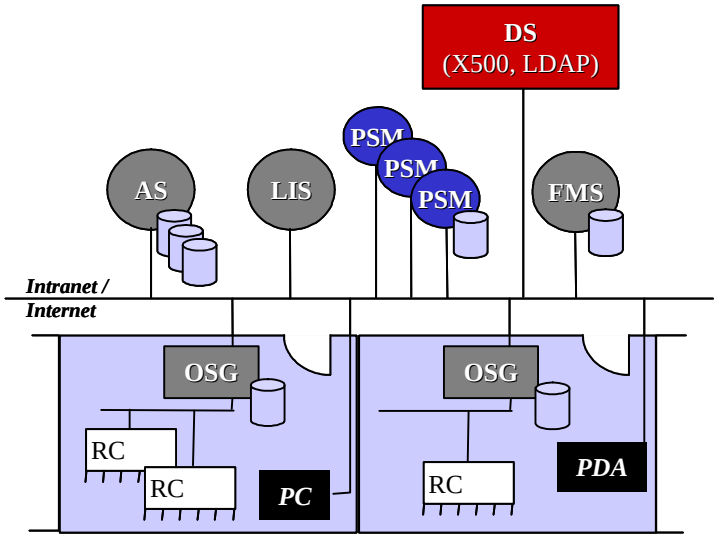


Figure 57 Location management

The information about the location of a person is sent via the network from the RoomComputer to a so-called location information server (LIS). The LIS hands over the data to the personal security managers (PSM) of the respective person. The PSM then stores the location information encrypted so that only this person can access it. A request

about the current location of a person within a building can be handled only via the LIS, and the respective PSM.

The implementation of personal security managers (PSM) together with local and central location managers in conjunction with appropriate public key cryptography methods serves to handle privacy protection, i.e. it prevents unauthorized knowledge about where people are located and/or have been located in the past, the resources they are using and/or have been using in the past etc.

8.7 Virtual Project Office

Within the Virtual Project Office, security measures are being provided within three modules:

- VPO Configuration,
- Access Control, and
- Rights Management.

In order to configure the security policy for a particular Virtual Project Office, the VPO configuration tool offers a variety of different security tools to choose. Therefore, as for all integrated tools and services within the Co-operation Platform, the configurator of a Virtual Project Office defines a subset fulfilling the security policy of his/her virtual team. A possible security policy could e.g. include the encryption of every message being exchanged within a VPO as well as a signature added to each official document but without any encryption for saving documents in the document repository. Depending on the security policy well known mechanisms like asymmetric cryptography, symmetric session keys, and using certification authorities support for example encryption of e-mails, signing of documents, and tracing & tracking.

After the configuration of a particular Virtual Project Office, each user who wants to get access to the Virtual Project Office has to be identified and authenticated as a member of this project. Identification and authentication provides every project member with particular rights regarding the access to different zones and the use of privileged tools within the Virtual Project Office (e.g. for administration purposes). The login process of a team member to the VPO consists of two steps. The first step is called identification: the user announces who he is, i.e. by showing his OIC. The second step is called authentication: the user proves that he is the one who he claims to be, i.e. by entering his PIN. Login will succeed if the user shows a valid OIC, the corresponding PIN and has appropriate access rights, otherwise access will be denied.

While access control identifies project members and assigns the appropriate access rights of the different zones within the Virtual Project Office to authorised members depending on their role (i.e. administrator, member, non-member, etc.), rights management deals with the document repository. The rights management system defines per-document rights, which guarantee that all documents of the VO are securely used of and distributed between the project members during the authoring phase. It controls and tracks any single movement, editing, and distributing of any single document.

A VPO provides so-called multilateral security services [FP97] [Gri97a], which are implemented as inherent services running in the background so that users can concentrate on their work instead of being confronted directly and constantly with specific security measures (e.g. identification and authentication). The provided security services include authentication of project members, single-sign-on, confidentiality and integrity of communication contents, service and document access control, as well as the attributability of project processes to team members.

Multilateral security means take into consideration the security requirements of all the parties (see above) involved. It also means considering all involved parties as potential attackers. This is especially important for open communication systems like the Internet, as one cannot expect the various parties to trust each other. Consequently, the requirements, which have to be fulfilled in order to achieve multilateral security, are particular high for networks that are intended to for universal use. Multilateral security requirements are not necessarily free of conflicts. Therefore, they have to be brought into a meaningful and acceptable balance [MR99].

8.7.1 Authentication of Project Members

Within this thesis, the unique and verifiable authentication of each individual project member at the beginning of every project-working phase has been considered as a primary goal of protection with respect to a trustworthy communication and cooperation among the registered project members. First, the user authentication against the VPO client software occurs by users using their OIC. That guarantees a suitable visualisation of the personal project office via its graphical user interface and activates, if required, the integrated services and tools necessary for individual project work. In addition, in the course of individual project work, an authentication with respect to the VPO server software is performed at every entry into a specific project environment or work zone.

An optimum transparency of the used protection mechanisms is achieved through the authentication of the project members by means of single-sign-on identification. By using the Office Identity Card introduced above, the identity of the participant is confirmed uniquely for the whole time of a working session. For every server-side connection establishment the authorisations of the corresponding user and, if necessary, their validity are being determined with the aid of a trustworthy certification authority.

After successful authentication, the work in the personal working environment can start. For this purpose, within the VPO a so-called Personal Office with access to all stored personal data and the possibility enter a selected project environment is being provided. The establishment of a connection to the respective VPO server occurs as transparently as the assignment of the individual project authorization with the aid of the authentication information depending on the user’s group identity (see section 8.7.3).

8.7.2 Confidentiality and Integrity of Communication Contents

A typical work process in a VPO is the establishment of contact to other team-members of that particular project, which, if they are also present in the project workspace, is initiated for example by means of suitable multimedia communication software (e.g. videoconferencing). In the case that team-members are not reachable in person, a message deposit for example via email, voice-email, or messaging system (e.g. SMS) is provided.

For the implementation of confidentiality and integrity of communication data one has to distinguish between measures in the application layer and below (in accordance with the layers of the OSI model). The application layer is in particular suitable for the efforts to use the VPO through dependability systems in a trustworthy environment for virtual teams. Encryption in the application layer concerns both, contributions by this thesis together with integrated third party software. Developments such as the digital office identity card or cryptographic procedures in Web browsers already support numerous mechanisms for encrypting, signing, and authenticating. Consequently, the configurable flexible use of these procedures makes the confidential communication and cooperation possible for all team members and secures the integrity of the connection depending on the respective team security policy.

A prerequisite is that the application programs support the procedures to be used, what often requires an adaptation of the used software. For this reason, not every application is equally suitable for the integrated use in the VPO. In addition, information which are relevant in deeper layers (e.g. sender and recipient addresses) cannot be encrypted, what can in turn be exploited for traffic flow analyses [Schm98].

Therefore, confidentiality and integrity of the transmitted data below the application layer is guaranteed by the use of well-known encryption protocols [Eck01]. SSL-extended project servers do guarantee for example an encrypted end-to-end transmission for TCP-based services. UDP-based services, which are not supported by SSL – such as IP telephony – require an encryption below the transport layer (layer 4) in the network layer (layer 3). IPSEC, which extends the conventional IP protocol by some components (integrated into the forthcoming IPv6), belongs to these procedures. Unlike SSL, IPSEC can also be used for store-and-forward between two routers and, therefore, is suitable for the creation of virtual private nets (VPN), on whose basis virtual project offices can also provide their services depending on the security requirements [Schm98]. For the protection of video conferencing sessions, the secure conferencing gateway together with its scalable partial encryption method is being used (see section 8.3).

8.7.3 Access Control

Providing secure services in a VPO is an important issue. Due to the lack of security in many of today's applications, there is a lot of work to be done to secure third party applications and services in the context of Virtual Project Offices.

The above-mentioned security measures for the encryption of communication data are without restrictions also suitable for the exchange of digital documents and the secure

use of the project management services. The authorized access to documents in the project-specific document repository as well as to services provided for the respective project is performed via the association of the individual project members with specific project-dependent roles and regulated by security policies.

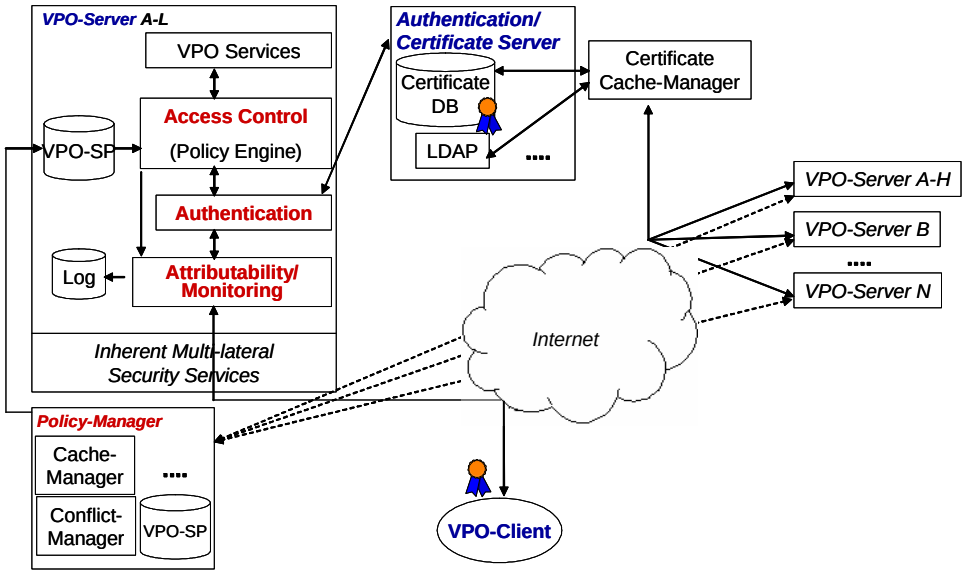


Figure 58 Access control to services and documents within a VPO

The VPO distinguishes different system-specific project-independent roles – such as project administrator, project manager, group members and guests - as well as project-specific roles – such as authors, editors, graphic designers, publishers. The assignment of the corresponding authorisations such as the maintenance of the system or the availability of a specific digital document is performed depending on these roles upon logon at a VPO server. Within a single Virtual Project Office, by the evaluation of the assigned authorizations, only specific services are accessible and the access to project documents is regulated.

8.7.4 Attributability of Project Processes

The support of cooperative, coordinated work on digital multimedia documents in teams distributed geographically makes it indispensable to implement individual project processes as reconstructable and finally non-repudiable processes. For this purpose, in addition to the authorization check, the access to services and documents is logged in accordance with privacy protection rules.

The integrity and attributability of documents is secured by signing the documents and/or the executed modifications. In addition to the document name the date and the time is stored in a document history. To the proof on both sides, the user finally receives a receipt for the modifications and the actions on the digital document. Furthermore,

recording forms the basis for accounting and billing of the used services. In this way, the costs for services provided by third party providers can be assigned to individual project members or projects.

8.8 RoomComputer

This section gives an overview of the security requirements with respect to the RoomComputer, applicable security functions. It describes various security aspects related to the integration of physical resources and rooms into the Co-operation Platform via the RoomComputer and how they have been approached.

Up to now, security aspects are being hardly ignored in conventional facility management systems. Little by little, they are coming into consciousness of system operators, designers, and manufacturers. Connecting the various facilities within an office building to open nets (as for example the Internet) is playing a rapidly increasing role in today's facility management systems. Security problems are especially evident when field busses are being linked to local area networks (e.g. Ethernet). In most field bus standards, security aspects are hardly foreseen, what in principle involves risks. New solutions are required here.

Therefore, connecting the RoomComputers to an Intranet or the Internet raises some security issues. The services provided by it (e.g. controlling the room devices) could thus be accessed from any computer on the same Intranet or Internet. However, it has to be prevented that users or computers may access such services without having the proper rights. Access to a room should only be granted under specific circumstances and with prior authorisation by the associated RoomComputer, i.e. services in a room must only be accessible via the associated RoomComputer, which encapsulates them. Direct access to the RoomComputers has to be prevented by applying standard procedures such as local "Firewalls", mutual authentication and securing the communication channels (e.g. by using the Secure Socket Layer, SSL). Beside unauthorized access to the room services, unauthorized access to a room itself must be prevented under certain circumstances.

However, the security measures for facility management applications have not to be focused on optimal protection of highly confidential data, but on protecting data and services against illegal access. Therefore, the security methods needed here have to be fast, with respect to real-time requirements, and to be cheap to be implemented in order to supply an emerging market of embedded devices and systems. The expense to break into such a system does not need be very high, but it should be more expensive than the legal access to the provided services. In all these facility management applications, the cryptographic functionalities must cover different aspects of security, like confidentiality, integrity, and authenticity. Therefore, different modules of encryption mechanisms are being available. This thesis provides an elegant way to combine these modules by a scalable and secured service gateway, providing different security functionalities, adaptive to the requirements of specific applications and the properties of special forms or even multimedia data.

8.8.1 Prevention of External Attacks

External attacks to a RoomComputer and the Open Service Gateways (OSGs) can be separated into two dimensions: the type of attack and the type of attacker. The following types of attacks can be distinguished:

- attacks to the communication link between two OSGs
- attacks to an individual OSG node
- attacks to an authentication device such as a smartcard reader

The type of attacker mainly involves:

- attackers passively listening (eavesdropping)
- attackers actively modifying data (spoofing)

Attacking the communication line between two OSGs can be prevented by using standard point-to-point encryption protocols such as the Secure Socket Layer (SSL) [FKK96]. Since the SSL protocol is based on public key cryptography, it introduces the problem of securely distributing public key certificates. Additionally the required individual private encryption keys have to be stored securely. The problem of key distribution refers to the provision of a Public Key Infrastructure (PKI).

A certification authority (CA) issues pairs of certificates and secret keys for every node within a RoomComputer installation, together with a CA root certificate, which is created in order to verify each nodes certificate. The secret key and the public key certificate of every node as well as the CA's root certificate can be stored securely on a smartcard. Then, each smartcard has to be installed inside the individual OSGs. Since the secret keys together with the CA root certificate normally have limited validity, the smardcards have to be exchanged in regular intervals. This approach prevents all kinds of attacks against the communication between two or more OSGs, but involves a high level of administrative overhead. A simpler, but less secure, method is to store the certificates and keys not on a smartcard, but directly on the RoomComputer (so-called Software-PSE).

Attacking individual OSG nodes means gaining access to the services provided by them (and therefore on of the corresponding RoomComputer), usually by using security holes in the operating system or by the installation of hostile software. This can only be prevented by carefully examining the operating system before using it on the OSG. In addition, the probability of having a security hole will be reduced if only such application software is installed that is really needed and certified. This particularly applies to network related software such as e.g. FTP, TFTP, Telnet, and similar daemons.

A third way of attack could be the manipulation of an authentication device, like a smartcard reader, attached to a RoomComputer or an end-user device such as e.g. a PDA or desktop PC. This involves attacking the communication link between the reader device and the corresponding host. Here smartcard-based security measures can be applied, for example secure messaging [ISO7816-4], to encrypt the communication between the smartcard reader and the corresponding host.

8.8.2 Access Control

This section describes how access control is integrated within the RoomComputer. Basically this is being done by attaching smartcard readers to the RoomComputer in combination with appropriate security policies implementing means for example to control the access to a single room (locking/unlocking doors), access to the services provided by the RoomComputer (as well as their appearance to the outside). The access control system for the RoomComputer has not been implemented from scratch. Instead, it can easily be integrated with existing solutions and security infrastructures.

As described earlier, users are only accessing the provided services via the generated user interfaces rendered from the AUI-ML documents, which are being stored in the AUI-ML repository and specifically composed dependent on a user's role. One possible solution with respect to access control could be the extension of the Abstract User Interface Language with specific elements supporting access control, i.e. supporting conditional rendering based on a user's role. Another solution would be that each user interface element gets a property, which describes the rights a user must have in order to access it. Both approaches do cause some overhead during the rendering process of an AUI-ML document, because there renderer component would continuously call the access control system in order to check user roles and access rights. In order to reduce this overhead, one could think of a kind of local cache on each RoomComputer, which would in turn increase the storage space and the processing power needed on a RoomComputer.

For the current implementation a simpler solution has been chosen, which expands the AUI-ML document repository with an access control system. Instead of conditionally rendering AUI-ML documents, access to AUI-ML documents within the repository will be granted based on the security model described in section 8.2. Under some circumstances different AUI-ML documents are being provided for different user roles, e.g. documents accessible by administrators, regular users or guests (as a general default role).

An important aspect with respect to the services provided by the RoomComputer is context awareness, i.e. a service representing a fire detection system would provide access to different functions during set-up, runtime, or in case of a fire. Another example would be that individual control of blinds by regular users is denied in case it storms.

Context awareness requires an extension of the used security model within this thesis. The Generalized Role-Based Access Control model (GRBAC) could be a possible basis [MA01]. GRBAC is based on the more traditional Role Based Access Control model (RBAC, see [San96] [BG98]) and extends it in a way, that it distinguishes three different roles:

- the subject role,
- the object role, and
- the context role.

The subject role corresponds to the subject role within the RBAC model. The object role abstracts the different properties of an object into categories. The context role contains information about the environmental situation, i.e. the context, sensed by

appropriate sensors). A context can for example is described by the following attributes, time, location, system load or whatever environmental factors might be of importance. Altogether, these three roles do offer a flexible and expressive way to describe the access conditions to the services offered by a RoomComputer.

Chapter 9

Conclusions and Future Work

Work environments of the future will increasingly be characterised by a high degree of diversity, flexibility and dynamics. Work performed within the individual office locale is being supplemented by tele-co-operative collaboration, and co-operation within organisations is being supplemented by cooperation across organisations. The traditional local office work goes together with various forms of fixed and mobile tele-work.

Co-operative Workplaces are a framework, which addresses the foreseeable requirements of future work environments from several perspectives. They constitute co-operative landscapes, in which information technology and physical work environments are highly integrated and can dynamically be re-configured to accommodate alternating use cases and different work situations.

In order to meet these objectives, the research preparatory to this thesis has followed an integrated and interdisciplinary approach at the intersection of information technology, work organisation and architecture. The overall objective has been to develop a framework, which can serve as a common basis for any application case within a Co-operative Workplace. The presented framework takes an integrated and systematic approach to providing support for various aspects of collaboration among people working together apart.

9.1 Co-operation Platform

Within this thesis, an architecture for an open, teamwork-oriented, and unified Co-operation Platform has been designed and developed; it has been verified with a prototype serving as a proof of concept. This architecture focuses on users, their work contexts, and usage scenarios by providing a unified and worldwide accessible platform based on Internet technologies and widespread open standards. The Co-operation Platform and its components have been prototypically implemented in order to support the creation of Co-operative Workplaces, which can be flexibly built, customised and run according to user needs. By the implementation of a prototype of the Co-operation

Platform, it has been proven that its architecture, based on component technologies, service oriented programming, and multi-agent systems. It is powerful and flexible enough to allow the easy integration of existing tools and services provided by third parties.

The Jini Technology has been used in the course of this research to build the service-oriented parts of the Co-operation Platform architecture. However, the UDDI technology is regarded from many sides as being the way of the future. With the support of a community of a greater number of companies (including Sun, IBM, Microsoft), UDDI might become the de-facto standard for web service registry. The major difference between UDDI and Jini is that UDDI is a document-based framework (using XML documents) whereas Jini follows an object-oriented approach (using Java interfaces) for distributed computing platforms. The advantage of UDDI is that it is independent of programming languages and operating systems, like as CORBA is. Until now, UDDI just defines the discovery protocol, but does not provide the concept of events and transactions, as Jini does. It seems that Jini could be an enabling technology used to implement UDDI web services components but they might also be in conflict.

In an organisation, which runs several projects in parallel, each project will have their own Virtual Project Office. Conceptually, anyone who is a member of several projects would go from VPO to VPO as he stops working on one project and switches to work on another. As projects are set up, modified, and closed, the corresponding Co-operative Workplaces have to be dynamically set up, modified and closed. This process is called Co-operative Workplace configuration. The ability to dynamically create Co-operative Workplaces becomes an important factor in creating virtual organizations rapidly and facilitating their operation in cyberspace.

Therefore, a component needed to be developed, a so-called Co-operative Workplace Configurer; its requirements being to provide an intuitive user interface, based on a reservoir of generic services, supporting the flexible configuration of Co-operative Workplaces by defining relevant workplaces and by associating required services and resources with them. Depending on the domain of work, this reservoir and this configuration must additionally be augmented with domain specific tools and services. In particular, business projects require the integration of business processes, e.g. by utilising available tools that allow developers to assemble business process components [BEA98].

