DYNAMIC GROUPWARE SERVICES

MODULAR DESIGN OF TAILORABLE GROUPWARE

ROBERT SLAGTER

Telematica Instituut,
On top of technology.

This publication is part of the Telematica Instituutís Fundamental Research Series. Telematica Instituut is a unique, co-operative venture involving businesses, scientific research institutions and the Dutch government, which carries out research into telematics, a field also known as information and communication technology (ICT).

The institute focuses on the rapid translation of fundamental knowledge into market-oriented applications such as electronic business practice, electronic co-operation and electronic data retrieval. We are also involved in the development of middleware.

In a financial, administrative and substantive alliance, the parties involved work on a joint mission, namely the development and application of high-quality telematics knowledge. By combining forces in an international network of this kind, we will be in a better position to deploy the opportunities of telematics so that ICT can make a contribution to strengthening the innovative capacity of business and industry.

In addition, telematics will fundamentally change and facilitate our way of living, working and even spending our leisure time. In all of these activities, technology remains the servant of man: On top of technology. The Dutch government also recognizes the importance of information and communication technology, and awarded Telematica Instituut the title 'leading technological institute'.

www.telin.nl

Dynamic Groupware Services

Modular design of tailorable groupware

Robert J. Slagter

Telematica Instituut

Enschede, The Netherlands, 2004

Telematica Instituut Fundamental Research Series, No. 011 (TI/FRS/011)

Cover Design: Studio Oude Vrielink, Losser and Jos Hendrix, Groningen
Book Design: Lidwien van de Wijngaert and Henri ter Hofte
Printing: Universal Press, Veenendaal, The Netherlands

Telematica Instituut Fundamental Research Series (see also: http://www.telin.nl/publicaties/frs.htm)

001 G. Henri ter Hofte, *Working apart together: Foundations for component groupware*
002 Peter J.H. Hinssen, *What difference does it make? The use of groupware in small groups*
003 Daan D. Velthausz, *Cost-effective network-based multimedia information retrieval*
004 Lidwien A.M.L. van de Wijngaert, *Matching media: information need and new media choice*
005 Roger H.J. Demkes, *COMET: A comprehensive methodology for supporting telematics investment decisions*
006 Olaf Tettero, *Intrinsic information security: Embedding security issues in the design process of telematics systems*
007 Marike Hettinga, *Understanding evolutionary use of groupware*
008 Aart T. van Halteren, *Towards an adaptable QoS aware middleware for distributed objects*
009 Maarten Wegdam, *Dynamic reconfiguration and load distribution in component middleware*
010 Ingrid J. Mulder, *Understanding designers, designing for understanding*

Samenstelling promotiecommissie:

Voorzitter, secretaris:	prof. dr. W.H.M. Zijm (Universiteit Twente)
Promotor:	prof. dr. ir. C.A. Vissers (Universiteit Twente)
Deskundige:	dr. ir. G.H. ter Hofte (Telematica Instituut)
Leden:	prof. dr. ir. E. Backer (Technische Universiteit Delft)
	prof. dr.-ing. J.M. Haake (FernUniversität Hagen, Duitsland)
	dr. A.I. Mørch (Universiteit van Oslo, Noorwegen)
	dr. ir. M.J. van Sinderen (Universiteit Twente)
	prof. dr. R.J. Wieringa (Universiteit Twente)

ISSN 1388-1795; No. 011
ISBN 90-75176-37-6

Telematica Instituut, P.O. Box 589, 7500 AN Enschede, The Netherlands
E-mail: info@telin.nl; Internet: http://www.telin.nl
Telephone: +31 (0)53-4850485; Fax: +31 (0)53-4850400

DYNAMIC GROUPWARE SERVICES
MODULAR DESIGN OF TAILORABLE GROUPWARE

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof. dr. F.A. van Vught,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op donderdag 26 augustus 2004 om 15.00 uur

door

Robert-Jasper Jurriaan Slagter
geboren op 31 augustus 1974
te Amersfoort

Dit proefschrift is goedgekeurd door de promotor, prof.dr.ir. C.A. Vissers.

Preface

The research presented in this book combines two of my personal interests: exploring and supporting human co-operation, and designing usable technology. I am intrigued by the dynamics of real-life co-operative settings, where exceptions seem to be the rule. Designing appropriate groupware to support such dynamic processes is quite a challenge. Based on my background in computer science and cognitive ergonomics, I focus on a solution that empowers the co-operating people themselves to adapt the behaviour of a groupware application. In this way, users can adapt the technology to match their changing needs and personal preferences.

My interest into computer supported co-operative work has been fostered by Henri ter Hofte: he introduced me to many interesting aspects of the research domain, as well as many interesting people. As a coach, he provided the trusted foundation and acted as a sparring partner throughout the whole process. Henri, thank you for being an inspirator, not a manager. I would like to thank Chris Vissers, my PhD supervisor, for finding the time for our discussions, for teaching me scrutiny in the use of concepts and abstraction levels, and illustrating all this by colourful anecdotes. Dick Quartel helped me to sharpen the descriptions and saw to it that the AMBER concepts were not misused. Furthermore, I would like to thank all members of the doctoral committee for their feedback on the dissertation. By programming the CoCoWare .NET platform, even when the specifications were still dynamic, Hans Kruse and Martin Snijders have permitted me to demonstrate my ideas. My gratitude goes to all test subjects for their valuable input during the evaluation.

The Telematica Instituut provided a stimulating breeding ground for the research, while the symbiosis between PhD research and project work allowed me to test my ideas in real-life settings. I would like to thank my Telematica Instituut colleagues, especially the TAO members, for providing an important part of the scientific as well as social context for my research.

Margit and Ynze, it is great to have you around for all the good fun and countless though-provoking discussions.

Of course, the foundation for all this has been provided by my family: Esther, Roel and Ruben. They have always believed in me and supported me to pursue my dreams. I thank Maarten, Remco, Bri, Jasper, Peter and all my other friends for being there and all the great fun we have had all these years. My thanks go to the TI borrelclub, for providing the essential distraction from the serious stuff on Fridays. This network of family and friends helped me to persevere in my PhD research, while not loosing touch with the more important things in life.

Robert Slagter
Enschede, the Netherlands, August 2004

Contents

Chapter 1

Introduction

Society acquires much of its character from the way in which people interact (Ellis, Gibbs, & Rein, 1991). Technology allows people to interact while bridging differences in location and time. Innovative technology helps us to interact in ways that fit our purposes; technology can even augment face-to-face collaboration, as demonstrated by group decision support systems that can improve group performance (Zigurs & Buckland, 1998).

Task-technology fit

However, it is not trivial to design appropriate technological support for groups of interacting people. One of the main reasons for this seems to be the required *task-technology fit* (Zigurs & Buckland, 1998). Applied to the domain of computer-supported co-operative work, the task-technology fit theory states that the support for co-operation provided by technology has to match the co-operative tasks people perform. People should have sufficiently rich support for their co-operative tasks, but at the same time information overload should be avoided. This theory is in line with research findings by McGrath & Hollingshead (1993) and Kahler, Mørch, et al. (2000). Based on the task-technology fit theory, we assume that in co-operative settings, different tasks require different types of technological support.

As a complicating factor, there seems to be a real and inherent gap between what we know we must support socially and what we can support technically. This gap is called the *social-technical gap*. It is considered one of the fundamental intellectual challenges in computer supported co-operative work research (Ackerman, 2000).

An important cause of this gap is the inherent dynamics in the way people interact: co-operating groups are dynamic (McGrath & Hollingshead, 1993), the tasks they co-operatively perform change over time, as does the context in which they perform these tasks (Kahler, Mørch, Stiemerling, & Wulf, 2000; Stiemerling, 2000; Hettinga & Heeren, 1998). As a result, the requirements for the technology to support co-operating people change over time. Although it is not possible to fully predict these

changes, groupware applications can be designed in such a manner that they can be adapted to match changing requirements.

The co-operating end users are the ones most affected when the technology no longer matches their tasks. In such cases, the end users could contact a groupware developer or a system administrator to solve the issue. Apart from the extra dependency, introducing an external person is an investment in terms of time, money and communication needed to explain the issue. If the end user is empowered to perform some frequently needed adaptations herself, the introduction of an external person may, in some cases, not be necessary.

For this reason, our research focusses on allowing the end users themselves to select and combine the groupware behaviour that fits their needs, rather then escalating each mismatch to groupware developers or system administrators. Such adaptations by end users are typically denoted as *tailoring* (Mørch, 1997b).

Main issue with current groupware applications

Chapter 2 illustrates that although various initiatives exist to design tailorable groupware, the main issue with current groupware applications is the lack of flexibility in the services they provide: *end-users should be provided with appropriate groupware services to support their dynamic co-operative tasks*. We assume that tailorability is beneficial to obtain and maintain a task-technology fit during computer-supported co-operation. Our work provides a structuring of groupware services in order to design tailorable groupware and to allow end users to select and compose the groupware behaviour they need while co-operating.

1.1 Research objective

Main objective

To solve the main issue, the research objective of this book is to design tailorable groupware services: groupware services that can be adapted by the co-operating end users.

This main objective has been split into two sub-objectives, with different target groups:

1. To provide co-operating *end users* with appropriate means to tailor the provided groupware service in order to suit their changing needs and personal preferences.
2. To provide groupware *designers* with support to design tailorable groupware services. Groupware designers should be provided with a structuring of groupware services, for instance in the form of a groupware service reference model, in order to support them in designing groupware services that can be tailored to match dynamic requirements.

The research takes the actual co-operation as a starting point: the behaviour provided by a groupware application should match the actual requirements of co-operating people and reflect their tasks and preferences. As such, the co-operating people, their co-operative activities as well as the dynamics in the way people co-operate form important sources of design requirements. Apart from this, our research focusses on the process of service design for computer-supported co-operative work.

1.2 Research questions

To achieve these objectives, we investigate the following main research question:

Main research question

How can groupware be designed in order to achieve tailorability of groupware services?

This main research question has been divided into four sub-questions:

1. What services does a groupware application have to provide to co-operating *end users*?
2. What adaptability of groupware services is needed to support real life co-operation?
3. How can groupware be designed in such a way that *end users* can adapt the provided groupware service?
4. Given a groupware service design based on functional modules:
   - 4a. What high-level, functional modules are appropriate to select and compose groupware behaviour?
   - 4b. What are the responsibilities of these modules, in terms of the behaviour they have to provide?
   - 4c. What are the relations between these modules?

1.3 Scope and focus

The scope of our research is to create a structuring of groupware services. The focus within this scope is to design tailorable groupware services: groupware services that can be adapted by end users to match the dynamic requirements of real life co-operation.

As the externally observable behaviour of groupware, i.e., the *service* groupware provides to co-operating end users, is of prime importance to these users, the service aspect should be the basis for groupware design. Curiously, one notices that currently groupware users are frequently confronted with details regarding the internal structure of groupware.

Moreover, as co-operating end users are most affected by the service groupware provides, they should be presented with tailoring options on a service level.

For these reasons, our research does not elaborate on implementation details or communication protocol standards for communication between groupware clients of different participants. Instead, the research supports designers to specify the *externally observable behaviour* of groupware applications, and to allocate this behaviour to coarse-grained modules that can be composed to form groupware services.

The target audience of this book includes groupware researchers, groupware designers and developers who wish to make informed design decisions, while putting the co-operating people first.

1.4 Research approach

We achieve the research objective via the following steps:

1. First, the State of the Art regarding groupware, groupware design, and groupware tailorability is explored. This exploration is followed by an analysis that motivates our research.
2. Chapter 3 describes the architectural toolkit: the relevant architectural concepts, groupware concepts, and design criteria. These concepts and criteria form the vehicle to describe groupware services in this dissertation.
3. Defining *a generic model of groupware services*. Chapter 4 describes criteria to define elementary units of groupware behaviour. Based on our experiences, groupware literature, as well as the dynamics in use of groupware services, a generic model of groupware services is derived. This model describes the elementary units of groupware behaviour, i.e., the Groupware Service Module Element (GSME) types, and their causal relations.
4. Describing the individual types of *elementary units of groupware behaviour*. The descriptions in chapter 5 yield detailed insight in behaviour associated with the various GSME types.
5. Defining a *groupware service reference model*. Chapter 6 describes criteria to define types of *units of composition of groupware services*. These units, denoted as Groupware Service Modules (GSMs), can be selected and composed by end users to form groupware services. The GSM types we distinguish are defined in a groupware service reference model. This model also describes their behaviour, in terms of the GSME types they consist of, and their relations on a service level.
6. Demonstrating the steps needed to map a service-level groupware specification onto an implementation. Apart from describing these

steps, chapter 7 also provides guidelines for a reference model for groupware *software* components. Finally, the chapter describes the CoCoWare .NET platform, as one example of a groupware application designed based on our groupware service reference model.

7. Evaluating the groupware reference model, in relation to the design criteria. The evaluation, described in chapter 8, is performed in four stages, focussing on different aspects of the design. Two important stages in the evaluation are the use of a questionnaire to discover the granularity and types of units tailors distinguish in groupware, and analyzing the groupware service reference model using the Software Architecture Analysis Method (SAAM). This evaluation provides evidence that our groupware service reference model adheres to the design criteria identified in section 3.7.

1.5 Research contributions

This dissertation describes a structuring of groupware services. This structuring allows co-operating end users to select and compose groupware modules in order to form groupware services that match their need. At the same time, this structuring helps designers design tailorable groupware services. Two important results of this structuring are:

1. A set of Groupware Service Module Element (GSME) types. These GSME types form *elementary units of groupware behaviour,* appropriate to describe and prescribe the behaviour of a large range of groupware applications;
2. A groupware reference model. This reference model defines Groupware Service Module (GSM) types and the relations between them. GSMs, which have been implemented based on these types, form *units of composition* of groupware services. These GSMs can be selected and composed by end users to determine the service provided by a groupware application.

Chapter 2

Groupware: State of the Art

This chapter motivates our research objective and places the research in context. To do so, it describes the state of the art regarding groupware and groupware tailoring, existing groupware reference models, and existing component groupware platforms and toolkits.

2.1 Introduction

Human beings are a social species. In general, people need to interact with others to feel good. At the same time, interaction also helps to share knowledge and experiences, and allows us to co-ordinate actions. As a result, an interacting group of people can achieve more than they could individually. However, there are natural hindrances for interaction: typically, people need to be together at one location in order to interact. After smoke signals, radio, and telephone, mankind now explores the possibilities of computers and computer networks to support human interaction.

Computer-mediated communication does not replace face-to-face communication, but in specific situations it may be preferable because certain difficulties, inconveniences, and breakdowns can be eliminated or minimized (Ellis et al., 1991). Most obviously, the use of computer-mediated communication eliminates the need for people to travel to each other to have a meeting. Just like the telephone and fax did not replace face-to-face communication, computer-mediated communication is also complementary to existing forms of communication. The challenge is to know in which situations computer-mediated communication provides a benefit, and how to design and apply appropriate technology to support co-operating people. The research presented in this dissertation focusses on the design of appropriate technology to co-operate.

2.2 Computer Supported Co-operative Work

In our work, the terms Computer Supported Co-operative Work, groupware and conferences are defined as follows:

Definition 1 Computer Supported Co-operative Work

Computer Supported Co-operative Work (CSCW) indicates the scientific study and theory of how people work together, how the computer and related technologies affect group behaviour, and how technology can best be designed and built to facilitate group work (Ellis et al., 1991). Despite what the name suggests, the field of study also examines competition, socialization, and play. The current work however, focusses on the co-operative aspects in a work-related context.

Definition 2 Groupware

Groupware applications denote any type of software application designed to support groups of people in communication and collaboration on shared information objects (Ellis et al., 1991; ter Hofte, 1998). The fundamental property of groupware is its ability to mediate the results of some user interactions to other people.

Well-known types of groupware include e-mail, videoconferencing, workflow, chat, and collaborative editing systems. Central to our notion of groupware is the *conference* concept:

Definition 3 Conference

A conference denotes a period of interaction by a group of co-operating people, supported by a particular groupware service. During such a conference, a set of rules may apply to regulate access to groupware behaviour. A conference has a beginning and an end, and is characterized by an explicit or implicit objective (e.g., a peer-review session, a project progress meeting, a design session). In literature, such conferences are sometimes denoted as sessions.

CSCW is an inherently multidisciplinary field of research; it can best be regarded as an umbrella collecting researchers from a variety of disciplines, including psychology, sociology, computer science, interaction design, and management, each contributing a different perspective and scientific methods.

As may be clear from the broad definition, introducing computer supported co-operative work in an actual work setting is a multi-faceted problem: it requires amongst others knowledge about the way individual people act, the tasks the co-operating people perform, the dynamics of a co-operating group, the possibilities of technological support, and human-computer interaction design. Some issues related to the introduction of CSCW are on the level of individuals; other issues are on the level of groups, organizations, or even networks of organizations. Typically, the way

to cope with this multi-faceted problem is to have a multi-disciplinary team that helps to design and introduce CSCW in an organization.

Although CSCW is a relatively young field of research, it is gaining maturity, boosted by the recent availability of high-bandwidth networks that allow people to interact using high-quality audio and video, while being on opposite sides of the world. However, the availability of broadband networks and fast computers did not solve many important issues related to CSCW.

2.2.1 The intellectual challenge of CSCW

One of the fundamental intellectual challenges of CSCW research is to bridge the social-technical gap. This social-technical gap is the divide between what we know we must support socially (i.e., the social requirements) and what we can support by technology. Exploring, understanding, and hopefully ameliorating this social-technical gap is the central challenge for the research field of CSCW and one of the central problems for Human Computer Interaction (HCI) (Ackerman, 2000).

The social-technical gap exists because the way people interact is complex, nuanced, and constantly changing (fluid) (Ackerman, 2000). Co-operating people may have different, even conflicting goals, exceptions are normal in work processes (Suchman & Wynn, 1984). Furthermore, the roles of people are often informal and fluid (Strauss, 1993). In contrast, technology typically is rigid, and especially good at supporting structured, well-defined processes.

However, the social-technical gap also represents the unique contribution of CSCW research. Its intellectual importance is at the confluence of technology-oriented aspects and social aspects, and its unique potential lies in its recognition of and attention to both (Ackerman, 2000).

2.2.2 CSCW design approaches

Given the social-technical gap, the design and introduction of CSCW is a complex problem. During the design of CSCW, a number of design approaches can be applied that focus on the way people actually co-operate. Two well-known examples are ethnography and participatory design. Without the intention to fully describe these approaches, their main characteristics, strengths and weaknesses are briefly summarized, as previously described in Mulder & Slagter (2002).

Ethnographic studies in CSCW focus on observing current work practices and analyzing the tasks that co-operating people perform in order to determine requirements on technology to support them. Ethnographic studies also reveal the social context of the work to support and the situatedness of it, which can be important factors in the design (Suchman,

1987). When the ethnographic study takes place concurrently with system development, users are exposed to the observations by ethnographers. This helps users to discover new ways of interacting, as they participate in a sort of action-learning experience (Agostini, De Michelis, & Susani, 2000). In our opinion, ethnographic studies typically result in a thorough analysis of the co-operation and expose users to an action-learning experience. However, it seems difficult to introduce innovative technology based on ethnographic studies, because of its focus on current work practices.

The other well-known approach is participatory design. As a Scandinavian reaction to traditional systems design, which puts functionality first, this approach takes the *user needs* as the starting point of a design process (Agostini et al., 2000; Kuhn & Muller, 1993). From the early stages in a design process, prospective users of the system are involved in the design. Designers e.g., discuss scenarios of use with future users, let them work with mock-ups and prototypes of the system. In participatory design, the design and development of a system is a process based on the interaction between users and system developers. The notion of putting the collaborating people first, and involving them from the early stages in the design is, in our opinion, an important benefit of participatory design.

2.2.3 CSCW introduction and use

The introduction of computer supported co-operative work in an organization requires special attention. Important questions include: who to deploy to, with what explicit or implied message is the technology introduced, and what or who does it replace. In CSCW, an important issue is achieving, and sustaining, a critical mass of users (Palen, 1998; Markus, 1990). When technology is to be applied for co-operation, a prerequisite is that all participating people have access to that technology, and are willing to use it. With an insufficient number of users, i.e. a lack of *critical mass*, technological support provides no benefit, and people will not use the technology. This has been found in e-mail, synchronous communication, and calendar systems (Palen, 1998).

The introduction of CSCW systems is often more difficult than for single-user systems, since CSCW systems often require initial buy-in from groups of people, rather than individuals, as well as continued buy-in (Ackerman, 2000). On the other hand, when a critical mass of users has been achieved subtle *peer pressure* can propel and maintain wider use (Palen, 1998).

After the successful introduction of CSCW in an organization it is important to keep evaluating the co-operation, and to assess how well the technology fits the current requirements. Hettinga describes the need for

such *reflection*, to allow for adaptations to the technology, and adaptations to the way of working (Hettinga, 2002).

2.3 Groupware

Contrary to optimistic predictions in the 1970s and 1980s (e.g., (Snyder, 1971)) groupware applications, especially videoconferencing systems, have not reached their anticipated potential. For instance, Snyder predicted that by the end of the 1970s a full 85% of all meetings would be electronically mediated. Obviously, this has not been the case. On the other hand, some types of computer-supported co-operation have become highly successful, such as e-mail and instant messaging (ter Hofte, Mulder, Grootveld, & Slagter, 2002).

Historically, groupware applications have been classified based on the time-place matrix described by Johansen (1988). This matrix, shown in *Table 2-1*, classifies groupware applications along two axis: whether the co-operating people have to work together at the same time (synchronously) or may work at different times (asynchronously), and whether the people work at the same physical location, or at different places.

Table 2-1 The time-place matrix of groupware

	Same time (synchronous)	**Different time (asynchronous)**
Same place (local)	Group decision support systems	Electronic project rooms
Different place (remote)	Videoconferencing, text-based chat, shared workspaces	E-mail, group scheduling, shared workspaces

However, this time-place matrix is very rigid, and does for instance not leave room for applications that support the transition between more synchronous and more asynchronous forms of co-operation.

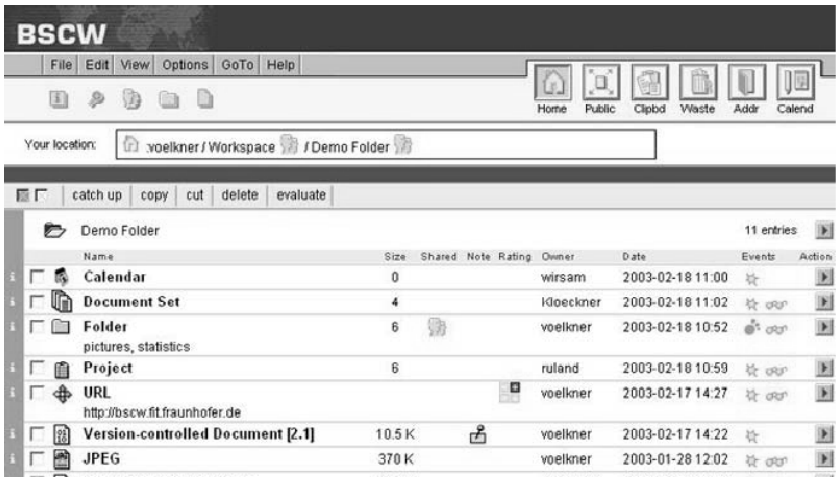
2.3.1 Some existing groupware applications

Many groupware applications have emerged, both in academia and in industry, each providing particular support for co-operating people. Typically, groupware applications are designed to support a particular work situation or a particular range of co-operative work situations (ter Hofte, 1998). Well-known examples of groupware applications include e-mail, audio- and videoconferencing applications, instant messaging applications, and applications to work together using shared virtual workspaces. This section introduces some well-known groupware applications that allow two or more people to interact for an extended period of time. The selected applications illustrate the diversity in current groupware applications.

BSCW

BSCW[1] (Basic Support for Co-operative Work) is a web-based environment to co-operate. BSCW, shown in *Figure 2-1*, started as a web-based application to share files, but gradually turned into a shared virtual workspace that supports for instance document sharing, online discussions, event notification, and group management. BSCW offers a fixed set of collaboration tools and depends on external tools for synchronous communication between workspace participants. The BSCW environment provides services to invite people to a shared virtual workspace, assign roles to people, and specify the rights associated with each role.

Figure 2-1 Screenshot of the BSCW interface



Lotus Notes

Lotus Notes[2], shown in *Figure 2-2*, is an integrated co-operative environment that combines messaging, shared databases, calendaring and scheduling functions with a platform to create collaborative applications. These collaborative applications are founded on replicated, shared databases that users can access and update through forms. Lotus Notes provides advanced security and role-based access control, related to workflow facilities. The Notes platform provides developers with integrated tooling to create tailor-made co-operative applications, based on the specific requirements of a company. Lotus Notes databases and forms can be accessed via the Notes client, or be turned into web applications using Lotus Domino[3]. Additionally, Lotus Web Conferencing and Lotus Instant Messaging (previously known in their combined form as Lotus Sametime)

[1] For more information, see: http://bscw.gmd.de
[2] For more information, see: http://www.lotus.com/notes
[3] For more information, see: http://www.lotus.com/domino

provide awareness services to discover which contacts are currently online, and to start communication using text-based messages, audio, video, or application sharing. Although Lotus Notes offers a highly flexible environment, it requires programming skills to create and adapt co-operative applications.

Figure 2-2 Screenshot of Lotus Notes

Microsoft Windows Messenger

Microsoft Windows Messenger[4] (as well as the related product MSN Messenger) is an instant messaging application that allows people to communicate in a synchronous manner using text messages. The application, shown in *Figure 2-3*, shows the presence status of a list of contacts, providing clues whether they are available for communication. The Messenger application includes facilities for voice and video communication, to share files, and for instance the option to collaborate using application sharing. The application also provides services to invite additional people to an online conference. The Messenger application relies on external tools for asynchronous forms of communication. The application does not provide support for access control or workflow management.

[4] For more information, see: http://messenger.microsoft.com

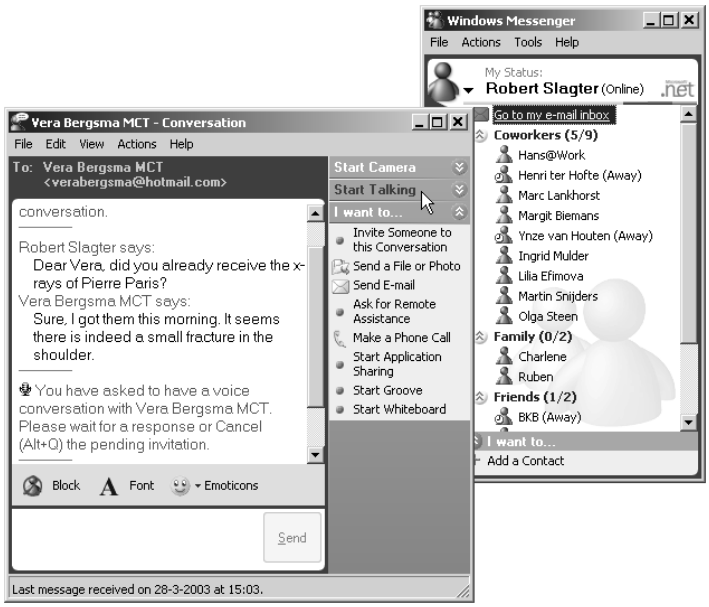


Figure 2-3 Screenshot of Microsoft Windows Messenger

Groove

The Groove Workspace[5] by Groove Networks, Inc. is a peer-to-peer groupware application that enables people to co-operate using shared virtual workspaces. Additionally, it provides synchronous communication via text-based chat and audio conferencing. Groove workspaces provide a co-operation environment where people can exchange information and collaborate using an extensible set of tools, including a shared schedule, a collaborative editor, a file sharing tool, and a shared presentation viewer. The Groove application, shown in *Figure 2-4*, provides services to see which of your contacts are online or active in the workspace, to invite (and uninvite) people to a workspace, to assign roles to people and specify the rights associated with a role. Additionally, the Groove Workspace application makes use of a groupware infrastructure that provides some value-added services, such as security, firewall transparency, and workspace synchronization after a participant has been offline. More details about this product can be found in section 2.7.3.

[5] For more information, see: http://www.groove.net

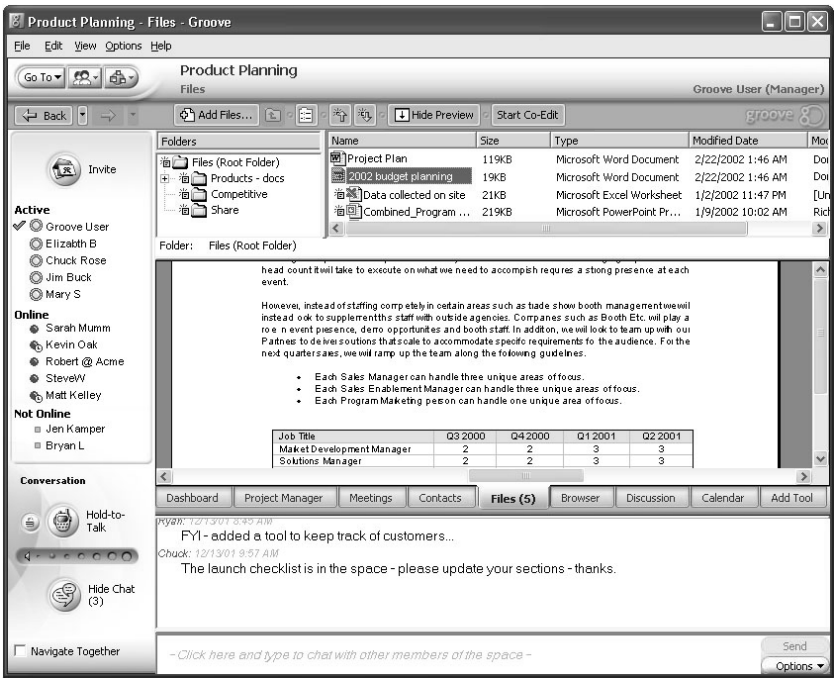


Figure 2-4 Screenshot of the Groove Workspace

2.4 Groupware tailorability

Based on our experience, and supported by literature, we derive that in groupware one size does not fit all: the behaviour provided by a groupware application has to match the requirements of the co-operative setting, in particular the task the people perform together. This dependency is frequently stated in literature, e.g., (Bardram, 1998; Bentley & Dourish, 1995; Kahler et al., 2000; Stiemerling, 2000; Koch & Teege, 1999; Zigurs & Buckland, 1998).