Additionally, winning the right team members for a project team and reorganising it during the lifetime of the project should also become integral parts of a Co-operative Workplace configuration. For instance, this will include contracting over the net, definition of responsibilities, rights, and roles within the project, as well as the provision for confidentiality, authentication and security profiles.

9.2 Virtual Workspaces

The Virtual Project Office implements an environment, which provides a customisable infrastructure for collaborating teams. It integrates and automates multiple services and tools and presents its users with a context oriented integrated view. Using the collaboration environment provided by the VPO in conjunction with the integrated human centred communication and collaboration services, the team members can initiate and conduct their collaborative tasks in a manner, which is appropriate to the problems they actually need to tackle. Specifying the collaboration partners together with the document(s) to which they require collaborative access, is sufficient input for the Co-operation Platform to set up the required communication channels, make the required reservations, and start the necessary tools at the right point in time. The users are relieved from the effort of having to control (and learn) a number of separate applications and can therefore concentrate on their actual work contexts and processes.

This thesis has presented the design and implementation of the VPO, a component of the Co-operation Platform designed to support collaborative work within geographically dispersed teams. It facilitates group cooperation and communication by integrating existing off-the-shelf components into a single comprehensive user interface, which allows easy control of the various functionalities available in the different components, whose combined usage in their previously non-integrated form has been too complex to employ in a productive manner. The adherence to the room metaphor as a means of interacting with the system and performing the cooperative tasks has been introduced and implemented subsequent to an analysis of the collaborative work requirements found to occur within the context of distributed governmental agencies.

Although the design and development of the VPO was partly driven by the application context of distributed governmental agencies, it is believed that this system is flexible and powerful enough to be usable in a wide range of application domains. Its approach for supporting the cooperation and communication tasks involved in joint work processes and meeting support as well as its foundation on commercially available system components makes it usable in a large number of distributed user communities. In the future, the practical usability of the VPO and its concepts need to be evaluated by social industrial scientist in different user communities. Such an evaluation has already been scheduled to start in the beginning of the year 2002.

Within the Co-operation Platform, the developed conference reservation and management software provides a simple and efficient method for end-users to schedule and set-up multimedia conferences. This software automates the necessary steps required to configure an MCU for a certain multipoint conference, which means that users no longer need the help of an operator to set-up or modify a multipoint conference. The conference reservation system can has a high degree of flexibility and an MCU independent architecture; it can easily be set-up on top of MCUs from different vendors.

So far, only two MCUs have been integrated into the MCRS system in order to support multipoint conferences: one is based on the ITU-T H.320 standard, and the other on the ITU-T H.323 standard. In between the both, an H.320/H323 gateway has been installed, in order to support mixed-type conferences. However, the MCRS system could benefit from the integration of more MCUs and gateways, such as e.g. the ones presented in [APWS99] and [ADG+01].

9.3 Physical Workspaces

The Science Club (SCLUB) designates as an office landscape, which has been set-up for in-company usage in combination with tele-working as an alternative work form. As such it comprises a set of non-territorial physical workspaces tailored to different kinds of activities. While being used as intended by the tele-workers, it additionally serves for demonstration purposes and has been the proving ground for the results achieved within the course of this thesis. The SCLUB environment needs to be upgraded on a regular basis to accommodate experiments with innovative technologies and to meet progressive demands on (tele-)co-operation, mobility, flexibility, and dynamic reconfiguration.

Within the context of this thesis research an innovative meeting room, called the CoMeet-Room, has been provided as part of the SCLUB arena; it is equipped with advanced digital information, communication, and media technology that creates a physical workspace for synchronous joint work of teams. The room supports communication and collaboration among individuals and between groups during meetings and (tele-)conferences. A Web-based control system, with an intuitive and context oriented user interfaces, has been developed for the CoMeet room in order to easily monitor and control all the media devices and technical components being used in the meeting room, both locally and remotely.

The Sm@rtLibrary represents a somewhat different development within the SCLUB and has served as a test and demonstration of a practical example with respect to ubiquitous technologies and their application. In this area, this thesis has confined itself to the localisation of persons and physical objects in combination with location-aware technologies, context-dependent services, platform and device independent access to the services and the information provided in such environments..

9.4 RoomComputer

Within the course of this research, the emerging need to integrate physical workspaces with virtual ones has been addressed by the development of the RoomComputer. This represents a significant achievement, with respect to advanced building automation and the management of facilities and services within physical workspaces. Additionally, it is the key component within the Co-operation Platform when it comes to seamlessly integrating physical and virtual workplaces under a unified view.

Through the introduction of the RoomComputers concept, a facility management system simply becomes a network of rooms, linked together via the Intranet or Internet. An expansion to include more rooms, devices, resources and services can easily and inexpensively be done, by adding RoomComputers and connecting them to the Intranet or Internet. The deployment of RoomComputer networks together with smart controllable room devices enables the development of compelling new facility management applications, which in turn moves us closer to fulfilling the vision of intelligent buildings that focus on and enhances users' comfort, and convenience.

The RoomComputer offers an open and flexible architecture with respect to the hardware, as well as to the software layer. Its 'Plug&Play' technology eases the installation and extension of the building automation and facilities management system, in particular with regard to all the devices and sensors. Compared to more conventional systems, the reconfiguration of a RoomComputer based installation is simple and smart. RoomComputers not only support building automation and facilities management, they rather bring new services into buildings, which allows for new business models with respect to building management. All services offered by a RoomComputer are accessible through Web-based technologies via the Intra-/Internet. The user interfaces are based solely on the developed Abstract User Interface Mark-up Language (AUI-ML), which allows RoomComputer services to be accessed from virtually any end-user device.

The software architecture of the RoomComputer is based on the concept of being an Open Service Gateway (OSG), using the OSGi framework specification, and being enriched by the Jini technology. This ensures the required degree of openness and flexibility of the RoomComputer concept.

The layered architecture based on generic drivers and interfaces has proven to be a simplification with respect to the plug-in of new devices. Originally, an important issue has not been addressed by the OSGi framework specification, namely the interoperation between two or more OSGs. In the research being described here, this interoperability has been affected using the Jini technology, which is also the basis for the integration of non device-based services, for example catering or car-rental. RoomComputers can be configured remotely by using the HTTP protocol and providing new configuration information as XML-based data.

Future extensions planned for the RoomComputer include:

- the integration of more devices into the OSG platform together with a refinement of the device hierarchy
- the extension of the remote configuration services, in particular with respect to the integration of CAD-based planning systems
- a better support for personalisation on the basis of user preferences and profiles, which requires an abstract way to define services and could potentially be based on the Web-Service Description Language (WSDL)

9.5 User interface

The development of the Abstract User Interface Mark-up Language (AUI-ML) arises from the fact that the user interfaces of Co-operative Workplaces need to be accessible from various end-user devices, such as PCs, laptops, PDAs, or mobile phones. Instead of developing a variety of different user interfaces, with each providing the same functionality for a different device, it was decided to formulate an abstract user interface description language, capable of being flexibly rendered into user interface representations for the target end-user devices and its user interface toolkit. This decision was motivated by the recognition that the existing approaches, while some were found to be very sophisticated, suffer from serious disadvantages, tending to be too complex to be implemented, too difficult to handle, too inflexible in the facilities provided, or too consumptive of system resources.

The AUI-ML, which was developed as a consequence, is sufficiently powerful, easy to learn, lean, and platform independent. A Java Servlet has been implemented, which can render AUI-ML based user interface descriptions into HTML, cHTML, and WML, by the provision of appropriate XSL stylesheets.

An issue, which needs to be addressed in the future, is the improvement of the performance of the Servlet, in particular the developed XSL stylesheets could benefit of some optimisation aimed toward generating less overhead for the SAX parser and the XSL engine. Another idea is the replacement of the used XML parser and XSL engine with ones, which are faster, but still have a small footprint. NanoXML would be a potential candidate [NanoXML]. Furthermore, a better caching mechanism needs to be implemented, which can reduce the amount of rendering necessary with respect to often-used user interfaces.

9.6 Security

For the secure identification and authentication of persons, an Office Identity Card (OIC) has been developed. It provides conventional identity card functions, i.e. the identity card is verifiable both optically as well as electronically and therefore serves as a means to ensure the authenticity of the cardholder. Supplementary applications, which have been developed, are admission control, single sign-on, library identity card, and privacy enhanced time recording. In addition, it can also securely store the personal profiles and preferences of its cardholder in order to protect his privacy.

Based on the OIC an extended role based access control model has been developed. This model has some limitations, as it only allows for differentiation based on organisational structures. What is needed is a more powerful approach, which can additionally handle constraints (e.g. time and/or location) and extend to the concept of work contexts. In a more general approach, access rights should be specified as sets of rules, whereas the concept of roles can be modelled by the definition of entity sets and classes.

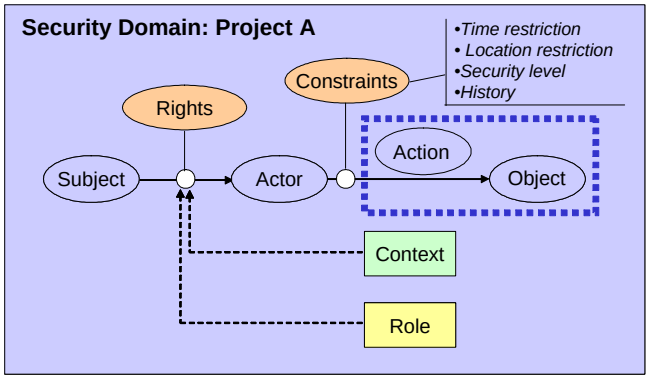


Figure 59 Extended access control model

The combination of the OIC together with a Public Key Infrastructure (PKI) has been established as the foundation upon which user identification is built throughout the Co-operation Platform. However, the application of PKIs is not making much headway in today's organisations. PKIs are often expensive, inflexible and difficult to handle. This is due to some system-inherent problems of contemporary PKI-structures. The validation of digital signatures and certificates or the retrieval of keys is complex, if PKIs are not strongly hierarchically structured, but more a kind of meshed network, and therefore, the application of PKIs is restricted to closed or simply structured environments.

In contrast, work structures are increasingly becoming project and team-oriented, with fluctuating member sets, these members sometimes being geographically dispersed and often originating from different organisations. This asks for the development of a new generation of highly flexible, scalable and context-oriented security infrastructures, which allow the protection of physical as well as information objects.

With respect to confidentiality and integrity, a special focus of this thesis was on protecting multimedia conferencing sessions, in particular video conferencing, between two or more participants. Therefore, a scalable secure conferencing gateway has been developed, using a special partial encryption method for video data streams. What is needed here in the future is a somewhat more advanced solution, creating less overhead with respect to the required co-ordination when session keys need to be changed, because participants have left the session.

The VPO provides for the definition of security policies, which are specific to the particular project and team. This thesis pursues an approach based on XML and based on that, an access control manager for the VPO has been designed.

What is required in the future with this respect is the development of a more advanced model supporting the creation, management, and enforcement of security policies in geographically dispersed teams. This model needs to result in a more powerful framework, supporting the flexible definition and enforcement of security policies, and the detection of conflicting policies, particularly when team members originate from different organisations and are therefore subject to potentially conflicting security policies.

In the case that conflicts are detected, it would be desirable that the framework be sufficiently extensive that it includes automated conflict resolution strategies and mechanisms. One aspect of this framework could be that security policies are specified using an XML schema, which would facilitate their exchange between different organisations and provide a basis for the detection and resolution of conflicts.

9.7 Scientific Evaluation

Planned future work includes, in addition to the above-mentioned points, the evaluation of the Co-operation Platform in a user community under observation of social industrial scientists in order to gather more feedback from the users in real work situations. This evaluation is expected to have a positive impact towards the further development and to deliver useful programmatic insights towards the directions to be taken while at the same time providing a scientific foundation for further efforts.

The concept of formative and summative evaluation will be used for this evaluation, as the terrain is occupied by different stakeholders with different interests. Usability evaluation will be complemented with an Extended Economic Efficiency Evaluation (E^4) method, as this focuses also on the costs and benefits incurred by usage of the Co-operation Platform.

E^4 is a combination of two methods, the utility value analysis and the work system evaluation, and is based on [FN88] and [A85]. This aims at providing an overall view of the costs and of the (monetary as well as non-monetary) benefits involved in implementing and using the Co-operation Platform. Among other things, this involves a comparison of work processes supported by the Co-operation Platform, as well as without.

Bibliography

[A85] Auch, M.: Menschengerechte Arbeitsplätze sind wirtschaftlich. Wirtschaftlichkeitsvergleich und Arbeitssystemwertermittlung - ein erweitertes Bewertungsverfahren. Eschborn, RKW, 1985

[AAH+97] Abowd, G. D.; Atkeson, C. G.; Hong, J., Long; S., Kooper, R.; Pinkerton, M.: Cyberguide: a mobile context-aware tour guide. In: ACM Wireless Networks, Vol. 3, No. 5, 1997, pp. 421-433

[AB96] Appelt, W.; Busbach, U.,: The BSCW System: A WWW-based Application to Support Cooperation of Distributed Groups. In: Proceedings of the 5th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'96), IEEE Computer Society Press, Stanford, CA, June 1996, pp. 304-309

[ABF+96] André, St.; Berger, A.; Faltin, U.; Dückminor, J.; Giehl, H.; Glöckner, P.; Hennecke, S.; Hetschold, Th.; Hühnlein, D.; Nüsseler, M.; Schneider, W.; Schupp, H.; Schwarz, O.; Surkau, Th.; Viebeg, U.: SECUDE-5.0 Documentation. Darmstadt, October 1996, http://www.secude.de

[ADG+01] Ralf Ackermann, Vasilios Darlagiannis, Manuel Goertz, Martin Karsten, and Ralf Steinmetz. An Open Source H.323-SIP Gateway as Basis for Supplementary Service Interworking. In: Proceedings of the 2nd IP Telephony Workshop, New York, 2001, pp. 169-175, April 2001.

[AHRI] Future Computing Environments (FCE) Group, Georgia Tech: Aware Home Research Initiative (AHRI). Future Computing Environments (FCE) Group, Georgia Tech, http://www.cc.gatech.edu/fce/house/house.html , Oct 28, 2001

[And94] Anderson, R.J.: Representations and requirements: the value of ethnography in system design. In: Human Computer Interaction, Vol. 9, No. 1, 1994, pp. 151-182

[AOLMes] AOL Instant Messenger, http://www.aol.com/aim/homenew.adp

[APWS99] Ralf Ackermann, Jörg Pommnitz, Lars Wolf, and Ralf Steinmetz. MBone2Tel - ein Gateway für die Teilnahme von Nutzern konventioneller Telefonendgeräte an MBone-Konferenzen. In: ITG/TKTG-Fachtagung Multimedia: Anwendungen, Technologie, Systeme, 8. Dortmunder Fernsehseminar, Dortmund, pp. 237-240, September 1999.

[BBH+99] Bahr, K.; Burkhardt, H.-J.; Hovestadt, L.; Reinema, R.: Integrating Virtual and Real Work Environements. In: Proceedings of IEEE Conference on Software in Telecommunications and Computer Networks (SoftCOM '99), IEEE, Split, Rijeka, Croatia, Trieste, Venice, Italy, October 1999

[BC01] Bieber, G.; Carpenter, J.: Introduction to Service-Oriented Programming, Whitepaper, Motorola ISD, April 2001, http://www.openwings.org/download/specs/ServiceOriented-Introduction.pdf , Oct 28, 2001

[BEK+00] Box, D.; Ehnebuske, D.; Kakivaya, G.; Layman, A.; Mendelsohn, N.; Nielsen, H. F.; Thatte, S.; Winer, D.: Simple Object Access Protocol (SOAP), World Wide Web Consortium (W3C), May 2000, http://www.w3.org/TR/2000/NOTE-SOAP-20000508 , Oct 28, 2001

[Bel94] Bell, D. E.: Modeling the Multipolicy Machine. In: Proceedings of the 1994 ACM SIGSAC on New security paradigms workshop, IEEE Computer Society Press Los Alamitos, CA, USA, 2-9. Aug. 1994, pp. 2-9

[BHJ+97] Bahr, K.; Hinsch, E.; Jaegemann, A.; Wang, Lan: Internet and Handling Confidential Internet Conferences by E-mail. In: Proceedings of 8th Joint European Networking Conference (JENC8), Edinburgh, Scotland, 12-15 May 1997

[BHS94] Bahr, K.; Hinsch, E.; Schulze G.: Incorporating Security Functions in Multimedia Conferencing Applications in the Context of the MICE Project. In: Proceedings of the 2nd International Workshop IWACA'94, Heidelberg, Germany, 1994

[BHR+97] Baumann, J.; Hohl, F.; Radouniklis, N.; et al.: Communication Concepts for Mobile Agent Systems. In: Rothermel, K.; Popescu-Zeletin, R. (Eds.): Mobile Agents – First International Workshop, Springer-Verlag, Berlin, Germany, LNCS 1219, 1997

[Bib77] Biba, K.J.: Integrity Considerations for Secure Computer Systems. Technical Report MTR – 3153, NTS AD A039324, MITRE Corp, Bedford, April 1997

[BIDSAVER] Business Integrator Dynamic Support Agents for Virtual Enterprise. EU IST-1999-10768, http://www.easist.sk/projects/BIDSAVER.shtml

[BG98] Barkley, J.; Gavrila, S.: Formal Specification for Role Based Access Control User/Role and Role/Role Relationship Management. In: Proceedings of third ACM Workshop on Role-Based Access Control, ACM, Fairfax, Virginia, 1998, see also: http://csrc.nist.gov/rbac

[BJN+98] Bohrer, K.; Johnson, V.; Nilsson, A.; Rubin, B.: Business Process Components for Distributed Object Applications. In: Communications of the ACM, Vol. 41, No. 6, June 1998, pp 43-48

[BK98] Brown, N.; Kindel, C., Distributed Component Object Model Protocol - DCOM/1.0. Internet Draft, IETF, Jan 1998, http://www.globecom.net/-ietf/draft/draft-brown-dcom-v1-spec-03.html

[Bla97] Blaszczak, M.: Professional MFC with Visual C++ 5. Wrox Press, May 1997, ISBN 1861000146

[BL73] Bell, D. E.; La Padula, L. J.: Secure Computer Systems: Mathematical Foundations and Model. Technical Report M74-244, Vol. 2, The MITRE Corporation, Bedford, Massachussetts, May 1973

[Blu01] Bluetooth SIG, Specification of the Bluetooth System - Core, February 2001, http://www.bluetooth.com

[Blu01e] Bluetooth SIG, Specification of the Bluetooth System, Part E, Service Discovery Protocol (SDP). February 2001, http://www.bluetooth.com/-files/Bluetooth_11_Specifications_Book.pdf

[BM98] Barrett, E.; Maglio, P.: Informative Things: How to Attach Information to the Real World. In: Proceedings of UIST ’98, 1998, pp. 81-88

[BMW00] BMW navigation system, 2000, http://www.netten.net/ROADSHOW/-nav.htm

[BN89] Brewer, D.; Nash, M.: The Chinese Wall Security Policy. In: Proceedings of the IEEE Symposium on Security and Privacy, Los Alamitos, CA, 1989, pp. 206-214

[BOP89] Burkhardt, H.J.; Ochsenschläger, P.; Prinoth, R.: Product Nets - A Formal Description Technique for Cooperating Systems. GMD-Studien, No. 165, September 1989, ISBN 3-88457-165-6, 1989

[BPR99] Bellifemine, F.; Poggi, A.; Rimassa, G.: Jade - a pa-compliant agent framework. In: Proceedings of PAAM'99, London, April 1999, pp. 97-108

[BRJ98] Booch, G.; Rumbaugh, J.; Jacobson, I.: The Unified Modelling Language User Guide. Addison-Wesley, 1998

[BRK97] Blecher, Th.; Reinema, R.; Kunkelmann, Th.: Kryptographieverfahren für Videoübertragung. GMD-Studie Nr. 319, GMD, Sankt Augustin, 1997, ISBN 3-88457-319-5

[BSC+97] Benford, S.; Snowdon, D.; Colebourne, A.; O'Brien, J.; Rodden, T.: Informing the Design of Collaborative Virtual Environments. In: Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work (Group'97), ACM Press, New York, NY, USA, Phoenix, Arizona USA, November 1997, ISBN 0-89791-897-5, pp. 71 - 80

[CCM+01] Christensen, E.; Curbera, F.; Meredith, G.; Weerawarana, S.: Web Services Description Language (WSDL) 1.1. World Wide Web Consortium, March 2001, http://www.w3.org/TR/wsdl , Oct 28, 2001

[Cha97] Chauhan, Deepika: JAFMAS: A Java-based Agent Framework for Multiagent Systems Development and Implementation. Thesis, http://www.ececs.uc.edu/~abaker/JAFMAS/JAFMAS.zip

[CHS+01] Clemm, G.; Hopkins, A.; Sedlar, E.; Whitehead, J.: WebDAV Access Control Protocol - Access Control Extensions to WebDAV. Jun 2001, http://www.webdav.org/acl/protocol/draft-ietf-webdav-acl-06.htm , June 21, 2001

[CGJP98] Chabert, A.; Grossman, E.; Jackson, L.; Pietrovicz, S.: NCSA Habanero - Synchronous collaborative framework and environment. http://havefun.ncsa.uiuc.edu/habanero/Whitepapers/index.html, 1998

[Coe98] Coen, Micheal H.: A Prototype Intelligent Environment. In: Cooperative Buildings: Integrating Information, Organization, and Architecture. Streitz, N. A., Konomi, S., Burkhardt, H.-J. (Eds.), Springer Verlag, LNCS 1370, 1998, pp. 41-52

[COM95] Microsoft Corporation: The Component Object Model Specification. 1995, http://www.microsoft.com/com

[Cooltown] Hewlett-Packard Cooltown Project: http://www.cooltown.com/

[Cookies] Netscape, Cookie Specification. http://home.netscape.com/newsref/-std/cookie_spec.html

[CoopWWW] Project CoopWWW. http://orgwis.gmd.de/projects/COOPWWW/ , Oct 28, 2001

[COVEN] COVEN: COllaborative Virtual Environments. http://www.crg.cs.nott.ac.uk/research/projects/Coven/

[CR87] Carroll, J.M.; Rosson, M.B.: The paradox of the active user. In: J.M. Carroll (Eds.), Interfacing Thought: Cognitive Aspects of Human-Computer Interaction. MIT Press, Cambridge, MA, 1987, http://www.cs.vt.edu/~rosson/papers/paradox.pdf

[CREANET] Creatives rights European Agency Network (CREA Net). EU IST-1999-10871, http://www.creativesrights.com/home.htm

[Crimson] Crimson XML parser, available from http://xml.apache.org

[CUSeeMe] First Virtual Communications: CUSeeMe, http://www.cuseeme.com/ , Oct 28, 2001

[CW87] Clark, D. D.; Wilson, D. R.: A Comparison of Commercial and Military Computer Security Policies. In: Proceedings of the IEEE Symposium on Security and Privacy, IEEE Computer Society Press, Los Alamitos, CA, 1987, pp. 184-194

[DAB+94] Dourish, P.; Adler, A.; Bellotti, V.; Henderson, A.: Your place or mine? Learning from long-term use of video communication. Technical Report EPC-1994-105, Rank Xerox Research Centre, Cambridge, 1994, http://www.xrce.xerox.com/publis/cam-trs/html/epc-1994-105.htm

[Dan01] Dana, P.H.: Global Positioning System Overview. Dptm. of Geography, Univ. of Texas at Austin, 2001, http://www.colorado.edu/geography/-gcraft/notes/gps/gps.html , Oct 28, 2001

[DCE98] Object Management Group, DCE Today. ISBN 1-85912-157-8, July 1998

[DEN+99] Duff, C.; Espey, J.; Neuf, H.; Rudinger, G.; Stapf, K.: Usability and Security. In: Müller, G., Rannenberg, K. (Eds.): Multilateral Security in Communications, Vol. 3, Addison-Wesley, 1999

[DH00] Doraswamy, N.; Harkins, D., IPSec: The New Security Standard for the Internet, Intranets, and Virtual Private Networks. Prentice Hall PTR, ISBN 0-13-011898-2, 2000

[DIN174] DIN NI 17.4, Specification of Interfaces for Smartcards with Digital Signature Functions (in German), 1998

[DM92] Davidow, W.; Malone, M.S., The Virtual Corporation – structuring and revitalizing the corporation for the 21st century. Harperbusiness, New York, 1992, ISBN 0887306578

[Dro99] Droms, Ralph: Automated Configuration of TCP/IP with DHCP. In: IEEE Internet Computing, Vol. 3, No. 4, July 1999, pp. 45-53

[DSA98] Dey, A. K.; Salber, D.; Abowd, G. D.: The Conference Assistant: Combining Context-Awareness with Wearable Computing. In: Proceedings of the 3rd International Symposium on Wearable Computers, 1998

[DYNOCA] A System to Realise Dynamic Networked Organisations on Heterogeneous Networks in the Consultancy/Agency Sector. EU IST-1999-11065, http://www2.dynoca.net/

[Eck01] Eckert, C.: IT-Sicherheit: Konzepte – Verfahren – Protokolle. Oldenbourg Verlag, München, 2001, ISBN 3-486-25298-4

[eColleg] E-Colleg - Advanced Infrastructure for Pan-European Collaborative Engineering. EU IST-1999-11746, http://alfa.iele.polsl.gliwice.pl/-~pawlak/E-Colleg/E-Colleg-index.htm

[Edw99] Edwards, W. Keith: Core Jini. Prentice Hall PTR, 1999, ISBN 0-13-0114469-X

[EKFS99] Abdulmotaleb El Saddik, Oguzhan Karaduman, Stephan Fischer, and Ralf Steinmetz. Collaborative Working with Stand-Alone Applets. In: Proceedings of the 12th International Symposium on Intelligent Multimedia and Distance Education(ISIMADE'99), Baden-Baden, Germany, pp. 203-209, August 1999. ISBN 0-921836-80-5.

[EKK97] Engel, A.; Kaack, H.; Kaiser, S.: Teamarbeitsräume zur Unterstützung verhandlungsorientierter Telekooperation. In: Proceedings of the Workshop „Rechnergestützte Kooperation in Verwaltungen und großen Unternehmen“, GI Jahrestagung, Aachen, 1997

[Eng97] Englander, R.: Developing Java Beans. O'Reilly, ISBN 15659228911997

[Eri94] Eriksson, H.: The Multicast Backbone. In: Communications of the ACM, 37(8), pp. 54-60, 1994

[eRoom] eRoom Technology, Inc.: eRoom - the digital workplace. http://www.instinctive.com , Oct 28, 2001

[eSCAPE] eSCAPE - Electronic landSCAPEs. http://escape.lancs.ac.uk/

[ESGS00] Abdulmotaleb El Saddik, Shervin Shirmohammadi, Nicolas D. Georganas, and Ralf Steinmetz. JASMINE: Java Application Sharing in Multiuser INteractive Environments. In: Proceedings of International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services 2000 (IDMS 2000), Enschede, Netherlands, pp. 214-226. Springer, October 2000.

[EW94] Etzioni, Oren; Weld, Daniel P., A softbot-based interface to the Internet, Communications of the ACM, Vol. 37, No. 7, pp. 72-76,

[Fai98] Faisst, W.: Die Unterstützung virtueller Unternehmen durch Informations- und Kommunikationssysteme – eine lebenszyklusorientierte Analyse. Dissertation, Friedrich-Alexander-Universität Erlangen-Nürnberg, 1998

[Fin00] Finkenzeller, K.: RFID-Handbuch. Grundlagen und praktische Anwendungen induktiver Funkanlagen, Transponder und kontaktloser Chipkarten. Hanser Fachbuchverlag, ISBN 3446212787, 2000

[FIP00] Foundation for Intelligent Physical Agents (FIPA): FIPA ACL Message Structure Specification.FIPA Document Number XC00061D, 2000, http://www.fipa.org/specs/fipa00061/

[FK92] Ferraiolo, D.; Kuhn, R.: Role-Based Access Control. In: Proceedings of 15th National Computer Security Conference, 1992

[FKK96] Freier, A. O.; Karlton, P.; Kocher, P. C.: The SSL Protocol Version 3.0. Internet Draft, Netscape Communications Corp., March 1996, http://home.netscape.com/eng/ssl3/ssl-toc.html , Oct. 28, 2001

[Fla94] Flanagan, D.: Motif Tools: Streamlined GUI Design and Programming. 1st Edition, O'Reilly & Associates, August 1994, ISBN 1-56592-044-9

[FN88] Fröschle, H.-P.; Niemeier, J.: Assessment of benefits of information technology in the office: A concept for economic valuation and case study. In: Proceedings of the First European Conference on Information Technology for Organisational Systems (Eurinfo '88), Protonatorios, E.N.; Bouwhuis, D.; Reim, F.; Bullinger, H.-J. (Eds.), 1988, Amsterdam: North-Holland Publishing, pp. 190-197

[FP97] Federrath, H.; Pfitzmann, A.: Bausteine zur Realisierung mehrseitiger Sicherheit. In: Mehrseitige Sicherheit in der Kommunikationstechnik. G. Müller, A. Pfitzmann (Eds.), Addison-Wesley, 1997

[FRR00] Fischer, S.; Rensing, C.; Rödig, U.: Open Internet Security: von den Grundlagen zu den Anwendungen. Springer Verlag, Reihe Xpert.press, Berlin, 2000, ISBN 3-540-66814-4

[FS98] Frécon, E.; Stenius, M.: DIVE: A scaleable network architecture for distributed virtual environments. In: Distributed Systems Engineering Journal, Vol. 5, No. 3, Sept 1998, pp. 91-100, http://www.sics.se/dce/dive/dive.html

[FW91] Finin, T.; Wiederholt, G.: An overview on KQML. Stanford University Logic Group, 1991, see also http://www.cs.umbc.edu/kqml/papers/

[GH96] Gehrke, M.; Hetschold, T.: Management of a public key certification infrastructure - Experiences from the DeTeBerkom project BMSec. In: Computer Networks and ISDN Systems 28, Elsevier, 1996, pp. 1901-1914

[GJ97] Grimm, R.; Johnstone, S.: End-to-End Security on the Internet. In: Datenschutz und Datensicherheit (DuD), Rubrik Forum, p.417

[GK94] Genesereth, M.; Ketchpel, S.: Software Agents. In: Communications of the ACM, Volume 37, No. 7, July 1994, pp. 48-53

[GMM+91] Gaver, W.; Moran, T. P.; MacLean, A.; Lovstrand, L.; Dourish, P.; Carter, K.; Buxton, W.: Working together in media space: CSCW research at EuroPARC. In: Computer-supported cooperative work: the multimedia and networking paradigm, Avebury Technical, London, 1991, pp. 185-205

[Gol99] Gollman, D.: Computer Security. Wiley & Sons Ltd, New York, ISBN 0471978442 , 1999

[GPS00] Global Positioning Systems. http://www.stanford.edu/group/GPS/ ,

[GQ96] Gong, L.; Qian, X.: Computational issue in secure interoperation. In: IEEE Transactions on Software Engineering, Vol.. 22, No.1, Jan. 1996, pp. 43-52,

[GR98] Greenberg, S.; Roseman, M.: Using a Room Metaphor to Ease Transitions in Groupware. Research report 98/611/02, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada, January 1998, http://pharos.cpsc.ucalgary.ca/Dienst/UI/2.0/Describe/-ncstrl.ucalgary_cs/1998-611-02

[Gra92] Grady, Robert B.: Practical Software Metrics for Project Management and Process Improvement. Prentice-Hall, Inc., New York, ISBN 0-13-720384-5, 1992

[Gra95] Gray., R. S.; Agent Tcl: A transportable agent system. In: Proceedings of the CIKM Workshop on Intelligent Information Agents, Baltimore, MD, December 1995

[Gri93] Grimm, R.: Non-Repudiation on Open Telecooperation. In: Proceedings of the 16th National Computer Security Conference, Baltimore, 1993

[Gri97a] Grimm, R.: Sicherheit für offene Kommunikationsnetze. In: Mehrseitige Sicherheit in der Kommunikationstechnik, G. Müller, A. Pfitzmann (Eds.), Addison-Wesley, 1997

[Gri97b] Grimm, R.: European Research Projects on Electronic Commerce and Open Networks Security. ERCIM News, No. 30, European Research Consortium on Informatics and Mathematics, July 1997, http://www-ercim.inria.fr/publication/Ercim_News/enw30/

[Gru88] Grudin, J.: Why CSCW applications fail: problems in the design and evaluation of organizational interfaces. In: Proceedings of CSCW '88, ACM, Portland, Oregon, 1988, pp. 85-93

[Gry00] Gryazin, Eugene A.: Service Discovery in Bluetooth, Group for Robotics and Virtual Reality, Helsinki University of Technology, 2000, http://www.cs.hut.fi/Opinnot/Tik-86.174/SD_in_Bluetooth.pdf , Oct 28, 2001

[Gut99] Guttman, Erik: Service Location Protocol: Automatic Discovery of IP Network Services. In: IEEE Internet Computing, Vol. 3, No. 4, July 1999, pp. 71-80

[GW91] Gurbaxani, V.; Whang, S.: The Impact of Information Systems on Organizations and Markets. In: Communications of the ACM, Vol. 34, No. 1, 1991, pp. 59-73

[GZ96] Grimm, R.; Zangeneh, K.: Cybermoney in the Internet: An Overview over new Payment Systems in the Internet. In: Horster, P. (Eds.): Communications and Multimedia Security II, IFIP, Chapman & Hall, London, 1996, pp. 183-195

[HAVi98] The HAVi architecture - Specification of the Home Audio/Video Interoperability (HAVi) Architecture, HAVi Consortium,1998

[HCF97] Hamilton, G.; Cattell, R. G. G; Fisher, M.: JDBC Database Access With Java: A Tutorial and Annotated Reference. Addison Wesley Longman, Inc., Java Series, 1997, ISBN 0201309955

[HCK95] Harrison, C.; Chess, D.; Kershenbaum, A.: Mobile Agents: Are they a good idea?", IBM Research Division, IBM Corporation, March 1995

[Her97] Herfert, M.: Security Enhanced Mailing Lists. In: IEEE Network - Network and Internet Security, Vol. 11, No. 3, May/June 1997, pp. 30-33

[Her98] Herfert, M.: Key Escrow and Message Recovery. In: S. Rill, M. Graf-Jahnke (Editors): Library on Congress Proceedings, Verlag Hochschule Bremen, Vol. 1, Bremen, June 1998, pp. 116-140

[HJR+96] Hinsch, E.; Jägemann, A.; Roper, I. C.; Wang, L.: The Secure Conferencing User Agent : A Tool to Provide Secure Conferencing with MBONE. Multimedia Conferencing Applications. In: Proceedings of IDMS '96, Springer-Verlag, Berlin, LNCS 1045, March 1996, ISBN 3-540-60938-5, pp. 131-142

[HJW96] Hinsch, E.; Jägemann, A.; Wang, L.: Experience with the Secure Conferencing User Agent: A Tool to Provide Secure. Conferencing with MBONE Multimedia Conferencing Applications. In: Proceedings of 7th Joint European Networking Conference (JENC7), Budapest, May 1996

[Huc95] Huchins, Ed: Cognition in the wild. MIT Press, 1995, ISBN 0262082314

[HZ00] Herrmann, K.; Zapf, M.: AMETAS. White Paper Series, July 2000, http://www.vsb.cs.uni-frankfurt.de/ametas/docs/white/AllWhite.pdf , Oct 28, 2001

[IAIK] IAIK-Java Group: IAIK-JCE Toolkit. Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Graz, http://jcewww.iaik.at/products/jce

[Ian01] Iannella, R.: Representing vCard Objects in RDF/XML. IPR Systems, 2001, Feb 2001, http://www.w3.org/TR/vcard-rdf, Oct. 28, 2001

[I-Code] Philips Semiconductor: I-Code – Technology for SmartLabels. Online: http://www.semiconductors.philips.com/acrobat/other/identification/i-code.pdf see also http://www.semiconductors.philips.com/identification/products/icode/

[ICQ] ICQ Homepage, http://web.icq.com/

[IEEE-610.12-1990] IEEE STD 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology

[IEEE-1394-1995] IEEE Std 1394-1995, Standard for a High Performance Serial Bus

[IETF] Internet Engineering Task Force (IETF), http://www.ietf.org

[IKV01] IKV++ Technologies AG: Grasshopper - The Agent Plattform. http://www.grasshopper.de/ , Oct 28, 2001

[iLand] i-LAND - An Interactive Landscape for Creativity and Iinnovation, http://www.darmstadt.gmd.de/ambiente/i-land.html

[IMC] Internet Mail Consortium: vCard, http://www.imc.org/pdi/ , See also RFC2425, RFC2426 (available at http://www.ietf.org/rfc/)

[ISO7498-2] OSI, OSI Reference Model - Part 2 : Security Architecture. ISO Information Processing Systems, ISO 7498-2, ISO, Geneva, Switzerland, 1988

[ISO7816-4] :ISO/IEC 7816 - Integrated circuits cards with contacts, Part 4: Interindustry command for interchange, 1st edition, ISO/IEC, Geneva, Switzerland, 1995

[ITSEC91] Commission of the European Communities: Information Technology Security Evaluation Criteria (ITSEC), 1991, http://www.iwar.org.uk/-comsec/resources/standards/itsec.htm

[ITU-H320] ITU-T Recommendation H.320 - Narrow-band visual telephone systems and terminal equipment. ITU, Geneva, 1996, http://www.itu.ch/itudoc/itu-t/rec/h/h320_23397.html

[ITU-H323] ITU-T Recommendation H.323: Visual telephone systems and equipment for local area networks which provide a non-guaranteed quality of service, ITU, Geneva, 1996, http://www.itu.ch/itudoc/itu-t/rec/h/h323_39297.html

[ITU-T.120] ITU-T, Recommendation T.120: Data protocols for multimedia conferencing. 1996, http://www.itu.int/rec/-recommendation.asp?type=folders&lang=e&parent=T-REC-T.120 , October 28, 2001

[Jar01] The on-line hacker jargon file, version 4.3.0, 2001-06-09, http://www.tuxedo.org/jargon/

[JCA97] JavaTM Cryptography Architecture - API Specification & Reference. Sun Microsystems, December 1997, http://java.sun.com/products/jdk/1.1/docs/guide/security/-CryptoSpec.html

[JCE00] JavaTM Cryptography Extension - API Specification & Reference. Sun Microsystems, June 2000, http://java.sun.com/products/jce/doc/guide/API_users_guide.html

[JDBC97] Hamilton, G.; Cattell, R. G. G.; Fisher, M.: JDBC Database Access With Java: A Tutorial and Annotated Reference (Java Series). Addison-Wesley, 1997

[Jin00a] Sun Microsystems: Jini Architecture Specification. Sun Microsystems, Palo Alto, California, October 2000, http://www.sun.com/jini/specs/jini1_1.pdf

[Jin00b] Sun Microsystems: Jini Technology Core Platform Specification. Sun Microsystems, Palo Alto, California, October 2000, http://www.sun.com/jini/specs/core1_1.pdf

[Jini99] Sun Microsystems: Why JINI Technology Now?, Sun Microsystems, Palo Alto, California, January 1999, http://www.sun.com/jini/-whitepapers/whyjininow.html

[Jini01] Jini Connectivity of Home and Building Networks, Department of Automation, Vienna University of Technology, http://www.auto.tuwien.ac.at/Projects/Jini/ , Oct 28, 2001

[K98] Kunkelmann, Th.: Sicherheit für Videodaten, Reihe Multimedia Engineering, Effelsberg, W.; Steinmetz, R. (Hrsg.), Vieweg Verlag, Braunschweig/Wiesbaden, 1998, ISBN 3-528-05680-0

[KLO97] Karjoth, G.; Lange, D.; Oshima, M.: A Security Model for Aglets, IEEE Internet Computing, Vol. 1, No. 4, 1997, pp. 68-77

[Kol96] Kolletzki, Stephan: Secure Internet banking with Privacy Enhanced Mail - A protocol for reliable exchange of secured order forms, (BAKO). In: Computer Networks and ISDN Systems 28, Elsevier Science Publishers B.V. (North-Holland), 1996, pp. 1891-1899

[KM96] Krause, S.; Magedantz, T.: Mobile Service Agents Enabling Intelligence on Demand in Telecommunications. In: Proceedings of IEEE GLOBECOM '96, London, U.K., Nov. 1996, pp. 18-22.

[Knu98] Knudsen, Jonathan: JAVA Cryptography, First edition, O'Reilly, 1998, ISBN: 1565924029

[KR97a] Kunkelmann, T.; Reinema, R.: A Scalable Security Architecture for Multimedia Communication Standards. In: Proceedings of the 4th IEEE International Conference on Multimedia Computing and Systems (ICMS97), Ottawa, Canada, pp. 660-661, IEEE Computer Society, 1997, ISBN 0-8186-7819-4

[KR97b] Kunkelmann, T.; Reinema, R.: A Scaleable Security Gateway for Distributed Multimedia Communication Tools. In: Proceedings of the IFIP WG 6.1 International Working Conference on Distributed Applications and Interoperable Systems (DAIS'97), Cottbus, Germany, pp. 89-94, Chapmann & Hall, 1997, ISBN 0-412-82340-3.