At the same time, one cannot predict exactly how people will co-operate, and what changes may occur in the course of co-operation. By investigating the dynamics of real life co-operation, it is possible to identify types of changes that are likely to occur. Based on this, one can derive which aspects of a groupware design have to be flexible to accommodate these changes (Bock & Marca, 1995).

To achieve a proper matching between the task and the technological support, typically denoted as the *task-technology fit*, it is possible to design an *adaptive* application: an application that adapts its own behaviour based on the observations it makes regarding the way it is used. The current research follows a different approach: that of an *adaptable* application. In the case of

an adaptable application a human being performs the adaptation. To achieve a task-technology fit in an adaptable system, two means can be distinguished:

1. A good initial selection of groupware behaviour by external experts, based on the characteristics of the collaboration "as is" and "to be". This includes the characteristics of the collaborating people, the tasks they perform together, and the context in which the collaboration takes place (e.g., in the office, on the road, or at home).
2. Allowing the collaborating people *themselves* to adapt the behaviour of their groupware applications. Such adaptations by end users in the context of system use are denoted as *tailoring*.

Definition 4 Tailoring

Tailoring is defined as the activity of adapting a computer application within the context of its use (Mørch, 1997b). As such, tailoring is performed by end users, not by programmers.

Definition 5 Tailorability

Tailorability denotes the capacity of a computer application to be tailored.

In both situations described above, people select, adjust and combine the groupware behaviour elements that suit their needs. However, the people who perform these operations differ in the two situations described above. In the first one *experts* do the adaptations, while in the second one the *end users* perform the adaptations. In the later situation, one cannot assume an expert will be available to assist the end user. So, it is important to select a mechanism to adapt the behaviour of groupware applications that is appropriate for experts as well as end users.

As indicated in *Figure 2-5*, tailoring is conceptually located between system use and programming: just like programming it is applied to adapt the behaviour of a system. However, the possibilities for adaptations are more limited than when programming. Tailoring is performed in the context of system use, but it is not the primary reason why people use the system: it is only a means, not the objective.

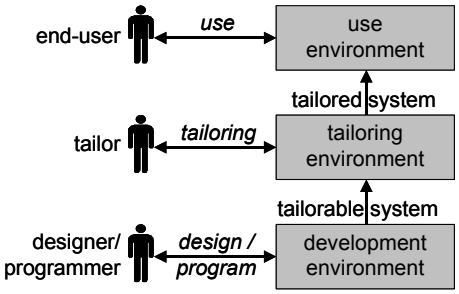


Figure 2-5 The relation between programming, tailoring and use (Biemans & ter Hofte, 1999)

Apart from dynamic requirements, diversity is another driving factor for tailorability. Today, completely custom-built systems are the exception, because it is much more economical to tailor off-the-shelf products to an organization's specific requirements (Stiemerling, 2000). On a smaller scale, also the different users in an organization may have different requirements for their technological support, given their specific tasks and personal preferences. From the vendor's perspective, a tailorable software system promises to reach a larger market segment than a rigid one.

A third reason for tailoring frequently mentioned in CSCW literature results from the so-called *critical mass* problem (Markus, 1990; Koch & Teege, 1999): some groupware applications can only be successful when many people use the application. This has been found in e-mail, synchronous communication, and calendaring applications (Palen, 1998). So, the acceptance of the software by all members of the group, rather than only by some individuals, is crucial to its success. This is, however, not always easy to achieve, since group members usually have different preferences (such as for a specific text editor), different experiences, and often heterogeneous skills and education (Koch & Teege, 1999). Tailoring can help to overcome these differences, as individual group members can adapt the application according to their individual needs and preferences (Oppermann & Simm, 1994), while still being able to co-operate.

In summary, we tentatively conclude that tailorability of groupware behaviour is a desirable property of groupware applications.

2.4.1 Tailoring types

Various mechanisms to tailor groupware services exist. This section briefly introduces these mechanisms, and describes how they can be applied to tailor groupware. Mørch (1997b) distinguishes three levels of tailoring, viz., *tailoring by integration, tailoring by customization*, and *radical tailoring*.

The level most relevant in our methodology, tailoring by integration, corresponds to being able to select and compose the behaviour that matches your needs. The selected and composed behaviour is integrated into a coherent application. In contrast, tailoring by customization involves making a selection from a pre-defined set of configuration options. An example of this is setting a dial to one of four possible values. In the third form of tailoring, radical tailoring, the behaviour provided by an application can be extended, typically via programming. Although this level of tailoring provides most opportunities for adaptations, we consider it to be too complex for many end users, since radical tailoring requires programming skills.

Selecting and combining groupware behaviour

In our approach, the primary mechanism to tailor groupware behaviour is *tailoring by integration*: our design allows people to select and combine the groupware behaviour that matches their needs. Based on the requirements of a specific co-operative setting a selection of behaviour-modules is composed to form a groupware service. This process of *selection* and *composition* may be performed by groupware programmers, system administrators or, in case of tailoring, by the co-operating end users themselves. This form of tailoring corresponds to the concept of *service composition*, as described in section 3.2.6.

In terms of a groupware design, this choice implies that the design should allow for composition at run-time, i.e., late composition: the design should include provisions for dynamic discovery and connection of implementations of behaviour-modules.

Extension: making new groupware behaviour available

Before groupware service users can make use of groupware behaviour, that behaviour has to be made available for them. Making groupware behaviour available can be done in many different ways, even though the user experience may be similar. The process starts with a groupware developer who creates one or more new implementations that realize groupware behaviour. Subsequently, these implementations can be made available for selection and composition in various ways. Without the intention to fully describe the various mechanisms, new implementations may for instance be made available in the following manners:

- *Global availability*: New implementations may be made centrally available, for instance as an online service. Possibly after authentication, any person can make use of the behaviour provided, without the need to download or install software;
- *Group-level availability*: Groupware applications may incorporate mechanisms to make new implementations available to other conference participants. The EVOLVE platform for instance includes a mechanism to share groupware building blocks with conference participants, based on a groupware extensibility pattern (Hummes & Merialdo, 2000).
- *Local availability*: New implementations may be made available to a particular groupware service user, for instance by downloading and installing a new software component. An example of this mechanism is the option in the Groove Workspace to download and install new groupware tools.

Effectively, when new groupware behaviour module implementations have been made available using one of these mechanisms, the groupware behaviour to choose from has been extended: the toolkit of units of

composition of groupware services has been extended with a new module. As such, this mechanism corresponds to the concept of *service extension*, as described in section 3.2.6.

In groupware, the fact that new groupware behaviour has become available does not necessarily impact other participants; only when the new behaviour is actually used there may be an impact for the other conference participants.

Extending the available groupware behaviour raises some interesting issues. When software components, which implement groupware behaviour, originate from different manufacturers, one cannot assume that the implementation details of the various software components are available for all programmers (Szyperski, 1998). As a consequence, programmers cannot perform a design-time analysis of unwanted interferences between system parts, unless precise service-level descriptions of the various parts are available.

Extensibility of groupware behaviour also has implications for tailorability design: For an individual groupware service module it is possible to specify the service it provides to groupware service users. For a composition of such modules it is far more difficult to "add up" the individual behaviours and predict the resulting service that is provided.

Fine-tuning groupware behaviour

Fine-tuning of groupware behaviour allows groupware service users to adapt the behaviour provided by a groupware application, without actually changing the composition of groupware service modules. This mechanism corresponds to the *tailoring by customization*, as identified by Mørch (1997b).

An example of fine-tuning groupware behaviour is changing the coupling level of a tool to share information objects. Such a change is needed to switch between tightly-coupled co-operation, where all participants have the same view of the shared information, and loosely-coupled co-operation, where participants work on individual versions of the shared information (Baker, Greenberg, & Gutwin, 2002).

In terms of a groupware design, fine-tuning operations typically involve changes to the behaviour of a single groupware service module. Given this containment, this type of adaptation only has limited consequences for the other modules that form the groupware service.

2.4.2 Tailors

End users who perform adaptations to an application in the context of system use are denoted as *tailors*. But although every tailor is, by definition, an end user, not every end user will be a tailor. Many groupware users will

use a predefined groupware service, without tailoring the provided groupware behaviour.

In a collaborative environment, many tailoring operations are the result of collaboration (Mackay, 1991), not just tailoring experience. Additionally, tailored artefacts can also be shared (Wulf, 1999b; Bentley & Dourish, 1995), so tailoring can be done by those users with most skills, motivation or time to tailor (sometimes called power users, super users, or mediators). Hence, a tailoring environment for groupware should itself also be designed as a groupware system. In some cases, successful adoption and use of a groupware application can even be largely attributed to the tailoring operations done by mediators (Okamura, Orlikowski, Fujimoto, & Yates, 1994).

2.4.3 The influence of tailoring operations

In a co-operative setting, interactions between one person and a groupware system may influence other people as well, both intentional and unintentional. For example adding an x-ray viewer to a groupware system during a conference affects the other conference participants as well. On the other hand, changing the position of the various windows on your screen does not have to affect the other users. These examples show that only certain tailoring operations need to be communicated to the other participants.

Tailoring operations lead to system changes, and typically require changes in user behaviour. In co-operative settings, system changes may affect the other participants, and consequently require behavioural changes of them as well. Based on this, it is possible to distinguish three types of tailoring operations:

1. *Changes that only affect the initiator*. These operations only have a local impact and do not affect other participants in a conference. Changing the colour scheme of your groupware application is an example of such a change: this does not affect the other people in a conference, unless using a "what I see is what you see" (WISIWYS) application.
2. *Changes that need to be communicated to the other participants, but do not require a response from them*. These operations require some form of tuning between participants to understand the changed situation, but require no response from the other participants. As one physician in a videoconference turns his image viewer to black and white (due to bandwidth restrictions), the other participants should be aware of this, although they do not have to act on the operation.
3. *Changes that need to be communicated to the other participants, and require a response from them*. These operations create a new situation for all participants, and therefore require both a response and a corresponding

behavioural change of all participants. An example of such a change is a patient who wants to show the spots on his skin and therefore adds a video connection to an ongoing audio conference. This requires a response at the receiving end as well, in order to see the transmitted video image. This response may be done automatically, or require some interaction between the physician and the groupware application.

2.4.4 Co-evolution of people and technology

Adaptations to work processes and adaptations to technology influence each other: while people tailor their systems to match changing work processes, people also adapt the way they work based on the support provided by technology. This process, typically denoted as *co-evolution*, has been investigated by Orlikowski (1992) and O'Day et al. (1996). Their research reveals that the adaptations to systems can be quite sophisticated. Moreover, people may use systems in ways that were not anticipated when the systems were originally designed. Based on their findings, they conclude that system designers should assume that people will try to tailor systems to match their use.

2.4.5 Current use of tailoring in groupware

Current groupware applications typically provide some form of tailorability, ranging from allowing users to select from a fixed set of communication tools to offering users the possibility to install and use new communication and collaboration tools. However, the provided tailoring options are currently not used to their full potential (see e.g., (Oppermann, 1994; Kahler et al., 2000; Hettinga, 2002)). In literature, several reasons have been proposed as the source of this lagging use: current systems do not provide appropriate support for selecting which tools actually to use in which situations (Tietze, 2001), users would not be able to find the relevant tailorable options (Wulf & Golombek, 2001) or groupware designers fail to communicate the tailorable options to their full extent (Mørch, 1994).

The research by Hettinga (2002) suggests another reason why tailoring is currently not frequently used: observations in a healthcare setting revealed that these healthcare professionals hardly reflected on their co-operative work practices to assess their way of working. As a result they also did not discuss possible improvements, let alone tailor the technological support.

Based on these findings we conclude that although tailoring is generally regarded an import capacity of groupware, it is currently not used to its full potential. Many groupware applications are too difficult to tailor. Furthermore, although tailoring operations can be regarded as separated from normal use, it should not be difficult to switch from normal system

use to tailoring. Finally, it seems important to stimulate end users to reflect on the co-operation, both during and between co-operative sessions, as such reflections may lead to tailoring activities and improved co-operation (Hettinga, 2002).

2.4.6 A conceptual model of tailorable groupware services

Ellis & Wainer (1994) have described a conceptual model of groupware. This conceptual model defines three views: a description of the objects and operations on these objects available in the system; a description of the activities (and their orderings) that the users of the system can perform; and a description of the interface of users with the system, and with other users. Given our service-oriented approach, we have adapted this conceptual model and introduce the following views: the description of the units of behaviour groupware provides to its users; the description of how the causal relations between these units of behaviour; and how are the participants to interact with the groupware application and each other. Our methodology only specifies in an abstract manner how groupware users interact with a groupware application and how they interact with each other via a groupware application.

It is also important to identify on a conceptual level the basic elements that constitute *tailorable* groupware services. Without subscribing to any particular style of architecture, one can identify three necessary elements in any tailorable system, not just in tailorable groupware (Stiemerling, 2000): The variable properties of the system have to be represented internally as data; there has to be functionality to change this data; and changes to the data representing a property have to cause the actual property exhibited by the system to change. These elements form the basis for our conceptual model of tailorable groupware services:

- Tailorable groupware services represent the behaviour they can provide to the co-operating end users in some form. This representation of the system is sometimes denoted as the meta-system (Stiemerling, 2000);
- Groupware service users have access to functionality to manipulate the groupware behaviour representation. The groupware design determines the freedom groupware service users have to manipulate the groupware behaviour representation;
- There exists a direct connection between the behaviour representation and the actually provided groupware behaviour: changes to one result in changes to the other.

The tailorability of a groupware application is characterized by the design choices made regarding these aspects. Our methodology focusses on structuring tailorable groupware services and specifying, in an abstract

manner, the interactions that take place between groupware service users and a groupware service provider, i.e., a groupware application. The design does not specify how a groupware service should be represented to groupware service users, or how these users should manipulate the representation.

2.4.7 Technical challenges regarding tailorable groupware

Designing and implementing tailorable groupware raises some interesting technical challenges.

- Run-time discovery of available groupware behaviour. The groupware application has to be able to discover what groupware behaviour is currently available. New groupware behaviour might become available in the course of a conference;
- Representing the currently active groupware behaviour in an appropriate manner to the end users;
- Representing the available optional groupware behaviour in an appropriate manner to the end users. Such a representation should also state the impact of including this behaviour in the groupware service: what are the consequences for the co-operating people and the provided groupware service;
- Run-time composition of groupware behaviour. Based on the manipulations by end users of the groupware behaviour representation, the appropriate groupware implementations should be activated and composed to provide a coherent groupware service. Note that when the complete set of groupware behaviour to choose from is not known at design time, for instance since this set can be extended, there is no possibility for full integration testing at the implementation level. Instead, service-level integration tests may be performed as part of the process of run-time composition.

2.5 Requirements for groupware

The characteristics of real life co-operation are an important source of requirements when designing groupware applications. This section summarizes the findings we consider most relevant for our research. These findings have been presented by other researchers who have investigated the way people prefer to interact, and the consequences for groupware design.

- *Provide matching support for co-operation*. The main requirement on a groupware application is that it should provide appropriate support given the tasks performed by the co-operating people (e.g., (Bentley & Dourish, 1995; Koch & Teege, 1999; Kahler et al., 2000; Stiemerling,

2000; Zigurs & Buckland, 1998)). Given the characteristics of co-operation, other important requirements on groupware design are: to support multiple work methods, to support the development of the group, to provide interchangeable interaction methods, to sustain multiple behavioural characteristics, to accommodate permeable group boundaries, and to be adjustable to the group's context (Mandviwalla & Olfman, 1994).

– *Groupware should not be restrictive*. During co-operation roles are often informal and dynamic (Strauss, 1993). Groupware applications should not be too restrictive, and leave much of the co-ordination (i.e., access control of communication and collaboration functions) to the human end users. Research by Beck and Bellotti (1993) indicates that academic co-authors apply great flexibility in the co-ordination of individual contributions to a paper; in three case studies, co-authors made opportunistic use of information made available to each other to determine when and what to contribute to the paper. However, when explicit roles have been assigned, a groupware system should be able to enact a co-ordination policy to enforce controlled access to sensitive options, such as the option to remove participants from a co-operative session.

2.6 Existing groupware reference models

Groupware reference models describe on a very high-level how groupware applications can be designed. They identify the modules that comprise a groupware application, and their responsibilities. The purpose of a reference model is to direct attention at an appropriate decomposition of the system without delving into details. The following sections introduce the CoMeCo Groupware Service Architecture, Dewan's generic collaborative architecture and the Clover architecture.

2.6.1 CoMeCo Groupware Service Architecture

The CoMeCo Groupware Service Architecture by Ter Hofte (1998) is a groupware reference model that decomposes groupware services into grouplet service classes. The Groupware Service Architecture is named after the three grouplet service classes it distinguishes:

– *Conference management grouplet services*, which provide users with the functionality to manage a set of online conferences, change conference membership, and for instance change the set of active media in the conference;

– *Medium grouplet services*, which provide users all services as far as required to support the interaction related to a single medium. In the CoMeCo Groupware Service Architecture, a medium may be a shared artifact, such as a shared whiteboard, a document, etc. It may also be a communication channel, such as an audio conferencing connection;
– *Co-ordination grouplet services*, which may be applied to active media in a conference. When applied, they provide a co-ordination policy that specifies which conference members may or must perform which medium actions at which moment.

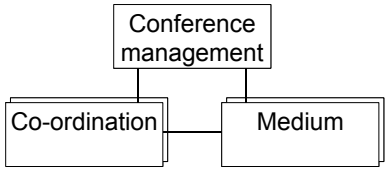


Figure 2-6 The CoMeCo Groupware Service Architecture

As illustrated in *Figure 2-6*, CoMeCo defines a conference as a collection of members, media, co-ordination policies, and sub-conferences. Media can for instance be shared virtual workspaces for indirect communication or conversation channels for direct communication. The CoMeCo Groupware Service Architecture has been implemented in a number of prototypes, including CoCoDoc and MediaBuilder. The latter has been professionalized by Lucent Technologies into an application for high-quality audio- and videoconferencing and application sharing. CoMeCo does not specify the actions and interactions that take place as part of units of groupware behaviour, nor does it describe the causal relations between units of groupware behaviour.

2.6.2 Dewan's generic collaborative architecture

Dewan has developed a generic collaborative architecture that can be considered a reference model for the internal structure of groupware applications. His architecture describes how collaborative applications can be designed using functional layers (Dewan, 1998). Such functional layers can be shared (i.e., centralized) or replicated. The bottom-most layers of the architecture, as depicted in *Figure 2-7*, are the workstation layers managing the screen and input devices attached to a workstation. The workstation layers are usually replicated to allow collaborators to use different workstations. The topmost layer in the architecture represents the semantic layer. Unlike the objects in the other layers, the objects in the semantic layer are themselves not an interactor for another object.

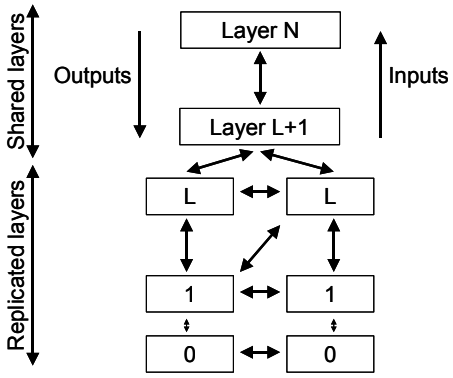


Figure 2-7 Dewan's generic collaborative architecture

Design choices regarding the sharing or replication of functional layers influence performance, ease of adaptation, and other properties desired by users and programmers of the application. Specifically, these design choices influence the following aspects of a collaborative application (Dewan, 1998):

- *Single-user architecture*: What is the architecture for implementing single-user semantics?
- *Concurrency*: What components of the application can execute concurrently?
- *Distribution*: Which of these components can execute on separate hosts?
- *Versioning/Replication*: Which of these components are replicated?
- *Collaboration awareness*: Which of these components are collaboration aware, i.e., implement collaboration semantics?

2.6.3 Clover architecture

Similar to Dewan's generic collaborative architecture, the Clover architecture for groupware defines several functional layers to implement groupware. As an extension, the Clover design model specifies at each layer a partitioning into production, communication and co-ordination functions (Laurillau & Nigay, 2002), as illustrated in *Figure 2-8*.

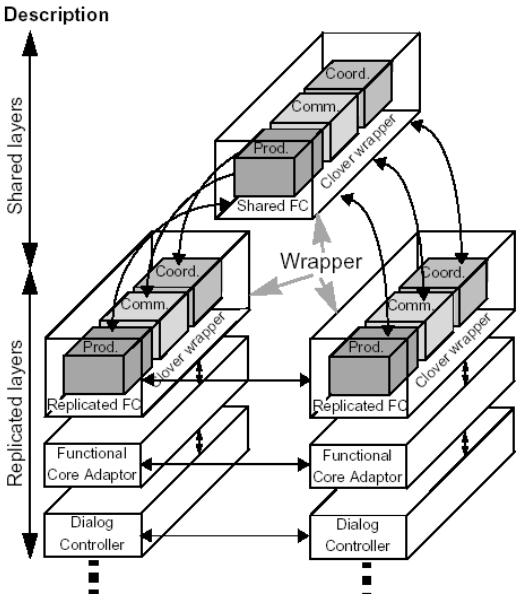


Figure 2-8 The Clover architectural model

These production, communication and co-ordination functions have been adopted from the PAC* architectural model (Calvary, Coutaz, & Nigay, 1997). In the Clover design model, *production functions* denote the functions needed to access and adapt a set of shared information objects (in Clover denoted as a *caddy*). The *communication functions* allow for direct communication between the users of Clover. Finally, the *co-ordination functions* in Clover encompass functions to join and leave sessions, to create, join or leave a working group, and to enter personal preferences. Note that these co-ordination functions correspond to conference management services identified in the CoMeCo Groupware Service Architecture (as described in section 2.6.1), not to the co-ordination functions identified in that model. The combined Clover production functions and communication functions correspond to the CoMeCo communication functions.

2.7 Component groupware platforms and toolkits

One possible way of creating adaptable groupware is to design these applications out of individual modules that provide a selection of groupware behaviour. In that case, the behaviour of an application is determined by the set of modules it is composed of. A groupware application that is composed out of such functional modules is denoted as *component groupware*. The

following sections introduce various component groupware platforms and toolkits from industry and academia.

2.7.1 EVOLVE

The EVOLVE tailoring platform by Stiemerling applies an implementation architecture for distributed component-based tailorability for CSCW applications (Stiemerling, 2000). This architecture defines an extension of JavaBeans, called FlexiBeans. FlexiBeans solves some important issues that arise when using JavaBeans for groupware, e.g. the fact that JavaBeans have no mechanism to share remote objects as interaction primitives. The EVOLVE platform is based on a client-server architecture: the central server maintains for example the repository with components, keeps descriptions of the hierarchical component architectures of the applications and manages the user accounts.

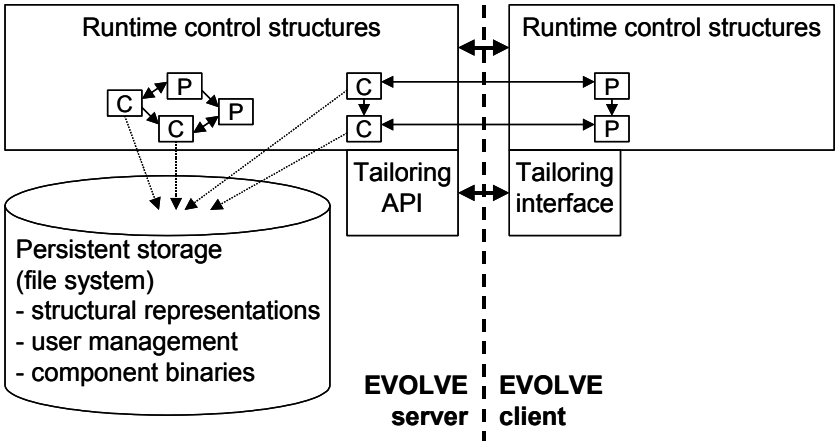


Figure 2-9 The EVOLVE groupware architecture

The EVOLVE architecture, depicted in *Figure 2-9*, defines both the client and the server side of a client-server groupware application. On the server side the run-time control structures contain structural representations, denoted by a *c*, that define what components are combined in what manner into a groupware application. Such component structures can have proxy objects, denoted by a *p*, that reside either on the server or on the various clients. Each component structure can be instantiated several times, resulting in an equivalent proxy structure for each instance structure. During tailoring, the *c* structures are manipulated. All changes are directly broadcasted to all instances, i.e., all *p* structures, which apply them to the running groupware application.

2.7.2 DISCIPLE

The DISCIPLE (DIstributed System for Collaborative Information Processing and LEarning) framework enables sharing of arbitrary JavaBeans applications in real-time synchronous group work (Marsic, 1999). DISCIPLE uses a generic collaboration bus providing a plug-and-play environment that enables collaboration with applications that may or may not be collaboration aware. The framework defines a client-server architecture, as depicted in *Figure 2-10*, which aims to offer both a toolkit for development of special purpose collaboration-aware applications and a framework for sharing existing single-user (collaboration transparent) applications. The framework is designed according to a layered architecture that describes e.g., at the client side a multimodal human/machine interface layer on top of a layer with application logic. The latter layer makes use of the collaboration bus layer, which is the lowest layer. Spanning all layers is the intelligent agents plane, which provides separate knowledge mechanisms. DISCIPLE uses a hybrid form of a toolkit and a platform approach, specific for a component groupware architecture.

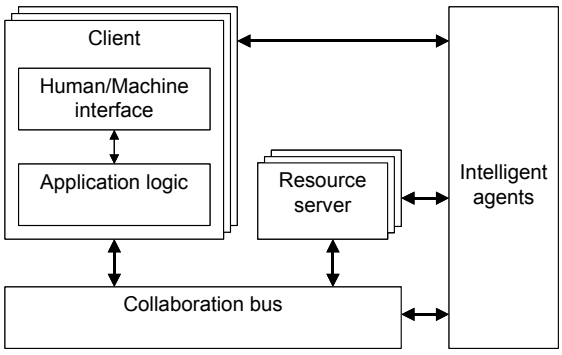


Figure 2-10 The DISCIPLE groupware architecture

2.7.3 Groove

The Groove Workspace by Groove Networks, Inc. is a commercial initiative for component-based peer-to-peer groupware. The peer-to-peer approach minimises the need for central servers and allocates as much functionality as possible at the systems of the groupware service users. In Groove, each participant in a conference has his own replica of the shared virtual workspace. This approach allows the users to fully control the application, while minimizing the dependency on other (economic) parties.

The Groove platform consists of a series of user interface components and a framework. The user interface components are connected by means of the framework, which provides basic functions. Such basic functions include security, firewall transparency, account and user id management,

and a basic service to synchronize the state of a workspace (described in XML) with the people you are co-operating with. The Groove architecture is shown in *Figure 2-11*.

Groove users manage and navigate workspaces through a so-called *transceiver*. Groove workspaces contain facilities for instant messaging, and may contain multiple Groove tools.

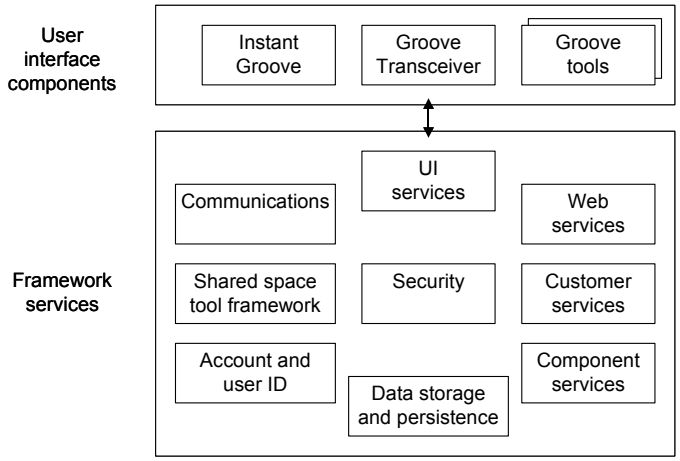


Figure 2-11 The Groove architecture

The Groove platform comes with a set of reusable groupware tools that provide frequently needed groupware functions. More information on the Groove Workspace can be found on: http://www.groove.net/products/

2.7.4 GroupKit

GroupKit is a groupware toolkit to support developers in creating applications for synchronous, distributed computer-based conferencing (Roseman & Greenberg, 1996). GroupKit reduces the implementation complexity by some key features. The run-time infrastructure automatically manages the creation, interconnection, and communication of the distributed processes that comprise conference sessions. A set of programming abstractions allow developers to control the behaviour of distributed processes, to take action on state changes, and to share relevant data.