[KRSB97] Kunkelmann, T.; Reinema, R.; Steinmetz, R; Blecher, Th.: Evaluation of Different Video Encryption Methods for a Secure Multimedia Conferencing Gateway. In: Proceedings of the 4th COST 237 Workshop, Lisboa, Portugal, pp. 75-89, Springer Verlag, Heidelberg, 1997, LNCS 1356, ISBN 3-540-63935-7

[KS93] Katzenbach, J. R.; Smith, D. K, The Wisdom of Teams: Creating the High-Performance Organisation. Harper Business, ISBN 0887306764, 1993

[LEAP] Lightweight Extensible Agent Platform. EU IST-1999-10211, http://leap.crm-paris.com/index.shtml

[LHOP] VA Linux Systems: The Linux Home Automation Project. http://linuxha.sourceforge.net

[LO98] Lange, D.; Oshima, M.: Programming and Deploying Java Mobile Agents with Aglets. Addison-Wesley, 1998

[LRH00] Ljungstrand, P.; Redström, J.; Holmquist, L.E.: WebStickers - Using Physical Tokens to Access, Manage and Share Bookmarks to the Web. In: Proceedings of DARE 2000 (Designing Augmented Reality Environments), Elsinore, Denmark, 2000

[LS97] Lipnack, J.; Stamps, J.: Virtual Teams: Reaching Across Space, Time and Organizations with Technology. Wiley & Sons, 1997

[LS00] Lee, R., Seligman, S.: JNDI API Tutorial and Reference: Building Directory-Enabled Java Applications. Addison Wesley Longman, 2000, ISBN 0201705028, see also http://java.sun.com/products/jndi/tutorial/

[Luc97] Lucent Technologies: Lucent Technologies Multipoint Conferencing Unit (MCU). Release 4.3 Documentation, Lucent Technologies, October 1997

[Luc98] Lucente, M.: Visualization Space: A Testbed for Deviceless Multimodal User Interface. In: Proceedings of Intelligent Environment 98, AAAI Spring Symposium Series, San Francisco, CA, AAAI Spring Symposium Series, March 23-25, 1998, pp. 87-92

[MA01] Moyer, M. J.; Ahamad, M.: Generalized Role-Based Access Control. In: Proceedings of the 21st IEEE International Conference on Distributed Computing Systems (ICDCS), 2001, p. 391-398

[Mae94] Maes, P.: Agents that reduce work and information overload. In: Communications of the ACM, Vol. 37, No. 7, July 1994, pp. 31-40

[MAM99] van der Meer, Sven; Arbanowski, Stefan; Magedanz, Thomas: An Approach for a 4th Generation Messaging System. In: Proceedings of The Fourth International Symposium on Autonomous Decentralized Systems (ISADS'99), Tokyo, March 21-23 1999

[MBB+98] Milojicic, D.; Breugst, M.; Busse, I. et al.: MASIF – The OMG Mobile Agent System Interoperability Facility. In: Mobile Agents – Second International Workshop MA'98, Springer-Verlag, Stuttgart, Germany, 1998

[Mei01] Meier, C.: Ethnografie. In: CSCW-Kompendium - Lehr- und Handbuch zum computerunterstützten kooperativen Arbeiten. Springer Verlag, Schwabe, Gerhard, Streitz, Norbert, Unland, Rainer (Eds.), Berlin, 2001, ISBN 3-540-67552-3

[MGR+00] Minar, N.; Gray, M.; Roup, O.; Krikorian, E.; Maes, P.: Hive: Distributed Agents for Networking Things. In: Proceedings of ASA/MA '99, IEEE Press, Palm Springs, CA, 2000, pp. 118-129

[MH97] McGraw, Karen L.; Harbison, Karan: User-centered requirements: The scenario-based engineering process, Lawrence Erlbaum Associates, ISBN 0805820655, 1997

[Min96] Minden, R.M.: IP next generation overview. Communications of the ACM, 39(6), pp. 60-71, June 1996

[Mon00] Monson-Haefel, Richard: Enterprise JavaBeans, 2nd Edition, O'Reilly, UK, ISBN 1565928695, March 2000

[MOTION] Mobile Teamwork Infrastructure for Organisations Networking. EU IST-1999-11400, http://www.motion.softeco.it

[MP97] Müller, G.; Pfitzmann, A.: Mehrseitige Sicherheit in der Kommunikationstechnik. Vol. 1, Addison-Wesley-Longman, 1997

[MP99] Miller, Brent; Pascoe, Robert: Mapping Salutation Architecture APIs to Bluetooth Service Discovery Layer. Doc. No. 1.C.118/1.0, White Paper, Bluetooth Special Interest Group, July 1999, http://www.bluetooth.com

[MR99] Müller, G.; Rannenberg, K.: Multilateral Security in Communications. Vol. 3, Addison-Wesley-Longman, 1999

[MRH00] Moschgath, M.-L.; Reinema, R.; Hoffmann, M.: A Security Infrastructure for the Virtual Project Office. In: Proceedings of the IEEE International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2000), Split, Croatia, October 2000

[MRH01] Moschgath, M.-L.; Reinema, R.; Hähner, J.: Sm@rtLibrary - An Infrastructure for Ubiquitous Technologies and Applications. In: Proceedings of 21st International Conference on Distributed Computing Systems Workshops (ICDCS 2001), International Workshop on Smart Appliances and Wearable Computing (IWSAWC), Mesa, Arizona, IEEE Computer Society, ISBN 0-7695-1080-9, April 2001, pp. 208-213

[MRK96] Magedanz, T.; Rothermel, K.; Krause, S.: Intelligent Agents: An Emerging Technology for Next Generation Telecommunications?, IEEE INFOCOM 1996, San Francisco, USA, March 24-28, 1996

[MS-ODBC97] Microsoft Corp.: Microsoft ODBC 3.0 Software Development Kit and Programmer's Reference. Microsoft Press, 1997

[MS96] Microsoft Corporation, Microsoft Authenticode. White Paper, October 1996, http://msdn.microsoft.com/workshop/security/-authcode/authwp.asp

[MSMes] Microsoft MSN Messenger, http://messenger.msn.de

[NanoXML] NanoXML parser, available at http://sourceforge.net/projects/nanoxml/

[NBB+98] Normand, V.; Babski, C.; Benford, S.; Bullock, A.; Carion, S.; Farcet, N.; Frécon, E.; Harvey, J.; Kuijpers, N.; Thalmann, N.; Raupp-Musse, S.; Rodden, T.; Slater, M.; Smith, G.; Steed, A.; Thalmann, D.; Tromp, J.; Usoh, M.; Van Liempd, G.; Kladias, N.: The COVEN project: Exploring Applicative, Technical and Usage Dimensions of Collaborative Virtual Environments. PRESENCE, MIT Press, 1998

[NBM95] Nunamaker, J. F.; Briggs, R. O.; Mittleman, D. D.: Electronic Meeting Systems: Ten Years of Lessons Learned. In: D. Coleman, R. Khanna (Eds.), Groupware: Technology and Applications. Prentice-Hall Inc., 1995, pp. 149-193

[NESSIE] NESSIE - A configurable awareNESS envIronmEnt. GMD-FIT, St. Augustin, November 1999, http://orgwis.gmd.de/projects/nessie/

[Net98] Netscape Communications Corporation: Netscape Directory SDK for Java. 1998, http://developer.netscape.com/docs/manuals/dirsdk/-jsdk40/contents.htm

[Nie93] Nielsen, J.: Usability Engineering. Academic Press, Boston, 1993

[NM94] Nielsen, J.: Mack, R. L. (Eds.), Usability Inspection Methods. John Wiley & Sons, New York, 1994

[NR96] Nagao, N.; Rekimoto, J.: Agent Augmented Reality: A Software Agent Meets the Real World. In: Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-96), 1996

[OCF99] OpenCard Framework 1.1.1: Programmer's Guide, Third Edition, IBM April 1999

[OH98] Orfali, R.; Harkey, D.: Client/server Programming with Java and CORBA. John Wiley and Sons, ISBN 047124578X, 1998

[OIC00] TeleTrusT Deutschland e.V.: German Office Identity Card (Elektronischer Dienstausweis) - Version 1.0, Jule 2000, http://www.teletrust.de/dokumente/oic_1-0.pdf

[OLDAP] openLDAP, OpenLDAP Foundation, 2001, http://www.openldap.org

[OMG-XMI00] Object Management Group (OMG): XML Metadata Interchange (XMI) Specification. November 2000, http://www.omg.org/cgi-bin/doc?formal/2000-11-02

[OSGI] OSGi Service Gateway Specification, Release 1.0. The Open Services Gateway Initiative, May 2000, http://www.osgi.org

[OSMOS] Open System for Inter-enterprise Information Management in Dynamic Virtual Environments. EU IST-1999-10491, http://cic.vtt.fi/projects/osmos/main.html

[Oxygen] MIT: Project Oxygen, http://www.oxygen.lcs.mit.edu

[Pha00] Phanouriou, C.: UIML: A Device-Independent User Interface Markup Language. Virginia Polytechnic Institue and State University, Blacksburg, Virginia, September 2000

[Pla00] Platt, David S.: The Essence of COM with ActiveX: A Programmer's Workbook. Prentice Hall, , ISBN 0130165816, May 2000

[PlaceWare] PlaceWare Web Conferencing: PlaceWare, http://www.placeware.com

[PreNet] Information about Presentation.Net, http://www.presentation.net , 07/15/99

[PRG+99] Pistoia, M.; Reller, D.F.; Gupta, D.; Nagnur, M.; Ramani, A.: JAVA 2 Network Security. 2 edition, Prentice Hall, August 1999, ISBN: 0130155926

[Prin99] Prinz, W.: NESSIE: An Awareness Environment for Cooperative Settings. In: Bødker, S.; King, M.; Schmidt, K. (Eds.): Proceedings of the Sixth European Conference on Computer Supported Cooperative

Work (ECSCW '99), 12-16 September, Kopenhagen. Dordrecht: Kluwer Academic Publishers, 1999, pp. 391-410.

[PPB99] Prinz, W.; Pankoke-Babatz, U.; Broll, W.: Nessie - eine Infrastruktur zur Gruppenwahrnehmung für virtuelle Teams. GMD Spiegel, Vol. 29, No. 1/2, GMD, St. Augustin, Germany, ISBN 0724-4339, March 1999

[PSR93] Patel, K.; Smith, B.C.; Rowe, L.A.: Performance of a Software MPEG Video Decoder. In: Proceedings ACM Multimedia, Anaheim, CA, 1993

[RBB+00] Reinema, R.; Bahr, K.; Burkhardt, H.-J.; Hovestadt, L.: Cooperative Rooms - Symbiosis of Real and Virtual Worlds. In: Proceedings of 8th International Conference on Telecommunication Systems, Modelling and Analysis, ATSMA, Nashville, Texas, March 2000, pp. 181-187

[RBB+98] Reinema, R.; Bahr, K.; Baukloh, M.; Burkard, H.-J.; Schulze, G.: Cooperative Buildings – Workspaces of the Future. In: Callaos, C., Omolayole, O., Wang, L. (Eds.), Proceedings of the World Multiconference on Systemics Cybernetics and Informatics (SCI'98), Vol. 1, Orlando, Florida, July 12-16, 1998, pp. 121-128

[REN+96] Rudinger, G.; Espey, J.; Neuf, H., and the LUSI Consortium: Human Factors Guidelines for Designers of Telecommunication Services for Non-Expert Users. Volume 1 & 2, Loughborough: HUSAT Research Institute, 1996

[RFC1831] Srinivasan, R.: RPC: Remote Procedure Call Protocol Specification Version 2. IETF RFC 1831, Network Working Group, 1995, http://www.ietf.org/rfc/rfc1831.txt

[RFC2068] Fieldings, R.; Gettys, J.; Mogul, J.; Frystyk, H.; Berners-Lee, T.: Hypertext Transfer Protocol – HTTP/1.1.. IETF RFC 2068, January 1997, http://www.ietf.org/rfc/rfc2068.txt

[RFC2169] Fraser, B.; et al.: Site Security Handbook. IETF RFC 2196, September 1997

[RFC2251] Wahl, M.; et al.: Lightweight Directory Access Protocol (Version 3). IETF RFC 2251, December 1997

[RFC2252] Wahl, M.; Coulbeck, A.; Howes, T.; Kille, S.: Lightweight Directory Access Protocol (Version 3): Attribute Syntax Definitions. IETF RFC 2252, December 1997, ftp://ds.internic.net/rfc/rfc2252.txt

[RFC2255] Howes, T.; Smith, M.: The LDAP URL format. IETF RFC 2255, December 1997, ftp://ds.internic.net/rfc/rfc2255.txt

[RFC2401] Kent, S.; Atkinson, R.: Security architecture for the Internet Protocol, IETF RFC 2401, November 1998, ftp://ds.internic.net/rfc/rfc2401.txt

[RFC2459] Housley, R.; Ford, W.; Polk, W.; Solo, D.: Internet x.509 public key infrastructure certificate and crl profile. IETF RFC 2259, January 1999, ftp://ftp.tu-chemnitz.de/pub/documents/rfc/rfc2459.txt

[RFC2510] Adams, C.; Farrell, S.: Internet X.509 Public Key Infrastructure Certificate Management Protocols. IETF RFC 2510, March 1999, ftp://ftp.tu-chemnitz.de/pub/documents/rfc/rfc2510.txt

[RFC2518] Goland, Y.; Whitehead, E.; Faizi, A.; Carter, S.; Jensen, D.: HTTP Extensions for Distributed Authoring – WEBDAV. IETF RFC 2518, Februar 1999, ftp://ftp.tu-chemnitz.de/pub/documents/rfc/rfc2518.txt

[RFC2527] Chokhani, S.; Ford, W.: Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework. IETF RFC 2527, March 1999, ftp://ftp.tu-chemnitz.de/pub/documents/rfc/-rfc2527.txt

[RFC2543] Handley, M.; Schulzrinne, H.; Schooler, E.; Rosenberg, J.: SIP: Session Initiation Protocol. RFC 2543, March 1999, http://www.faqs.org/rfcs/rfc2543.html

[RFC2589] Yaacovi, Y.; Wahl, M. T.; Genovese, T., Lightweight directory access protocol (v3): Extensions for dynamic directory services. IETF RFC 2589, May 1999, ftp://ftp.tu-chemnitz.de/pub/documents/rfc/rfc2589.txt

[RFC2608] Guttman, E.; Perkins, Ch. E.; Day, M.: Service Location Protocol, Version 2. IETF RFC 2608, June 1999, http://www.ietf.org/rfc/rfc2608.txt

[RFC2609] Guttman, E., Perkins, Ch. E., Kempf, J.: Service Templates and Service Schemes. IETF RFC 2609, June 1999, http://www.ietf.org/rfc/rfc2609.txt

[RFC2610] Perkins, Charles E.; Guttman, Erik: DHCP Options for Service Location Protocol. IETF RFC 2610, June 1999, http://www.ietf.org/rfc/rfc2610.txt

[RFC2616] Berners-Leeet, T.; et. al.: Hypertext Transfer Protocol -- HTTP/1.1. IETF RFC 2616, June 1999, ftp://ftp.tu-chemnitz.de/pub/documents/rfc/rfc2616.txt

[RFC2829] Wahl, M.; Alvestrand, JH.; Hodges, J.; Morgan, R.: Lightweight directory access protocol (v3): Authentication Methods for LDAP. IETF RFC 2829, May 2000, ftp://ftp.tu-chemnitz.de/pub/documents/-rfc/rfc2829.txt

[RG96] Roseman, M.; Greenberg S.: TeamRooms: Network Places for Collaboration. In: Proceedings of CSCW'96 (ACM Conference on Computer-Supported Cooperative Work), ACM Press, Cambridge, MA, 1996

[RMH00] Richard Monson-Haefel, Enterprise JavaBeans, 2nd Edition, O'Reilly, UK, March 2000, ISBN 1565928695

[RS99] Reinema, R.; Steinmetz, R.: MCRS - A Multimedia Conference Reservation and Management System. In: Proceedings of TENCON'99, Cheju, Korea, September 15-17, 1999

[RT01] Reinema, R.; Türpe, S. (Eds.): UNITE Architecture – Basic Platform. internal project report, The UNITE Consortium, Darmstadt, September 2001, http://www.unite-project.org/public/index.html

[RTNotes] Lotus Notes: Lotus Realtime Notes, http://www.lotus.com/realtime ,

[RTS98a] Reinema, R.; Tietze, D.A.; Steinmetz, R.: IMCE - An Integrated Multimedia Conferencing and Collaboration Environment. In: Proceedings of the First International Chinese CSCW Workshop (C-CSCW98), Publishing House of Electronics Industry, Beijing, China, December 1998, ISBN 7-5053-5066-8, pp. 95-100

[RTS98b] Reinema, R.; Tietze, D.A.; Steinmetz, R.: IMCE - An Integrated Multimedia Collaboration Environment. In: Proceedings of ACM Multimedia 98, Bristol, September 1998

[RW01] Reinema, R.; Wolf, R. (Eds.): UNITE Solution Concept. Internal project report, The UNITE Consortium, Darmstadt, April 2001, see also http://www.unite-project.org

[SA97] Schneider, W.; André, S.: ICE-TEL: Interworking Public Key Certification Infrastructure for Europe. In: Datenschutz und Datensicherheit (DuD), Vol. 21, No. 7, 1997

[SA99] Stajano, F.; Anderson, R.: The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In: B. Christianson, B. Crispo and M. Roe (Eds.). Security Protocols, 7th International Workshop, LNCS 1796, 1999

[SAL98] Salutation Consortium: Salutation Architecture: Overview. White Paper, 1998, http://www.salutation.org/whitepaper/originalwp.pdf

[Sametime] Lotus Sametime - Realtime Collaboration Standards, Whitepaper, Lotus Development Corporation, October 1999, http://www.lotus.com/home.nsf/welcome/sametime

[San96] Sandhu, R.S.: Role Hierarchies and Constraints for Lattice-Based Access Controls. In: Proceedings of the European Symposium on Research in Security and Privacy, 1996

[SAW94] Schilit, B.N.; Adams, N.; Want, R.: Context-Aware Computing Applications. In: Proceedings of the Workshop on Mobile Computing Systems and Applications, IEEE Computer Society, Santa Cruz, CA, December 1994, pp. 85-90

[SC94] Scrivener, St.A.R.; Clark, S.: Introducing computer-supported cooperative work. In: Computer-supported cooperative work: the multimedia and networking paradigm, Avebury Technical, S.A.R. Scrivener (Eds.), Aldershot, 1994, pp. 19-38

[Schi98] Schiller, R.: Unternehmensnetzwerke bei kleinen und mittleren Unternehmen – Ergebnisse einer empirischen Studie. In: Winand, U., Nathusius, K. (Eds.): Unternehmungsnetzwerke und virtuelle Organisationen, Stuttgart, 1998

[Schm98] Schmeh, Klaus: Safer Net - Kryptographie im Internet und Intranet. dpunkt.Verlag, ISBN 3-932588-23-1, 1998

[Schn00] Schneier, B.: Secrets and Lies. John Wiley and Sons Inc., ISBN 0471253111, September 2000

[Schn96] Schneier, B.: Applied Cryptography. 2nd Edition, John Wiley and Sons Inc. ISBN 0-471-11709-9, 1996,

[Scr00] Scribner, Kennard: Understanding SOAP: Simple Object Access Protocol. Sams Publishing, ISBN 0672319225, 2000

[SDA99] Salber, D.; Dey, A. K.; Abowd, G. D.: The Context Toolkit: Aiding the Development of Context-Enabled Applications. In: Proceedings of CHI'99, ACM Press, Pittsburgh, PA, 1999

[Secude] Secude Homepage: Secude Toolkit, http://www.secude.com

[SGH+94] Streitz, N.A.; Geißler, J.; Haake, J.M.; and Hol, J.: DOLPHIN: Integrated Meeting Support across LiveBoards, Local and Remote Desktop Environments. In: Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW '94), ACM Press, Chapel Hill, N.C., October 22-26, 1994, pp. 345-358

[SGH98] Streitz, N.A.; Geissler, J.; Holmer, T.: Roomware for Cooperative Buildings: Integrated Design of Architectural Spaces and Information Spaces. In: Cooperative Buildings: Integrating Information, Organization, and Architecture, Streitz, N. A., Konomi, S., Burkhardt, H.-J. (Eds.), Springer Verlag, LNCS 1370, 1998, pp. 4-21

[Sha97] Shaked, E.: Intelligent MCU, 1997, http://www.internettelephony.com/-archive/2.03.97/Features/feature4.html