GroupKit uses an architecture in which session management, conference applications and a central registrar are decoupled, as shown in *Figure 2-12*. All end users each have a replica of the session manager and the used conference applications, while the registrar resides at a central server. GroupKit, created in Tcl/Tk, offers abstractions for multicast remote procedure calls, an event mechanism, and reading and setting environment variables. Furthermore, GroupKit offers a number of multi-user groupware

widgets, which can be used by programmers in conference applications to satisfy some user-centred design requirements.

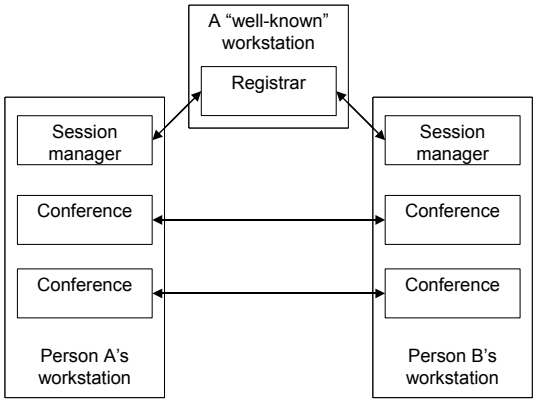


Figure 2-12 The GroupKit groupware architecture

Currently, the commercial version of GroupKit, TeamWave, comes with a large number of collaborative tools: e.g. a shared whiteboard, a shared web browser, a message board, and a shared database. More information on GroupKit can be found on: http://www.groupkit.org

2.7.5 JViews

JViews is a toolkit for constructing view and repository components for multi-view systems (Grundy, Mugridge, & Hosking, 1997; Grundy, Mugridge, Hosking, & Apperley, 1998; Grundy & Hosking, 2002). A multi-view system permits different, possibly overlapping views on shared information. These different views may correspond to different end users. Consistency between the different views is maintained by automatically propagating changes to a shared repository and then to all affected views, according to the Model-View-Controller design pattern (logically only one model exists, which is associated with multiple views). JViews includes abstractions for view and repository components. The repository component corresponds to the model component in the Model-View-Controller design pattern (logically that is; in fact each user may have an instance of the model). Furthermore, JViews defines inter-component relationships. The inter-component relationships are used for data structures, to aggregate components and to maintain inter-component and intra-component consistency.

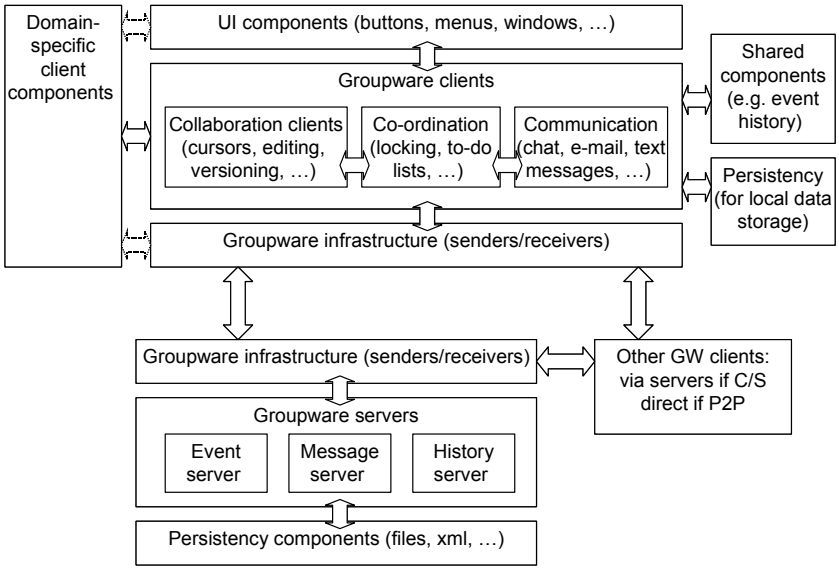


Figure 2-13 The JViews groupware architecture

The JViews groupware architecture, as depicted in *Figure 2-13*, defines that a groupware client consists of three types of components with different responsibilities: collaboration clients (handling shared cursors, editing & versioning of shared information objects), co-ordination components (handling for instance locking) and communication components for direct communication between conference participants. More information on JViews can be found on: http://www.cs.auckland.ac.nz/~john-g/jviews.html

2.7.6 COAST

COAST (COoperative Application Systems Toolkit) is a toolkit that offers developers an architecture for co-operative applications and corresponding classes that can be used to implement such applications (Schuckmann, Kirchner, Schuemmer, & Haake, 1996). The implementation of the toolkit does not require any specific platform (such as an operating-system, user interface management system, or communication channel). User requirements are reflected in the following features:

- A replication mechanism combined with a fully optimistic concurrency control, which enables immediate processing of user actions independent of network latency,
- A synchronization mechanism for replicated objects ensuring fast propagation of other users' actions,
- Dynamic sessions, supporting the flexible coupling of shared objects and users.

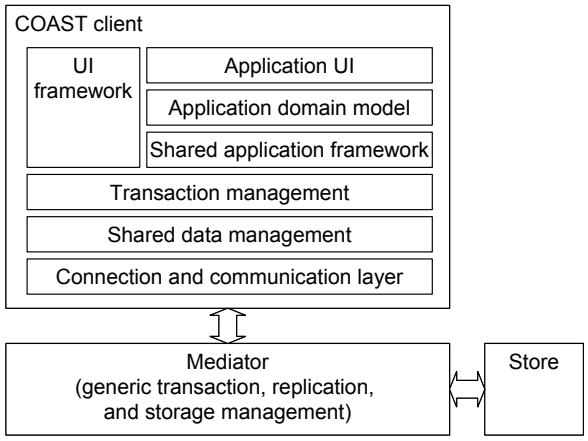


Figure 2-14 The COAST groupware architecture

The COAST groupware architecture, as depicted in *Figure 2-14*, defines a layered structure of groupware functions. End users can make use of the groupware functions through the user interface framework and the application user interface. These application user interface layer makes use of an application domain model, which in turn relies on a shared application framework. The clients of the various co-operating participants in a conference communicate through centralized mediator components, which provide generic mechanisms to transport information, for replication, and to persistently store information.

2.8 Analysis and research motivation

Although the research domain of Computer Supported Co-operative Work is gaining maturity, and a wide range of groupware applications has been developed, the groupware design landscape is still fragmented and some important issues are not sufficiently solved yet.

2.8.1 The lack of a standard to design groupware services

The exploration of groupware applications reveals that different applications have been developed based on different paradigms: Some products are based on the notion of shared virtual workspaces, some focus on direct communication between pairs of people, while others focus on information storage and workflows. The lack of a shared standard to design groupware services has contributed to a lack of conceptual consistency between groupware applications. As a result, different groupware services are not presented to groupware users in a similar manner. Having such conceptual consistency would make it easier for groupware users to switch between groupware applications (Kellogg, 1987).

Moreover, groupware applications from different manufacturers are typically not interoperable; in order to co-operate, people have to use the same product.

2.8.2 Current groupware design methodologies

The exploration of current groupware design methodologies and associated reference models reveals that these methodologies are typically oriented at the implementation. As such, the existing groupware reference models do not provide insight in the elementary units of groupware behaviour as well as the units to compose groupware services. However, we consider the *service* a groupware application provides the most important aspect of the system, not the way in which this behaviour is realized.

Existing groupware design methodologies insufficiently focus the designer's attention on the service aspect; instead, they typically focus on implementation aspects. As a result, these methodologies do not help designers to obtain the "bigger picture", showing the key functions and key relations in the design. Therefore, we conclude there is a need for a service-oriented methodology to design tailorable groupware services. When such a methodology includes a structuring of groupware services, for instance in a service reference model, that structuring can become a standard approach to design groupware services.

Recently, a methodology for the design of component-based groupware systems has been published that identifies three different concern levels (Guareis de Farias, 2002). In this methodology a groupware application is modelled on an enterprise level, on a system level, and on a component level. At each level, the methodology identifies interrelated perspectives and views to design the structural, behavioural, and interactional aspects in a stepwise manner. Guareis de Farias argues that these different concern levels allow for a better control over the design process of a groupware application, such that the system designer can focus on the relevant set of concerns at each step along the design trajectory.

The current lack of interoperability, conceptual consistency, and a shared paradigm illustrate the need for standards to design groupware services. A groupware service reference model is an important part of such a service-oriented design approach.

Definition 6 Reference model

A reference model defines the functional modules that form a class of applications (typically related to a specific domain), together with the data flow between these modules.

A groupware service reference model defines a structuring of groupware services, and as such provides a basis for design decisions. Additionally, such

a reference model promotes interoperability and conceptual consistency between groupware applications from different manufacturers. Interoperability between groupware applications increases the freedom for end users to choose a groupware application that fits the specific needs of a co-operative setting, since people are not restricted to use the same groupware application in order to co-operate. Conceptual consistency on the other hand makes it easier for end users to switch between groupware applications (Kellogg, 1987).

2.8.3 Issues regarding adaptability of groupware services

An adaptable groupware service is characterised by a design that allows for adaptations to the provided groupware service depending on changes regarding the context in which the service is being used. As a result, the design of such a service embeds few assumptions about the context in which it is going to be used. In general, three options exist to perform such adaptations:

1. Programmers or system administrators perform the adaptations;
2. The co-operating end users perform the adaptations themselves;
3. The groupware application performs the adaptations automatically.

Our approach advocates the second mechanism: allowing co-operating end users to adapt the groupware service they use. Given the complexity to automatically derive the exact behaviour co-operating end users require, the third option is not considered realistic given the current stage of groupware technology.

The co-operating end users are likely to be motivated to tailor groupware services: they are the ones most affected when the provided groupware service no longer matches their needs. When such a mismatch occurs, the end users could also contact a groupware developer or a system administrator to solve the issue. Apart from the extra dependency, introducing an external person is an investment in terms of time, money and communication needed to explain the issue. If the end users are empowered to perform some classes of adaptations, the introduction of an external person may not always be necessary.

Section 2.4 illustrates that many CSCW researchers acknowledge the need for tailorability of groupware applications. However, our state of the art exploration shows that although tailorability is generally regarded a desirable property of groupware, tailoring is currently not used to its full potential. Various reasons have been proposed in literature to explain this discrepancy. One of these reasons is that current groupware applications are typically difficult to tailor: users are not properly supported in tailoring tasks, such as selecting the groupware behaviour they need.

The underlying reason for this may be that current groupware design methodologies typically focus on implementation aspects. However, end users are more affected by what the system does for them, i.e. the service the system provides, than how this service is implemented; it is, after all, the *service* a groupware application provides to support co-operating people that is the most relevant aspect of the application. So, end users should be presented with tailoring options on a service level.

Chapter 3

Concepts and formal notation

This chapter describes the concepts and formal notation applied in this dissertation to describe groupware behaviour. In our research, the *service* a groupware application provides to co-operating end users is the most relevant aspect of the system. Consequently, this chapter defines this architectural concept, as well as the related concepts of behaviour, actions, interactions and causality relations. The chapter also introduces the formal notation of these concepts, which helps to create structured descriptions that can be assessed based on design criteria. These are introduced at the end of this chapter.

Although we apply these concepts to describe groupware services, the architectural concepts are generic and could be applied to describe the behaviour of a wide range of interactive applications. The concepts that are specific to groupware and groupware services are described in sections 3.3 and 3.4.

3.1 Introduction

In the context of our research the most important aspect of a system is its *externally observable behaviour*, informally described as *what the system can do*. To express the externally observable behaviour of an application, architectural concepts are applied, such as actions, interactions and causality relations. The following section describes these concepts.

3.2 Architectural concepts

Architectural concepts are abstractions, i.e., models, of frequently occurring aspects of technical objects. Such concepts are manipulated during the

design and implementation process (Ferreira Pires, Vissers, & Van Sinderen, 1993).

The architectural concepts most relevant for our research are actions, interactions, causality relations, and services. This set of architectural concepts stems from our focus on describing the *behaviour* of groupware applications.

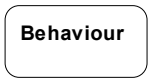
3.2.1 Behaviour

The behaviour of a system models the activities that system can perform: it states what the system can do. An example of an activity is a specific function the system can perform. In our research, the concepts of *actions* and *interactions* are applied as abstractions to model activities. Similarly, *causality relations* are applied to model relations between activities.

A system operates within an environment, which consists of users and other systems that are capable of interacting with the system. The behaviour of a system models a collection of possibly related activities, such that each activity is either performed by the system alone, or by multiple systems in co-operation.

The concepts that model such activities and their relationships are the basic building blocks for defining behaviours (Vissers, Ferreira Pires, Quartel, & Van Sinderen, 2002). The formal language AMBER (Eertink, Janssen, Oude Luttighuis, Teeuw, & Vissers, 1999) has been selected to graphically describe groupware behaviour. AMBER was selected since it allows one to describe the various relevant aspects in a clear and structured manner. In particular, the language applies the concepts of actions, interactions and causality relations to describe behaviour and it allows us to specify in an abstract manner what interactions take place between a groupware application and its users.

Figure 3-1 AMBER behaviour description



In AMBER, behaviour is depicted by a rounded rectangle, as depicted in *Figure 3-1*. Behaviour may be expressed in terms of multiple related sub-behaviours: in that case a rounded rectangle that depicts the total behaviour contains one or more other rounded rectangles that depict the sub-behaviours.

3.2.2 Actions and interactions

Performing activities is the essential purpose of systems: by performing activities systems do what they are supposed to do (Vissers et al., 2002).

The *action* concept is applied to model an activity performed by a single system.

To model an activity that is performed by multiple systems in co-operation, the *interaction* concept (Vissers et al., 2002; Vissers, Lankhorst, & Slagter, 2003) is adopted. In correspondence with these works, interactions are defined as follows:

Definition 7 Interaction

An interaction in a system is a unit of common activity shared by multiple system parts, through which co-operation between these system parts takes place for the purpose of establishing and exchanging information (Quartel, Ferreira Pires, Van Sinderen, Franken, & Vissers, 1997).

Figure 3-2 depicts the AMBER notation for actions and interactions. As described in the definiton, an interaction is a process in which multiple system parts participate to pass information, or to create new information. Such system parts can be software systems or human beings. In the context of our research, interactions typically take place between software systems and human end users.

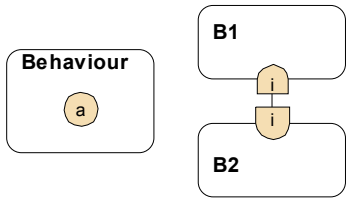


Figure 3-2 AMBER descriptions of an action a and an interaction i. Behaviours B1 and B2 perform interaction i in co-operation

An interaction can only take place if all parties involved have enabled the interaction, and all constraints imposed by the various contributing parties have been met. When the interaction has been completed, all parties involved receive information. As a minimal result, all parties involved receive an acknowledgement that the interaction has been completed. Apart from enabling the interaction, all parties involved may provide additional input for the interaction: the input parameters. Apart from receiving the acknowledgement, all parties involved may receive additional information when the interaction has been completed: the output parameters, denoted by the ι symbol. Since the various parties may have different concerns, they may refer to different parts of the result of an interaction. In any case, all participating parties are informed about the fact that an interaction has been completed.

As depicted in *Figure 3-2*, an action can be considered *internal behaviour* of an entity. As our approach focusses on the service perspective, we typically abstract from internal details. However, if that internal behaviour influences the externally observable behaviour and helps to increase the

understanding of the provided behaviour, it may be helpful to specify internal actions. For this reason, some of the descriptions of groupware behaviour in the remainder of this dissertation also specify associated actions performed by one entity.

3.2.3 Causality relations

A causality relation defines the condition for the occurrence of an action (Vissers et al., 2002). The arrow in *Figure 3-3* indicates a causal relation between action *a* and action *b*: only when action *a* has been completed can action *b* take place. In other words, action *a enables* action *b*.

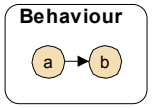


Figure 3-3 AMBER description of a causality relation between actions a and b

AMBER also defines how an action can enable multiple other actions. A diamond, such as the one depicted in *Figure 3-4*, indicates such a split. A solid diamond indicates an AND-split: all subsequent actions are enabled. An open diamond indicates an OR-split: only one of the subsequent actions is enabled. The arrows departing an OR-split typically state the conditions under which the following action will be enabled. Conditions can also be stated for causality relations between actions: only when action *a* has been completed and the condition is true can action *b* take place.

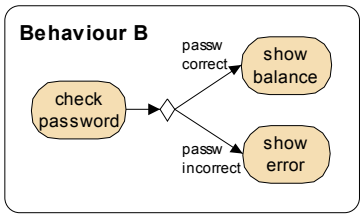


Figure 3-4 AMBER specification of an OR split

An action may also depend on multiple other actions, as illustrated in *Figure 3-5*. In that case the join operator, graphically denoted by a square, should be applied. The solid square, the AND-join, is applied to denote that *all* input actions have to be finished to enable the action, while the open square, the XOR-join, is applied to denote that only *one* of the input actions has to occur to enable the action.

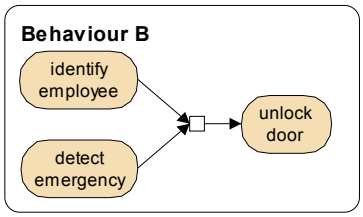


Figure 3-5 AMBER specification of an XOR join.

Note that when an action or interaction has been enabled, there is no guarantee that it will occur: the fact that an action or interaction has been enabled only indicates it is *allowed to* start. The sufficient conditions for an interaction to take place are that it has been enabled by *all* entities contributing to the interaction, and all conditions have been met.

3.2.4 Replicated and recursive behaviour

Some behaviours in real life are executed multiple times: either since multiple replicas of the behaviour may exist next to each other, denoted as *replicated behaviour*, or because similar behaviour is executed multiple times over time, denoted as *repeated behaviour* (Eertink, Janssen, Oude Luttighuis, Teeuw, & Vissers, 1999). When a behaviour causes a repetition of its own behaviour, it is denoted as *recursive behaviour*. The following figure shows the applied notation for a behaviour that is replicated an undefined number of times, recursive behaviour, and the notational shortcut applied to denote repeated behaviour.

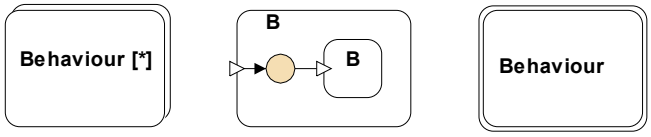


Figure 3-6 Replicated behaviour, recursive behaviour, and the notational shortcut for repeated behaviour

In our designs, replicated behaviour is typically applied to denote behaviour that is to be executed for a group of conference participants: for instance the behaviour to notify the existing participants about a new participant has to be executed for each of the existing participants.

3.2.5 Service and service primitive

In our work, a *service* is defined as follows:

Definition 8 Service

A service can be defined and expressed as a set of possible interactions between user and system that the system is capable of supporting (Vissers et al., 2002). As such, a service denotes the capabilities of a system as can be observed and experienced by its users.

The service concept is of prime importance since it determines the purpose of a system: it defines what benefit a system provides to its users. Apart from defining a service only in terms of observable behaviour, the service should be designed and expressed in a way that provides the most suitable basis for further design steps.

Structuring principles are the basic conceptual tools to achieve this, and to allow for assessing whether a service conforms to certain quality criteria, such as the ones expressed in section 3.6.

The service provided by a system can be defined and expressed as a set of possible interactions the system is capable of supporting. Such interactions take place between the system itself, denoted as the *service provider*, and users of that system. In general, *service users* can either be human users or other systems; in the context of our research it is essential that groupware services are used by human end users. Service users make use of services through *service access points* (SAPs). A SAP is uniquely identified by its location, denoted by the λ symbol. The interaction of a user with a service at a SAP is always in terms of one or more units of interaction, denoted in our research as *service primitives*.

3.2.6 Composability and extensibility

Definition 9 Composability

Composability is the property of a system that denotes how easy a function of that system can be composed by selecting and combining more basic component functions.

In the context of a service-oriented design approach, the term *service composition* denotes the activity of selecting and combining service modules to form new services.

Definition 10 Extensibility

Extensibility is the property of a system that denotes how easy new functions can be added to the system, without interference with existing functions.

In the context of a service-oriented design approach, *extending a service* denotes the activity of making a new service element available for service composition.

3.3 Groupware concepts

This section describes and relates the various concepts that are frequently applied when describing groupware and groupware services. A groupware application provides a groupware service to co-operating people. By means

of this service people can interact with groupware applications in order to establish contact with other people and start conferences. Subsequently, the groupware service allows the co-operating people to communicate and collaborate using shared information objects. The following figure applies an entity-relationship diagram notation to link the conference concept and related concepts.

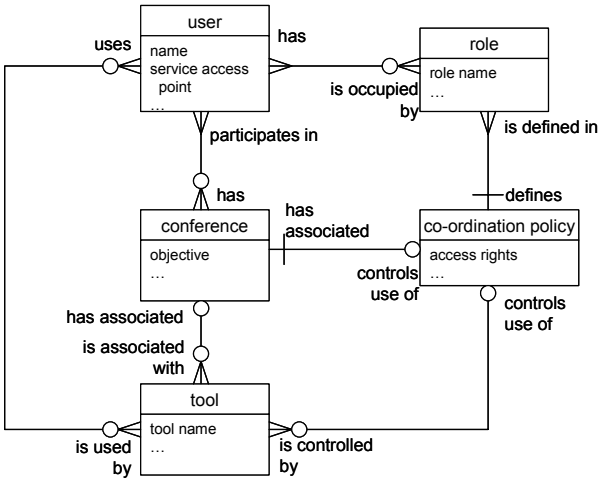


Figure 3-7 Groupware concepts and their relations

Conferences may have attributes, such as an objective. *Users* participate in zero or more conferences at the same time, as indicated by the crow's foot symbol and the circle: the crow's feet denotes the many-relationship while the circle denotes the optional part of the relation. According to this model, a conference has one or more users that participate.

The users in a conference may use the *tools* that are associated with a conference. An active tool is associated with zero or one conferences. A conference includes zero or more active tools. Any of these active tools may be used by zero or more conference participants.

A participant in a conference may have one or more *roles*, which are defined in a *co-ordination policy*. However, not every conference is associated with a co-ordination policy. A co-ordination policy controls the use of the conference and any of its associated tools.

Users in this model can be identified through their names, while each user has a Service Access Point through which he accesses the service provided by a groupware application.

3.4 Towards describing groupware services

To support people in their online collaboration, a groupware application provides a groupware service. As shown in *Figure 3-8*, a groupware service may consist of multiple active Groupware Service Modules (GSMs). GSMs form *units of composition* of groupware services: end users can select and compose GSMs to determine the provided groupware service. GSMs are designed and implemented based on the GSM *types* defined in chapter 6.

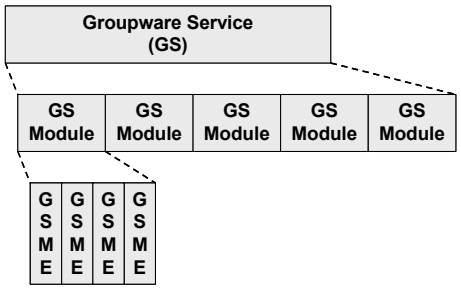


Figure 3-8 Consists-of relations between Groupware Service, Groupware Service Module and Groupware Service Module Element

A GSM type is defined by a grouping of multiple Groupware Service Module Element (GSME) types. GSMEs represent *elementary units of groupware behaviour* towards human end users. Chapter 5 describes these GSME types. An example of a GSME type is the StartConf GSME, which describes the behaviour to start conferences. The behaviour associated with GSME types can be expressed in terms of actions, interactions and causality relations: the basic architectural concepts introduced in section 3.2.

Specific to our approach is the existence of GSME types to adapt the composition of GSMs: the groupware service provides behaviour to adapt the groupware service.

Before a GSM can be included in a groupware composition it has to be made active. *Figure 3-9* illustrates the various stages in the lifecycle of a GSM. The CooPS groupware reference model, described in chapter 6, defines GSM types. Groupware programmers create GSM implementations that adhere to one of these GSM types. In the remainder of this dissertation, we sometimes apply the term GSM to denote a GSM implementation. After creation, GSMs can be made available to groupware service users. Available GSMs have to be activated in order to make use of the behaviour they provide. Activating one available GSM multiple times results in multiple instances of the activated GSM. The latter mechanism is for example needed when two shared whiteboards are active in the same conference.

To tailor a groupware service, groupware service users may select *available* as well as *active* GSMs for inclusion in the groupware composition:

the composition of GSMs associated with one conference. However, when an available GSM has been selected it will be activated before it can be composed with other GSMs.

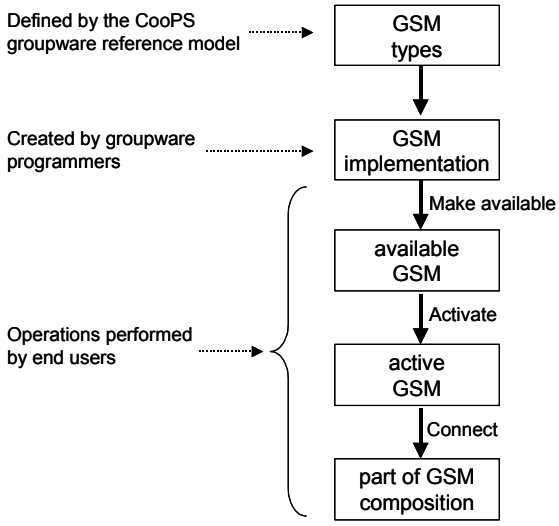


Figure 3-9 Stages in the lifecycle of a GSM

An example of selecting an *available* GSM for a groupware composition is the case where a groupware service user selects a new shared whiteboard for use in a conference. Similarly, an example of selecting an *active* GSM is the case where a groupware service user decides to share a document he is currently editing, and the application to edit the document, with a group of other people: in that case the already activated editing application is included in the GSM composition.

3.4.1 Two levels of GSME type descriptions

The GSME type descriptions in chapter 5 consist of two levels that focus on different aspects of GSME types:

1. A *local service interface* description. This description focusses on the interactions that take place at a single Service Access Point (SAP) between the groupware service user and the groupware service provider, as well as the relations between these interactions. It describes the ordering of interactions, the parameters that are established as a result of these interactions, and parameter value dependencies;
2. A high-level *description of the groupware service provider behaviour*. This level focusses on a high-level description of the relations between interactions that take place at distinct SAPs.

These two levels of descriptions have been selected as they provide complementary information needed for our design: the first level expresses the interaction between the end user and a groupware application in abstract terms. The second level describes the behaviour provided by a groupware application, including the distribution aspects on the service level.

3.4.2 Notational shortcuts

For clarity reasons, some figures leave out the details of individual GSMEs or combinations of GSMEs that are described in other figures as well. This is denoted as a *collapsed* view, as depicted in *Figure 3-10*.

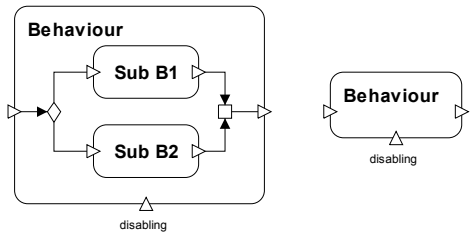


Figure 3-10 A view and its corresponding collapsed view

Causal relations that cross the boundary of a behaviour description are indicated by triangles, as indicated in *Figure 3-10*. This occurs when one behaviour enables another behaviour. Triangles pointing inward into a behaviour description denote an input relation: another behaviour enables the specified behaviour. Triangles pointing outward from a behaviour description denote an output relation: the specified behaviour enables another behaviour.

The triangle denoted as *disabling* that points inward into a behaviour specification is a shorthand notation, which is not part of the AMBER language. This notation indicates a disabling relation towards all actions and interactions inside the specified behaviour. Using this shorthand notation it is for instance possible to indicate that when a conference is ended, all GSMEs related to that conference are not available anymore.

3.5 Elementary interaction patterns

When multiple service users interact to pass information or to create new information, some generic interaction patterns can be distinguished that form the basis for many more complex interaction types. These elementary interaction patterns are: *unconfirmed message transfer*, *provider-confirmed message*

transfer, *user-confirmed message transfer*, and the *action-feedback-feedthrough* pattern. This section describes these four elementary interaction patterns.

3.5.1 Unconfirmed message transfer

The unconfirmed message, schematically depicted in *Figure 3-11*, is an elementary interaction patterns that involves multiple service users and a service provider: one service user sends a message to one or more other service users via the service provider. The sending service user does not receive an acknowledgement that the message is actually received by the other service users. The following figure depicts an unconfirmed message transfer between two service users and an unconfirmed message transfer involving more than two service users.

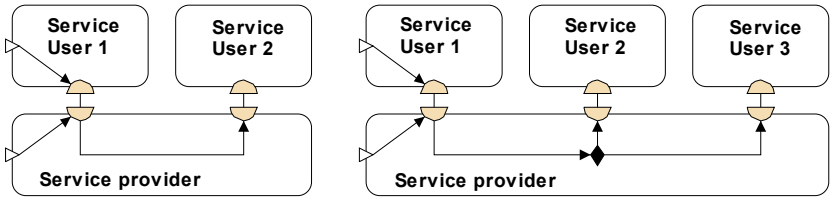


Figure 3-11 Unconfirmed message transfer

When applying an unconfirmed message transfer in a groupware service design, the address of the target service users, i.e., the target Service Access Points, should be specified as an established parameter of the first interaction.