[SHM+00] Shannon, B.; Hapner, M.; Matena, V.; Davidson, J.; Pelegri-Llopart, E.; Cable, L.: Java 2 Platform: Platform and Component Specifications (Enterprise Edition). Addison-Wesley, May 2000, ISBN: 0201704560

[Shn98] Shneiderman, B.: Designing the User Interface, Addison Wesley, ISBN 0201694972, 1998

[Sie96] Sieber, P.: Ein KMU in der Informatikbranche. In: Griese, J., Sieber, P. (Eds.): Internet – Nutzung für Unternehmungen, Bern, 1996

[Sie98] Sieber, P.: Virtuelle Unternehmen in der IT-Branche – die Wechselwirkung zwischen Internet-Nutzung, Organisation und Strategie. Dissertation, Reihe: Griese, J., Kühn, R./Thom, N. (Eds.): Berner betriebswirtschaftliche Schriften, Vol. 19, Bern, 1998

[SigG97] Bundestag: „Gesetz zur Regelung der Rahmenbedingungen für Informations- und Kommunikationsdienste – Artikel 3 Gesetz zur Digitalen Signatur (Signaturgesetz SigG)“, 13.06.1997, online unter http://www.bsi.bund.de/aufgaben/projekte/pbdigsig

[SigV97] Bundesregierung: „Verordnung zur Digitalen Signatur (Signatur-verordnung – SigV)“, 08.10.1997, online unter http://www.iid.de/rahmen/sigv.html

[SKB98] Streitz, N. A.; Konomi, S.; Burkhardt, H.-J. (Eds.): Cooperative Buildings - Integrating Information, Organization, and Architecture. Proceedings of the First International Workshop on Cooperative Buildings, Springer Verlag, Darmstadt, LNCS 1370, February 1998

[Socialweb] The Social Web Research Program - Linking people through virtual environments, GMD-FIT, February 2000, St. Augustin, Germany http://orgwis.gmd.de/projects/SocialWeb/

[Soo97] Soor, M.: The Information Building (in German). Dissertation, ifib, University of Karlsruhe, 1997

[SRI00] SRI International: Home page of The Open Agent Architecture, 2000, http://www.ai.sri.com/~oaa

[SSD+99] Goland, Yaron Y.; Cai, Ting; Leach, Paul; Gu, Ye: Simple Service Discovery Protocol/1.0. October 1999, http://www.globecom.net/ietf/draft/draft-cai-ssdp-v1-02.html

[ST94] Schilit, B.N.; Theimer, M.M.: Disseminating Active Map Information to Mobile Hosts. In: IEEE Network, Vol. 8, No. 5, IEEE Computer Society, Sept/Oct. 1994, pp. 22-32

[Sta99] Stapf, K.-H.: Psychologische Betrachtungen zum Sicherheitsbegriff. In: Müller, G., Rannenberg, K. (Eds.): Mehrseitige Sicherheit in der Kommunikations-technik, Bd. 2, Erwartungen, Akzeptanz, Nutzung, Addison-Wesley-Longman, Massachusetts, 1999

[Ste98] Stephens, G. K.: Organization Structures and Design. 1998, http://voltaire.is.tcu.edu/~stephens/teaching/mana3353/OrgStructure

[Ste99] Steinmetz, R.: Multimedia-Technologie: Grundlagen, Komponenten und Systeme. Springer Verlag, Berlin, 1999, ISBN 3-540-62060-5

[TBR98] Tietze, D. A.; Bapat, A.; Reinema, R.: Document-Centric Groupware for Distributed Governmental Agencies. In: Proceedings of CAiSE'98, Pisa, Italy, June 1998

[teamHos] Methodology and Tools for World-best Teamwork in Hospitals. EU IST-1999-11567, http://www.team-hos.net

[TeamWave] TeamWave Software Ltd.: TeamWave. http://www.teamwave.com , 07/15/99

[Tie01] Tietze, D.: A Framework for Developing Component-based Co-operative Applications. PhD Thesis, Darmstadt University of Technology, Darmstadt, Germany, 2001

[TOWER] Theatre of Work Enabling Relationships. EU IST-1999-10846, http://orgwis.gmd.de/projects/tower/

[TTT00] Teletrust TTT-AG2: TTT-OIC: German Office Identity Card, Version 1.0, 06.07.2000

[TTT98] Teletrust TTT-AG2: DIA - Digital Identity Card Specification Version 3 (in German), 1998

[UDDI00] IBM: UDDI. Technical White Paper, September 2000, http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.PDF

[UpnP00] Universal Plug and Play Forum: Universal Plug and Play Device Architecture. March 2000, http://www.upnp.org

[VAJ] IBM Visual Age for Java - Java IDE, see http://www.ibm.com/software/vajava/

[Ven91] Venkatraman, N.: IT-Induced Business Reconfiguration. In: Scott Morton, M.S. (Editor), The Corporation of the 1990's: Information technology and organizational Transformation, New York, 1991

[VXML] VoiceXML, VoiceXML Forum, http://www.voicexml.org

[W3C-XKMS] Ford, W.; Hallam-Baker, Ph.; Fox, B.; Dillaway, B.; LaMacchia, B.; Epstein, J.; Lapp, J.: XML Key Management Specification (XKMS). World Wide Web Consortium (W3C), March 2001, http://www.w3.org/TR/xkms/ , 30. March 2001

[W3C-XML] Bray, T.; Paoli, J.; Sperberg-McQueen, C. M.: Extensible Markup Language (XML) 1.0 Specification. World Wide Web Consortium (W3C), February 1998, http://www.w3.org/TR/1998/REC-xml-19980210

[Wei93] Weiser, Mark: Some Computer Science Issues in Ubiquitous Computing. In: Communications of the ACM, Vol. 36, No. 7, July 1993, pp. 75-84

[WFG+99] Want, R.; Fishkin, K.; Gujar, A.; Harrison, B.: Bridging Physical and Virtual Worlds with Electronic Tags. In: Proceedings of CHI '99, Pittsburgh, PA, 1999, pp. 370-377

[WHF+92] Want, R.; Hopper, A.; Falcao, V.; Gibbons, J.: The Active Badge Location System, ACM Transactions on Information Systems, Vol. 10, No. 1, January 1992

[WJ95] Wooldridge, M.; Jennings, N.R.: Intelligent Agents: Theory and Practice. In: The Knowledge Engineering Review, Vol. 10, No. 2, 1995, pp. 115-152

[WJH97] Ward, A.; Jones, A.; Hopper, A.: A New Location Technique for the Active Office. In: IEEE Personal Communications, Vol 4, No. 5, pp. 42-47

[Woo97] Wooldridge, M. J.: Agent-based software engineering. In: IEEE Proceedings on Software Engineering, Vol. 144, No. 1, 1997, pp. 26–37

[WSW95] Want, R.; Schilit, B.N.; Weiser, M.; et al.: An Overview of the Parctab Ubiquitous Computing Experiment. In: IEEE Personal Communications, Vol. 2, No. 6, December 1995, pp. 28-43

[WY00] Weltman, R.; Yoshinaga, D.: LDAP Programming with Java. Addison Wesley, 2000, ISBN 0201657589

[X.509] ITU Recommendation X.509 (1997E). Information Technology – Open Systems Interconnection: The Directory Authentication Framework, June 1997

[Xalan] Xalan XSLT processor, available from http://xml.apache.org

[Xerces] Xerces XML parser, available from http://xml.apache.org

[XML] Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation, The World Wide Web Consortium, October 2000, http://www.w3.org/TR/2000/REC-xml-20001006

[XMLS] XML Schema - Part 2: Datatypes. W3C Recommendation, The World Wide Web Consortium, October 2000, http://www.w3.org/TR/xml/xmlschema-2/

[XP] XP XML parser, available from http://www.jclark.com/xml/

[XSLT] XSL Transformations, W3C Recommendation, November 1999, http://www.w3.org/TR/xslt.html

[XT] XT XSLT processor, available from http://www.jclark.com/xml/

[YaMes] Yahoo!-Messenger, http://messenger.yahoo.com/

[Ylo95] Ylonen, T.: The SSH (Secure Shell) Remote Login Protocol. 1995, http://www.massconfusion.com/ssh/ssh_rfc.txt

[ZH99] Zapf, M.; Herrmann, K.: The AMETAS-Tutorial. 1999, http://www.ametas.de/docs/Tutorial/index.html

[ZHG99] Zapf, M.; Herrmann, K.; Geihs, K.: Decentralized SNMP Management with Mobile Agents. In: Sloman, M.; Mazumdar, S.; Lupu, E. (Eds.): Integrated Network Management VI (IM'99), IEEE/IFIP, May 1999, ISBN 0–7803–5748–5, pp. 623–635

[ZMG98] Zapf, M.; Müller, M.; Geihs, K.: Security Requirements for Mobile Agents in Electronic Markets. In: Trends in Distributed Systems for Electronic Commerce – International IFIP/GI Working Conference, TREC'98, Hamburg, Springer, Lecture Notes of Computer Science (LNCS 1402), June 1998 ISBN 3–540–64564–0, pp. 205–217

List of publications

The following publications have appeared so far (December 2001) or have already been accepted for publication.

Books

Reinema, Rolf: WWFM – Worldwide Facility Management. In: O. Gassmann, H. Meixner (Eds.): Sensors in Intelligent Buildings. Volume 2, WILEY-VCH Verlag, ISBN 3-527-29557-7, 2001, pp. 469-481

Kraetzschmar, G. K.; Reinema, R.: VKI Tools und Experimentierumgebungen. In: Jürgen Müller (Ed.): Verteilte Künstliche Intelligenz: Methoden und Anwendungen. B.I. Wissenschaftsverlag, Mannheim, ISBN 3-411-16181-7, 1993, pp. 222-256

Journals

Kloos, R.; Reinema, R; Schroeder, M.: Intelligent Traders for Co-operative Rooms. To appear in: Journal of Applied Systems Studies, Cambridge International Science Publishing, Cambridge, England

Reinema, R.; Thielmann, H.: RoomComputer - Ubiquitous Computing in Cooperative Rooms. To appear in: thema Forschung, Technische Universität Darmstadt, Darmstadt, Germany, January 2002

Conference papers

Kloos, R.; Reinema, R; Schroeder, M.: UNITE - Modern Teamwork with Adaptive Communications. In: A. Canning, T. Stock (Eds.): E-Working: How to Improve Effectiveness. DTI, Software Technology Outreach, London, November 2001

Woodcock, A.; Reinema, R.: Enhancing the Distribution of Tacit Knowledge Through a Ubiquitous and Integrated Teamwork Environment. In Proceedings of European CSCW 2001, Workshop On Managing Tacit Knowledge, Bonn, Germany, September 16th, 2001

Kloos, R.; Reinema, R; Schroeder, M.: An Adaptive Trading Framework based on Agents supporting a Geographically Distributed Team. In: Proceedings of the Third European Agent Systems Summer School (EASSS 2001), Advanced Course on Artificial Intelligence (ACAI-01), Pechoucekpp, M. (Ed.), 80-87, Czech Technical University in Prague, Czech Republic, July 2001

Moschgath, M.-L.; Reinema, R.; Hähner, J.: Sm@rtLibrary - An Infrastructure for Ubiquitous Technologies and Applications. In: Proceedings of 21st International Conference on Distributed Computing Systems Workshops (ICDCS 2001), International Workshop on Smart Appliances and Wearable Computing (IWSAWC), Mesa, Arizona, IEEE Computer Society, ISBN 0-7695-1080-9, April 2001, pp. 208-213

Reinema, R.: Die Rolle der SmartCard im Kooperativen Raum. In: Tagungsband des 11. GMD-SmartCard Workshop, Bruno Struif (Ed.), Darmstadt, 06./07.Februar 2001

Kloos, R.; Reinema, R; Schroeder, M.: Intelligent Traders for Communication in Cooperative Rooms. In: Audrey Canning and Tony Stock (Eds.): E-Working: How to Improve Effectiveness. DTI, Software Technology Outreach, London, December 2000

Hoffmann, M.; Reinema, R.: A Multi Agent Architecture for a Virtual Project Office. In: Proceedings of The International ICSC Symposium on Multi-Agents and Mobile Agents in Virtual Organizations and E-Commerce (MAMA'2000), Wollongong, Australia, December 2000

Moschgath, M.-L.; Reinema, R.; Hoffmann, M.: A Security Infrastructure for the Virtual Project Office. In: Proceedings of the IEEE International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2000), Split, Croatia, October 2000

Reinema, R.; Moschgath, M.-L.; Hoffmann, M.: Eine Sicherheitsinfrastruktur für das Virtuelle Projektbüro. In: Tagungsband Informatik 2000 (GI Jahrestagung), Workshop "Sicherheit in Mediendaten", Gesellschaft für Informatik (GI), Berlin, Germany, September 2000

Kloos, R.; Reinema, R.; Schroeder, M.: Intelligent Traders for Communication in Cooperative Rooms (COR). In: Proceedings of The Fourth International Conference on Autonomous Agents (Agents 2000), Workshop on Intelligent Agents for Computer Supported Co-operative Work: Technology and Risks, Barcelona, Spain, June 2000

Reinema, R.; Steinmetz, R.: MCRS - A Multimedia Conference Reservation and Management System. In: Proceedings of 8th International Conference on Telecommunication Systems, Modelling and Analysis, ATSMA, Nashville, Texas, March 2000, pp. 174-180

Reinema, R.; Bahr, K.; Burkhardt; H.-J., Hovestadt, L.: Cooperative Rooms -- Symbiosis of Real and Virtual Worlds. In: Proceedings of 8th International Conference on Telecommunication Systems, Modelling and Analysis, ATSMA, Nashville, Texas, March 2000, pp. 181-187

Reinema, R.; Moschgath, M.-L.; Bahr, K., Hovestadt, L.: roomComputers – Bridging Spaces. In: Proceedings of 7th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS '99), Cape Town, South Africa, December 1999, pp.75-80

Bahr, K.; Burkhardt, H.-J.; Hovestadt, L.; Reinema, R.: Integrating Virtual and Real Work Environements. In: Proceedings of IEEE Conference on Software in Telecommunications and Computer Networks (SoftCOM '99), Split, Rijeka, Croatia, Trieste, Venice, Italy, October 1999, pp.225-232

Reinema, R.; Zarcula, A.: Virtuelles Büro für Zusammenarbeit in Teams. In: Tagungsband der GWS Jahrestagung '99, Saarbrücken, Oktober 1999

Reinema, R.; Steinmetz, R.: MCRS - A Multimedia Conference Reservation and Management System. In: Proceedings of IEEE TENCON'99, September 15-17, 1999, Cheju, Korea, University of Stony Brook, Stony Brook, NY, ISBN 0-7803-5739-6, 1999, pp. 950-953

Reinema, R.; Tietze, D.A.; Steinmetz, R.: IMCE - An Integrated Multimedia Conferencing and Collaboration Environment. In: Proceedings of the First International Chinese CSCW Workshop (C-CSCW98), December 1998, Beijing, China, ISBN 7-5053-5066-8, Publishing House of Electronics Industry, 1998, pp. 95-100

Tietze, D.; Bapat, A.; Reinema, R.: Document-Centric Groupware for Distributed Governmental Agencies. In: Fankhauser, P.; Ockenfeld, M. (Eds.): Integrated Publication and Information Systems, ISBN 3-88457-968-1, GMD, Sankt Augustin, October 1998, pp. 75-90

Reinema, R.; Tietze, D.A.; Steinmetz, R.: IMCE - An Integrated Multimedia Collaboration Environment. Poster Presentation at: The Sixth ACM International Multimedia Conference (MULTIMEDIA'98), September 12-16, 1998, Bristol, UK

Bahr, K.; Burkhardt, H.-J.; Reinema, R.: Cooperative Buildings - Workspaces of the Future. In: Balzer et al. (Eds.), Telecooperation Technology, GMD, Sankt Augustin, August 1998, pp. 60-63

Reinema, R.; Bahr, K.; Baukloh, M.; Burkhardt, H.-J.; Schulze, G.: Cooperative Buildings - Workspaces of the Future. In: Callaos, C.; Omolayole, O.; Wang, L. (Eds.), Proceedings of the World Multiconference on Systemics Cybernetics and Informatics (SCI'98), July 12-16, 1998, Orlando, Florida, Vol. 1, pp. 121-128

Tietze, D.; Bapat, A.; Reinema, R.: Document-Centric Groupware for Distributed Governmental Agencies. In: Pernici, B.; Thanos, C. (Eds.), Proceedings of the 10th Conference on Advanced Information Systems Engineering (CAiSE'98), June 8-12, 1998, Pisa, Italy, pp. 173-190

Kunkelmann, T.; Reinema, R.; Steinmetz, R.: Evaluation of Different Video Encryption Methods for a Secure Multimedia Conferencing Gateway. In: Proceedings of

the 4th COST 237 Workshop, Lisboa, Portugal, pp. 75-89, Springer Verlag, Heidelberg, 1997, LNCS 1356, ISBN 3-540-63935-7

Kunkelmann, T.; Reinema, R.: A Scaleable Security Gateway for Distributed Multimedia Communication Tools. In: Proceedings of the IFIP WG 6.1 International Working Conference on Distributed Applications and Interoperable Systems (DAIS'97), Cottbus, Germany, 1997, pp. 89-94, Chapmann & Hall, ISBN 0-412-82340-3.

Kunkelmann, T.; Reinema, R.: A Scaleable Security Architecture for Multimedia Communication Standards. In: Proceedings of the 4th IEEE International Conference on Multimedia Computing and Systems (ICMS97), Ottawa, Canada, pp. 660-661, IEEE Computer Society, 1997, ISBN 0-8186-7819-4

Geißler, J.; Haake, J.M.; Hilgert, P.; Reinema, R.: Virtual Meetings - Unterstützung verteilter Arbeitsgruppen in virtuellen Sitzungen. In St. Uellner (Ed.): Tagungsband Workshop „Computer Supported Cooperative Work (CSCW) in großen Unternehmungen“, Darmstadt, May 1996

Kraetzschmar, G. K.; Reinema, R.: Pede-lab - a knowledge-based modeling and simulation environment and its application in business process reengineering. In: W. Hamscher (Ed.): Working Notes of the AAAI-94 Workshop on Artificial Intelligence in Business Process Reengineering, Seattle, WA, August 1994

Kraetzschmar, G. K.; Reinema, R.: Pede-lab: Testbeds for manufacturing applications. In: D. Goldstein (Ed.): Proceedings of SIGMAN Workshop on Intelligent Manufacturing Applications, Washington, DC, July 1993.

Kraetzschmar, G. K.; Reinema, R.: Pede-lab - Eine Experimentierumgebung für die Verteilte Künstliche Intelligenz. In: Jürgen Müller (Eds.): Beiträge zum Gründungsworkshop der Fachgruppe Verteilte Künstliche Intelligenz, Document D-93-06, DFKI, Saarbrücken, April 1993

Technical Reports

Blecher, Th.; Reinema, R.;Kunkelmann, Th.: Kryptographieverfahren für Videoübertragung. GMD-Studie Nr. 319, GMD, Sankt Augustin, 1997, ISBN 3-88457-319-5.

Bahr, K.; Burkhardt, H.-J.; Hovestadt, L.; Hovestadt, K.; Friedrichs, K.; Nabert, U.; Reinema, R.; Streitz, N.: Das Kooperative Gebäude. Studie, GMD, Darmstadt, November 1996.

Geißler, J.; Haake, J.M.; Hilgert, P.; Reinema, R.: Virtual Meetings - Unterstützung verteilter Arbeitsgruppen in virtuellen Sitzungen. Arbeitspapiere der GMD, Nr. 1016, Sankt Augustin, Juli 1996.

Reinema, R.: Pede-lab - Aufbau und Entwicklung einer Experimentierumgebung für Multi-Agenten-Systeme. Technical Report FR-1993-009, FORWISS, Erlangen, 1993.

Appendix A -

Acronyms and Abbreviations

ACL	Agent Communication Language
ActiveX	A brand name for a set of technology and services, all based on the Component Object Model (COM)
ADSL	Asymmetric Digital Subscriber Line
AMETAS	Asynchronous Message Transfer Agent System
Apache	Apache is the most popular web server. Apache is "A PAtCHy server". It was based on some existing code and a series of "patch files"
API	Application Programming Interface
ASP	Active Server Pages
ATM	Asynchronous Transfer Mode
AUI	Abstract User Interface
AUI-ML	AUI Mark-up Language
AV	Audio and Video
BACnet	Building Automation and Control Network
BAV	Basic Audio Visual Device within HAVi
BC	Broadcast Client
BOA	Basic Object Adapter
CA	Certification Authority
CAD	Computer Aided Design
cHTML	Compressed HTML
Cobol	Programming Language
Cocoon	Open Source Java publishing framework Servlet that relies on technologies like DOM, XML, and XSL to provide web content.