3.5.2 Provider-confirmed message transfer

The provider-confirmed message transfer, schematically depicted in *Figure 3-12*, is an elementary interaction pattern that involves multiple service users and a service provider: one service user sends a message to one or more other service users via the service provider.

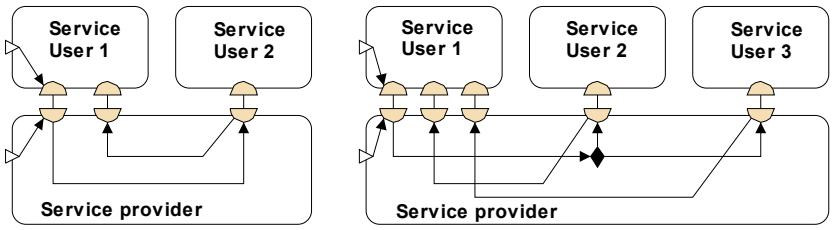


Figure 3-12 Provider-confirmed message transfer

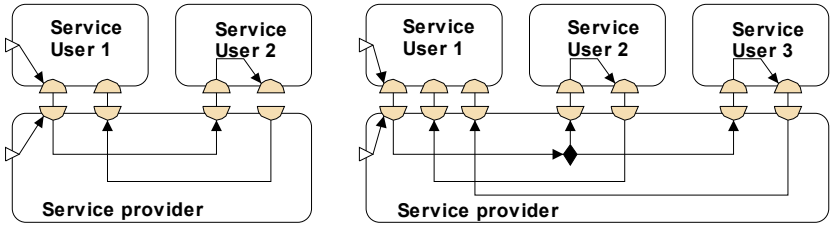
The sending service user receives an acknowledgement by the service provider when the message is actually received by the addressed service users, i.e., the receive interaction has been completed. When applying a provider-confirmed message transfer in a groupware service design, the

address of the target service users, i.e., the target Service Access Points, should be established as part of the first interaction. Additionally, the address of the initiator has to be passed on during the process in order to return the confirmation to the initiator.

3.5.3 User-confirmed message transfer

The user-confirmed message transfer is an elementary interaction pattern that involves multiple service users and a service provider: as schematically depicted in *Figure 3-13*, one service user sends a request to one or more other service users using the service provider. The sending service user receives a response from the addressed service users, based on some action by these users.

Figure 3-13 User-confirmed message transfer

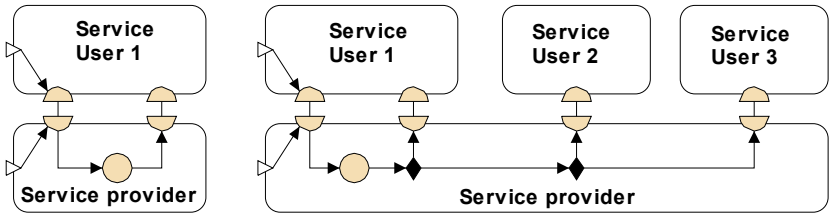


When applying a user-confirmed message transfer in a groupware service design the address of the targets, i.e., the target Service Access Points, should be a result of the first interaction. Additionally, the address of the initiator has to be passed on during the process in order to return the response to the initiator.

3.5.4 Action-feedback-feedthrough

The action-feedback-feedthrough pattern is an elementary interaction pattern that involves one or more service users and a service provider. As illustrated in *Figure 3-14*, one service user performs an interaction that results in some internal action in the service provider.

Figure 3-14 Action-feedback-feedthrough pattern



Such an internal action may for instance denote a state change in the service provider. As a result of the internal action the initiator receives feedback

information via a feedback interaction, while any other service users receive feedthrough information about the result of the action via a feedthrough interaction. When applying an action-feedback-feedthrough pattern, the addresses of the other service users, i.e., the other Service Access Points, should be established as part of the first interaction.

3.5.5 Expressing high-level behaviour using elementary interaction patterns

The previously described elementary interaction patterns form the basis for all complex interaction patterns in distributed services, such as groupware services. In order to express high-level behaviour using these elementary interaction patterns two mechanisms are applied: specializing the interactions by defining their meaning and parameters, and combining multiple elementary interaction patterns to construct a more complex pattern.

It is possible to express groupware services directly in terms of these elementary interaction patterns, just as it is possible to express the behaviour of any software application in terms of a Turing machine. However, by choosing an appropriate level of abstraction and defining appropriate levels of composition we aim to express the behaviour of groupware services in a more meaningful manner, appropriate for the purpose of our research.

3.6 Groupware behaviour patterns

Two basic patterns, i.e., sequences of actions and interactions, frequently occur in the groupware behaviour descriptions in chapter 5:

1. *GW_LocalRequest*. This pattern describes a series of actions and interactions that occur when a participant makes use of groupware behaviour that does not directly impact other participants in the conference. The person who initiates the groupware behaviour, denoted as the *initiator*, receives feedback, i.e., a response, based on his action. The GW_LocalRequest pattern for instance occurs when a participant requests the list of current conference participants;
2. *GW_RemoteInteraction*. This pattern describes a series of actions and interactions that occur when a participant makes use of groupware behaviour that may impact other participants in the conference. The person who initiates the behaviour receives feedback based on his action, while the other participants in the conference receive feedthrough based on the results of the action.

Most groupware behaviour descriptions in chapter 5 are based on these two patterns. The patterns may be extended with additional actions and interactions to accomplish additional behaviour. Additional actions can for instance be applied to denote internal activities, such as storing or retrieving state information.

3.6.1 GW_LocalRequest

The GW_LocalRequest pattern is applied when modelling groupware behaviour that impacts only the user who initiated the behaviour. The pattern, illustrated in *Figure 3-15*, is based on the action-feedback-feedthrough pattern: it starts with an interaction between the groupware service user and the groupware service provider. In the context of our research, a groupware service user is a human end user. In this interaction the groupware service user requests some behaviour by the groupware service provider, and specifies any possible parameters. When the groupware service provider completes the behaviour, the groupware service user receives an indication, i.e., *feedback*, about the result.

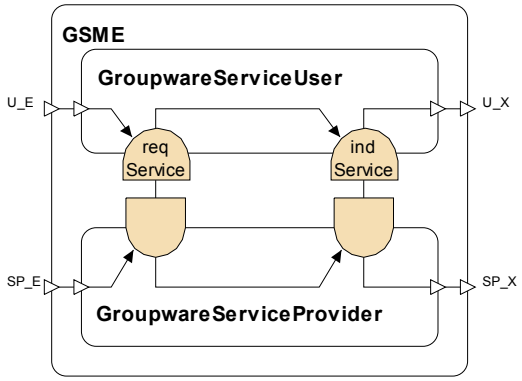


Figure 3-15 GW_Local Request: Local service interface description

On the local service interface level, the pattern shows that one groupware service user interacts with the groupware service provider by means of two related interactions. The groupware behaviour descriptions in chapter 5 state the precise interactions that take place, the parameters that are established as a result of these interactions and the relations between the established parameter values.

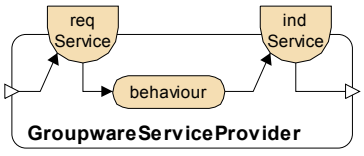


Figure 3-16 GW_ LocalRequest: Groupware service provider behaviour

Figure 3-16 shows the high-level description of the behaviour of the groupware service provider. It shows that the groupware service provider performs some internal behaviour as a result of the reqService interaction. The result of this interaction is input for the indService interaction between the groupware service user and the groupware service provider.

3.6.2 GW_RemoteInteraction

Similar to the *GW_LocalRequest*, the *GW_RemoteInteraction* pattern is also based on the action-feedback-feedthrough pattern. The pattern describes how a groupware service user requests an action by the groupware service provider. In contrast to the previous groupware service pattern, other groupware service users (e.g., the other participants in a conference) are notified about the result of the action as well (denoted as *feedthrough*).

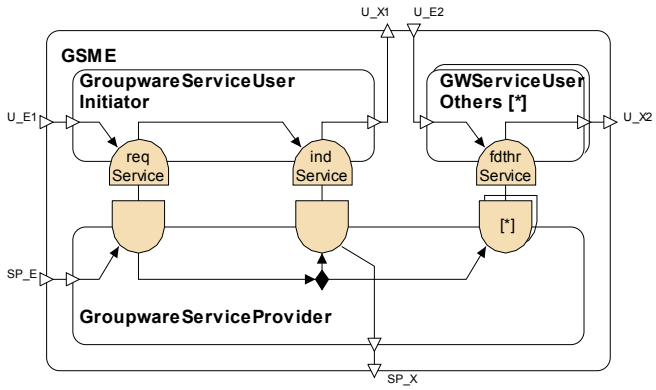


Figure 3-17 GW_ RemoteInteraction: Local service interface descriptions

The GW_RemoteInteraction pattern is typically applied in groupware behaviour that impacts not just the person who initiated the behaviour, but also other people. *Figure 3-17* shows that some groupware service user, denoted as the initiator, requests some service by means of the reqService interaction. The figure shows that the indService interaction can only take place after the reqService interaction has finished. Similarly, the various interactions with feedthrough information, the fdthrService interactions, can only take place between the groupware service provider and the other people after the reqService interaction has been completed.

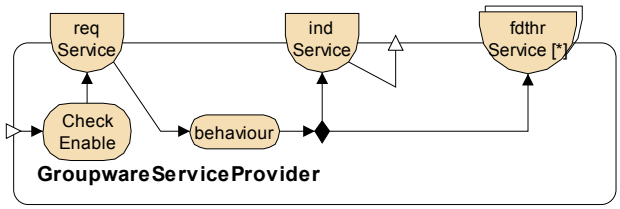


Figure 3-18 GW_ RemoteInteraction: Groupware service provider behaviour

Figure 3-18 illustrates, at an abstract level, the behaviour provided by the groupware service provider as part of the GW_RemoteInteraction pattern: the reqService interaction is enabled after a check has successfully been performed by the groupware service provider. This mechanism is applied to model a check to assess whether a given groupware service user is allowed to use the specified groupware behaviour in a given conference. Such a check has been omitted from the GW_LocalRequest pattern since none of the GSME types described in chapter 5 that adhere to this pattern do include such a check.

As a result of the reqService interaction some behaviour is performed by the groupware service provider. The result of this behaviour is input to the indService interaction with the service initiator. Additionally, the result of this behaviour is input for the feedthrough interaction between the groupware service provider and the other people in the conference.

The output triangle following the indService interaction denotes a causal relation: after the indService interaction has taken place, other behaviour is enabled. In such cases, the textual description of the GSME type states what behaviour is enabled by this GSME type.

3.6.3 Regulating access to groupware behaviour

Groupware applications typically require mechanisms to regulate access to groupware behaviour. Three main situations exist that require a mechanism to check whether a user is allowed to access groupware behaviour:

1. *Role-based access control*. Based on the role of a participant access is granted to specific GSMEs. As an example, a groupware application may enact a policy that only participants with the role of teacher are allowed to change the address of the website that is being shared, while all participants are allowed to ask questions using the chat tool;
2. *Workflow*. Based on the phase of the co-operative process and the role of a participant, access is granted to specific GSMEs. As an example, a groupware application may enact a policy that during the brainstorm phase of the meeting all participants are allowed to add ideas to the shared database, while during the selection phase they are only allowed to use the voting tool, and they no longer are allowed to add items to the shared database;
3. *Concurrency control*. If multiple users co-operate on shared information objects their actions can potentially conflict. Groupware applications may implement mechanisms to detect, avoid, or resolve such conflicts.

CheckEnable

Many GSME type descriptions in chapter 5 include an internal action denoted as CheckEnable. The purpose of this check is to see whether the

associated groupware behaviour should be enabled for a given groupware service user in the given conference. Before a groupware user is able to make use of a GSME, the groupware service provider has to enable the associated interaction. This check helps groupware designers to create applications that offer groupware service users only the groupware behaviour they can actually use.

3.6.4 Applying GSME patterns

The GSME patterns in the previous sections describe patterns of actions and interactions that frequently occur as part of groupware behaviour. However, GSME types that follow these patterns also have specific aspects:

- The type of interactions between the groupware service user and the groupware service provider are specific for a particular GSME type. Similarly, the parameters associated with these interactions are specific;
- The behaviour provided by the groupware service provider as a result of the service request is specific;
- GSME types based on the GW_RemoteInteraction pattern include a specific check. This check is applied to assess whether the given groupware behaviour may currently be used by the given groupware service user in the given conference. As an exception, this check is omitted in the UpdatePersonalInfo GSME type, even though it is designed based on the GW_RemoteInteraction pattern.

3.7 Groupware service design criteria

One of the objectives of the architectural concepts and notation introduced in this chapter is to be able to assess the quality of the provided design. As described in the introduction, the *behaviour* a system provides towards its users is the most important aspect of that system. We assume that the user's prime interests are summarized in the following questions:

> *Does the system provide the functions I need for my task, does it react to my input in a timely manner, and can I interact with the system in an effective, efficient and pleasing way?*

However, the quality of a groupware design is not just determined by the functionality it provides to its users and its usability: also generic architectural criteria determine the quality of a design. This section outlines the relevant design criteria and describes their impact on our research.

3.7.1 Overview of usability criteria

One of the most important criteria for any interactive system is usability. Usability denotes how well end users can use the system. A contributing factor to increase usability is a consistent and intuitive user interface that matches the user's mental model of the system and fits the task for which the system is used. According to ISO standard 9241, usability is defined as the effectiveness, efficiency, and satisfaction with which specified users achieve specified goals in particular environments. These usability metrics, which are not orthogonal, are defined as follows:

- *Effectiveness* ("does it do the right things?"). Defined as the accuracy and completeness with which specified users can achieve specified goals in particular environments. In other words, does the system help the user to reach his objective? In our research, this corresponds to the requirement that a groupware application should support the co-operating people with the right set of GSMEs to perform their co-operative tasks, given their particular contexts.
- *Efficiency* ("does it do the things right?"). Defined as the resources expended in relation to the accuracy and completeness of goals achieved. In other words, does the system provide adequate support? Does it help the user avoid taking unneeded steps and making mistakes?
- *Satisfaction*. Defined as the comfort and acceptability of the work system to its users and other people affected by its use. In other words, does the user feel well using the system, or does using the system frustrate the user? Given the abstract level of our design, this criterion is only affected indirectly by our design.

3.7.2 Overview of run-time criteria

Run-time criteria are those criteria that are observable when using the system. In fact, the usability criteria can be regarded as part of the run-time criteria. Given the context of our research, other run-time criteria are:

- *Functionality*. The ability of the system to do the work for which it was intended. This requirement, even though it is non-architectural in nature, is important in our research as it corresponds to the ability of a groupware application to properly support co-operating people in their co-operative tasks. More precisely, the groupware application should be able to provide co-operating people with the groupware behaviour needed for their co-operative tasks.
- *Tailorability*. The ability of a system to be adapted by end users within the context of system use (Mørch, 1997a). Essential to our research is the fact that end users are empowered to tailor the *behaviour* of their groupware applications within the context of system use.

- *Performance*. The responsiveness of the system - the time required to respond to stimuli (events) or the number of events processed in some interval of time. In groupware, this translates for instance to the time it takes before changes to a shared information object become visible for the other participants in a conference. Although this aspect is important for groupware in general, it is not a focal point of our research.
- *Security*. A measure of the system's ability to resist unauthorized attempts at usage and denial of service while still providing its behaviour to legitimate users. In groupware, this corresponds to how well a groupware application protects the co-operation and prevents unauthorized people to eavesdrop on the communication, or access restricted groupware functions. Although this aspect is important for groupware in general, it is not a focal point of our research.
- *Data consistency management*. Data consistency management in a distributed system (such as a groupware application) indicates the system's ability to maintain consistency between distributed replicas of the same data. The fundamental cause of the emergence of inconsistency is that mapping actions at one location onto actions at another location consumes (an unpredictable amount of) time. This may result in different order of actions at different locations (ter Hofte, 1998). Although this aspect is important for groupware in general, it is not a focal point of our research.
- *Reliability*. Traditionally, a measure of the system's ability to keep operating over time. In groupware, reliability also encompasses aspects such as fault tolerance, handling (temporal) network unavailability, and handling the unavailability of one or more participants in a conference. Although this aspect is important for groupware in general, it is not a focal point of our research.
- *Availability*. Availability measures the proportion of time the system is up and running. It is measured by the length of time between failures as well as by how quickly the system is able to resume operation in the event of failure. In groupware, this corresponds to the proportion of time a groupware application is available to support co-operation. Although this aspect is important for groupware in general, it is not a focal point of our research.
- *Scalability*. The ability of a system to serve many clients. In groupware, this corresponds to the ability of a groupware application to handle large groups of concurrent users, possibly in a single online conference. Although this aspect is important for groupware in general, it is not a focal point of our research.
- *Interoperability*. In a component-based design of a distributed system, such as groupware, one can distinguish two forms of interoperability: interoperability *within* an application and interoperability *between* the

applications of different users.
The first form of interoperability denotes the ability of two or more parts of the application to exchange information and use that information. Interoperability within an application is increased when system parts interact through well-defined interfaces only and share the semantics of the exchanged information. This form of interoperability is required to be able to compose a groupware application out of systems parts that may originate from different manufacturers. Our research forms a basis to achieve this form of interoperability.
The second form of interoperability denotes the ability of two or more applications or systems to exchange information and use the information that has been exchanged. This type of interoperability is increased when systems co-operate using generally accepted communication protocol standards and through well-defined interfaces only. In groupware, this corresponds to the ability of a groupware application to support co-operation with people who have a different type of groupware application. Although this aspect is important for groupware in general, it is not a focal point of our research.

3.7.3 Overview of generic architectural criteria

There is no such thing as an inherently good or bad architecture. Architectures are either more or less fit for some stated purpose (Bass, Clements, & Kazman, 1999, p. 17). In our research the following definition of a software architecture is applied:

Definition 11 Software architecture

The software architecture of a program or computing system is the structure of the system, which describes the software components, the externally visible properties of those components, and the relationships among them (Bass et al., 1999).

Although an architecture is not inherently good or bad, there exist some generic criteria that hold for any software architecture:

- *Generality*. This denotes the degree of generic applicability of an architecture. In contrast to a design that is created for a specific purpose, a generic design can be applied for a wide range of purposes.
- *Conceptual consistency*. This denotes the presence of one underlying theme or vision that unifies the design of the system, at all levels. This property promotes the architecture to do similar things in the similar ways (Bass et al., 1999). Conceptual consistency is regarded a very important consideration in systems design (Brooks, 1975).
- *Correctness* and *completeness* of the architecture. Correctness and completeness of a design are needed in order to meet the system's requirements and run-time resource constraints.

- *Orthogonality* in the architecture. This denotes the level of separation of concerns: does the architecture separate different concerns to allow for independent solutions. In an electronic transaction system for instance, the used transaction mechanism should be independent from the used security mechanism, to allow for independent updates of both.
- *Propriety* and *parsimony*. Propriety denotes that the architecture's features are *proper* to the to the purpose of the system, where the purpose of the system is determined by the user requirements. As a result, the architecture does not describe features that do not belong there, or are in some way extraneous. Propriety implies parsimony. Not only should designs only define relevant functions, corresponding to essential, well understood user and domain requirements, they also should define each function only once (Van Sinderen, 1995).
- *Adaptability* (by programmers). Denotes the degree to which a system or component facilitates the incorporation of changes by programmers, once the nature of the desired change has been determined. This aspect is most closely aligned with the architecture of a system. The ability to make changes quickly and cost effectively follows directly from the architecture; it is largely a function of the locality of any change. Since an architecture defines the functional entities that form an application, an architecture also defines which changes are local. In groupware, this translates for instance to the degree to which an application allows for adaptations to the set of available GSMEs.
- *Reusability*. The degree to which a software module or other work product can be used in more than one software system. Designing for reusability means that the system is structured so that its components can be chosen from previously built products. Reusability is related to the software architecture since architectural components are the units of reuse, and how reusable a component is depends on how tightly coupled it is with other components. In groupware, this corresponds to the ability to reuse groupware components in groupware applications for new domains. Although this aspect is important for groupware in general, it is not a focal point of our research.
- *Buildability*. This allows the system to be implemented in a timely manner and to be open to certain changes as the system development progresses (Bass et al., 1999). Typically, the buildability of a system architecture refers to the ease of constructing a system according to this architecture, and is measured in terms of cost and time. The buildability of our design is assessed by the proof-of-concept demonstrator, presented in chapter 7.

A system architecture identifies the functional entities that comprise a system, the responsibilities of these functional entities, and the relations

between them. Regarding the identified functional entities, a system architecture should possess the following qualities:

- *High coherence*. Functions and responsibilities that belong together should be grouped in one functional entity. In our research, this corresponds to grouping all functions related to one type of communication (e.g., audio conferencing) in one functional entity, and separating them from other groupware functions.
- *Low coupling*. The interdependencies between different functional entities should be minimised. Low coupling increases the possibilities to replace system parts, without affecting other functional entities. In our research, this corresponds to minimizing the exchange of information between functional entities. If two functional entities exchange much information, it may be needed to combine them.
- *Low complexity*. The resulting groupware architecture should have a low complexity, in terms of the amount of functional entities, the number of different types of functional entities, and the number of relations between the functional entities. In our research, this corresponds to an architecture that identifies those functional entities that are *required* and *sufficient* to create a large range of groupware services.

3.7.4 Most relevant groupware service design criteria

Combining the previous criteria, we derive the claim that groupware should provide co-operating people with *appropriate functionality*, to work together in an *effective* and *efficient* manner. Given our assumption that co-operating end-users require tailorable groupware services to cope with the dynamics of co-operative settings, the principle of *tailorability* is also very important.

In our approach, the most relevant groupware service design criteria are:

1. *Generality*. A groupware service design should be applicable for a wide range of co-operative settings;
2. *Conceptual consistency*. In groupware service design, similar interactions between the groupware service user and the groupware service provider should be done in a similar fashion;
3. *Correctness and completeness*. A groupware service design should be correct and complete in the sense that it should perform all required functions in a correct manner;
4. *Orthogonality*. In groupware service design, different concerns should be handled by independent system parts;
5. *Propriety and parsimony*. A groupware service design should only include essential functionality and avoid redundant functionality;
6. *Interoperability within applications*. As we strive for groupware services that can be formed by selecting and composing groupware service modules, interoperability between groupware service modules is essential.

Chapter 4

Generic model of groupware services

As depicted in *Figure 3-8* on page 44, a groupware service is formed by a composition of Groupware Service Modules (GSMs). The various GSM types are defined in chapter 6. GSMs are formed by grouping more elementary units of groupware behaviour, denoted as Groupware Service Module Element (GSME) types. This chapter describes a generic model of groupware services that identifies the various GSME types, while chapter 5 provides the details regarding the individual GSME types. The groupware service model is derived and described in the following steps:

1. By describing the applied criteria for defining GSME types;
2. By discussing groupware literature, as a basis to derive a set of GSME types;
3. By discussing the dynamics in groupware behaviour use, in order to derive what aspects of groupware services have to be tailorable;
4. By presenting a generic model of groupware services. This model identifies the various GSME types and the causal relations between them.

4.1 Introduction

The primary purpose of a groupware application is to support people in performing co-operative work. To achieve this purpose, a groupware application provides a *groupware service* to its users. As described in section 3.4, a groupware service is formed by a composition of Groupware Service Modules (GSMs). In turn, one GSM consists of multiple Groupware Service Module Elements (GSMEs).

A GSME represents an *elementary unit of groupware behaviour* towards end users. Although, in theory, an infinite number of GSME implementations

may exist, this chapter describes the finite set of GSME *types*. The first part of the chapter describes a model of groupware services, and the process to derive this model. In the second part of the chapter, the individual GSME types are described in detail, using the concepts defined in chapter 3. These GSME types allow groupware designers:

1. To *prescribe* the behaviour of a groupware application;
2. To *describe* the behaviour of existing groupware applications, thus providing a basis for comparing groupware applications.

The structured approach applied to specify GSME types allows one to state the actions and interactions that take place as part of the behaviour, as well as the associated dependencies. At the same time, the formal descriptions allow for analysis of the design based on the criteria defined in section 3.7.

4.1.1 Towards informed groupware design decisions

The main purpose of structured descriptions of GSME types is to be able to take informed groupware service design decisions. The descriptions have to be detailed enough to state what actions and interactions take place as a result of each GSME type, and the relations between these actions and interactions. On the other hand, the descriptions have to be coarse-grained enough to reveal dependencies between various GSME types, for instance as a result of causal relations. The GSME type descriptions focus on three aspects:

1. The *local service interface*, describing the abstract interface between groupware service users and the groupware service provider;
2. *Interaction details*, describing the details of each interaction in the local service interface in terms of the parameters established and their values;
3. *Groupware service provider behaviour*, focussing on the distributed behaviour. This part of the description states actions that are performed by the groupware service provider and the relations between these actions and the interactions defined in the local service interface.

These descriptions form the basis to group GSME types in order to form Groupware Service Modules (GSM) types. This process, as well as the motivation to group certain GSME types, is described in chapter 6.

4.2 Criteria for defining GSME types

Groupware can potentially provide many different types of functions to end-users. To select an appropriate set of Groupware Service Module Element types the following rule is applied:

A GSME describes a unit of groupware behaviour: it describes a recognizable unit of groupware behaviour essential to describe and prescribe the behaviour associated with groupware services.

Consequently, a GSME type has to be *necessary* to express groupware services, and that the complete set of GSME types should be *sufficient* to express the groupware service provided by any groupware application.

The requirement that a GSME type has to be a *recognizable* to tailors is quite subjective. Nevertheless, we consider this requirement to be important since we assume that when tailors recognize GSMEs, the likelihood of successful tailoring increases. In our design, tailors have to be able to select and compose the groupware behaviour they need. So, on a service level, individual GSME types should be recognizable: a GSME type needs to have *unique externally observable behaviour*: every GSME type should include a unique combination of interactions between the groupware service user and the groupware service provider.

A GSME type description should state its *relations with other GSME types*. Our methodology distinguishes two types of causal relations between GSME types on a service level: First, a GSME type may be enabled or disabled by one or more other GSME types. Second, a GSME type may use status information that is established as part of another GSME type. *Figure 4-1*, *Figure 4-2* and *Figure 4-3* on page 72 and further depict the enabling and disabling relations between GSME types. All causal relations between GSME types are made explicit in the individual GSME type descriptions.

A GSME represents a *unit of allocation*: the complete behaviour associated with one GSME is allocated to one Groupware Service Module (GSM) type. However, one GSME type may be allocated to multiple GSM types.

4.2.1 Influence of criteria

The individual GSME type descriptions adhere to the criteria described in section 3.7. In particular, the following criteria have shaped the design:

- *Generality*: the set of GSME types in our design has to be sufficiently generic to describe the behaviour of a large set of groupware services;
- *Effectiveness*: the identified set of GSME types has that allow for effective co-operation between groupware service users, given their particular contexts;
- *Efficiency*: the design avoids unneeded interactions;
- *Tailorability*: the design allows groupware service users to select and compose the groupware behaviour that suits their needs;
- *Conceptual consistency*: in the design, similar GSME types are designed in a similar fashion, requiring similar groupware user interactions. In order

to represent similar information, similar parameters are applied in GSME type descriptions;
– *Completeness*: the design should include all groupware behaviour needed for effective co-operation.

To illustrate how these criteria have influenced the design, consider the aspect of *conceptual consistency*: all groupware behaviour descriptions in this dissertation are variations or combinations of the elementary interaction patterns, described in section 3.5. As a result, all groupware behaviour descriptions start with a *request* interaction in which both the groupware service user and the groupware service provider (i.e., the groupware application) participate. This request interaction is always followed by an *indicate* interaction, or – when another user confirmed the request – a *confirm* interaction.

Similarly, always upon receiving feedthrough information, the groupware user is informed by the groupware service provider by means of a *feedthrough* interaction.

4.3 Groupware behaviour identified in literature

The last decade, many scientists have published findings regarding the behaviour a groupware application should provide to co-operating people. This section summarizes the findings of frequently cited researchers, without the intention to completely describe their work. We have combined their findings with our own experiences to derive a set of GSME types.

4.3.1 Ellis, Gibbs, and Rein

In an early paper on issues and experiences related to groupware, Ellis, Gibbs and Rein distinguish three main categories of groupware functions: communication, collaboration and co-ordination functions (Ellis et al., 1991). Without specifying detailed functions, they state that groupware applications should be able to support (direct) inter-personal communication and collaboration by means of shared information objects such as a whiteboard or a document. Furthermore, groupware applications should provide functions to co-ordinate group activities, to increase the effectiveness of communication and collaboration. This co-ordination can be viewed as an activity in itself, as a necessary overhead when several parties are performing a task together.

Co-operation can be synchronous, as in spoken conversation, when people interact in real time. In asynchronous communication, such as

postal correspondence, people interact over an extended period of time. Additionally, intermediate forms of communication exist, and people can switch between synchronous discussions to asynchronous work on a shared document. Ellis et al. focus on synchronous forms of groupware, and introduce the term *session* to indicate a period of synchronous interaction supported by a groupware system (Ellis et al., 1991). Their term *session* corresponds to our concept of a *conference*.