COM	Microsoft’s Component Object Model
CORBA	Common Object Request Broker Architecture
COS	CORBA Naming Standard
CPU	Central Processing Unit
CRL	Certificate Revocation List
CSCW	Computer Supported Cooperative Work
CSM	Client Security Manager
CSS	Cascading Style Sheet
CTI	Computer Telephony Integration
CVE	Collaborative Virtual Environments
DA	Deputy Agent
DAL	Data Access Layer
DBMS	Database Management System
DCE-RCP	DCE-RCP is the Distributed Computing Environment's implementation of RPC (Remote Procedure Call). The Open Software Foundation is the developer of DCE, which is a set of distributed computing technologies. As well as an RPC mechanism, DCE includes security services, LAN namespace services and network time services.
DCT	Discrete Cosine Transform
DCOM	Microsoft’s Distributed Component Object Model
DCM	Device Control Module within HAVi
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DocBook	A DTD maintained by the DocBook Technical Committee of OASIS
DTD	Document Type Definition
EIB	European Installation Bus
EJB	Enterprise Java Beans
FAV	Full Audio Visual Device within HAVi
FIPA	Foundation for Intelligent Physical Agent
FM	Facility Management
FURPS	Functionality, Usability, Reliability, Performance, and Supportability
FURPS+	FURPS plus Design constraints, Implementation requirements, Interface requirements and Physical requirements
G.711	Audio Encoding
G.723	Audio Compression/Decompression
GCC	General Call Control (developed by Parlay Standard)

GENA	Generic Event Notification Architecture
GSM	Standard Voice Compression
GUI	Graphical User Interface
H.263	Video Compression/Decompression
H.320	Narrow-Band Visual Telephone Systems and Terminal Equipment
H.323l	Packed-Based Multimedia Communication Systems
HAVi	Home Audio Video Interoperability
HDML	Handheld Device Markup Language
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Secure Hypertext Transfer Protocol
IAV	Intermediate Audio Visual Device within HAVi
ICL	Inter-agent Communication Language
ICQ	'I Seek You'. ICQ is a user-friendly Internet program that notifies who is online at all times. With ICQ one can chat, send messages and files, and much more.
IDEA	International Data Encryption Algorithm
IDL	Interface Definition Language
IIOP	Internet Inter-ORB Protocol
ILS Server	MS Internet Locator Service Server
IM	Instant Manager
IMAP4	Mail Standard for retrieving e-mails from mail server
IP	Internet Protocol
IPSEC	An Internet protocol designed to provide connectionless data integrity service and data origin authentication service for IP datagrams, and (option-ally) to provide protection against replay attacks.
IPv6	Internet Protocol Version 6
ISP	Internet Service Provider
ISDN	Integrated Services Digital Network
IVR	Interactive Voice Response
J2ME	Java 2 Micro Edition
JAIN	Java API for Integrated Network
JCC	Java Call Control (developed under JAIN)
JDBC	Java Database Connectivity
JDK	Java Development Kit
JINI	Java Intelligent Network Infrastructure

JMS	Java Messaging System
JRE	Java Runtime Environment
JRMP	Java Remote Method Protocol
JSAPI	Java Speech API
JTS	Java Transaction Service
JTAPI	Java Telephony API
JTSPI	Java Telephony Specific Provider Interface
JVM	Java Virtual Machine
KQML	Knowledge Query and Manipulation Language
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LIS	Location Information Server
LON	Local Operation Network
LUS	Lookup Service within Java JINI
Luxmate	Luxmate lighting control management systems
	A bus protocol developed by Luxmate, a manufacturer of lighting control and bus systems.
MAPI	Messing Application Programming Interface
MASIF	Mobile Agent Systems Interoperability Facilities
MathML	Mathematical Markup Language
Mbeans	Managed Beans. An Mbean is a Java object that implements specific interfaces and conforms to certain design patterns.
MC	Multipoint Controller
MCU	Multipoint Control Unit
MFC	Microsoft Foundation Classes: C++ class library from Microsoft, e.g. used for the development of graphical user interfaces based on Windows-OS
MGCP	Media Gateway Control Protocol
Microsoft Project™	allows you to create and manage projects quickly and effectively
MIME	Multipurpose Internet Mail Extension
MP	Multipoint Processor
MRC	Meeting Room Client
MUX	Multiplexer, Multiplexing Unit
NetMeeting	MS NetMeeting is a real-time collaboration and conferencing tool
Network-OLE	Network Object Linking and Embedding
NMS	Network Management System

NPS	Network Provider Services
OA	Object Adapter
OAA	Open Agent Architecture
Object Pascal	Programming Language
OCSP	Online Certificate Status Protocol
ODBC	Open Database Connectivity
OIC	Office Identity Card
OMG	Object Management Group
OOP	Object Oriented Programming
OSG	Open Service Gateway
OSGi	Open Services Gateway Initiative: Industry consortium to define a software platform that integrates devices and services from many different vendors. See also http://www.osgi.org
OTS	OMG Transaction Service
ORB	Object Request Broker
ORPC	Object Remote Procedure Call
OS	Operating System
PA	Personal Agent
PABX	Private Automatic Branch Exchanges
PBX	Private Branch Exchange
PC	Personal Computer
PCM	Personal Communication Management
PDA	Personal Digital Assistant (e.g. PalmPilot, Handspring Visor, Psion MX5, HP Journada, Compaq iPaq, Nokia Communicator)
PIM	Personal Information Manager
PCK	Public Key Cryptography
PKI	Public Key Infrastructure
PMH	Personal Message Handling
PO	Personal Office
POA	Portable Object Adapter
POP3	Mail Standard
POTS	Plain Old Telephony System
PR	Personal Agent Representative
PRE	Personal Office Runtime Environment
PS	Personal Office Services
PSA	Personal Security Assistant
PSE	Personal Security Environment

PSM	Personal Security and Privacy Manager
PSTN	Public Switched Telephone Network
QCIF	Quarter Common Intermediate Format
QoS	Quality of Service
RA	Room Agent
RAM	Random Access Memory
RC2	Rivest Cipher #2, a very low encryption algorithm designed by Ronald Rivest.
RCP	Remote procedure Call
RDBMS	Relational Database Management System
RFID	Radio Frequency Identification
RM	Resource Manager
RMI	Remote Method Invocation interface provide by Java
RMS	Rights Management System
RSA	Asymmetrical Encryption Algorithm
RTCP	Real-Time Control Protocol
RTP	Real-Time Transport Protocol
RTSP	Real-Time Streaming Protocol
SCN	Switched Circuit Network
SDP	Service Discovery Protocol used by Bluetooth
SECUDE	Software development kit for cryptographic applications.
SG	Security Guidelines
SIP	Session Initiation Protocol
SLP	Service Location Protocol
SMIL	Synchronized Multimedia Integration Language
SMP	Session Management Protocol
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Remote Access Protocol
SOCKS4 / SOCKS5	Protocols, which relay TCP sessions at a firewall host to allow application users transparent access across the firewall.
SPE	Service Provisioning Environment
SPS	Service Provider Services
SSDP	Simple Service Discovery Protocol
SSL	Secure Socket Layer
T.120	The International Telecommunications Union (ITU) T.120 standard consists of a suite of communication and application

	protocols developed and approved by the international computer and telecommunications industries.
TCP	Transmisson Control Protocol
TCSEC	Trusted Computer System Evaluation Criteria
TM	Tool Manager
Tomcat	Tomcat is a Servlet container and JavaServer Pages (JSP) environment. A Servlet container is a runtime shell, which manages and invokes Servlets on behalf of users. It may be used stand alone, or in conjunction with several popular web servers (most likely with the Apache Web-server).
TS	Team Service
TUI	Telephone User Interface
UDDI	Universal Description, Discovery, and Integration
UDP	User Datagram Protocol
UI	User Interface
UM	Unified Messaging
UML	Unified Modeling Language
UPnP	Universal Plug and Play
URI	Universal Resource Identifier
URL	Universal Resource Locator
USB	Universal Serial Bus
Visual Basic	Programming Language
VML	Voice Markup Language
VoIP	Voice over IP
VPIM	Voice Profile for Internet Messaging
VPO	Virtual Project Office
VSM	VPO Security Manager
VXML	Voice Extensible Markup Language
W3C	World Wide Web Consortium
WAP	Wireless Application Protocol
WebDAV	Web-based Distributed Authoring and Versioning
WIDL	Web Interface Definition Language
WML	Wireless Mark-up Language used by WAP
WSDL	Web Service Description Language
WWW	World Wide Web
X.10	Categories of access for DTE’s (Data Terminal Equipment) accessing the PSPDN (Packet Switched Public Data Networks)

X.509	ITU standard, which defines a framework for public key infrastructures
XMI	XML Metadata Interchange
XKMS	XML Key Management Specification
XML	Extensible Markup Language
XML/EDI	Using XML for Electronic Data Interchange
XSL	Extensible Stylesheet Language

Appendix B -

Terms and Definitions

Abstract Service	A service class, which is defined through a device-independent functionality, as apposed to specific services, which are usually defined through specific device capabilities and features.
Access Control	Protection of system resources against unauthorised access; a process by which use of system resources is regulated according to a security policy and is permitted by only authorised entities (users, programs, processes, or other systems) according to that policy. See also Role-base Access Control.
Active Directory (AD)	AD is Microsoft's version of directory services, a feature that stores information about network resources. Directory services offer a consistent method for administrators to manage and secure resources and centrally organise and control access to network resources. AD supports several industry standard protocols and APIs, including DHCP, DNS, Kerberos 5, LDAP, and X.509 certificates.
Application Server	An application server provides an optimised execution environment for server-side application components. A Java application server delivers a high-performance, highly scalable, robust execution environment specifically suited to support Internet-enabled application systems.
Application Sharing	Application Sharing permits the simultaneous use of off-the-shelf applications like word processors and spreadsheets by a group of cooperating users.
Asymmetric Cryptography	A modern branch of cryptography (popularly known as "public key cryptography") in which the

	algorithms employ a pair of keys (a public key and a private key) and use a different component of the pair for different steps of the algorithm.
Auditing	A security service that records information needed to establish accountability for system events and for the actions of system entities that cause them.
Authentication	The process of verifying an identity claimed by or for UNITE known users.
Beans Technology	Beans is the portable, platform-independent software component model used in Java (Java Beans, Enterprise Java Beans). It enables developers to write reusable components once and run them anywhere – benefiting from the platform-independent nature of Java.
Components	are self-contained, reusable software units that can also be visually composed. Beans are Java classes that are created using property and event interface conventions.
Enterprise JavaBeans	EJB technology defines a model for the development and deployment of reusable JavaTM server components. The EJB architecture logically extends the JavaBeans component model to support server components.
Server components	are application components that run in an application server. EJB can support the rigorous demands of large-scale, distributed, mission-critical application systems. EJB technology supports application development based on a multitier, distributed object architecture in which most of an application's logic is moved from the client to the server. The application logic is partitioned into one or more business objects that are deployed in an application server.
Bell/LaPadula, Biba, Clark/Wilson,Chinese Wall	Security models, that express a security policy in a formal way. See [BeLaP73], [ClWi87] and [BrNa89].
Certificate	See: Public Key Certificate
Certificate Revocation	The event that occurs when a CA declares that a previously valid digital certificate issued by that CA has become invalid; usually stated with a revocation date.
Certificate Revocation List (CRL)	Certificate revocation lists are data structures within the directory service which contain information such as the date together with the reason for revoked certificates whose validity period has not yet expired. See also Certificate Revocation.

Certification Authority (CA)	An entity that issues digital certificates and vouchers for the binding between the data items in a certificate.
Confidentiality	The property that information is not made available or disclosed to unauthorised individuals, entities, or processes (i.e., to any unauthorised system entity).
Co-operative Workplace	The co-operative workplace integrates physical and virtual workplaces under a unified view and user interface.
DB query language	(e.g. SQL, XQL)
Digital Signatures	A value computed with a cryptographic algorithm and appended to a data object in such a way that any recipient of the data can use the signature to verify the data's origin and integrity.
Encryption / Decryption	Cryptographic transformation of data (called "plaintext") into a form (called "ciphertext") that conceals the data's original meaning to prevent it from being known or used. If the transformation is reversible, the corresponding reversal process is called "decryption", which is a transformation that restores encrypted data to its original state.
Firewall	An internetwork gateway that restricts data communication traffic to and from one of the connected networks (the one said to be "inside" the firewall) and thus protects that network's system resources against threats from the other network (the one that is said to be "outside" the firewall).
Gantt chart	Gantt chart is a project planning tool that can be used to represent the timing of tasks required to complete a project.
IBM’s NuOffice	NuOffice is a networked office system based on Lotus Domino for large corporate offices and workgroups with many mobile or telecommuting users.
Integrity	The property that data has not been changed, destroyed, or lost in an unauthorised or accidental manner.
Jakarta Open Source	The Apache developer community, Sun, IBM and others are working together in the Jakarta Project with the goal to provide commercial-quality server solutions based on the Java Platform that are developed in an open and cooperative fashion.
JavaCard	JavaCard is the Smartcard platform (operating system plus virtual machine) based on the Java programming language developed by Sun Microsystems.

JDK	The JavaTM Development Kit (JDKTM) contains the software and tools that developers need to compile, debug, and run applets and applications written using the Java programming language.
JavaSpaces	JavaSpaces technology is a simple unified mechanism for dynamic communication, coordination, and sharing of objects between Java technology-based network resources like client and server. In a distributed application, JavaSpaces technology acts as a virtual space between providers and requesters of network resources or objects.
Key Management	How to transfer secret keys used to provide the security without showing them to others.
Lotus Sametime™	The Sametime family includes the Sametime Server, the Sametime Connect client, and a range of Application Developer Tools. The Sametime Server supports the T.120 standard and is designed to work smoothly with third-party clients like Microsoft's NetMeeting. The Sametime Server also works seamlessly with any browser or with Lotus Notes, and has audio and video capabilities to enhance your online experience. Lotus Sametime supports immediate communication with people across the hall or around the world. Sametime supports the three foundations of real-time collaboration - Awareness, Conversation, and Shared Objects with Audio and Video.
Non-Repudiation	To create evidence that data has been sent or received, so that the sender (or receiver) cannot later falsely deny this fact.
Office Identity Card (OIC)	The digital office identity card (OIC) will be provided as a secure means for identification, authentication and authorisation of people together with privacy enhanced auditing, and location management.
Personal Agent	The representative of a person (an individual user) inside the UNITE system.
Personal Office (PO)	Personal Office is an instantiation of a virtual workplace which supports personal work and where users find their personal resources (e.g. personal address book, personal calendar, personal documents).
Personal Security Environment (PSE)	Personal Security Environment will be used to store private keys and sensitive public keys of a user. A PSE can be implemented by an encrypted file in the local file

	system of a user (software PSE) or by a smartcard (hardware PSE).
Personal Security Manager	Via the Personal Security Manager a UNITE user can determine how he would like to deal with his personal data, e.g. is current physical location.
PhoneSoft	A Lotus Domino-based unified messaging solution that delivers all of your messages-voice, fax, and e-mail-into your Lotus Notes inbox. It lets you access and manage all your messages using the device of your choice-a PC, touchtone telephone, or over the Internet-making you more productive and efficient.
Physical Workplace	the physical side of co-operative workplaces
Project Creation Wizard	The Project Creation Wizard helps the project initiator to create a project office.
Project Non-member	A supported user who doesn't play an active role in that project (e.g. an unregistered or anonymous customer).
Project Portal	A Project Portal denotes a specific instantiation of the user interface of a virtual project office.
Project Team Member	A Virtual Project Office user who plays an active role in that project (e.g. an employee of the project).
Proposal Portal	access to a team workplace for initiating a project
Public Key Certificate	A digital certificate that binds a system entity's identity to a public key value, and possibly to additional data items; a digitally-signed data structure that attests to the ownership of a public key.
Public Key Infrastructure (PKI)	A system of certification authorities (and, optionally, registration authorities and other supporting servers and agents) that perform some set of certificate management, archive management, key management, and token management functions for a community of users in an application of asymmetric cryptography.
Registration Authority (RA)	Maybe an optional PKI entity (separate from the CAs), if no CA is locally available. It acts as a mediator between the user and the CA. The RA does not sign either digital certificates or CRLs but has responsibility for recording or verifying some or all of the information (particularly the identities of subjects) needed by a CA to issue certificates and CRLs and to perform other certificate management functions.
Rights Management	Rights management deals with the document repository. The rights management system defines per-document rights, which guarantee that all documents of the Virtual Project Office are securely used of and distributed

	between the project members during the authoring phase. It controls and tracks any single movement, editing, and distributing of any single document.
Role-based Access Control	A form of identity-based access control where the system entities that are identified and controlled are functional positions in an organisation or process.
RoomComputer	A RoomComputer is an autonomous installation unit which will be used to sense and control equipment as well as services offers in physical rooms. It consists of a single board PC with a touch sensitive LCD-display as local interface, and an Intranet/Internet connection for remote access.
Salutation	Salutation is an approach to service discovery. This architecture is developed by an open industry consortium, called Salutation Consortium.
Secure Sockets Layer (SSL)	An Internet protocol (originally developed by Netscape Communications, Inc. [FKK96]) that uses connection-oriented end-to-end encryption to provide data confidentiality service and data integrity service for traffic between a client (often a web browser) and a server, and that can optionally provide peer entity authentication between the client and the server.
Security Guidelines	The security guidelines define a uniform security standard and guarantee for the utilisation and the management of the public key infrastructure.
Security Policy	A set of rules and practices that specify or regulate how a system or organisation provides security services to protect sensitive and system or organisation critical resources.
Service Middleware	Service announcement discovery, JINI, SLP, UPnP, SDP.
Smartcard	A Smartcard is a secure, tamper-resistant device consisting of a single –chip microcomputer, which is mounted on a plastic card of the size of a standard credit card. Very reliable authentication, electronic signature, and cryptography are tasks where Smartcards are superior to traditional technologies like magnetic stripe cards.
SMS Gateway	A computer transporting SMS messages from the Internet to a mobile phone network and vice versa.
System Administrator	A technical person who Administers, manages, maintains the system, sets up Co-operative Worplaces, configures them, assigns users.

Team operator	A person who administers a specific Co-operative Workplace, (re)configures it, (re)assigns users compared to the system administrator. A Virtual Project Office administrator's scope of responsibilities and rights is restricted to one (or more) Virtual Project Offices.
Team Agent	The representative of a team of collaborating users inside the UNITE system.
Team Initiator (or Project Initiator)	A non-technical person who initiates a new project, selects team members, initiates setting up a new Virtual Project Office.
Time-stamp Service	The time stamp service certifies the presence of digital data at a specific date.
Trusted Computer System Evaluation Criteria	European security evaluation schema. See also [TCSEC]
Unicode	Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language. The Unicode Standard has been adopted by such industry leaders as Apple, HP, IBM, JustSystem, Microsoft, Oracle, SAP, Sun, Sybase, Unisys and many others.
User Portal	A User Portal denotes a specific instantiation of the user interface of a personal office.
vCalendar	vCalendar defines a transport and platform-independant format for exchanging calendaring and scheduling information in an easy , automated and consistent manner.
vCard	vCard automates the exchange of personal information typically found on a traditional business card. VCard is used in applications, such as Internet mail, voice mail, Web browsers, telephony applications, call centres, video conferences, PIMs (Personal Information Managers), PDAs (Personal Data Assistants), pagers, fax, office equipment, and Smartcards.
Virtual Office	The term Virtual Office generally denotes an instantiation of a virtual workplace. We are distinguishing two types: personal office and virtual project office. A virtual project office is synonymous with Virtual Organisation, because a Virtual Office is considered the home of a Virtual Organisation.
Virtual Project Office (VPO)	VPO is an instantiation of a virtual workplace which supports teamwork, where people meet and collaborate on a common subject and/or project and where users find

	the resources shared among the corresponding team members (e.g. team calendar, team document repository)
Virtual Workplace	the work context side of co-operative workplaces
WAP Gateway	A computer translating between WAP and Internet protocols.
Web-Technologies	Technologies designed for usage within the World Wide Web (e.g. HTTP, HTML, CSS, XML, XSL, XSLT, Servlets, EJB, ASP, JSP, XSP, Apache, Tomcat, Cocoon)
Work zones	A work zone is defined as a set of resources (tools and data) which are grouped together in a Virtual Office to serve a particular activity, i.e. work context.

Appendix C -

Abstract User Interface Mark-up Language - DTD

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- ################ Preface ################    -->
<!--
        This is the document type definition (DTD) for the abstract user
        interface mark-up language, reference to as AUI-ML
-->
<!-- ################ Entities ################    -->
<!--
        The Widgets entity contains all widgets defined in this version of the
        DTD from the abstract user interface (AUI-ML) mark-up language.
-->
<!ENTITY % Widgets "( container | choose | textinput | slider | setselection | output | link  )">
<!--
        The JavaTypes entity contains all supported java return types from
        calls to a java method.
-->
<!ENTITY % JavaTypes "( void | bool | string | char | byte | short | int | long | float | double )">
<!--
        The BorderStyle entity declares attributes, which describe the
        appearance of abstract elements of a user interface.
-->
<!ENTITY % BorderStyle
"       border                  ( none | line | bevelup | beveldown )          'none'
        highcolor               CDATA
        '#FFFFFF'
        lowcolcor               CDATA
        '#000000'
        background      CDATA
        '#C0C0C0'
">
<!--
```

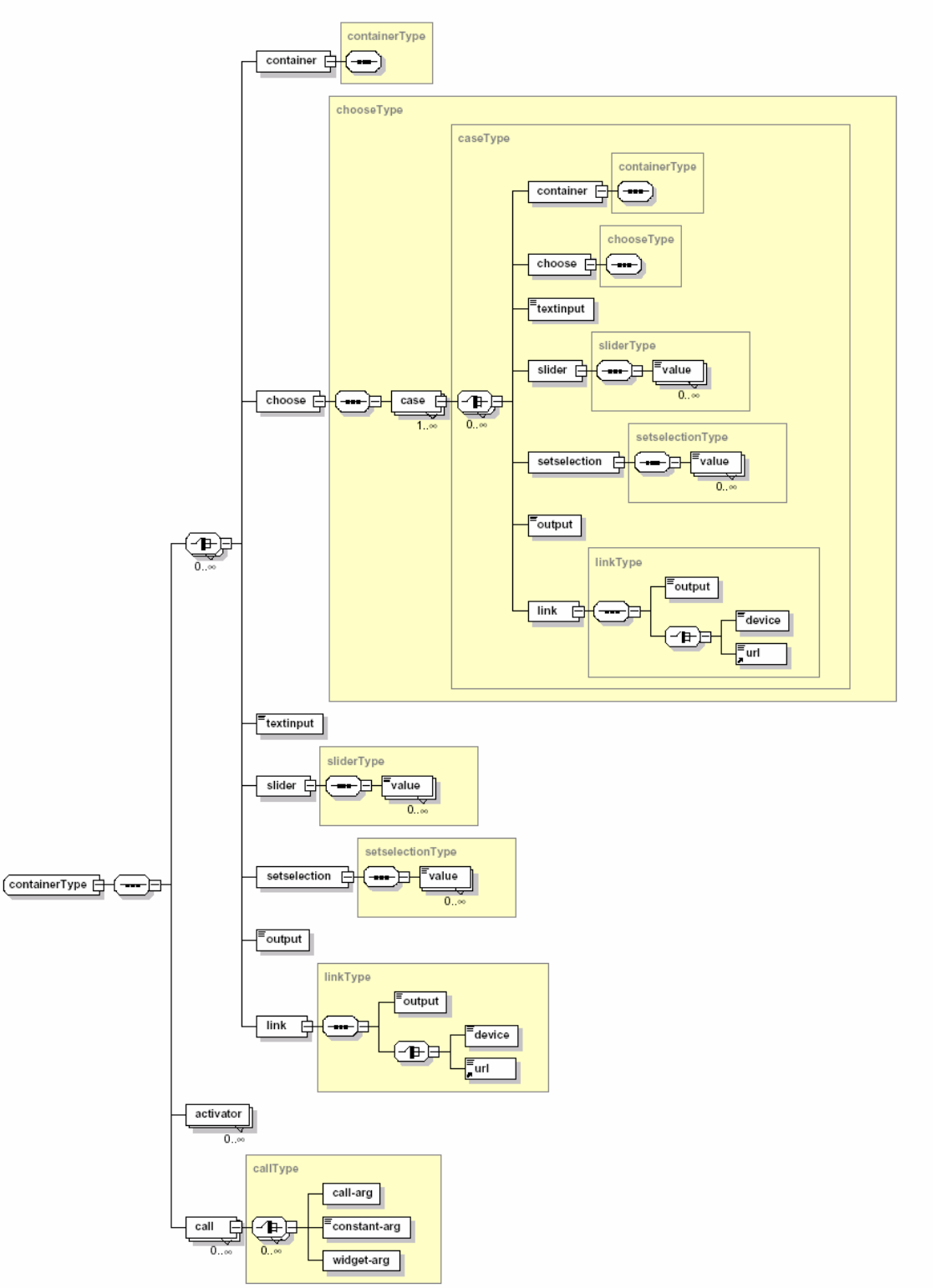
```
        The TextStyle entity declares attributes, which describe the
        appearance of text on a user interface..
-->
<!ENTITY % TextStyle
"       textalign       ( left | center | right )                                   'center'
        textcolor       CDATA
        '#000000'
        fontface        CDATA
        #IMPLIED
        fontsize        CDATA
        #IMPLIED
">
<!--
        The NamedId Entity declares an attribute, which if supplied to an element,
        ensures that it must be unique throughout the document. Thus the author
        of the document can give an element a unique identifier, which makes
        it easier to refer to it from another element,
-->
<!ENTITY % NameId
"       name                    ID
                        #IMPLIED
">
<!-- ################ Elements ################    -->
<!--
        The container element is the root element. It can contain a header,
        it should have one if it is the root element, a lot of widgets (a container
        is also a widget), activator definitions and calls.
-->
<!ELEMENT     container ( (%Widgets;)*, activator*, call*)>
<!ATTLIST               container
        layout          (horizontal | vertical)
        #REQUIRED
        stretch         (yes|no)
        'yes'
        size            CDATA
        #IMPLIED
        span            CDATA
        '1'
        %BorderStyle;
>
<!--
        The header element describes a link. The link
        can be of type device or href. Href is a link
        to another URL. Device type links to devices,
        which can be a device named or a device class.
        The containing output element is rendered as link.
-->
<!ELEMENT     link (output?,(device | url))>
<!ATTLIST               link
        type            (device | url)
        'device'
>
<!--
        The href element contains the url to which the link points
-->
<!ELEMENT url (#PCDATA)>
```

```
<!--
        The device element describes a device. It can describe an instance
        of a device or a class of  devices. The class attribute contains
        the full qualified java interface name of the device class. If the device
        element contains any text, it will be interpreted as the name of an instance
        from a device belonging to the device class defined by this attribute.
-->
<!ELEMENT     device (#PCDATA)>
<!ATTLIST               device
        class           NMTOKEN
        #IMPLIED
>
<!--
        The choose element is similar to a switch case statement in C/C++.
        The value of the referenced variable will be used to decide which case
        should be rendered.
-->
<!ELEMENT     choose (case+)>
<!ATTLIST               choose
        call                    IDREF
                        #REQUIRED
>
<!--
        The case element is similar to a case within a switch statement in C/C++.
        The value attribute contains the value for which this case should be
        rendered. If there is no value attribute the case will be the default case.
        And if there is more than one case within a choose without a value
        attribute the first one will be taken as default.
-->
<!ELEMENT     case ((%Widgets;)*)>
<!ATTLIST               case
        value           CDATA
        #IMPLIED
>
<!--
        The text element describes a widget for textinput. The pattern attribute
        contains, if supplied, a regular pattern, which the input must match. The
        length attribute constraints the maximum input length of the text. If the textinput
        element contains any text, the text will be used as default input value, if
        it conforms to the any supplied constraints from pattern and/or length
        attribute. The
 -->
<!ELEMENT     textinput (#PCDATA)>
<!ATTLIST               textinput
        pattern         CDATA
        #IMPLIED
        length          CDATA
        #IMPLIED
        call                    IDREF
                        #IMPLIED
        value           CDATA
        #IMPLIED
        %NameId;
        %BorderStyle;
>
<!--
```

```
        The slider element describes a widget for continues selection, out
        of a range of data. Therefore only three containing value elements
        are necessary: one for the start, end, and optional a default value.
        If there are more containing value elements with a type of start/end or
        default, the first one appearing with that type will be used.
-->
<!ELEMENT     slider (value*)>
<!ATTLIST                slider
        %NameId;
        %BorderStyle;
>
<!--
        The setselction element describes a widget for discrete selection of data,
        out of a given set of values. With the multiple attributes it is possible to
        allow a user to select more than one value from the selection.
 -->
<!ELEMENT     setselection (value*)>
<!ATTLIST                setselection
        multiple (yes | no)
        'no'
        %NameId;
        %BorderStyle;
>
<!--
        The value element describes a value, which will be used from a parent
        element like setselection or slider. The element can be of specific type.
        E.g. default, to tell the parent element that this value should be supplied
        as a default value, if the user does not make any other selection. Or of start/end type
        for the slider to define the starting value and end value of the value range.
        The javatype attribute declares the java type of the value.
-->
<!ELEMENT     value (#PCDATA)>
<!ATTLIST                value
        type             (start | end | default)
        #IMPLIED
        javatype %JavaTypes;
        #REQUIRED
        display          CDATA
        #IMPLIED
>
<!--
        The output element describes a widget for output of any text or an image.
        If the image cannot be accessed, it will be tried to render the supplied text, contained
        within the output element. The span attribute lets the output element span
        multiple cells within a container, which ones depends on the layout of parent
        container.
-->
<!ELEMENT     output (#PCDATA)>
<!ATTLIST                output
        image            CDATA
        #IMPLIED
        span             CDATA
        '1'
        %BorderStyle;
        %TextStyle;
        %NameId;
```

```
>
<!--
        The activator element describes a hot spot within the user interface.
        It links an output element with a java method call element together. With this,
        it is possible to tell that the referenced output element should be rendered
        in a different way, mark it that user recognize it easier as a hot spot.
-->
<!ELEMENT     activator EMPTY>
<!ATTLIST                 activator
        output            IDREF
                  #REQUIRED
        call                        IDREF
                          #REQUIRED
        border            (none | line | bevelup)
        'none'
        markcolor         CDATA
        '#0000FF'
        highcolor         CDATA
        '#FFFFFF'
        lowcolcor         CDATA
        '#000000'
>
<!--
        The call element describes a java method, which can be called.
        The name of the method is defined with the method attribute.
        The containing elements will be passed as arguments to the method.
        The first children element will be the first argument, the second the second,...
-->
<!ELEMENT     call (call-arg | constant-arg | widget-arg)*>
<!ATTLIST                 call
        method  ID
                          #REQUIRED
        javatype %JavaTypes;
        #REQUIRED
>
<!--
        The call-arg defines that this argument should be taken from the result
        of another java call.
-->
<!ELEMENT     call-arg EMPTY>
<!ATTLIST                 call-arg
        call                        IDREF
                          #REQUIRED
>
<!--
        The constant-arg defines that this argument should be a constant, supplied
        within this element, and from the type define with the javatype attribute.
-->
<!ELEMENT     constant-arg (#PCDATA)>
<!ATTLIST                 constant-arg
        javatype %JavaTypes;
        #REQUIRED
>
<!--
        The widget-arg defines that this argument should be taken from a widget
        where the user made a selection or supllied a text, if the user didn't supply
```

```
        anything the default value will be taken. If even no default value exists
        an error will be forwarded to the user, telling him to make a decision.
-->
<!ELEMENT     widget-arg EMPTY>
<!ATTLIST                 widget-arg
        widget            IDREF
                  #REQUIRED
        javatype %JavaTypes;
        #REQUIRED
>
```


containerType
container
containerType
chooseType
caseType
containerType
container
chooseType
choose
textinput
sliderType
slider
value
0..∞
choose
case
1..∞
0..∞
setselectionType
setselection
value
0..∞
output
linkType
link
output
device
url
0..∞
textinput
sliderType
slider
value
0..∞
containerType
setselectionType
setselection
value
0..∞
output
linkType
link
output
device
url
activator
0..∞
callType
call
0..∞
0..∞
call-arg
constant-arg
widget-arg

Appendix D - AUI-ML Document for the SCLUB Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE container SYSTEM "/aui.dtd">
<container layout="horizontal" size="2">
    <container span="2" layout="horizontal" stretch="no">
        <link type="url">
            <output textalign="left" fontface="Arial, Helvetica, sans-serif" fontsize="2" image="/images/home.gif">
                Home
            </output>
            <url>/</url>
        </link>
        <link type="url">
            <output textalign="left" fontface="Arial, Helvetica, sans-serif" fontsize="2" image="/images/links.gif">
                Zurück
            </output>
            <url>..</url>
        </link>
    </container>
    <output textalign="left" fontface="Arial, Helvetica, sans-serif" fontsize="1">GMD-SIT COR</output>
    <output textalign="right" fontface="Arial, Helvetica, sans-serif" fontsize="1">SCLUB</output>
    <container span="2" layout="horizontal" size="5">
        <output span="2"/>
        <output textalign="right" image="/images/window_hori.gif"/>
        <output textalign="right" image="/images/window_hori.gif"/>
        <output/>
        <container layout="vertical">
            <output image="/images/window_vert.gif"/>
            <output image="/images/window_vert.gif"/>
```

```
        </container>
        <container span="2" layout="horizontal" background="#ADADAD">
            <link type="device">
                <device>Conference</device>
            </link>
            <link type="device">
                <device>Middle</device>
            </link>
        </container>
        <container layout="vertical" background="#ADADAD">
            <link type="device">
                <device>Window</device>
            </link>
            <link type="device">
                <device>Coffee</device>
            </link>
        </container>
        <container layout="vertical" background="#ADADAD">
            <output span="4" background="#848484"/>
            <output background="#ADADAD" fontsize="1" fontface="Arial, Helvetica, sans-serif">SCLUB</output>
        </container>
        <output span="3"/>
        <output textalign="left" image="/images/door.gif"/>
        <output/>
    </container>
    <link type="url">
        <output textalign="left" fontface="Arial, Helvetica, sans-serif" fontsize="2" image="/images/raumag.gif">
            RaumComputer AG
        </output>
        <url>http://www.raumcomputer.com</url>
    </link>
    <link type="url">
        <output textalign="center" fontface="Arial, Helvetica, sans-serif" fontsize="2" image="/images/help.gif">
            Help
        </output>
        <url>/help.html</url>
    </link>
</container>
```

Appendix E - Co-operation Platform – Software Packages

Remark: This Appendix gives an overview of the most important software packages developed within the context of this thesis. It only lists the package names and the class hierarchy inside. A more complete documentation, only listing the packages, the class hierarchy, the classes, and attributes and methods of each class, would have filled more than 1200 pages.

E.1 Co-operation Platform Packages – gmd.sit.vpo (Co-operation Platform, VPO, Human Centred Communication Services)

These packages do contain the code of the Co-operation Platform, the Virtual Project Office and the JTAPI-based human centred communication modules for handling media rich point-to-point communication (audio, video, and data) and unified messaging (email, voice mail, and fax).

Summary of Packages

- gmd.sit.vpo.client
- gmd.sit.vpo.client.imsconf
- gmd.sit.vpo.events
- gmd.sit.vpo.matrix
- gmd.sit.vpo.matrix.chat
- gmd.sit.vpo.matrix.deck
- gmd.sit.vpo.matrix.deck.eai
- gmd.sit.vpo.matrix.deck.sd
- gmd.sit.vpo.matrix.gmdsit
- gmd.sit.vpo.matrix.server
- gmd.sit.vpo.matrix.util
- gmd.sit.vpo.matrix.vrml
- gmd.sit.vpo.multiculture.beans
- gmd.sit.vpo.multiculture.db
- gmd.sit.vpo.multiculture.jini
- gmd.sit.vpo.multiculture.jtapi
- gmd.sit.vpo.multiculture.taglib
- gmd.sit.vpo.multiculture.transfer
- gmd.sit.vpo.multiculture.util
- gmd.sit.vpo.server
- gmd.sit.vpo.world

Class Hierarchy: gmd.sit.vpo

```
Class
|- - - Area
|- - - Avatar
|- - - Backend
|- - - BrowserSniffer
|- - - Component
|- - - - - - - Container
|- - - - - - - - - - - Panel
|- - - - - - - - - - - - - - - Applet
|- - - - - - - - - - - - - - - - - - - Chat
|- - - - - - - - - - - - - - - - - - - VotecClient
|- - - - - - - - - - - - - - - BorderPanel
|- - - - - - - - - - - - - - - - - - - LoginPanel
|- - - - - - - - - - - - - - - - - - - RunPanel
|- - - - - - - - - - - - - - - Frontend
|- - - - - - - - - - - Window
|- - - - - - - - - - - - - - - Frame
|- - - - - - - - - - - - - - - - - - - ChatFrame
|- - - - - - - - - - - - - - - - - - - InfoScreen
|- - - - - - - - - - - - - - - - - -
ReceptionWindow
|- - - ConfWrapper
|- - - Connection
|- - - - - - - ServerThread
|- - - EAIViewer
|- - - Environment
|- - - Field
|- - - - - - - FieldArray
|- - - - - - - MFColor
|- - - - - - - MFFloat
|- - - - - - - MFInt
|- - - - - - - MFRotation
|- - - - - - - MFString
|- - - - - - - MFVecf
|- - - - - - - MFVecf
|- - - - - - - SFBool
|- - - - - - - SFColor
|- - - - - - - SFFloat
|- - - - - - - SFImage
|- - - - - - - SFInt
|- - - - - - - SFRotation
|- - - - - - - SFString
|- - - - - - - SFTime
|- - - - - - - SFVecf
|- - - - - - - SFVecf
|- - - InputStream
```

|- - - - - - - FilterInputStream
|- - - - - - - - - - - - DataInputStream
|- - - - - - - - - - - - - - - - FieldInputStream
|- - - JTAPIClient
|- - - Log
|- - - MC_CallDealer
|- - - MC_InCallDealer
|- - - MC_InCallNotifier
|- - - MC_JTelServant
|- - - MC_OutCallDealer
|- - - MC_OutCallNotifier
|- - - MC_ProviderDealer
|- - - MC_ProviderNotifier
|- - - MathUtil
|- - - Message
|- - - - - - - ControlMessage
|- - - - - - - RealTimeMessage
|- - - OutputStream
|- - - - - - - FilterOutputStream
|- - - - - - - - - - - - DataOutputStream
|- - - - - - - - - - - - - - - - FieldOutputStream
|- - - PersonalAgent
|- - - RealTimeHandler
|- - - RemoteObject
|- - - - - - - RemoteServer
|- - - - - - - - - - - - UnicastRemoteObject
|- - - - - - - - - - - - - - - - BasicUnicastAdmin
|- - - - - - - - - - - - - - - - Listener
|- - - RoomAgent
|- - - RoomManager
|- - - TeamAgent
|- - - TelephonyProxy
|- - - Thread
|- - - - - - - MatrixD
|- - - Throwable
|- - - - - - - Exception
|- - - - - - - - - - - - ConsumerException
|- - - - - - - - - - - - IOException
|- - - - - - - - - - - - - - - -
UnknownMessageException
|- - - - - - - - - - - - InvalidFieldException
|- - - - - - - - - - - - RuntimeException
|- - - - - - - - - - - - - - - -
UnsupportedFieldTypeException
|- - - Timetable
|- - - Tools
|- - - VotecEvent
|- - - - - - - CloseDocEvent
|- - - - - - - EnterZoneEvent
|- - - - - - - LeaveZoneEvent
|- - - - - - - LogoffEvent
|- - - - - - - LogonEvent
|- - - - - - - OpenDocEvent
|- - - - - - - ShareDocEvent
|- - - - - - - TelephoneConnection
|- - - - - - - UnshareDocEvent
|- - - VotecServer

MC_OutCallObserver
MC_InCallObserver
MC_UserInCallListener
MC_UserProviderListener
MC_ProviderObserver
MC_UserOutCallListener
BasicUnicastService.RestoreListener
Telephony
RemoteTelephony
BasicUnicastService
|- - - JTAPIService
TKListener
JTAPIServiceEvent
Reception
CoffeeHandler
DeskHandler
VideoHandler
VRMLEventHandler
UserAvatar
WalkControl
BetaX
BetaXPanel
Beta
Gate
Route
UserAvatar

SharedObject
Viewer
UserInterfaceListener
NetworkListener
ViewerListener
RealTimeConsumer
GateListener
Network
UserInterface
INmCall
INmSysInfo
tagNmShareState
tagNmCallState
INmMember
INmChannelData
INmChannel
tagNmCallType
IEnumNmCall
NmManager
tagNmSysProp
INmConference
IEnumNmChannel
tagNmConferenceState
IEnumNmMember
tagNmAddrType
IEnumNmConference
INmManage

E.2 Co-operation Platform Packages – com.mcrs (Multimedia Conference Reservation and Management System)

These packages do contain the code for the Multimedia Conference Reservation and Management System (MCRS).