4.3.2 Ter Hofte

The CoMeCo Groupware Service Architecture by Ter Hofte (1998) describes the behaviour of a groupware application in terms of *conference management* functions (starting and stopping conferences, participation management, media management), *media* (corresponding to the communication and collaboration functions distinguished by Ellis), and *co-ordination* functions (floor control and access control functions). Since the CoMeCo Groupware Service Architecture lists groupware functions that are meaningful and important for co-operating end users, these functions have been a starting point for the GSME types defined in this chapter. Some functions identified in the CoMeCo Groupware Service Architecture are considered outside the scope of our research and have been omitted: for instance, the functions to manipulate conference hierarchies is considered to be too specific given the context of our design.

4.3.3 Baker, Greenberg and Gutwin

Baker, Greenberg and Gutwin propose eight heuristics to evaluate the usability of a special type of groupware applications, namely shared virtual workspaces (Baker et al., 2002). These groupware heuristics are based on a theoretical framework called *the mechanics of collaboration* (Gutwin & Greenberg, 2000), which states seven major activities that comprise collaboration, viz., explicit (or direct) communication, consequential (or indirect) communication, co-ordination of action, planning, monitoring, assistance, and protection. The eight groupware heuristics can be used by evaluators to inspect shared virtual workspace groupware to see how well the applications support teamwork. In particular, the heuristics can be used to assess the usability of these applications. According to these eight heuristics, a groupware application has to:

1. Provide the means for intentional and appropriate verbal communication;
2. Provide the means for intentional and appropriate gestural communication;
3. Provide consequential communication of an individual's embodiment, for instance the individual's facial expression;

4. Provide consequential communication of shared artefacts (i.e., artefact feedthrough);
5. Provide protection;
6. Manage the transition between tightly and loosely-coupled collaboration;
7. Support people with the co-ordination of their actions;
8. Facilitate finding collaborators and establishing contact.

Although Baker, Greenberg and Gutwin state that their groupware heuristics are aimed at shared virtual workspaces, the heuristics can also be applied to a wider range of groupware applications; they also apply to synchronous forms of communication (see heuristics 1, 2 and 3). This makes these heuristics for instance also valuable to evaluate audio- and videoconferencing applications.

The given heuristics reveal what functions a groupware application should provide from the end user perspective. For instance, heuristic 1 reveals that a groupware application should provide a function to convey verbal communication. As such, the heuristics form a source to identify what behaviour groupware applications should provide to end-users.

4.3.4 Greenberg and Roseman

Greenberg and Roseman identify the following conference management functions a groupware application should be able to provide (Greenberg & Roseman, 1999):

- Creating new conferences;
- Naming conferences;
- Deleting conferences;
- Locating existing conferences;
- Finding out who is in a conference;
- Joining people to conferences;
- Controlling access to conferences;
- Allowing latecomers;
- Allowing people to leave conferences;
- Deciding whether conferences persist when all users exit.

As described by Greenberg & Roseman (1999), the interface to access these conference management functions could present these functions as explicit steps a user takes to begin and maintain collaboration.

4.3.5 Dewan

Dewan decomposes the behaviour of a collaborative application into the following categories (Dewan, 2001):

1. Single-user interaction – determines the effect of users' commands on their own displays;
2. Coupling – determines the effect of users' commands on the displays of other users;
3. Undo – allows users to cancel/repeat previous actions;
4. Diffing – finds the differences in independent versions of an object;
5. Merging – combines independent versions into a single object;
6. Concurrency control – prevents inconsistent concurrent authorized interactions;
7. Process/workflow management – prevents execution of authorized commands that are inconsistent with the current process/workflow step and also automatically enacts commands according to the current process/workflow step;
8. Awareness – makes users aware of "out of band" activities of their collaborators, that is, activities not deducible from the application feedback they receive from coupling, concurrency control, and other functions above;
9. Session management – allows users to start/stop interaction with a collaborative application.

The objective in this research by Dewan is to facilitate the implementation and evaluation of collaborative applications and infrastructures. As such, the identified functions are those that need to be implemented by groupware programmers. Although his objective is a valuable one, the current research focusses on the service a groupware application needs to provide, not on how that service is to be implemented.

Consequently, not all groupware functions identified by Dewan may be suitable for the current research. When creating a detailed design based on our groupware reference model, functions such as undo, diffing and merging are indispensable. However, in our service-level design we choose not to include tool specific functions, as this large set of functions increases the complexity of the model. Tool specific functions to send and receive information, create, read, update and delete data, as well as functions such as undo, diffing and merging should be considered after the key functions and key relations in the design.

Nevertheless, the decomposition by Dewan reveals important functions related to conference management, process & workflow management, and awareness.

4.3.6 Kausar and Crowcroft

Kausar and Crowcroft specify functions a groupware application needs to present to the collaborating people to manage a conference (Kausar &

Crowcroft, 1999). These conference control functions are divided into different functional groups: functions related to conference configuration, participation management, floor control and security.

The functions related to conference configuration essentially allow end users to define a conference profile. They provide a means for specifying and enforcing conference policies. Functions for charging and billing to join a conference may also be included. A conference profile can define permissible participants, available roles (e.g. chair, speaker) and associated permissions. Roles may be assigned at the start of a conference, or may be requested during the conference and then granted, denied, or shared.

Participation management comprises functions for setting up conferences, inviting people, and terminating conferences. Furthermore, functions for charging the membership of a conference may be included. Participants may join or leave a conference on their own, or they may be invited or excluded from a conference. Kausar and Crowcroft also identify functions to switch from one conference to another. Our design focusses on the behaviour needed to support a co-operation setting, not the behaviour to switch between various co-operative settings. As such, our design does not include GSME types that support this behaviour.

Kausar and Crowcroft identify functions for floor control. In CSCW research, "assigning the floor to a speaker" is a metaphor for granting a specific user or group of users the right to access and manipulate a sharable resource, such as a shared drawing or a shared application. Floor control prevents for instance conflicting concurrent changes to shared information objects by multiple participants.

The security functions Kausar and Crowcroft identify are value-added options for conference control and are a part of participation management as well as conference configuration. Authentication is performed before a participant enters a conference and may be repeated arbitrarily during the conference course.

4.3.7 Edwards

The research by Edwards (1994) identifies groupware functions related to starting co-operation. He distinguishes explicit and implicit mechanisms to start interaction. An example of an explicit mechanism is a person who invites another person for a co-operative session. Implicit mechanisms can for instance be based on similarities in (virtual) location, activities or simultaneous use of an information object. For instance, the fact that two people concurrently try to access the same document may be a trigger to start online communication between them.

4.4 Dynamics of groupware behaviour use

The way people co-operate is likely to change over time (Kahler et al., 2000). People may for instance switch between tasks in ways that cannot be foreseen, the context in which they perform a task may change (e.g., since they switch between their office, their car, and their home), or the constitution of the co-operating group may change (Mandviwalla & Olfman, 1994).

Section 2.4 suggests that to properly support people in their co-operative work, the behaviour of a groupware application has to match the requirements of that specific setting. Combining these two aspects, groupware has to be able to provide different behaviour to support co-operation over time (Bardram, 1998). In order to identify what aspects of groupware services should be made flexible, the dynamics of real life co-operation have been investigated.

4.4.1 Communication media

Depending on their tasks, preferences, and context, people should be able to select appropriate media to communicate (Cockburn & Greenberg, 1993; Mandviwalla & Olfman, 1994; Zigurs & Buckland, 1998). The noisy environment of a factory may require text-based communication, while audio communication may be preferable for quick discussions in the office.

One of the most influential and predominating theories in the area of communication media selection is the Media Richness Theory (MRT) (Daft & Lengel, 1984; Daft, Lengel, & Trevino, 1987). This theory considers media selection as a rational process to match characteristics of a medium and message content in order to reduce ambiguity. MRT suggests that media can be ranked according to their richness. The theory considers face-to-face the richest and written communication as the leanest medium. It is proposed that effective use of a medium occurs when the richness of this medium matches the information requirements and ambiguity of the tasks people perform (Daft et al., 1987).

Social Influence Theory (Fulk, Schmitz, & Steinfield, 1990) can be seen as complementing or opposing MRT. Social Influence Theory proposes media selection as a process based on an individuals' perception, preference of a particular medium and social context. Related to this theory is the notion of "media stickiness" (Steinfield et al., 2001). Due to media stickiness, teams tend to continue using the media they have initially chosen, even when these tools are not ideal for their co-operative tasks (anymore).

The latter observation can be regarded evidence *against* the need for tailoring. Nevertheless, both Media Richness Theory and Social Influence

Theory reveal the need to select communication media: in Media Richness Theory this selection is based on task characteristics, in Social Influence Theory this selection is based on personal preferences and peer pressure. The current research aims to reduce media stickiness by making it as easy as possible to switch between various communication media. In related research, Hettinga suggests that media stickiness can be reduced by stimulating co-operating teams to reflect on their way of working, possibly with help from an external facilitator (Hettinga, 2002).

4.4.2 Shared information objects

Similarly, different tasks may require sharing different types of information objects (Mandviwalla & Olfman, 1994). While physicians may need to share high-resolution x-ray images at one moment, they may need to share documents with treatment plans during a next stage of the meeting. A groupware application should be flexible enough to support such switches in types of shared information objects. In order to effectively and efficiently use shared information objects during a conference, these shared information objects have to be accessible in an easy and quick manner. All conference participants must be able to view, point at, and if appropriate, adapt such shared information objects (Fish, Kraut, Root, & Rice, 1992).

4.4.3 Co-ordination of co-operation

As commonly found in groupware literature (e.g., (Kausar & Crowcroft, 1999; Edwards, 1996; ter Hofte, 1998)), some types of co-operation require explicit co-ordination of the manner in which people work together. Communication within large groups may for instance be more efficient with a chairman who can grant the floor to specific people. On the other hand, explicit user roles and access rights may restrict communication in unwanted ways.

There exist fundamentally different mechanisms to co-ordinate co-operation. A well-known example is role-based access control, where groupware service users are granted access to groupware behaviour based on their role. Another example is a workflow that specifies the order of actions and the people to perform these actions. Co-ordination policies are defined and enacted using co-ordination engines.

As a result, a groupware application should allow groupware service users to specify whether a co-ordination engine should be associated with the conference, and if so, which co-ordination engine to use.

4.4.4 Meeting styles

As described by Edwards (1994) and Kristoffersen & Ljungberg (1999) people apply different styles to join together into a collaborative session. Meeting styles range from explicit invitations to implicit chance encounters in a shared (virtual or physical) location. Research by Erickson and Kellogg (2000) indicates that people prefer to know who else is available for communication (presence awareness) and use this information to find appropriate occasions for communication. From this we derive that groupware applications should be able to support various mechanisms to start communication, corresponding to different meeting styles.

4.4.5 Conference management styles

The term conference management refers to the process of starting, stopping, joining and leaving online conferences. In CSCW literature conference management is sometimes also referred to as session management. Different types of online conferences are likely to require different conference management styles: aspects such as the duration of the conference, the number of conference participants, and the nature of the conference membership (static or dynamic, open or closed) determine the appropriate conference management style. For instance, an application for large public discussions on the Internet may apply a different model for joining and leaving than an application that supports close collaboration in small groups.

Additionally, groupware service users may need different implementations of conference management to be able to make contact with other people using different communication protocol standards. Current groupware applications use different communication protocol standards to exchange conference management information. Examples of such communication protocol standards are the Internet Engineering Task Force (IETF) Session Initiation Protocol (SIP) (Rosenberg et al., 2002) and the Telecommunication Standardization Sector (ITU-T) H.320 series of standards (IMTC, 2003). Since these standards are not interoperable, groupware service users need to use the same communication protocol standard to be able to co-operate.

4.5 A generic model of groupware services

In abstract terms, a groupware service is defined as:

Definition 12
Groupware service

The whole of behaviour to support a group of co-operating people during a period of interaction. The key aspect of a groupware service is the ability to convey the results of actions by one person to a set of other people.

A groupware service may include behaviour to bring people together for co-operation. A groupware service includes behaviour to start and end the period of interaction and to determine the set of people to interact with. A groupware service may encompass behaviour to define rules that are enacted by the groupware service provider during the co-operation, in order to co-ordinate the co-operation.

Finally, a groupware service also encompasses all behaviour needed to support the dynamics of co-operation, as identified in section 4.4.

At any given moment in the lifecycle of a groupware application, only some groupware behaviour is enabled to be used by the end users. For instance, the behaviour to make use of a communication tool can only be used after the user has entered a conference and that specific tool has been added to the conference.

This section describes a generic process model in AMBER (Eertink et al., 1999) that states the relations between the availability of GSMEs: it states what GSMEs a groupware application offers to the end user at any given stage in its lifecycle. The behaviour associated with the individual GSME types is described in chapter 5. The figures in this section apply the concepts and notation described in chapter 3.

4.5.1 A manual to GSME usage

In our design there exist causal dependencies between GSME types: some GSME types can only be used after other GSMEs have successfully been completed. Section 4.5.2 describes these dependencies. The current section presents some additional generic dependencies that apply to the GSME types in our design.

- GSMEs are provided to groupware service users. In the context of our research these service users are co-operating human end users.
- According to the CooPS groupware reference model, described in the next chapter, GSME types are grouped into GSM types.
- Before any GSME can be used, the associated GSM has to be made available and has to be activated. This relationship is also stated as part of the lifecycle of a GSM, as depicted in *Figure 3-9* on page 45.
- *Figure 4-1*, *Figure 4-2* and *Figure 4-3* on page 72 and further state the causal relations between GSMEs: it defines the order in which GSMEs can be used, related to the lifecycle of a groupware application.
- All relations between GSME types are described with the individual GSME type descriptions. An example of a relation that is not shown in

the figures is the fact that a UseTool GSME can only be used after the corresponding tool, i.e., a communication / collaboration GSM, has been added to a conference using an AddTool GSME.

4.5.2 Relations between GSMEs

Figure 4-1, *Figure 4-2* and *Figure 4-3* depict the relations between GSMEs in terms of the lifecycle of a groupware application: they represent a local perspective, illustrating the order in which GSMEs are available to a specific user. The arrow-shaped symbol in *Figure 4-1* denotes a trigger: it indicates the start of the behaviour. The grey blocks indicate individual GSMEs. The white blocks represent compositions of GSMEs or other behaviour that is described separately. The diamonds and squares indicate splits and joins, as introduced in section 3.2.3.

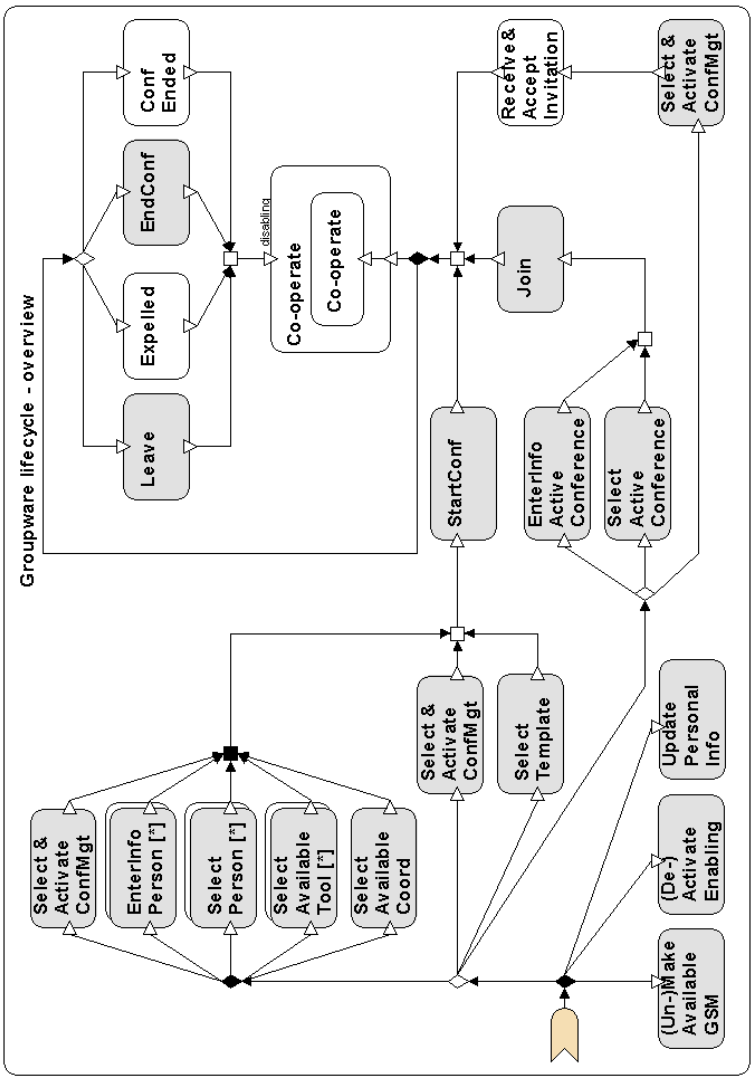


Figure 4-1 Overview of GSMEs and their relations (local perspective)

The connected triangles in *Figure 4-1* indicate causal relations: the completion of the source GSME enables one or more other GSMEs. However, in groupware the *completion* of a GSME can have different meanings: the *local completion*, indicating that the last interaction related to that GSME has taken place for the local groupware service user; the *distributed completion*, indicating that the last interaction related to the GSME has taken place for the other groupware service users; the *global completion*, indicating that both the local and distributed behaviour has been completed. Given the local perspective of the figure, the causal relations denote the local completion of GSMEs.

Although the figure is a local perspective, illustrating the relations between GSME types for a specific user, the figure also has distributed implications: the EndConf GSME type for instance ends a conference. As such, it disables the behaviour to co-operate not just for the user who initiated the EndConf behaviour, but also for the other conference participants.

Figure 4-1 contains four white blocks. In contrast to the other blocks in the figure, these blocks do not represent single GSME types: the *Co-operate* block represents a complex set of GSMEs that is described in *Figure 4-2*. The *Receive & Accept Invitation* block represents the remote behaviour associated with the Invite GSME type; as such, it is addressed together with that behaviour. Similarly, the *Expelled* block and the *ConfEnded* block represents remote behaviour associated with the Expel GSME type and the EndConf GSME type, respectively.

Details about the groupware service lifecycle

The *Co-operate* behaviour defines the GSMEs that are provided to a participant by a groupware application in the course of a conference. The interaction of a user with the groupware application determines which of these GSMEs is actually executed.

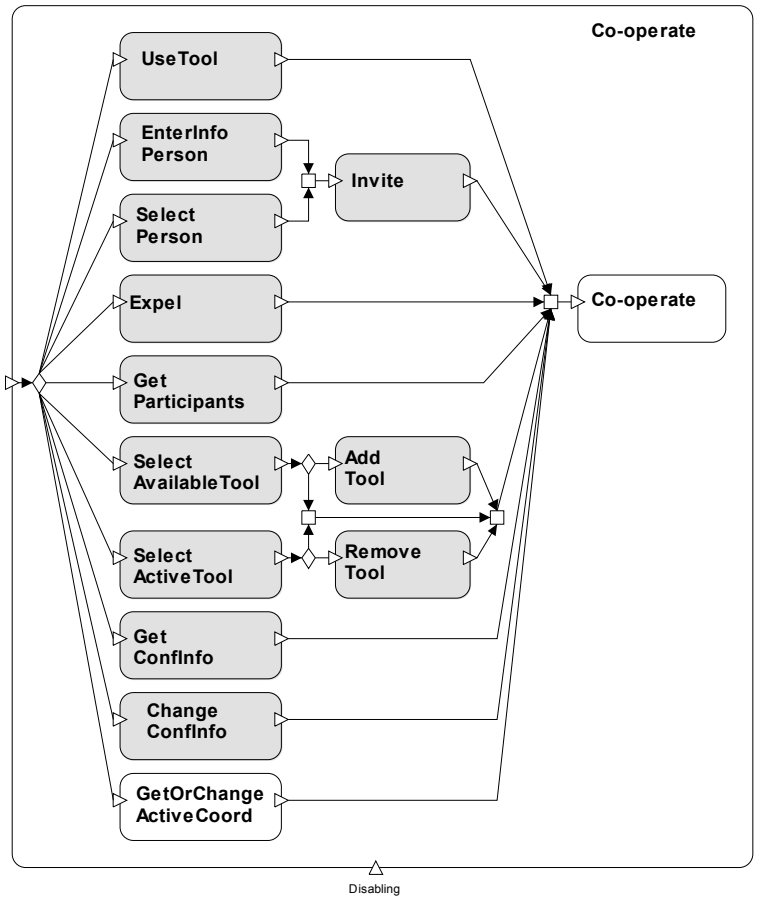


Figure 4-2 Overview of GSMEs: Co-operate behaviour

The *Co-operate* behaviour is available to a groupware participant, until the participant leaves the conference or the conference is ended: this is indicated by the *disabling* triangle at the bottom of the figure. To reduce the complexity of *Figure 4-2*, the details regarding GetOrChangeActiveCoord are provided in a separate figure.

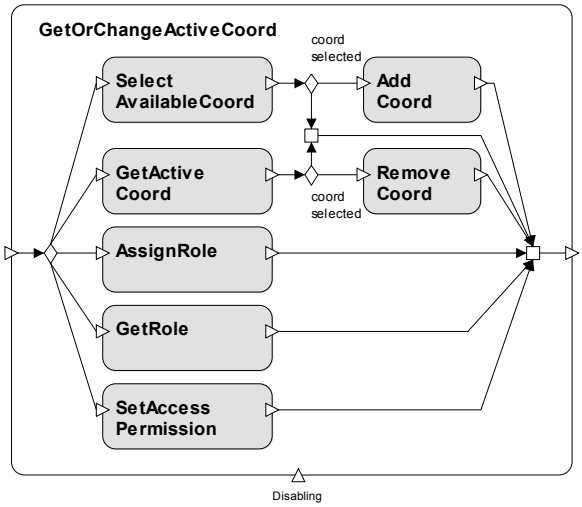


Figure 4-3 Overview of GSMEs: GetOrChangeActive Coord behaviour

The previous figures reveal some of the complexity of the relations between GSMEs. In order to reduce this complexity, GSMEs have been clustered in behaviour descriptions. The following figure illustrates the five clusters of behaviour a groupware service user can access, after starting a groupware application.

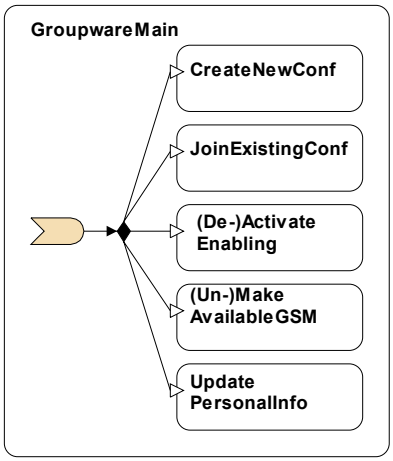


Figure 4-4 Groupware main process

When a person has started a groupware application, she can create a new conference, join an existing conference, activate or de-activate Enabling GSMs (defined in the next chapter), make additional GSMs available for activation (or reverse this process), and update the information regarding herself. This section zooms in on the processes of creating a new conference and joining an existing conference and shows the relations with the various identified GSMEs.

CreateNewConf

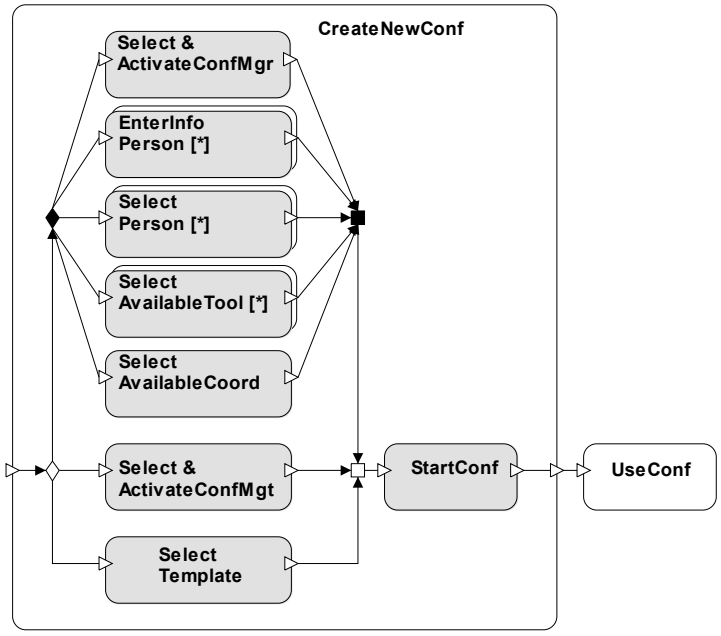


Figure 4-5 Create new conference

Three alternatives exist when creating a new conference:

1. One selects and activates a conference management GSM to be able to configure the conference, one enters or selects zero or more people, zero or more tools (denoted as communication / collaboration GSMs in the following chapters) and zero or one co-ordination GSMs before starting the actual conference. This procedure is for instance applied in Windows Messenger. The details about the Select&ActivateConfMgt, SelectPerson, SelectAvailableTool and SelectAvailableCoord behaviour are described in sections 5.2, 5.6, 5.4 and 5.5, respectively;
2. One selects and activates a conference management GSMs and starts an empty conference and subsequently invites people and adds communication / collaboration GSMs. This procedure is for instance applied in BSCW or upon starting a shared virtual workspace in Groove;
3. One starts a new conference based on a template. Such a template defines a conference management GSM, may optionally define a default set of people to invite, a set of communication / collaboration GSMs to include in the conference and a co-ordination GSM that should be associated with the conference. This procedure is for instance supported by Groove templates. The details about the SelectTemplate behaviour are described in section 5.6.

After a conference has been started, the conference participants can make use of a series of GSMEs, grouped in *Figure 4-5* as the UseConf behaviour. The behaviour associated with UseConf is described later on in this section.

JoinExistingConf

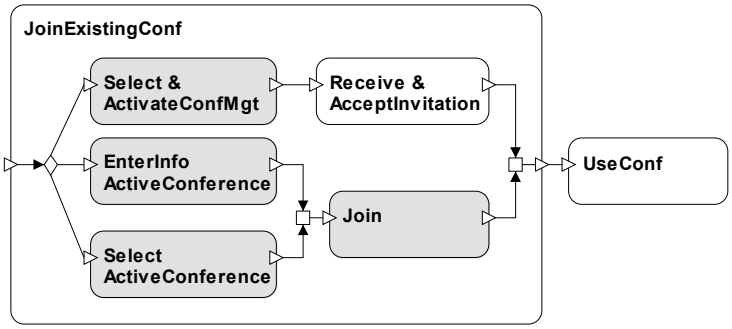


Figure 4-6 Join existing conference

As shown in *Figure 4-6*, existing conferences can be joined by a groupware user in multiple ways: the user either has to receive and accept an invitation, or the user has to enter or select an active conference and request one of the existing participants permission to join that conference. Incoming invitations can only be handled if the appropriate conference management GSM has previously been activated. The behaviour associated with receiving and accepting invitations is described with the Invite GSME.

To join an existing conference, a user either has to enter all required information, or select the active conference from a list. A new person is only included as a participant in an existing conference after an existing participant, who is authorized to do so, has accepted the request to join. Afterwards, the new participant can make use of the GSMEs associated with the conference, within the boundaries specified by the co-ordination policy that may apply.

Many conferences cannot be joined: for instance since their existence is not made public. It is beyond the scope of our design to provide details about the different mechanisms that exist to communicate the existence of a conference.

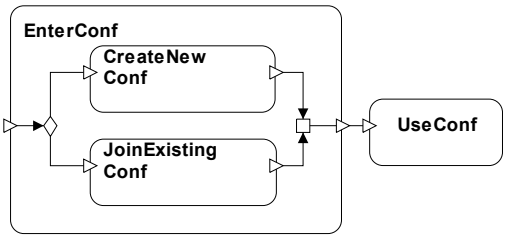
The Join behaviour is also needed to allow latecomers in a conference, and to model public meeting places with dynamic participation.

EnterConf

The EnterConf behaviour is introduced as a shorthand for the various alternatives that exist to enter a conference. As illustrated in *Figure 4-7*, two alternatives exist to enter a conference: starting a new conference using the

CreateNewConf behaviour or joining an existing conference using the JoinExistingConf behaviour.

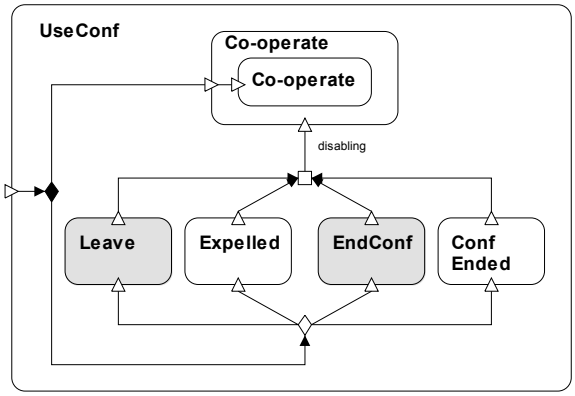
Figure 4-7 EnterConf



UseConf

When a person has entered a conference, using one of the alternatives described in the EnterConf behaviour, the UseConf behaviour defines that a person can use the Co-operate behaviour until the person leaves the conference, is expelled from the conference, or the conference is ended. This is illustrated in *Figure 4-8*.

Figure 4-8 UseConf



When the local user has left a conference, has been expelled from a conference, if the local user has ended a conference, or the conference has been ended by some other user, all groupware behaviour related to that conference is disabled for the local user.

4.5.3 (Un-) MakeAvailableGSM

One of the essential properties of our groupware design is the existence of GSMEs to tailor the provided groupware service. These GSMEs do so by adjusting the composition of Groupware Service Modules (GSMs) that form the groupware service. In order to add a specific GSM to a groupware composition, that GSM has to be activated. Only GSMs that have been

made available can be activated. The behaviour to make a GSM available for activation is depicted in *Figure 4-1* and *Figure 4-4* as (Un-) MakeAvailable GSM. Section 5.7 provides details about this behaviour, and the behaviour to reverse the process.

In our design, a groupware service provider has to be able to establish what GSMs are currently available for activation. For this purpose, the groupware service provider incorporates the *GetAvailableGSMs* behaviour, described in section 5.8.1.

4.5.4 (De-) ActivateEnabling

The behaviour depicted in *Figure 4-1* and *Figure 4-4* as (De-) ActivateEnabling represents the behaviour to activate *enabling GSMs* and to reverse this process. Enabling GSMs are a type of GSMs that help people when starting co-operation by providing information regarding people who can be invited to a conference, or by providing information regarding conferences that can be joined. More information regarding enabling GSMs is provided in sections 6.8 and 6.9.

To establish what GSMs are currently active, the groupware service provider also makes use of the GetActiveGSMs behaviour, as described in section 5.8.2.

Chapter 5

Groupware Service Module Element types

This chapter describes the details of the individual Groupware Service Module Element (GSME) types, as identified in chapter 4. It starts by introducing the GSME categories, and stating how the various identified GSME types have been allocated to these categories. The individual GSME type descriptions consist of:

- A textual description of the GSME types, stating the behaviour it provides to co-operating end users, the behaviour that enables this GSME type (the input relation), and the behaviour that is enabled by this GSME type (the output relation);
- A local service interface description, focussing on the interactions that take place between the groupware service user and the groupware service provider, and the relations between these interactions. This part of the description describes the ordering of interactions, the parameters that are established as a result of these interactions, and parameter value dependencies;
- A description of the groupware service provider behaviour. This description focusses on the information flow between distinct, and possibly geographically distributed, Service Access Points (SAPs) as well as the actions performed by the groupware service provider based on this information.

This chapter includes tailoring GSME types, i.e., GSME types that adjust the composition of Groupware Service Modules (GSMs) that form a groupware service. As a result, the chapter already introduces the various GSM types identified in our design. Chapter 6 describes the details regarding the individual GSM types.

5.1 Overview: GSME categories

Based on the type of support GSME types provide to co-operating users, five GSME categories are distinguished:

1. *Conference management GSMEs*. These GSMEs are primarily aimed at managing the lifecycle of an online conference. This includes, for instance, GSMEs to start and stop a conference, and to obtain information about a conference;
2. *Participation management GSMEs*. These GSMEs are primarily aimed at monitoring and changing the set of conference participants. This category includes, for instance, GSMEs to invite additional people to a conference, or to leave a conference;
3. *Communication and collaboration GSMEs*. These GSMEs are primarily aimed at supporting direct and indirect communication between conference participants. These GSMEs provide the fundamental behaviour of groupware applications: they allow the conference participants to communicate and to access shared information objects. Additionally, this category includes GSMEs to change the set of active communication and collaboration tools;
4. *Co-ordination GSMEs*. These GSMEs are primarily aimed at enacting and changing a role-based co-ordination policy during a conference. This category includes GSMEs to include a co-ordination policy in the conference, to assign roles to participants, to associate rights with a specific role, and to actually enact a co-ordination policy during a conference;
5. *Conference enabling GSMEs*. These GSMEs are primarily aimed at bringing people together for co-operation. This category includes GSMEs to provide information regarding people who can be invited to a conference, and information regarding conferences that can be joined;
6. *Bootstrapping GSMEs*. These GSMEs are aimed at bootstrapping the actual groupware service.

Table 5-1 provides an overview of the GSME categories and the GSME types that have been allocated to them.

Table 5-1 GSME categories and GSME types

GSME category	GSME types
Conference management GSMEs (see section 5.2)	StartConf, EndConf, GetConfInfo, ChangeConfInfo
Participation management GSMEs (see section 5.3)	GetParticipants, Join, Leave, Invite, Expel
Communication and collaboration GSMEs (see section 5.4)	SelectActiveTool, SelectAvailableTool, AddTool, RemoveTool, UseTool
Co-ordination GSMEs (see section 5.5)	GetActiveCoord, SelectAvailableCoord, AddCoord, RemoveCoord, AssignRole, GetRole, SetAccessPermission
Conference enabling GSMEs (see section 5.6)	EnterInfoPerson, SelectPerson, EnterInfoActiveConference, SelectActiveConference
Bootstrapping GSMEs (see section 5.7)	(Un-) MakeAvailableGSM, Select&ActivateConfMgt, (De-) ActivateEnabling, SelectTemplate, UpdatePersonalInfo

Note that this table does not include the behaviour denoted Receive & Accept Invitation, Expelled, and ConfEnded in *Figure 4-1*. This behaviour is not included as a GSME since it indicates the remote, reacting part of the Invite, Expel and EndConf GSME types, respectively.

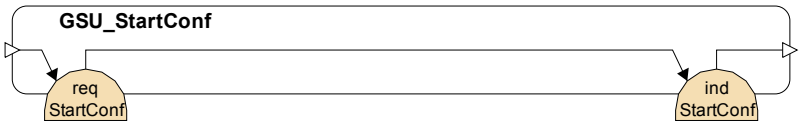
5.2 Conference management GSMEs

The GSMEs categorized as *conference management GSMEs* are primarily aimed at managing the lifecycle of an online conference.

5.2.1 StartConf

This GSME allows end users to create a new online conference. After successful completion of the GSME a new conference exists, that can be applied to co-operate. *Figure 5-1* depicts the behaviour provided by the groupware service user (GSU) who makes use of the StartConf GSME. The figure illustrates that the GSU participates in two related interactions, the reqStartConf interaction and the indStartConf interaction. Moreover, the indStartConf interaction is enabled after the reqStartConf interaction has been completed.

Figure 5-1 StartConf: Local interface description



The following table describes the details regarding the interactions that take place as part of the behaviour. It states the parameters that are established

as part of the interaction, indicated by the ι symbol. In some groupware behaviour descriptions, the λ symbol is applied in order to specify the Service Access Point (SAP) where the interaction takes place. Typically, one needs to specify the SAP if multiple groupware system users interact with the groupware service provider as part of groupware behaviour. The "source" value indicates the source of the established parameter: *GSU* denotes the groupware service user at the specified SAP, *GSP* denotes the groupware service provider, *entry* denotes that the parameter value is passed via the entry of this GSME from the previous GSME.

The *Enabled by* and *Enables* fields indicate by what behaviour this GSME type is enabled and what behaviour it enables. The notation "<behaviour name>" is applied to denote a clustering of behaviour, specified in section 4.5

Table 5-2 StartConf: Details

Interaction	**Established parameters**	**Source**
reqStartConf	ι1: people to invite (optional)	Entry
	ι2: tools to add (optional)	Entry
	ι3: co-ordinator to add (optional)	Entry
indStartConf	ι: conference id	GSP
Source in literature	(Ellis et al., 1991)	
Pattern applied	GW_LocalRequest	
Enabled by	Select&ActivateConfMgt, SelectTemplate, EnterInfoPerson, SelectPerson, SelectAvailableTool, SelectAvailableCoord	
Enables	<UseConf[6]>	

The StartConf GSME is applied to create a new conference, with the groupware service user as the initial conference participant. *Figure 5-2* depicts the behaviour provided by the groupware service provider (GSP) as part of the StartConf GSME. As explained in section 3.2.2, this behaviour specification also includes actions, i.e., internal behaviour, by the GSP as these actions influence the externally observable behaviour of the GSP. *Figure 5-2* illustrates the relations between the various interactions and the actions provided by the GSP.

Parameters may be provided during the reqStartConf interaction, for instance as a result of the template that was chosen. These parameters may specify a set of people to invite, a set of tools to include in the conference, and a co-ordinator to include in the conference. The AddCoord, Invite and AddTool actions in *Figure 5-2* are in fact not internal actions, but shorthand notations for the behaviour associated with these GSME types. The details regarding these GSME types, as well as the specification of the interactions with other users, are provided in sections 5.5, 5.3, and 5.4.

[6] The UseConf behaviour is described in section 4.5.2

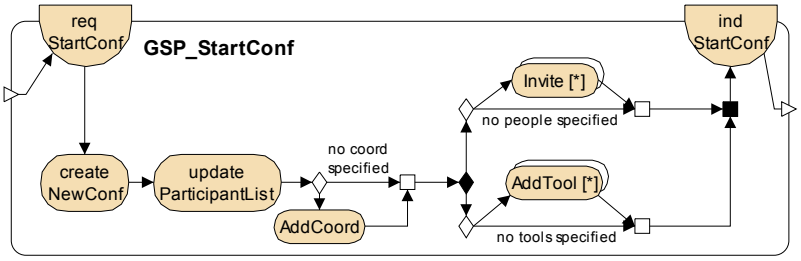


Figure 5-2 StartConf: Groupware service provider behaviour

5.2.2 EndConf

This GSME is the opposite of the StartConf GSME: it allows groupware service users to end the existence of an online conference. Access to this GSME may be limited to participants with specific roles. Successful completion of this GSME disables all GSMEs associated with the specified conference for *all* participants in the conference, as depicted in *Figure 5-3*.

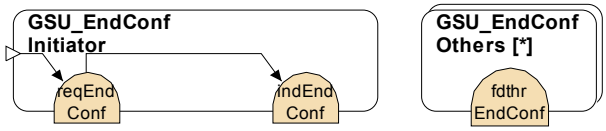


Figure 5-3 EndConf: Local service interface description

Interaction	Established parameters	Source
reqEndConf	ι: conference id	GSP
	| CheckEnable (conference c, user initiator, service EndConf) == true	
indEndConf	ι: notification of the end of the conference	GSP
fdthrEndConf	ι: notification of the end of the conference	GSP
	λ (Service Access Point of interaction)	GSP: participants addresses - address initiator
Source in literature	(Ellis et al., 1991)	
Pattern applied	GW_RemoteInteraction	
Enabled by	<EnterConf>	
Enables	No behaviour: EndConf disables all GSMEs related to the specified conference for all conference participants	

Table 5-3 EndConf: Details

The EndConf GSME first checks whether this user is allowed to end the specified conference. If so, the reqEndConf interaction is enabled. This constraint is indicated by the "|" symbol in *Table 5-3*. Although the various types of Groupware Service Modules (GSMs) are introduced in the next chapter, the details of this GSME already reveal some relationships between GSMs: when a conference is ended, participants can no longer co-operate

using the communication / collaboration GSMs in a conference, so these communication / collaboration GSMs have to be disconnected. Additionally, participants can no longer fulfil specific roles in the conference, so the optional co-ordination GSM also has to be disconnected.

When this process is completed, the conference is terminated: all GSMEs related to the conference are disabled for all conference participants and the conference ceases to exist. The RemoveTool and RemoveCoord actions in *Figure 5-4* are in fact not internal actions, but shorthand notations for the behaviour associated with these GSME types. The details regarding these GSME types are provided in sections 5.4 and 5.5, respectively.

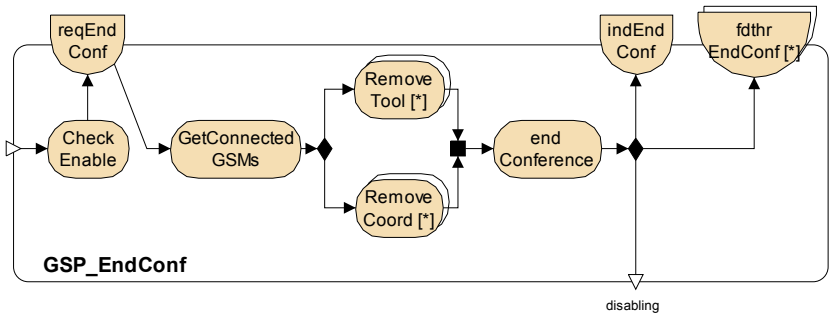


Figure 5-4 EndConf: Groupware service provider behaviour

The set of associated communication / collaboration GSMs and the possible co-ordination GSM associated with the current conference can be obtained using the GetConnectedGSMs behaviour, which is specified in section 5.7.

5.2.3 GetConfInfo

This GSME allows conference participants to view the information that is available about a conference. This information may include information about the purpose of the conference and its organizer. The GetConfInfo GSME and the ChangeConfInfo GSME are grouped in *Figure 5-5* as the GetOrChangeConferenceInfo behaviour.

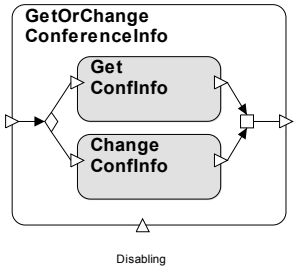


Figure 5-5 GetOr ChangeConferenceInfo behaviour

A conference description has to provide sufficient information to join the conference: it should at least include the service access point (SAP) of one conference participant, typically the organizer, to send the request to join to. After successful completion of the GetConfInfo GSME the end user has been presented with the information regarding the conference. *Figure 5-6*, *Table 5-4*, and *Figure 5-7* provide details of the GetConfInfo GSME.

Figure 5-6 GetConfInfo: Local service interface description

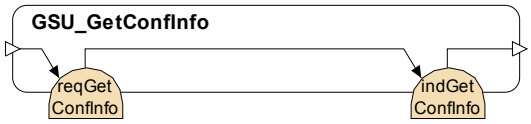
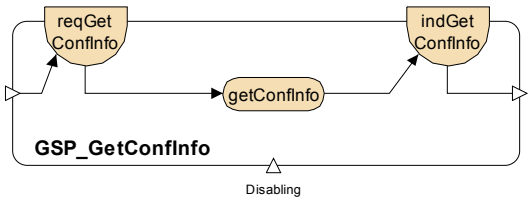


Table 5-4 GetConfInfo: Details

Interaction	Established parameters	Source
reqGetConfInfo	ι: conference id	GSP
indGetConfInfo	ι: conference info	GSP
Source in literature	(Greenberg & Roseman, 1999)	
Pattern applied	GW_LocalRequest	
Enabled by	<EnterConf>	
Enables	<Co-operate>	

The information regarding the conference is stored by the groupware service provider and communicated to the groupware service user as part of the indGetConfInfo interaction.

Figure 5-7 GetConfInfo: Groupware service provider behaviour



5.2.4 ChangeConfInfo

As illustrated in *Figure 5-8*, *Table 5-5* and *Figure 5-9*, this GSME allows end users to adapt (parts of) the information associated with a conference, such as its purpose and organizer. Access to this GSME is typically limited to participants with specific roles. After successful completion of the GSME all conference participants have been notified about the new conference information, and any subsequent uses of the GetConfInfo GSME return the new information associated with the conference.

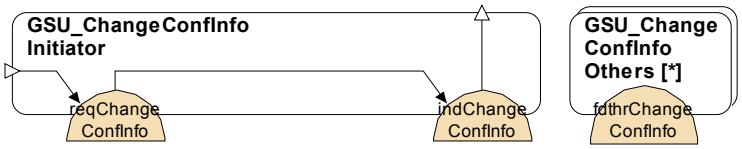


Figure 5-8 ChangeConfInfo: Local service interface description

Table 5-5 ChangeConf Info: Details

Interaction	Established parameters	Source
reqChangeConfInfo	ι: new conference info	GSU
	| CheckEnable (conference c, user initiator, service ChangeConfInfo) == true	
indChangeConfInfo	ι: conference info	reqChangeConfInfo.ι
fdthrChangeConfInfo	ι: conference info	reqChangeConfInfo.ι
	λ	GSP: participants addresses - address initiator
Source in literature	(Greenberg & Roseman, 1999)	
Pattern applied	GW_RemoteInteraction	
Enabled by	<EnterConf>	
Enables	<Co-operate>	

Before the reqChangeConfInfo interaction is enabled, a check is performed to assess whether this groupware service user is currently allowed to make use of the specified GSME in the specified conference.

Based on the new value of the conference information, established during the reqChangeConfinfo interaction, the conference information is updated. The new conference information is also propagated to the other participants.

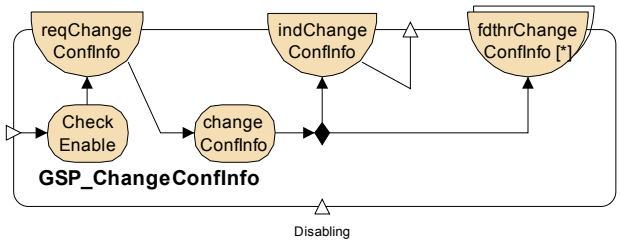


Figure 5-9 ChangeConfInfo: Groupware service provider behaviour

5.3 Participation management GSMEs

The participation management GSMEs in our design are primarily aimed at monitoring and changing conference participation. The GSMEs to inspect and change the set of participants during a conference have been grouped

into the InviteOrExpelParticipants behaviour. This behaviour is illustrated in *Figure 5-10*.

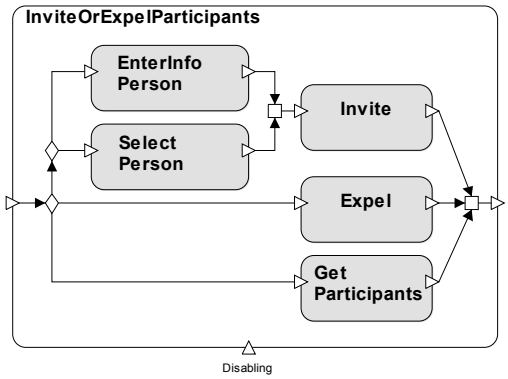


Figure 5-10 InviteOr ExpelParticipants behaviour

This behaviour shows that during a conference participants can invite additional people, expel people from a conference, or obtain a list of current conference participants.

The two other participation management GSMEs discussed in this section are the Join and Leave GSMEs. The EnterInfoPerson GSME and SelectPerson GSME are enabling GSMEs, discussed in section 5.6.

5.3.1 GetParticipants

As illustrated in *Figure 5-11*, *Table 5-6*and *Figure 5-12*, this GSME allows end users to obtain a list of participants in the current conference. A groupware user needs to participate in the conference to access this GSME. After successful completion of this GSME the user has been presented the list of current conference participants.

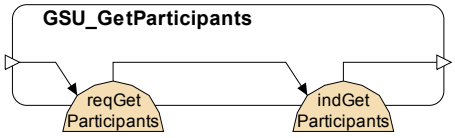
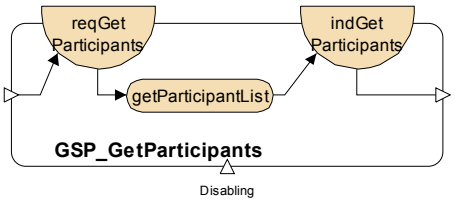


Figure 5-11 Get Participants: Local service interface description

Table 5-6 Get Participants: Details

Interaction	Established parameters	Source
reqGetParticipants	ι: conference id	GSP
indGetParticipants	ι: info about the conference participants	GSP
Source in literature	(Greenberg & Roseman, 1999; Kausar & Crowcroft, 1999).	
Pattern applied	GW_LocalRequest	
Enabled by	<EnterConf>	
Enables	<Co-operate>	

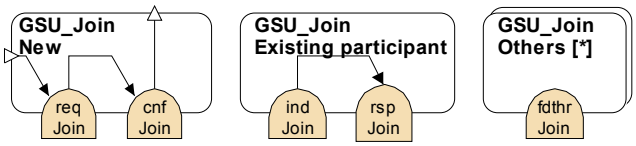
Figure 5-12 Get Participants: Groupware service provider behaviour



5.3.2 Join

This GSME, illustrated in *Figure 5-13*, *Table 5-7* and *Figure 5-14*, allows groupware service users to join an existing conference. A user can only join a conference that has been created previously using the StartConf GSME. A user cannot join a conference in which he already participates. Access to this GSME can be limited to specific users, or groups of users. After successful completion of this GSME the joining groupware service user is either accepted as a new participant in the conference, or has received a rejection. After joining a conference, the groupware service user can access all GSMEs related to the conference, within the boundaries specified by any co-ordination policy that may apply.

Figure 5-13 Join: Local service interface description



After information regarding the conference to join has been collected using the SelectActiveConference GSME, the actual behaviour to join the conference can take place. This involves sending a request to join the conference to an existing conference participant and receiving a reply.

Table 5-7 Join: Details

Interaction	Established parameters	Source
reqJoin	ι1: info about the new participant	GSU
	ι2: address of an existing participant	Entry
	ι3: conference id	GSP
cnfJoin	ι1: <accept, reject, not authorized>	rspJoin.ι
	ι2: conference state {participants, conference info, active co-ordination GSM + state, active communication / collaboration GSMs + state} | cnfJoin.ι1 == accept	GSP
indJoin	ι: info about the new participant	reqJoin.ι1
	λ	reqJoin.ι2
	| CheckEnable (conference c, user existing, service Join) == true	
rspJoin	ι: <accept, reject>	GSU
fdthrJoin	ι: info about the new participant	reqJoin.ι1
	λ	GSP: participants addresses - address new GSU - address existing GSU
Source in literature	(ter Hofte, 1998), (Greenberg & Roseman, 1999; Kausar & Crowcroft, 1999)	
Pattern applied	User-confirmed message transfer (extended)	
Enabled by	EnterInfoActiveConference, SelectActiveConference	
Enables	<UseConf>	

Before the new person is able to send a request to join a given conference, an appropriate conference management GSM has to be activated. The details about conference management GSMs and why different conferences may require different conference management GSMs are provided in section 6.5.

Before an existing conference participant can accept or reject the request to join, a check is performed to see whether that specific participant is allowed to accept or reject the request. When the existing participant is authorized to do so, the indJoin interaction is enabled, otherwise the new person is notified.

When the existing conference participant rejects the request to join the conference, the invitor is notified about the rejection.

When the existing conference participant accepts the request, the new person is included in the list of conference participants and receives a notification of acceptance as well as an updated regarding the current state of the conference. This state information encompasses information about the current set of conference participants, conference information, the set

of communication / collaboration GSMs that are active in the conference and their state, as well as the state of the co-ordination GSM, if one is active in the conference.

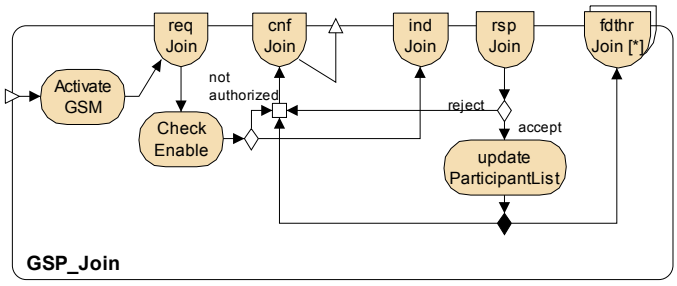


Figure 5-14 Join: Groupware service provider behaviour

5.3.3 Leave

This GSME, illustrated in *Figure 5-15*, *Table 5-8* and *Figure 5-16*, allows groupware service users to leave a specified ongoing conference. After successful completion of this GSME the groupware service user is no longer participating in the specified conference and can, as a result, no longer access any groupware GSMEs related to that conference. Note that a person may concurrently participate in multiple conferences

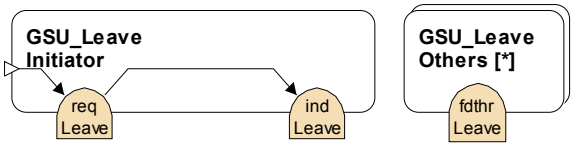
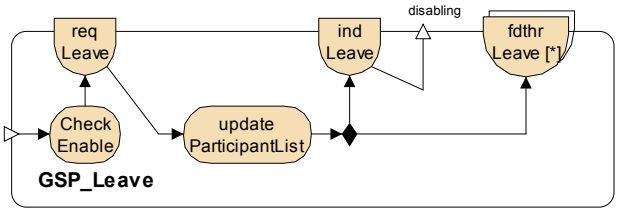


Figure 5-15 Leave: Local service interface description

Table 5-8 Leave: Details

Interaction	Established parameters	Source
reqLeave	ι: conference id	GSP
	| CheckEnable (conference c, user invitor, service Leave) == true	
indLeave	ι: notification of leaving the conference	GSP
fdthrLeave	ι: the new set of conference participants	GSP
	λ	GSP: participants addresses - address initiator
Source in literature	(ter Hofte, 1998; Greenberg & Roseman, 1999; Kausar & Crowcroft, 1999)	
Pattern applied	GW_RemoteInteraction	
Enabled by	<EnterConf>	
Enables	No behaviour is enabled; all GSMEs associated with the specified conference are disabled for the initiator of the Leave GSME.	

Figure 5-16 Leave: Groupware service provider behaviour



Typically, a groupware service user is allowed to leave a conference at any time. As a result of the reqLeave interaction the initiator of the leave GSME is removed from the list of participants of the specified conference. The fact that the person has left the conference is also conveyed to the remaining conference participants.

As a result of the Leave GSME, all GSMEs related to the specified conference are disabled for the person who left the conference.

5.3.4 Invite

The Invite GSME, illustrated in *Figure 5-17*, *Table 5-9* and *Figure 5-18*, allows current participants in a conference to invite other people to an existing conference. To access the GSME, the invitor needs to be a participant in the conference he invites the invitee to. Upon accepting the invitation, the invitee is added to the set of conference participants. As a result, the invited user can access the GSMEs related to the conference, within the boundaries specified by any co-ordination policy that may apply.

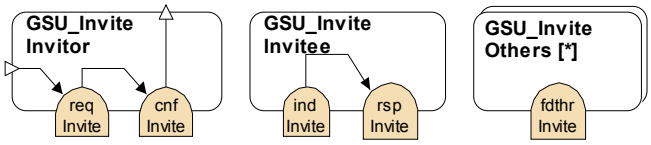


Figure 5-17 Invite: Local service interface description

Table 5-9 Invite: Details

Interaction	Established parameters	Source
reqInvite	ι1: address invitee	Entry
	ι2: conference info	GSP
	| CheckEnable (conference c, user invitor, service Invite) == true	
cnfInvite	ι1: <accept, reject>	rspInvite.ι1
	ι2: info about the invitee | rspInvite.ι1 == true	rspInvite.ι2
indInvite	ι: conference info	reqInvite.ι2
	λ	reqInvite.ι1
rspInvite	ι1: <accept, reject>	GSU
	ι2: info about the invitee	GSU
	ι3: conference state {participants, conference info, active co-ordination GSM + state, active communication / collaboration GSMs + state} | rspInvite.ι1 == accept	GSP
fdthrInvite	ι: the new set of conference participants	GSP
	λ	GSP: participants addresses - address invitor - address invitee
	| rspInvite.ι1 == accept	
Source in literature	(ter Hofte, 1998), (Greenberg & Roseman, 1999), (Kausar & Crowcroft, 1999) and (Edwards, 1994)	
Pattern applied	GW_RemoteInteraction	
Enabled by	EnterInfoPerson, SelectPerson	
Enables	<Co-operate>	

Based on the information obtained using the SelectPerson GSME a specific person can be invited to join the conference. Before the reqInvite interaction is enabled a check is performed to assess whether the invitor is currently allowed to invite other people to the specified conference.

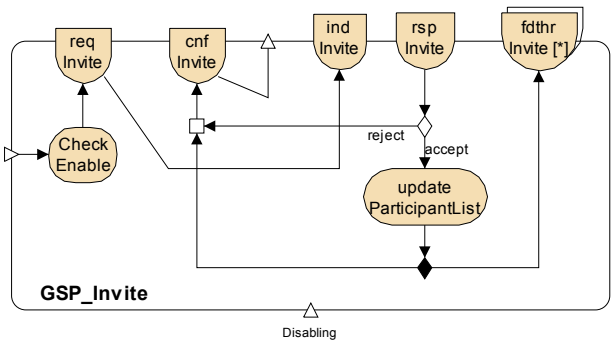


Figure 5-18 Invite: Groupware service provider behaviour

The relation between the behaviour to receive and accept invitations and the other groupware GSMEs has been depicted in the groupware GSME overview in *Figure 4-1*.

When the invitee rejects the invitation, the invitor is notified. If the invitee accepts the invitation, the list of conference participants is updated. Subsequently, the invitor is notified about the acceptance, the other conference participants are notified about the new participant, and the new participant is updated regarding the state of the conference.

5.3.5 Expel

This GSME, illustrated in *Figure 5-19*, *Table 5-10* and *Figure 5-20*, allows end users to remove another participant from a conference. Both the expelling person and the expelled person have to be participants in the conference. Access to this GSME can be limited to participants with specific roles. After successful completion, the expelled person can no longer access any GSME related to the specified conference.