Summary of Packages

- com.mcrs.applet
- com.mcrs.bauer.comm
- com.mcrs.conference
- com.mcrs.directory_service
- com.mcrs.file
- com.mcrs.ldap
- com.mcrs.mcu
- com.mcrs.modem
- com.mcrs.protocol
- com.mcrs.protocol.sdp
- com.mcrs.protocol.sip
- com.mcrs.rmi
- com.mcrs.server
- com.mcrs.system
- com.mcrs.text
- com.mcrs.time
- com.mcrs.utility

Class Hierarchy: com.mcrs

```
Class
|--- CBasicConference
|--- CBasicConferenceMember
|--- CBasicIETFProtocolReader
|--- CBasicIETFProtocolWriter
|--- CBitOption
|--- CClient
|--- CConferenceSettings
|--- CConnection
|------ CClientConnection
|------ CServerConnection
|--- CConvenerPanel
|--- CDate
|--- CFile
|--- CMemberPanel
|--- CModem
|--- CMyPanel
|--- CPropertyFile
|--- CReservation
|--- CSDPMediaDescription
|--- CSDPTimeDescription
|--- CServer
|--- CSettingPanel
|--- CString
|--- CSystem
|--- CTable
|------ CDebug
|------ CProperty
|--- CTime
|--- CTimePanel
|--- CTimer
|--- CommDemon
|--- Component
|------ Container
|--------- Panel
|------------ Applet
|--------------- CApplet
|--------------- CRCSRMIMain
|--------------- CWEBPage
|--------------- WebReservation
|--------- Window
|------------ Dialog
|--------------- RMIdlg
|------------ Frame
|--------------- CInternetHost
|---------------
CReservationServer
|--------------- CWEBServer
|--------------- MCUServer
|--- MCUError
```

```
|--- MCUReservation                    |--------- CTimeException
|--- MCUResponse                       |--------- MCUException
|--- RAI                               |--------- ModemException
|--- RemoteObject                      |--- WinPort
|------ RemoteServer                   |--- Writer
|--------- UnicastRemoteObject         |------ PrintWriter
|------------ CDaemon                  |--------- CCallStack
|------------ CRMIServer               CServerInterface
|------------ Server                   DirectoryInterface
|--- Thread                            CRCSRMIRoot
|------ CSIPServer                     ReservationInterface
|--- Throwable                         CSIPReader
|------ Exception                      CSIPDaemon
|--------- CClientServerException      SIPInterface
|--------- CDateException              CSDPWriter
|--------- CFileException              SDPInterface
|--------- CPropertyException          CSDPReader
|--------- CPropertyFileException      LDAPServerInterface
|--------- CProtocolException          CDirectoryService
|--------- CRMIException                ServerInterface
|--------- CStringException            CLDAPServer
|--------- CTableException             ReservationServerInterface
```

E.3 Co-operation Platform Packages – de.gmd.de.darmstadt.dia (Office Identification Card)

These packages do contain the code for the Office Identification Card (OIC).

Summary of Packages

- de.gmd.darmstadt.dia
- de.gmd.darmstadt.dia.api
- de.gmd.darmstadt.dia.api.icc
- de.gmd.darmstadt.dia.api.ldap
- de.gmd.darmstadt.dia.model
- de.gmd.darmstadt.dia.view
- de.gmd.darmstadt.dia.exceptions
- de.gmd.smartcard.terminal.GDMICT
- de.gmd.smartcard.terminal.MIFARE

Class Hierarchy: de.gmd.darmstadt.dia

```
Class
|- - - AbstractListModel
|- - - - - - DefaultListModel
|- - - - - - - - - DefaultCmdListModel
|- - - AccessCondition
|- - - BasicAttribute
|- - - - - - TupelVVAttribute
|- - - Component
|- - - - - - Container
|- - - - - - - - - JComponent
|- - - - - - - - - - - - JPanel
|- - - - - - - - - - - - - - - - CardHolderPanel
|- - - - - - - - - - - - - - - - CmdListPanel
|- - - - - - - - - - - - - - - - DimensionPanel
|- - - - - - - - - - - - - - - - EnabledPanel
|- - - - - - - - - - - - - - - - - - - - BorderedPanel
|- - - - - - - - - - - - - - - - - - - - MaskPanel
|- - - - - - - - - - - - - - - -
RRRelationSettingsPanel
|- - - - - - - - - - - - - - - - RightSettingsPanel
|- - - - - - - - - - - - - - - - RoleSettingsPanel
|- - - - - - - - - - - - - - - - ShowCardPanel
|- - - - - - - - - - - - JScrollPane
|- - - - - - - - - - - - - - - - EnabledScrollPane
|- - - - - - - - - Window
|- - - - - - - - - - - - Frame
|- - - - - - - - - - - - - - - - JFrame
|- - - - - - - - - - - - - - - - - - -
FileChooserDemo
|- - - - - - - - - - - - - - - - - - - MainFrame
|- - - - - - - - - - - - - - - - - - - PanelToFrame
|- - - CreateData
|- - - - - - CreateDFData
|- - - - - - CreateEFData
|- - - - - - CreateEndData
|- - - - - - CreateMFData
|- - - FileConfigSource
|- - - IccDia
|- - - LdapClient
|- - - - - - LdapConfigSource
|- - - - - - LdapCryptoKeySource
|- - - - - - LdapDia
|- - - ListCtrl
|- - - - - - RightCtrl
|- - - - - - RoleCtrl
|- - - Observable
|- - - - - - CmdListModel
|- - - - - - - - - Berechtigungsraum
|- - - - - - - - - Dimension
|- - - - - - - - - Rollenraum
|- - - - - - DiaData
|- - - - - - Rolle
|- - - Throwable
|- - - - - - Exception
|- - - - - - - - - DiaException
|- - - - - - - - - - - - DataSourceException
|- - - - - - - - - - - - - - - -
DataCannotBeSignedException
|- - - - - - - - - - - - - - - -
DataCannotBeVerifiedException
|- - - - - - - - - - - - - - - -
DataNotCompleteException
|- - - - - - - - - - - - - - - -
DataNotValidException
|- - - - - - - - - - - - WrongValueException
```

|- - - TupelVV
|- - - TupelValueName
|- - - - - - DimTupelValueName
|- - - Utilities
HasValueAndName
LdapConstants
GDCardServiceFactory
IccConstants
WriteKeyAPDU
GDFileSystemCardService
GDFileAccessCardService
ConfigDataSource
DiaDataSource
Constants
CryptographicKeySource
DiaDataMultiSource

E.4 Co-operation Platform Packages – com.raumcomputer (RoomComputer)

These packages do contain the code for the RoomComputer.

Summary of Packages

- com.raumcomputer
- com.raumcomputer.configuration
- com.raumcomputer.configuration.parser.bus
- com.raumcomputer.device
- com.raumcomputer.device.sct
- com.raumcomputer.driver
- com.raumcomputer.driver.room
- com.raumcomputer.interfaces
- com.raumcomputer.repository
- com.raumcomputer.service
- com.raumcomputer.servlet
- com.raumcomputer.util

Class Hierarchy: com.raumcomputer

Class
|- - - AbstractCollection
|- - - - - - AbstractList
|- - - - - - - - - Vector
|- - - - - - - - - - - - SortedQueue
|- - - Argument
|- - - - - - CallArg
|- - - - - - ConstantArg
|- - - - - - WidgetArg
|- - - BaseConfigContentHandler
|- - - - - - BusConfigContentHandler
|- - - BaseDriver
|- - - - - - GroupingContainerDriver
|- - - - - - RoomDriver
|- - - BaseService
|- - - - - - DriverLocator
|- - - - - - DumpOsgiEnv
|- - - BitsNBytes
|- - - BusConfigurationImpl
|- - - ConfigChangedEvent
|- - - CoreActivator
|- - - Device
|- - - DeviceConfigurationImpl
|- - - Entry
|- - - - - - Activator
|- - - - - - Call
|- - - - - - TransformResult
|- - - - - - Variable
|- - - GenericServlet
|- - - - - - HttpServlet
|- - - - - - - - - RoomServlet
|- - - - - - - - - TestServlet
|- - - OSGIServiceCache
|- - - ReconfigureServiceActivator
|- - - Repository
|- - - RoomServletActivator
|- - - TestServletActivator
|- - - Throwable
|- - - - - - Exception
|- - - - - - - - - RaumComputerException
|- - - - - - - - - - - - BusDeviceException
|- - - - - - - - - - - - IncompatibleSettingException
|- - - - - - - - - - - - SctException
|- - - - - - - - - UndefinedArgument
|- - - VirtualDevice
|- - - - - - BaseDevice
|- - - - - - - - - ReconfigureService
|- - - - - - ContainerDevice
|- - - - - - - - - GroupingContainerDevice
|- - - - - - - - - - - - RoomDevice
|- - - - - - SymbolicDevice
AUIContentHandler
BusConfiguration
DeviceConfiguration
ConfigChangeListener
IReconfigureService
ConfigContentHandler
ISensorDevice
IMifareSctDevice

IThermoSensorDriver
IRoomDriver
IActorDevice
ISymbolicDevice
IRSDriver
IBusDriver
ISubjectDevice
ISctDriver
ILightDriver
ISensorDriver
IMifareSctDriver
IGroupingContainerDevice
IPollBusDriver
IPollBusDevice

IActorDriver
BusDeviceListener
IVirtualDevice
ISubjectDriver
IThermoSensorDevice
IRoomDevice
IContainerDevice
IBaseDevice
IBusDevice
IRSDevice
ISctDevice
ILightDevice

E.5 Co-operation Platform Packages – congo (Media Control System for the CoMeet Room)

These packages do contain the code for the CoMeet Mediacontrol System (congo).

Summary of Packages

- congo
- congo.broker
- congo.broker.deviceagents
- congo.broker.deviceagents. audiocontrol
- congo.broker.deviceagents. cameracontrol
- congo.broker.deviceagents.diacontro
- congo.broker.deviceagents.foliacame racontrol
- congo.broker.deviceagents. genericcontrol
- congo.broker.deviceagents. microphoncontrol
- congo.broker.deviceagents. projectioncontrol
- congo.broker.deviceagents.telephonc ontrol
- congo.broker.deviceagents.videoplay control
- congo.broker.profile
- congo.broker.scenario
- congo.broker.session
- congo.client
- congo.client.controller.agents
- congo.client.controller.global
- congo.client.view.agents
- congo.client.view.global
- congo.common
- congo.server.admin
- congo.server.admin.panel
- congo.server.admin.render
- congo.server.controll
- congo.server.database
- congo.server.database.table
- congo.server.device
- congo.server.hardware
- congo.server.management
- congo.server.mediatypes
- congo.util

Class Hierarchy: congo

Class
|- - AbstractDocument
|- - - - PlainDocument
|- - - - - -
WholeNumberField.WholeNumberDocument
|- - AbstractListModel
|- - - - DeviceList
|- - - - NamedVector
|- - - - SetEditor.ScenarioModel
|- - AbstractTableModel
|- - - - TableMap
|- - - - - - TableSorter
|- - ActionReport
|- - Admin
|- - AdvertisementInfo
|- - Analyst
|- - Barrier
|- - ButtonFactory
|- - Client
|- - Component
|- - - - Container
|- - - - - - JComponent
|- - - - - - - - JLabel
|- - - - - - - - - - DeviceRenderer
|- - - - - - - - - - MediatypeRenderer
|- - - - - - - - - - ObjectRenderer
|- - - - - - - - - - PortRenderer
|- - - - - - - - JPanel
|- - - - - - - - - - AdjusterPanel
|- - - - - - - - - - DevicePanel
|- - - - - - - - - - - - AudioControlPanel
|- - - - - - - - - - - - CameraControlPanel
|- - - - - - - - - - - - DiaControlPanel
|- - - - - - - - - - - - FoliaCameraControlPanel
|- - - - - - - - - - - - GenericControlPanel
|- - - - - - - - - - - - MicrophonControlPanel
|- - - - - - - - - - - - ProjectionControlPanel
|- - - - - - - - - - - - TelephonControlPanel
|- - - - - - - - - - - - VideoPlayControlPanel
|- - - - - - - - - - Panel
|- - - - - - - - - - - - ListPanel
|- - - - - - - - - - - - - - PanelObject
|- - - - - - - - - - - - PanelDatabase
|- - - - - - - - - - - - PanelProfile
|- - - - - - - - - - - - TablePanel
|- - - - - - - - - - - - - - PanelDevice
|- - - - - - - - - - - - - - PanelHandle
|- - - - - - - - - - - - - - PanelHandle
|- - - - - - - - - - - - - - PanelPort
|- - - - - - - - - - - - - - PanelResource
|- - - - - - - - - - - - - - PanelScenario
|- - - - - - - - - - - - - - PanelWire
|- - - - - - - - - - - - - - PanelWire
|- - - - - - - - - - ScenarioPanel
|- - - - - - - - - - SelectionPanel
|- - - - - - - - JTextComponent
|- - - - - - - - - - JTextField
|- - - - - - - - - - - - WholeNumberField
|- - - - - - Window
|- - - - - - - - Dialog
|- - - - - - - - - - JDialog
|- - - - - - - - - - - - SetEditor
|- - - - - - - - - - - - StringListEditor
|- - - - - - - - Frame
|- - - - - - - - - - JFrame
|- - - - - - - - - - - - LoginFrame
|- - - - - - - - - - - - MainFrame
|- - Controller
|- - - - Mute
|- - - - - - MultiMute
|- - - - Volume
|- - - - - - VolumeAbsolute
|- - Database
|- - - - AdminDatabase
|- - - - MakeWorld
|- - DeviceAgentManager
|- - DeviceConfig
|- - DeviceManager
|- - - - DebugDeviceManager
|- - DeviceManager.DriverInit
|- - DeviceManager.WireInit
|- - DeviceVisuals
|- - Dump
|- - DumpDriver
|- - HandleCache
|- - HandleCache.HandleEntry
|- - Jojodatabase
|- - LoginFrameController
|- - LoginInfo
|- - MainFrameController
|- - Mediatype
|- - MessageRecorder
|- - NamedItem
|- - - - NamedStateItem
|- - - - - - DevicePanelController
|- - - - - - - - AudioControlDPC
|- - - - - - - - CameraControlDPC
|- - - - - - - - DiaControlDPC
|- - - - - - - - FoliaCameraControlDPC

|- - - - - - - - GenericControlDPC
|- - - - - - - - MicrophonControlDPC
|- - - - - - - - ProjectionControlDPC
|- - - - - - - - TelephonControlDPC
|- - - - - - - - VideoPlayControlDPC
|- - - - - - ScenarioPanelController
|- - - - - - SelectionPanelController
|- - - - ParticipantInfo
|- - - - ScenarioAdvertisement
|- - - - ScenarioAmb
|- - - - Set
|- - ObjTest
|- - Observable
|- - - - Connector
|- - - - - - Device
|- - - - - - - - ControlableDevice
|- - - - - - Handle
|- - - - DeviceAgentAmb
|- - - - - - AudioControlAmb
|- - - - - - CameraControlAmb
|- - - - - - DiaControlAmb
|- - - - - - FoliaCameraControlAmb
|- - - - - - GenericControlAmb
|- - - - - - MicrophonControlAmb
|- - - - - - ProjectionControlAmb
|- - - - - - TelephonControlAmb
|- - - - - - VideoPlayControlAmb
|- - - - NamedStateVector
|- - - - Port
|- - - - - - TCPIP
|- - Parser
|- - - - ParserArray
|- - - - ParserChar
|- - - - ParserGroup
|- - - - ParserInteger
|- - - - ParserString
|- - PortFactory
|- - Profile
|- - Range
|- - RemoteObject
|- - - - RemoteServer
|- - - - - - UnicastRemoteObject
|- - - - - - - - ProfileManager
|- - - - - - - - RMIObservable
|- - - - - - - - - - DeviceAgent
|- - - - - - - - - - - - AudioControl
|- - - - - - - - - - - - CameraControl
|- - - - - - - - - - - - DiaControl
|- - - - - - - - - - - - FoliaCameraControl
|- - - - - - - - - - - - GenericControl
|- - - - - - - - - - - - MicrophonControl
|- - - - - - - - - - - - ProjectionControl
|- - - - - - - - - - - - TelephonControl
|- - - - - - - - - - - - VideoPlayControl
|- - - - - - - - RMIObserver
|- - - - - - - - Scenario
|- - - - - - - - - - ScenarioAudioConference
|- - - - - - - - - - ScenarioAudioPlay
|- - - - - - - - - - ScenarioAudioProtocol
|- - - - - - - - - - ScenarioComputerShow
|- - - - - - - - - - ScenarioDiaShow
|- - - - - - - - - - ScenarioFoliaShow
|- - - - - - - - - - ScenarioVideoConference
|- - - - - - - - - - ScenarioVideoProtocol
|- - - - - - - - - - ScenarioVideoShow
|- - - - - - - - Selection
|- - - - - - - - Session
|- - - - - - - - SessionManager
|- - SQLColumn
|- - SQLEntry
|- - - - SQLEntryNew
|- - - - SQLEntryOld
|- - SQLHelper
|- - SQLTable
|- - - - SQLTableModel
|- - - - - - DeviceTable
|- - - - - - HandleTable
|- - - - - - HandleTable
|- - - - - - ObjectList
|- - - - - - PortTable
|- - - - - - ResourceTable
|- - - - - - ResourceTable
|- - - - - - ScenarioTable
|- - - - - - WireTable
|- - - - - - WireTable
|- - Scenario.MyConnection
|- - ScenarioDescriptor
|- - ScenarioListener
|- - ScenarioManager
|- - SelectionAmb
|- - Server
|- - SessionAmb
|- - StringTokenizer
|- - Suite
|- - Switchbox
|- - - - DebugSwitchbox
|- - ThreadLister
|- - Throwable
|- - - - Exception
|- - - - - - CongoException
|- - - - - - - - FatalException
|- - - - - - - - - - ExternalException
|- - - - - - - - - - InternalException
|- - - - - - - - - - - - WrongStateException
|- - - - - - - - NonFatalException
|- - - - - - - - - - AccessDeniedException
|- - - - - - - - - - AgentNotFoundException
|- - - - - - - - - - DPCNotFoundException

|- - - - - - - - - -
DevicesNotConnectableException
|- - - - - - - - - -
NoDevicesAvailableException
|- - - - - - - - - -
ScenarioNotAvailableException
|- - - - - - ControllerException
|- - - - - - DeviceCommunicationException
|- - - - - - IllegalItemException
|- - - - - - ItemExistsException
|- - - - - - ItemNotFoundException
|- - - - - - MissingMediaException
|- - - - - - NoSuchAdvertisementException
|- - - - - - NoSuchConnectionException
|- - - - - - ResourceConflictException
|- - - - - - ScenarioNotFoundException
|- - - - - - UnconnectableException
|- - - - - - UnknownDeviceException
|- - - - - - WrongMediatypeException
|- - UserManager
|- - Wire
|- - WireVisitor
NamedItemInterface
NamedStateItemInterface
ControllableConnection
Connection
ReceiveListener
PortFactoryInterface
HandleReceiveListener
ResourceEntry
ScenarioEntry
HandleCallback
DeviceAgentCallback
PortCallback
WireCallback
DeviceCallback
DatabaseInterface
Subcontroller
DeviceDescription
ClientRemoteInterface
SessionManagerInterface
SessionInterface
SessionAdminInterface
ScenarioInterface
SelectionInterface
ProfileAdminInterface
VideoPlayControlInterface
TelephonControlInterface
ProjectionControlInterface
MicrophonControlInterface
GenericControlInterface
FoliaCameraControlInterface
DiaControlInterface
CameraControlInterface
AudioControlInterface
RMIObservableInterface
DeviceAgentInterface
RMIObserverInterface

E.6 Co-operation Platform Packages – smartlibrary (Sm@rtLibrary)

These packages do contain the code for the Sm@rtLibrary.

Summary of Packages

- smartlibrary.jdia
- smartlibrary.rfid
- smartlibrary.rfid.kronegger
- smartlibrary.sitxsputils

Class Hierarchy: smartlibrary

Class

DiaData
DiaString
EscapeChars
ICodeAccess
RfidReaderCom
|- - ICodeMicro
RfidReaderFactory
|- - ICodeMicroFactory
RfidApplet
Throwable
|- - Exception
|- - - - RfidException
|- - - - RfidReaderException
|- - - - - - - - - RfidInsufficientStorageException
|- - - - - - - - - RfidNoTagException
|- - - - - - - - - RfidNoWriteAccessException
RfidReader
Dia
IDiaConst