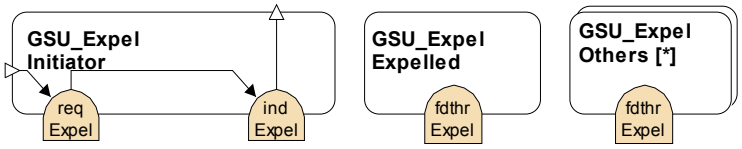


Figure 5-19 Expel: Local service interface description

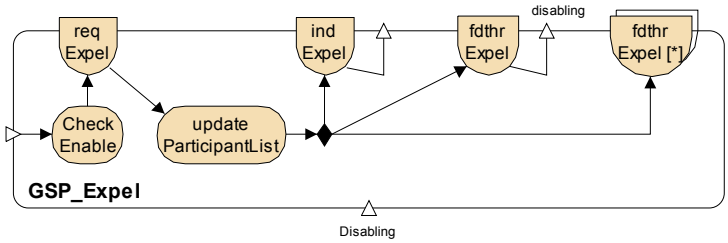
Table 5-10 Expel: Details

Interaction	Established parameters	Source
reqExpel	ι1: address expelled participant	GSP: the set of conference participants GSU: the selection of one participant
	ι2: conference id	GSP
	| CheckEnable (conference c, user initiator, service Expel) == true	
indExpel	ι: the new set of conference participants	GSP
fdthrExpel	ι: notification of being expelled	GSP
	λ	reqExpel.ι1
fdthrExpel	ι: the new set of conference participants	GSP
	λ	GSP: participants addresses - address initiator - address expelled
Source in literature	Ter Hofte (1998) and Kausar & Crowcroft (1999)	
Pattern applied	GW_RemoteInteraction	
Enabled by	<EnterConf>	
Enables	<Co-operate> for the GSME initiator Disables all GSMEs related to the specified conference for the expelled participant	

After a check has been performed to assess whether the initiator is currently allowed to expel people from the specified conference the reqExpel interaction is enabled. During this interaction the initiator selects a person from the set of current conference participants. Subsequently, this person is removed from the set of conference participants and a notification is sent to both the expelled person and the remaining conference participants.

As a consequence of being expelled, the expelled person can no longer access any GSME associated with the specified conference. *Figure 4-1* shows this relation between the behaviour of being expelled and the other GSMEs.

Figure 5-20 Expel: Groupware service provider behaviour



5.4 Communication and collaboration GSMEs

The communication and collaboration GSMEs are primarily aimed at supporting direct and indirect communication between conference participants. Indirect communication, typically denoted as *collaboration*, takes place via shared information objects: the various participants in a conference share access to the state of a shared information object. They can collaborate by changing and observing changes to the state of this object. Examples of such collaboration tools include application sharing, collaborative document editors, and collaborative web browsers.

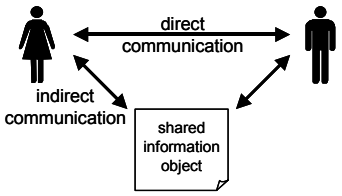


Figure 5-21 Direct versus indirect communication between participants

The distinction between direct and indirect communication, illustrated in *Figure 5-21*, is not very strict: some tools such as e-mail and text-based chat tools have some properties of a shared information object, even though they are designed for direct communication between participants. Instead of focussing on a precise definition of the boundary, we aim to design groupware that can handle direct, indirect and intermediate forms of communication. Apart from the communication and collaboration services, a groupware application also provides the GetOrChangeActiveTools behaviour. This behaviour can be specified as follows.

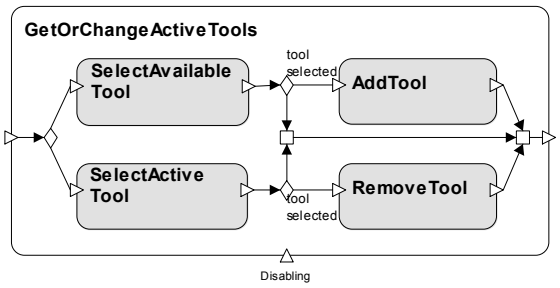


Figure 5-22 GetOr ChangeActiveTools behaviour

The SelectAvailableTool GSME presents the groupware service user the communication / collaboration GSMs (CC-GSMs) that are currently available for inclusion in the conference. The SelectAvailableTool GSME may also be applied to obtain the list of available CC-GSMs, without actually including any new tools in the conference.

The SelectActiveTool GSME presents the groupware service user the tools that are currently active in the conference. The groupware service user can make a selection from these CC-GSMs and remove them from the conference. The SelectActiveTool GSME may also be applied to obtain the list of active CC-GSMs, without actually removing any from the conference.

5.4.1 SelectActiveTool

This GSME, illustrated in *Figure 5-23*, *Table 5-11* and *Figure 5-24*, allows groupware service users to obtain a list of CC-GSMs that are currently active in the specified conference: a CC-GSM is active in a given conference if it is connected to the CM-GSM associated with that conference. A user needs to participate in the conference to access this GSME. After successful completion the user has been presented a list of active conference CC-GSMs. This GSME is practical when combined with the RemoveTool GSME.

Figure 5-23 Select ActiveTool: Local service interface description

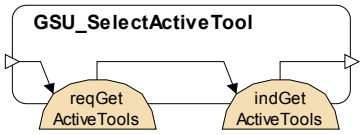


Table 5-11 SelectActive Tool: Details

Interaction	Established parameters	Source
reqGetActiveTools	ι: conference id	GSP
indGetActiveTools	ι: zero or one selected active communication / collaboration GSMs	GSP: the set of active communication / collaboration GSMs GSU: selection of zero or one communication / collaboration GSM
Source in literature	No source in literature	
Pattern applied	GW_LocalRequest	
Enabled by	<EnterConf>	
Enables	RemoveTool, <Co-operate>	

The groupware service provider is responsible for maintaining an overview of the communication / collaboration GSMs that are currently active in a conference. If the groupware service user has selected an active tool during the indGetActiveTools interaction, the result of that interaction includes sufficient information to remove the tool using the RemoveTool GSME.

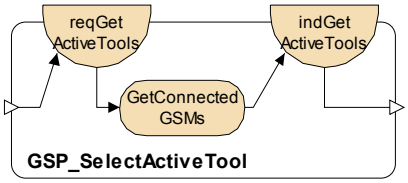


Figure 5-24 SelectActiveTool: Groupware service provider behaviour

5.4.2 SelectAvailableTool

This GSME, illustrated in *Figure 5-25*, *Table 5-12* and *Figure 5-26*, allows groupware service users to obtain a list of tools, i.e., CC-GSMs, that are available to be added to the conference. Based on this list the user may select a specific tool to add to the composition of active GSMs. A CC-GSM is available for selection and composition if it has been made available and is not yet connected to a CM-GSM: the CC-GSM may already have been activated. After successful completion the groupware service user has been presented a list of available CC-GSMs. This GSME type is practical when combined with the AddTool GSME.

When the groupware service provider has access to additional information regarding the preferences of the groupware service user, these preferences may used to propose a default CC-GSM during the indGetAvailTools interaction. A groupware service user may store such preferences using the UpdatePersonalInfo GSME.

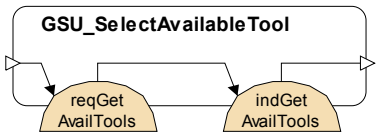


Figure 5-25 Select AvailableTool: Local service interface description

Table 5-12 Select AvailableTool: Details

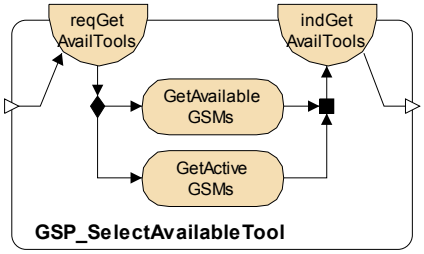
Interaction	Established parameters	Source
reqGetAvailTools		
indGetAvailTools	ι: zero or one selected communication / collaboration GSMs that are available	GSP: the set of available and activated communication / collaboration GSMs GSU: the selection of zero or one of these tools
Source in literature	No source in literature	
Pattern applied	GW_LocalRequest	
Enabled by	<process start[7]>, <EnterConf>	
Enables	StartConf, AddTool, <Co-operate>	

The groupware service provider is able to determine what GSMs are available via two sources: it combines the list of available, not yet activated, GSMs and the list of activated GSMs. The results are filtered by specifying the type of the GSM: in this case CC-GSMs. Both sources are needed as users may wish to include an already activated tool in a conference as well as a tool that has not been activated yet.

An example of the first case is a person who wishes to share a design she is currently working on, and the application to edit the design, with a group of other people. In that case, the application to edit the design has already been activated, and can be included in the groupware composition.

When a user selects a tool that has not been activated yet, it needs to be activated before it can be included in the groupware composition. For this reason the list of available CC-GSMs presented during the indGetAvailTools interaction includes both available and active tools. Note that in the groupware reference model presented in chapter 6, an active tool can only be associated with one conference at a time.

Figure 5-26 SelectAvailableTool: Groupware service provider behaviour



[7] The process start is indicated by the arrow-shaped symbol in *Figure 4-1*.

5.4.3 AddTool

As illustrated in *Figure 5-27*, *Table 5-13* and *Figure 5-28*, this GSME allows groupware service users to add a communication / collaboration GSM to the composition of GSMs that determine the provided groupware service. As a result of this GSME, the groupware service is extended with the GSMEs provided by a communication tool or a tool to access shared information objects. To make use of this GSME, the groupware service user has to be a participant in the conference. The AddTool specification assumes that the newly added tool is available for all participants in the conference. Access to this GSME can be limited to participants with specific roles. After successful completion all participants in the conference can access the behaviour provided by the newly added tool, within the boundaries specified by the co-ordination policy that may apply. The AddTool GSME is required given the dynamics in co-operative settings, as described in sections 4.4.1 and 4.4.2.

Figure 5-27 AddTool: Local service interface description

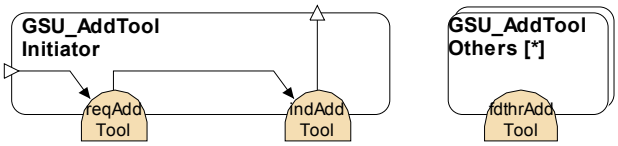


Table 5-13 AddTool: Details

Interaction	Established parameters	Source
reqAddTool	ι1: communication / collaboration GSM to add to the conference	Entry
	ι2: conference id	GSP
	| CheckEnable (conference c, user initiator, service AddTool) == true	
indAddTool	ι: the new set of active communication / collaboration GSMs	GSP
fdthrAddTool	ι: the new set of active communication / collaboration GSMs	GSP
	λ	GSP: participants addresses - address initiator
Source in literature	(ter Hofte, 1998)	
Pattern applied	GW_RemoteInteraction	
Enabled by	SelectAvailableTool	
Enables	<Co-operate> Enables the UseTool GSMEs provided by the specified communication / collaboration GSM	

Before a user can request to add a specific tool to a specific conference, a check is performed to assess whether that user is currently allowed to do so.

If the user selected an activated tool, that tool is included in the current GSM composition associated with this conference. Otherwise, the tool is first activated before it is connected. When the tool has been included in the conference, the other conference participants are notified about the new situation.

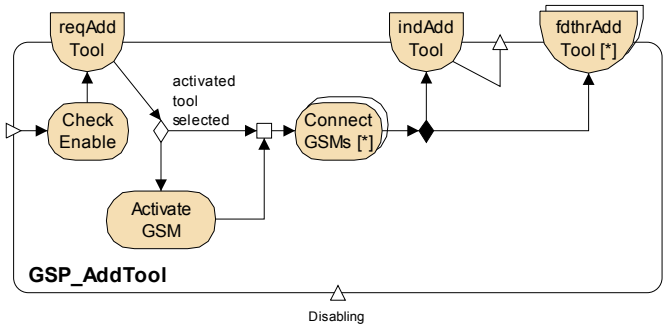


Figure 5-28 AddTool: Groupware service provider behaviour

5.4.4 RemoveTool

As illustrated in *Figure 5-29*, *Table 5-14* and *Figure 5-30*, this GSME allows groupware service users to remove a communication / collaboration GSM from the composition of GSMs associated with a conference. As a result, the GSMEs provided by the removed tool are no longer available to the participants in the conference. The service initiator has to be a participant in the conference to make use of this GSME. The tool to remove has to be active in the conference at the start of the operation. Access to this GSME can be limited to participants with specific roles.

After successful completion of the RemoveTool GSME, the specified tool is no longer active in the specified conference. However, participants may remain using the behaviour provided by the tool in single-user mode, as may be desirable for stateful tools such as a shared editor. The *RemoveTool GSME* forms the counterpart of the AddTool service.

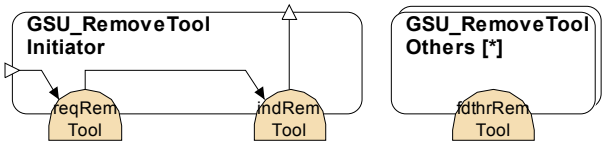


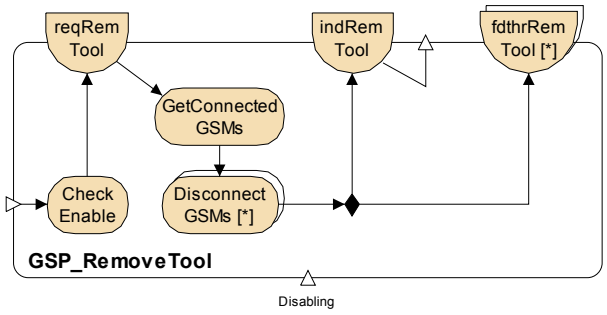
Figure 5-29 RemoveTool: Local service interface description

Table 5-14
RemoveTool: Details

Interaction	Established parameters	Source
reqRemTool	ι1: communication / collaboration GSM to remove from the conference	Entry
	ι2: conference id	GSP
	| CheckEnable (conference c, user initiator, service RemoveTool) == true	
indRemTool	ι: the new set of active communication / collaboration GSMs	GSP
fdthrRemTool	ι: the new set of active communication / collaboration GSMs	GSP
	λ	GSP: participants addresses - address initiator
Source in literature	(ter Hofte, 1998)	
Pattern applied	GW_RemoteInteraction	
Enabled by	SelectActiveTool	
Enables	<Co-operate> Disables the UseTool GSMEs provided by the specified communication / collaboration GSM	

The reqRemTool interaction is enabled after a check has been performed to assess whether the initiator is currently allowed to use this GSME in the specified conference. After the reqRemTool interaction, the groupware service provider looks up to what other GSMs the selected tool is connected and disconnects the tool.

Figure 5-30
RemoveTool: Groupware service provider behaviour



5.4.5 UseTool

This GSME allows groupware service users to actually co-operate using a communication / collaboration GSM. This GSME is a generic behaviour description: it does not provide details of specific tools. As such, the UseTool GSME description can be applied both for stateful and stateless communication / collaboration GSMs. To make use of the GSME, the

groupware service user has to be a participant in the conference and the associated tool has to be active in the conference. Access to this GSME can be subject to a co-ordination policy. After successful completion, the groupware service user has received feedback about the results of the action she performed on the tool. The other participants in the conference may have received feedthrough about the results of the performed action as well, depending on the settings of the tool.

Figure 5-31 UseTool: Local service interface description

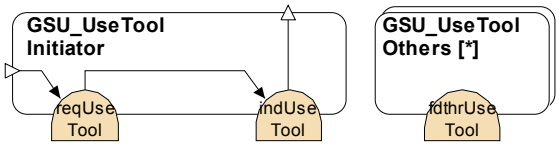


Table 5-15 UseTool: Details

Interaction	Established parameters	Source
reqUseTool	ι1: activated communication / collaboration GSM to use	GSP: the active communication / collaboration GSMs GSU: the selection of one active tool
	ι2: operation to perform on the communication / collaboration GSM	GSP: the available operations on the selected tool GSU: the selection of an operation to perform
	ι3: conference id	GSP
	| CheckEnable (conference c, user initiator, service UseTool, tool t) == true	
indUseTool	ι: the new state of the communication / collaboration GSM	GSP
fdthrUseTool	ι: the new state of the communication / collaboration GSM	GSP
	λ	GSP: participants addresses - address initiator
Source in literature	(Ellis et al., 1991; ter Hofte, 1998; Baker et al., 2002)	
Pattern applied	GW_RemoteInteraction	
Enabled by	AddTool: a communication / collaboration GSM has to be added to a conference, before the UseTool GSMEs it consists of can be used by the conference participants	
Enables	<Co-operate>	

Many aspects of this GSME type are specific for individual tools. In any case, before a tool can be used in a conference, it has to be added to that conference using the AddTool GSME.

Additionally, before a user can make use of the reqUseTool interaction, a check may be performed to assess whether the specified user is currently allowed to perform the specified action on the tool in the current conference. If the user is authorized, the action on the tool is performed and the initiator receives feedback based on the result of this action. In case of a stateful tool, the user is typically notified about the new state of the tool.

The other conference participants may receive feedthrough based on the result of the action. In case of a stateful tool, other participants are typically notified about the new state of the tool. In a stateless tool, such as a tool for audio conferencing, actions on the tool are directly conveyed to other participants. The next section provides more details about the co-ordination of tool usage.

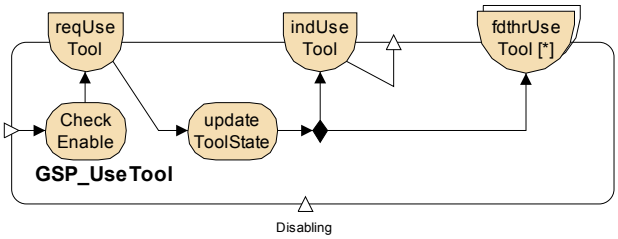


Figure 5-32 UseTool: Groupware service provider behaviour

5.5 Co-ordination GSMEs

Co-ordination GSMEs are primarily aimed at enacting and adapting a role-based co-ordination policy during a conference. In our design, co-ordination GSMEs are provided by co-ordination GSMs, which are described in detain in the next chapter. As specified in section 3.6.3, a co-ordination policy defines the roles users may occupy in a conference and associated access rights. Co-ordination GSMEs are optional: a groupware service does not have to include these GSMEs.

In section 4.5 the various co-ordination GSMEs have been grouped in the behaviour GetOrChangeActiveCoord. The following figure depicts how this behaviour is specified.

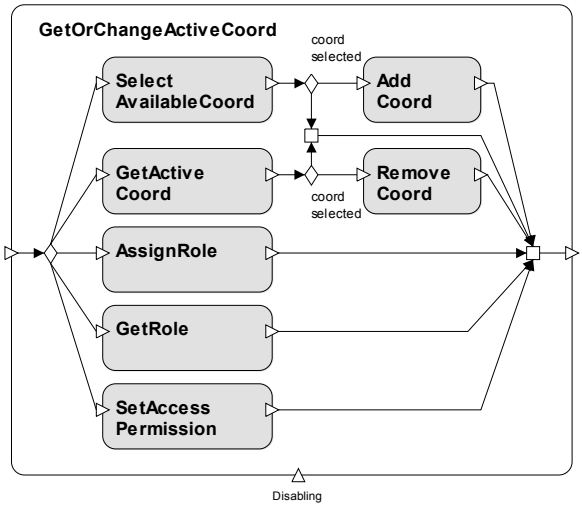


Figure 5-33 GetOr ChangeActiveCoord behaviour

The individual co-ordination GSMEs are defined in the sections below.

5.5.1 GetActiveCoord

This GSME, illustrated in *Figure 5-34*, *Table 5-16* and *Figure 5-35*, allows groupware service users to obtain a reference to the co-ordination GSM (CO-GSM) that is currently active in the conference, if one is active. The resulting reference can, for instance, be applied to remove the CO-GSM using the RemoveCoord GSME. A CO-GSM is active in a given conference if it is connected to the CM-GSM associated with that conference.

The groupware service user needs to participate in the conference to access this GSME. After successful completion, the initiator has been presented a reference to the currently active CO-GSM or has received a notification that no CO-GSM is currently active in the conference.

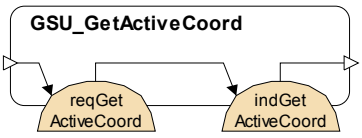


Figure 5-34 GetActive Coord: Local service interface description

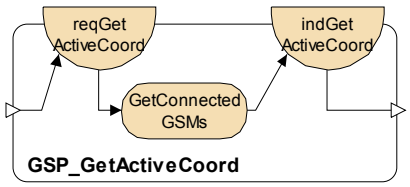
Table 5-16 GetActive Coord: Details

Interaction	Established parameters	Source
reqGetActiveCoord	ι: conference id	GSP
indGetActiveCoord	ι: zero or one selected co-ordination GSMs that are active in the conference	GSP: the active co-ordination GSMs GSU: whether or not the co-ordination GSM is selected
Source in literature	(ter Hofte, 1998; Baker et al., 2002)	
Pattern applied	GW_LocalRequest	
Enabled by	<EnterConf>	
Enables	RemoveCoord, <Co-operate>	

The groupware service provider is responsible for maintaining information regarding the CO-GSMs that are associated with a conference. One is able to obtain the list of GSMs in a groupware composition using the GetConnectedGSMs behaviour; each conference is associated with one conference management GSM, as is described in detail in the next chapter. As a result, one can obtain the set of GSMs associated with a conference by specifying that conference management GSM as a parameter of the GetConnectedGSMs behaviour. The GetConnectedGSMs behaviour is described in section 5.7.

If the groupware service user has selected a specific co-ordination GSM as part of the indGetActiveCoord interaction, the result of that interaction should include sufficient information to remove the specified CO-GSM using the RemoveCoord GSME.

Figure 5-35 GetActive Coord: Groupware service provider behaviour



5.5.2 SelectAvailableCoord

This GSME allows groupware service users to obtain a list of currently available co-ordination GSMs (CO-GSMs), in order to select one and connect it with the GSMs associated with the conference . As depicted in *Figure 3-9* on page 45, a CO-GSM can be connected with other GSMs after it has been made available and has been activated. The availability of this GSME is practical in conjunction with the AddCoord GSME.

When the groupware service provider has access to additional information regarding the preferences of the groupware service user, these

preferences may be used to propose a default co-ordination GSM during the indGetAvailCoords interaction. A user may store such preferences using the UpdatePersonalInfo GSME. The following figures and table specify the details of this GSME.

Figure 5-36 Select AvailableCoord: Local service interface description

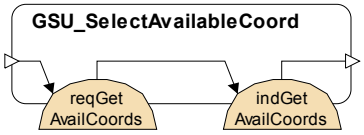
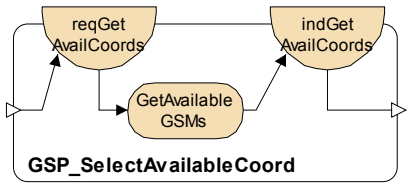


Table 5-17 Select AvailableCoord: Details

Interaction	Established parameters	Source
reqGetAvailCoord		
indGetAvailCoord	ι: zero or one selected co-ordination GSMs	GSP: the set of available co-ordination GSM classes GSU: the selection of zero or one co-ordination GSM
Source in literature	(ter Hofte, 1998; Baker et al., 2002)	
Pattern applied	GW_LocalRequest	
Enabled by	<process start>, <EnterConf>	
Enables	StartConf, AddCoord, <Co-operate>	

The groupware service provider is able to determine what GSMs are available using the GetAvailableGSMs behaviour. The list of results can be filtered by specifying the GSM type: in this case co-ordination GSMs. The resulting list of available co-ordination GSMs is presented to the groupware service user as part of the indGetAvailCoords interaction.

Figure 5-37 Select AvailableCoord: Groupware service provider behaviour



If the groupware service user selects a co-ordination GSM as part of the indGetAvailCoords interaction, that GSM is passed on as a result of the interaction. This result includes sufficient information to add the selected co-ordination GSM using the AddCoord GSME.

5.5.3 AddCoord

This GSME, illustrated in the following figures and table, allows groupware service users to add a co-ordination GSM to the composition of GSMs that determine the provided groupware service. As a result of this GSME, the groupware service is extended with the GSMEs provided by a co-ordination GSM. To make use of the AddCoord GSME, the groupware service user has to be a participant in the conference. In our design, a groupware composition includes zero or one active co-ordination GSMs. Repetitive use of the AddCoord GSME replaces the active co-ordination GSM.

The AddCoord specification assumes that the added co-ordination GSM is available for all participants in the conference. Access to this GSME can be limited to participants with specific roles. After successful completion all participants in the conference can access the GSMEs provided by the newly added co-ordination GSM, within the boundaries specified by the co-ordination policy. This GSME is required given the dynamics in co-operative settings, as described in sections 4.4.1 and 4.4.2.

Figure 5-38 AddCoord: Local service interface description

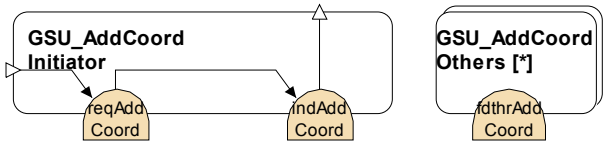


Table 5-18 AddCoord: Details

Interaction	Established parameters	Source
reqAddCoord	ι1: co-ordination GSM to add to the conference	Entry
	ι2: conference id	GSP
	| CheckEnable (conference c, user initiator, service AddCoord) == true	
indAddCoord	ι: the new active co-ordination GSM	GSP
fdthrAddCoord	ι: the new active co-ordination GSM	GSP
	λ	GSP: participants addresses - address initiator
Source in literature	(ter Hofte, 1998; Baker et al., 2002)	
Pattern applied	GW_RemoteInteraction	
Enabled by	SelectAvailableCoord	
Enables	<Co-operate> Enables the GetRole, AssignRole and SetAccessPermission GSME for the specified co-ordination GSM	

Before a user can request to add a specific co-ordination GSM to a specific conference, a check may be performed to assess whether the specified user is currently allowed to do so.

Based on the co-ordination GSM the user selected during the SelectAvailableCoord GSME, a co-ordination GSM is activated and connected to the current GSM composition associated with the conference. When the co-ordinator has been added to the conference, the other conference participants are notified about the addition.

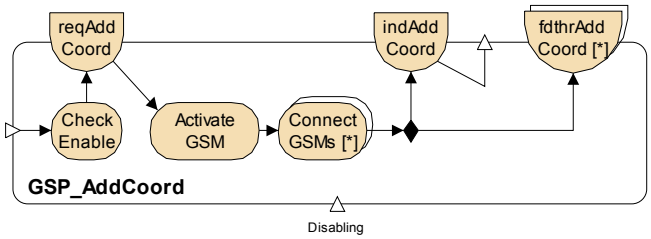


Figure 5-39 AddCoord: Groupware service provider behaviour

5.5.4 RemoveCoord

This GSME, illustrated in the following figures and table, allows groupware service users to remove a co-ordination GSM from the composition of GSMs associated with a conference. As a result, the behaviour provided by the removed co-ordination GSM is no longer available to the participants in the conference: it does no longer enact a co-ordination policy during the conference. The service initiator has to be a participant in the conference to make use of this GSME. The removed co-ordination GSM has to be active in the conference: it has to be added previously using the AddCoord GSME. Access to the RemoveCoord can be limited to participants with specific roles.

After successful completion of the RemoveCoord GSME, the specified co-ordination GSM is no longer active in the specified conference. The RemoveCoord GSME forms the counterpart of the AddCoord service.

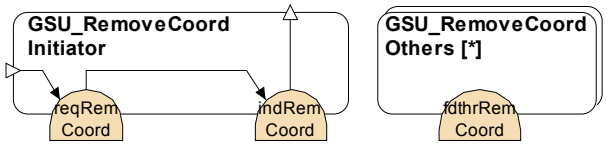


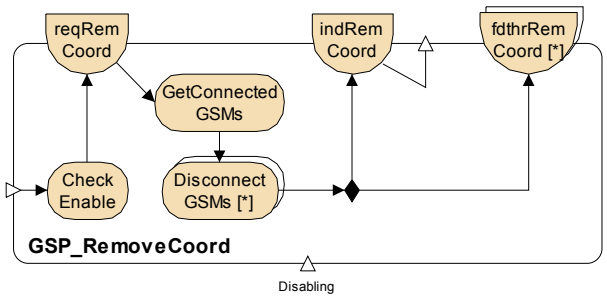
Figure 5-40 RemoveCoord: Local service interface description

Table 5-19 RemoveCoord: Details

Interaction	Established parameters	Source
reqRemCoord	ι1: co-ordination GSM to remove from the conference	Entry
	ι2: conference id	GSP
	| CheckEnable (conference c, user initiator, service RemoveCoord) == true	
indRemCoord	ι: notification that no co-ordination GSM is active in the conference	GSP
fdthrRemCoord	ι: notification that no co-ordination GSM is active in the conference	GSP
	λ	GSP: participants addresses - address initiator
Source in literature	(ter Hofte, 1998; Baker et al., 2002)	
Pattern applied	GW_RemoteInteraction	
Enabled by	SelectActiveCoord	
Enables	<Co-operate> Disables the GetRole, AssignRole and SetAccessPermission GSME for the specified co-ordination GSM	

The reqRemCoord interaction is enabled after a check has been performed to assess whether the initiator is currently allowed to use this GSME in the specified conference. After the reqRemCoord interaction, the groupware service provider looks up to what other GSMs the selected co-ordination GSM is connected and disconnects the co-ordination GSM. After being disconnected from a conference, an active co-ordination GSM deactivates itself.

Figure 5-41 Remove Coord: Groupware service provider behaviour



5.5.5 AssignRole

This GSME, illustrated in the following figures and table, allows groupware service users to change the role of any participant in the conference. The initiator and the person whose role is changed have to be participants in the conference, and a co-ordination GSM has to be active in the conference.

The available roles are defined by the active co-ordination GSM. Access to the AssignRole GSME can be limited to participants with specific roles. After successful completion the specified participant has a new role, and any subsequent groupware service requests are handled accordingly. Additionally, any subsequent invocations of the GetRole GSME return the new role for the specified participant.

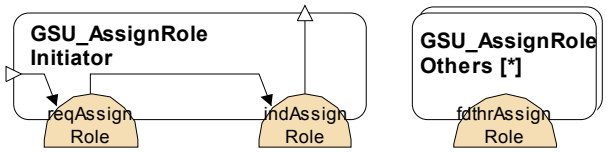


Figure 5-42 AssignRole: Local service interface description

Table 5-20 AssignRole: Details

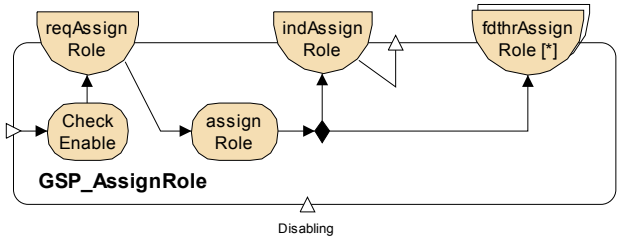
Interaction	Established parameters	Source
reqAssignRole	ι1: participant to change the role of	GSP: set of conference participants GSU: selection of one participant
	ι2: new role	GSP: set of available roles GSU: selection of one role
	ι3: conference id	GSP
	| CheckEnable (conference c, user initiator, service AssignRole) == true	
indAssignRole	ι1: the participant with a new role	GSP
	ι2: the new role	GSP
fdthrAssignRole	ι1: the participant with a new role	GSP
	ι2: the new role	GSP
	λ	GSP: participants addresses - address initiator
Source in literature	(Ellis et al., 1991; Baker et al., 2002; Dewan, 2001; Kausar & Crowcroft, 1999)	
Pattern applied	GW_RemoteInteraction	
Enabled by	<EnterConf> and AddCoord	
Enables	<Co-operate>	

Before the reqAssignRole interaction is enabled, the active co-ordination policy is applied to check whether the initiator is currently allowed to assign roles to conference participants.

As part of the reqAssignRole interaction, the groupware service provider presents the groupware service user with the set of conference participants and the set of roles a participant can have in the conference. The active co-

ordination GSM determines the set of available roles. As part of the interaction, the groupware user requests to assign one of these roles to a specified participant. After a role has been assigned, the other conference participants are notified about this assignment.

Figure 5-43 AssignRole: Groupware service provider behaviour



5.5.6 GetRole

This GSME allows groupware service users to retrieve the role of any conference participant. To access this GSME the groupware service user needs to be a participant in the conference and a co-ordination GSM needs to be active in the conference. After successful completion the user has been presented the role of the specified participant.

Figure 5-44 GetRole: Local service interface description

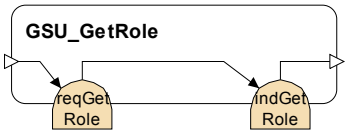


Table 5-21 GetRole details

Interaction	Established parameters	Source
reqGetRole	ι1: participant to obtain the role of	GSP: set of conference participants GSU: selection of one participant
	ι2: conference id	GSP
indGetRole	ι: the role of the specified participant	GSP
Source in literature	No source in literature	
Pattern applied	GW_LocalRequest	
Enabled by	<EnterConf> and AddCoord	
Enables	<Co-operate>	

The groupware service provider is responsible for maintaining the associations between actual conference participants and user roles.

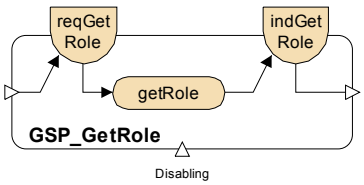


Figure 5-45 GetRole: Groupware service provider behaviour

5.5.7 SetAccessPermission

This GSME, illustrated in *Figure 5-46*, *Table 5-22* and *Figure 5-47*, allows groupware service users to associate user roles and GSMEs. A role-based co-ordination policy specifies what role a groupware service user should have to be allowed to access specific GSMEs. Access to the SetAccessPermission GSME is typically limited to participants with specific roles: if a co-ordination policy is active the right to set access right is typically reserved to a specific group of participants. After successful completion, the new access permissions are in place, regulating future requests to access specific groupware services in the conference.

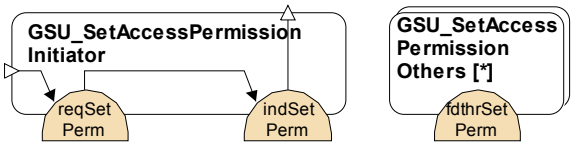


Figure 5-46 SetAccess Permission: Local service interface description

Table 5-22 SetAccess Permission details

Interaction	Established parameters	Source
reqSetPerm	ι1: type of GSME to set the access permission of	GSP: possible GSMEs GSU: selection of one GSME
	ι2: affected role	GSP: set of available roles GSU: selection of one role
	ι3: <allowed, not allowed>	GSU
	ι4: conference id	GSP
	| CheckEnable (conference c, user initiator, service SetAccessPermission) == true	
indSetPerm	ι1: type of GSME to set the access permission of	reqSetPerm.ι1
	ι2: affected role	reqSetPerm.ι2
	ι3: <enabled, disabled>	reqSetPerm.ι3
	ι4: conference id	reqSetPerm.ι4
fdthrSetPerm	ι1: type of GSME to set the access permission of	reqSetPerm.ι1
	ι2: affected role	reqSetPerm.ι2
	ι3: <enabled, disabled>	reqSetPerm.ι3
	ι4: conference id	reqSetPerm.ι4
	λ	GSP: participants addresses - address initiator
Source in literature	Ellis, Gibbs, et al. (1991), Dewan (2001), and Kausar & Crowcroft (1999)	
Pattern applied	GW_RemoteInteraction	
Enabled by	<EnterConf> and AddCoord	
Enables	<Co-operate>	

As part of the reqSetPerm interaction, the co-ordinator presents the groupware user with the various user roles, and the GSMEs it can control access to. The groupware service user can now select the GSME to set the access permission of, the role concerned, and whether or not a participant with the specified role is allowed to perform that GSME.

Co-ordination policies may control access to all GSMEs that include the CheckEnable behaviour. In our design, the following GSMEs can be subject to co-ordination: EndConf, ChangeConfInfo, Join, Leave, Invite, Expel, AddTool, RemoveTool, UseTool (see below), AddCoord, RemoveCoord, AssignRole, SetAccessPermission.

The UseTool GSME is a special case; an active communication / collaboration GSM may provide different sub-types of the UseTool GSME.

A shared editor may, for instance, provide behaviour to create, read, update and delete shared objects. In that case it may be required to assign different access permissions for these different types of operations. The section in this chapter, entitled *Performing access control on tool usage* elaborates on this aspect.

Note that a check to assess whether the initiator is allowed to change access permissions is performed *before* the new values are set. When new access permissions are set, the other conference participants are notified as well.

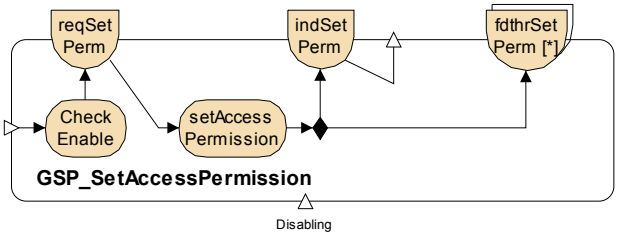


Figure 5-47 SetAccess Permissions: Groupware service provider behaviour

5.5.8 Supporting behaviour: CheckEnable

The CheckEnable behaviour is provided to check whether a given GSME should currently be enabled for a given user in a given conference. This check takes place before the user can actually request to use the GSME. This supporting behaviour helps to create groupware applications that offer groupware service users only those services they are actually allowed to use.

As described in section 3.6.3, the CheckEnable behaviour is generic behaviour provided by the groupware service provider as part of the behaviour associated with GSMEs. As such, the CheckEnable behaviour is not regarded a GSME. However, we consider the CheckEnable behaviour to be a relevant part of the groupware service for co-operating end users.

5.5.9 Performing access control on tool usage

In our design, a communication / collaboration GSM (CC-GSM) provides variations of the UseTool GSME to allow people to communicate or collaborate using shared information objects. These variations are specific for a given CC-GSM: a CC-GSM may, for instance, provide behaviour to create, read, update and delete shared information objects.

However, these variations of the UseTool GSME may have different access permissions associated with them. Given the example of the create, read, update and delete behaviour, the groupware service users may wish to specify that delete operations can only be performed by conference

participants with a specific role. In that case different access control rules are required for the different variations of the UseTool service.

On the other hand, the roles participants are allowed to occupy depend on the active co-ordination policy: different co-ordination GSMs may define different roles to choose from.

This means that in a conference the combination of the active co-ordination GSM and CC-GSMs determines what access control options can be set. To obtain this information a co-ordination GSM may for instance query CC-GSMs regarding the subtypes of the UseTool GSME they distinguish. Our model does not specify implementation details regarding the manner in which GSMs exchange such information.

Based on the UseTool subtypes a CC-GSM provides the co-ordination GSM allows groupware service users to specify the relevant access rights for these subtypes as well. The right to set access rights is also co-ordinated by the active co-ordination GSM.

Although the access control options depend on the active co-ordination GSM and the set of active CC-GSMs, the associated access rights can be stored and reused. If certain compositions of a co-ordination GSM and CC-GSMs are frequently used, the composition can be stored in a template, together with the associated access rights. Such templates facilitate the use and tailoring of groupware applications.

Consider a template for a medical peer-review session that includes a co-ordination GSM that distinguishes the roles of *treating physician* and *consulted physician*. Similarly, the template specifies that the groupware composition includes an audio conferencing CC-GSM and a CC-GSM to share treatment plans. The latter CC-GSM distinguishes the *view treatment plan* and *adapt treatment plan* GSMEs, both variations of the UseTool GSME. The template may now also specify that only participants with the role of *treating doctor* are allowed to perform the *adapt treatment plan* GSME.

5.6 Enabling GSMEs

The enabling GSMEs are primarily aimed at bringing people together for co-operation, for instance by showing what people are available for communication, and what conferences can be joined.

5.6.1 EnterInfoPerson

The EnterInfoPerson GSME allows groupware service users to enter all information about a person needed to invite that person to the current conference. After successful completion of this behaviour, the groupware

service provider has sufficient information regarding the invitee to invite that person to a given conference.

The EnterInfoPerson GSME can be applied when starting a new conference or during a conference, to provide input for the Invite GSME. The details of this GSME type are provided in the following figures and table.

Figure 5-48 EnterInfo Person: Local service interface description

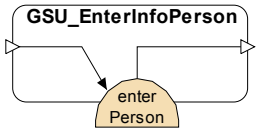
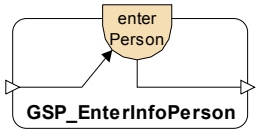


Table 5-23 EnterInfo Person: Details

Interaction	Established parameters	Source
enterPerson	ι: address of the person to invite	GSU
Source in literature	No source in literature	
Pattern applied	No pattern applied	
Enabled by	<process start>, <EnterConf>	
Enables	StartConf, Invite	

In this GSME the groupware service user enters sufficient information to invite a given person to a conference: it has to be possible to derive the Service Access Point of the person to invite from the provided information.

Figure 5-49 EnterInfo Person: Groupware service provider behaviour



5.6.2 SelectPerson

The SelectPerson GSME allows groupware service users to select a person to invite to a conference from a list of available people. After successful completion of this behaviour, the groupware service provider has sufficient information regarding the invitee to invite that person to a given conference.

The SelectPerson GSME can be applied when starting a new conference or during a conference, to provide input for the Invite GSME.

When the groupware service provider has access to additional information regarding the people who can be invited, this information may be presented to the groupware service user, as part of the reqSelectPerson interaction. Such additional information may include the name of the person, her presence status, and any other information that person has

stored using the UpdatePersonalInfo GSME. The details of this GSME type are provided in the following figures and table.

Figure 5-50 Select Person: Local service interface description

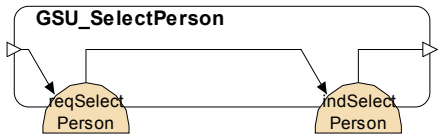
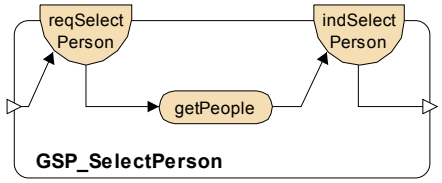


Table 5-24 Select Person: Details

Interaction	Established parameters	Source
reqSelectPerson		
indSelectPerson	ι: address of the person to invite	GSP: list of people GSU: selection from list
Source in literature	(Baker et al., 2002; Edwards, 1994)	
Pattern applied	GW_LocalRequest	
Enabled by	<process start>, <EnterConf>	
Enables	StartConf, Invite	

The groupware service provider presents the groupware service user with a list of people. When the groupware service user selects a person from this list, the selection has to provide sufficient information to invite that person to a conference: it has to be possible to derive the Service Access Point of the person to invite from the provided information.

Figure 5-51 Select Person: Groupware service provider behaviour



5.6.3 EnterInfoActiveConference

The EnterInfoActiveConference GSME allows groupware service users to provide information regarding an ongoing conference to join. After successful completion of this behaviour, the initiator has provided the groupware service provider sufficient information regarding the conference to send a request to join that conference. The EnterInfoActiveConference GSME is applied to provide input for the Join GSME.

Figure 5-52 EnterInfo ActiveConference: Local service interface description

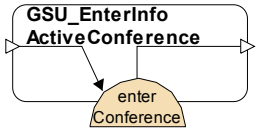
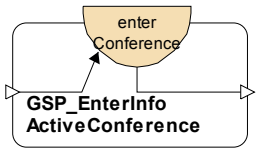


Table 5-25 EnterInfo ActiveConference: Details

Interaction	Established parameters	Source
enterConference	ι1: address of a person to send the request to join the conference to	GSU
	ι2: type of conference management GSM to use during the conference	GSU
Source in literature	No source in literature	
Pattern applied	No pattern applied	
Enabled by	<process start>	
Enables	Join	

In this GSME the groupware service user enters sufficient information to derive the Service Access Point of one of the existing conference participants. Additionally, the information has to be sufficient to activate an appropriate conference management GSM during the Join GSME, as different conferences may require different conference management GSMs.

Figure 5-53 EnterInfo ActiveConference: Groupware service provider behaviour



5.6.4 SelectActiveConference

The SelectActiveConference GSME allows groupware service users to specify an ongoing conference to join, typically by selecting a conference from a list of active conferences. After successful completion of this behaviour, the initiator has provided the groupware service provider with sufficient information regarding the conference to send a request to join that conference. The SelectActiveConference GSME is applied to provide input for the Join GSME. The details of this GSME type are provided in the following figures and table.

Figure 5-54 Select ActiveConference: Local service interface description

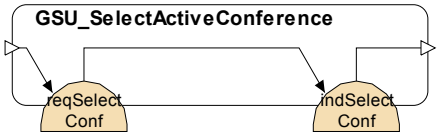
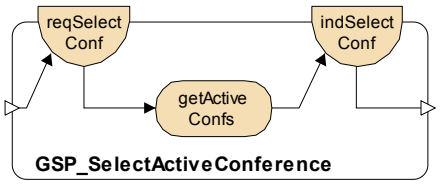


Table 5-26 Select ActiveConference: Details

Interaction	Established parameters	Source
reqSelectConf		
indSelectConf	ι1: address of a person to send the request to join the conference to	GSP: list of active conferences GSU: selection from list
	ι2: type of conference management GSM to use during the conference	GSP & GSU (see above)
Source in literature	(Edwards, 1994)	
Pattern applied	GW_LocalRequest	
Enabled by	<process start>	
Enables	Join	

As part of the indSelectConf interaction, the groupware service provider presents the groupware service user a list of active conferences. Based on the selection by the groupware service user, the groupware service provider obtains sufficient information to send a request to join the selected conference to one of the existing conference participants: it has to be possible to derive the Service Access Point of one of the existing conference participants from the provided information. Additionally, the list has to provide sufficient information to activate an appropriate conference management GSM during the Join GSME, as different conferences may require different conference management GSMs.

Figure 5-55 Select ActiveConference: Groupware service provider behaviour



5.7 Bootstrapping GSMEs

The GSME types described in this section are at the borderline of a groupware application: they provide behaviour to bootstrap groupware services. Using these GSME types groupware service users can make GSMs available for activation, and activate the GSMs needed to start a conference.

5.7.1 (Un-)MakeAvailableGSM

Before groupware service users can make use of the services provided by a GSM or include that GSM in a groupware composition, the GSM has to be made available and activated, as depicted in *Figure 3-9*. The

MakeAvailableGSM GSME allows groupware service users to make a GSM available for activation. As described in section 2, making a GSM available can take many forms: from installing a software component at a user's computer to making an online service available for a group of users. It is beyond the scope of this research to list all such mechanisms and their details. Independent of the manner in which a GSM has been made available, the behaviour it provides remains the same.

The UnMakeAvailableGSM GSME is the counterpart of the MakeAvailableGSM GSME: it allows groupware service users to make a GSM unavailable for activation. Details of both are provided below.

Figure 5-56 (Un-) Make AvailableGSM: Local service interface description

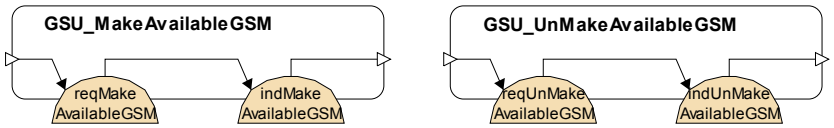
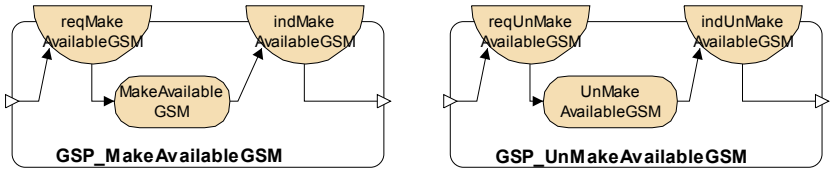


Table 5-27 (Un-) Make AvailableGSM: Details

Interaction	Established parameters	Source
reqMakeAvailableGSM / reqUnMakeAvailableGSM	ι: GSM to make (un-) available	GSP
indMakeAvailableGSM / indUnMakeAvailableGSM	ι: success or failure	GSU
Source in literature	No source in literature	
Pattern applied	GW_LocalRequest	
Enabled by	<process start>	
Enables	Making a GSM available enables its activation. Making a GSM unavailable disables its activation.	

Groupware users may at any time make additional GSMs available for activation. Similarly, groupware service users may at any time make GSMs unavailable for activation. The groupware service provider keeps track of all available GSMs.

Figure 5-57 (Un-) Make AvailableGSM: Groupware service provider behaviour



5.7.2 Select&ActivateConfMgt

The Select&ActivateConfMgt GSME allows a groupware user to select and activate a specific conference management GSM. As will be described in detail in the next chapter, such a GSM provides the groupware behaviour to

manage online conferences and their participation. Different conference management GSMs may apply different interaction styles to interact with the end user and apply different communication protocol standards for their internal communication. As a result, one may wish to use different conference management GSMs for different occasions.

The Select&ActivateConfMgt GSME presents the groupware service user with a list of available conference management GSMs. Based on the user's selection from this list, one specific conference management GSM is activated in order to start a new conference using this conference manager.

When the groupware service provider has access to additional information regarding the preferences of the groupware service user, these preferences may used to propose a default conference management GSM during the indGetAvailConfMgrs interaction. A user may store such preferences using the UpdatePersonalInfo GSME. Details of this GSME type are provided in the following figures and table.

Figure 5-58 Select& ActivateConfMgt: Local service interface description

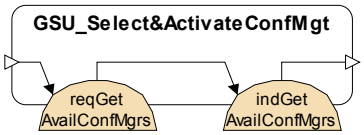


Table 5-28 Select& ActivateConfMgt: Details

Interaction	Established parameters	Source
reqGetAvailConfMgrs		
indGetAvailConfMgrs	ι: selected conference management GSM | selected conference management GSM ∈ available conference management GSMs	GSP: available conference management GSMs GSU: selection of one of these conference management GSMs
Source in literature	No source in literature	
Pattern applied	GW_LocalRequest	
Enabled by	<process start>	
Enables	StartConf, Join, <Receive&AcceptInvitation[8]>	

The groupware infrastructure is able to determine what conference management GSMs are available using the GetAvailableGSMs behaviour, with the GSM type specified as a parameter. As part of the indGetAvailConfMgrs interaction, the groupware service provider presents the groupware service user with the list of available conference management GSMs. Based on the selection by the groupware service user from this list, one conference management GSM is activated.

[8] Receive&AcceptInvitation is sub-behaviour of the Invite GSME

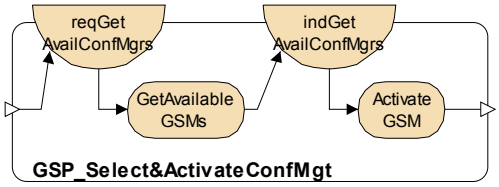


Figure 5-59 Select& ActivateConfMgt: Groupware service provider behaviour

The GetAvailableGSMs and ActivateGSM behaviour are described in section 5.8.

5.7.3 (De-)ActivateEnabling

The ActivateEnabling GSME allows groupware service users to activate an available People Listing GSM or Conference Listing GSM. These Enabling GSMs, which are described in detail in the next chapter, provide behaviour to facilitate the process of inviting additional people to a conference or joining other conferences. Before a user can make use of the behaviour provided by an Enabling GSM, that GSM has to be activated.

The DeActivateEnabling GSME is the counterpart of the ActivateEnabling GSME: it allows groupware service users to deactivate a previously activated Enabling GSM. After deactivation, an Enabling GSM no longer provides any GSMEs to the groupware service user.

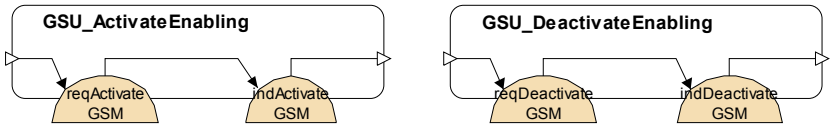


Figure 5-60 (De-) ActivateEnabling: Local service interface description

Table 5-29 (De-) ActivateEnabling: Details

Interaction	Established parameters	Source
reqActivateGSM / reqDeactivateGSM	ι: PLIST-GSM or CLIST-GSM to (de-) activate	GSU
indActivateGSM / indDeactivateGSM	ι: success or failure	GSP
Source in literature	No source in literature	
Pattern applied	GW_LocalRequest	
Enabled by	<process start>	
Enables	(De-) Activating a people listing GSM influences the SelectPerson GSME (De-) Activating a conference listing GSM influences the SelectActiveConference GSME	

Groupware users may at any time, also before any conference has been started, activate and deactivate Enabling GSMs. The groupware service provider keeps track of all activated GSMs. Note that the generic behaviour

to activate and deactivate any type of GSMs is also applied for Enabling GSMs.

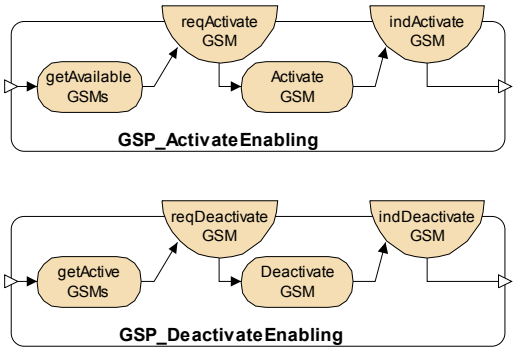


Figure 5-61 (De-) ActivateEnabling: Groupware service provider behaviour

5.7.4 SelectTemplate

This GSME allows groupware service users to select a template as a basis for a new conference. Groupware templates, in our research also denoted as *groupware patches*, are named after the task they are designed to support, and describe a selection of GSMs. These GSMs provide the set of GSMEs typically needed for the specified task. Groupware templates form valid starting points for groupware use and tailoring: they describe a valid composition of GSMs, according to the rules described in chapter 6.

Groupware templates may include a set of people to invite to the conference, a set of communication / collaboration GSMs that should be included in the conference, as well as a co-ordination GSM that should be included. In any case, the template has to state what conference management GSM is associated with the conference, as this GSM needs to be activated before the StartConf GSME can be used. Details of this GSME type are provided in the following figures and table.

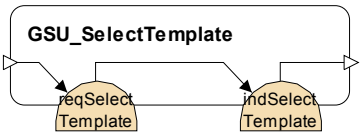


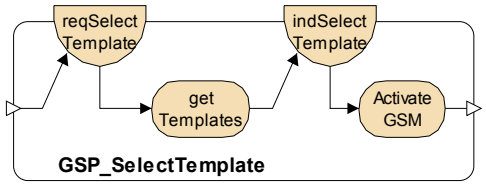
Figure 5-62 Select Template: Local service interface description

Table 5-30 Select Template: Details

Interaction	Established parameters	Source
reqSelectTemplate		
indSelectTemplate	ι1: people to invite (optional)	GSP: list of available templates GSU: selection from list
	ι2: communication / collaboration GSMs to add (optional)	GSP (see above)
	ι3: co-ordination GSM to add (optional)	GSP (see above)
Source in literature	No source in literature	
Pattern applied	GW_LocalRequest with extensions	
Enabled by	<process start>	
Enables	StartConf	

Using the SelectTemplate GSME the groupware service user can select a previously stored groupware template. Our design does not specify how groupware templates are to be stored.

Figure 5-63 Select Template: Groupware service provider behaviour



5.7.5 UpdatePersonalInfo

This GSME allows groupware service users to store and update information regarding themselves in the groupware service provider. This information may, for instance, encompass their user name, preferences and presence status. Although this behaviour is at the borderline of the definition of a groupware service, it is included in our design since the results of such an update may optionally be communicated to other groupware service users. Since this behaviour is available outside the scope of a conference, the set of people to distribute the update to should be specified as a parameter of the reqUpdatePersInfo interaction. Details of this GSME type are provided in the following figures and table.

Figure 5-64 Update PersonalInfo: Local service interface description

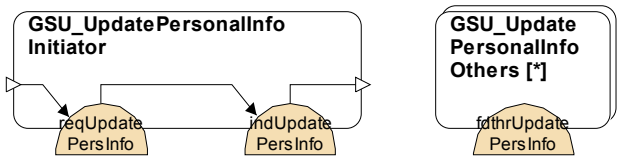


Table 5-31 Update PersonalInfo: Details

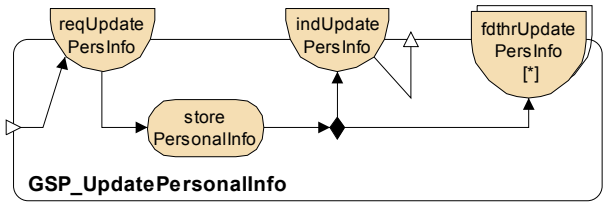
Interaction	Established parameters	Source
reqUpdatePersInfo	ι1: name of entry to update	GSU
	ι2: new value of the entry	GSU
	ι3: addresses of people to inform	GSU
indUpdatePersInfo	ι1: name of updated entry	reqUpdatePersInfo. ι1
	ι2: new value of the updated entry	reqUpdatePersInfo. ι2
fdthrUpdatePersInfo	ι1: initiator	GSP
	ι2: name of updated entry	reqUpdatePersInfo. ι1
	ι3: new value of the updated entry	reqUpdatePersInfo. ι2
	λ	reqUpdatePersInfo. ι3
Source in literature	No source in literature	
Pattern applied	GW_RemoteInteraction	
Enabled by	<process start>	
Enables	<no behaviour>	

Using the UpdatePersonalInfo GSME the groupware service user can update the information that is stored about them by the groupware service provider. Optionally, the updates may be distributed to other groupware service users as well. Even though this GSME type is designed based on the GW_RemoteInteraction pattern, it does not include a CheckEnable action, since groupware service users should always be allowed to change their personal information.

Although not prescribed by our design, a groupware service provider may store various types of information regarding a groupware service user for various reasons. The information that may be stored includes:

- Name. The name of the groupware service user may for instance be needed to allow others to invite a person by his name as part of the SelectPerson GSME.
- Preferences. This information is for instance needed in order to propose a default communication tool as part of the SelectAvailableTool GSME.
- Presence status. This information can for instance be applied as part of the SelectPerson GSME.

Figure 5-65 Update PersonalInfo: Groupware service provider behaviour



5.8 Generic behaviour applied by GSMEs

The GSME type descriptions make use of generic behaviour that is provided by the groupware service provider. Although this behaviour does not represent individual GSMEs, it is described in this chapter as it determines the behaviour associated with the individual GSMEs that make use of it.

5.8.1 GetAvailableGSMs

This GetAvailableGSMs behaviour returns the list of GSMs that are currently available for activation and possibly inclusion in a groupware composition. In contrast to the GetActiveGSMs behaviour, described in the following section, this groupware behaviour returns GSMs that have been made available, but have not been activated yet. The GetAvailableGSMs behaviour is part of the SelectAvailableTool, SelectAvailableCoord, Select&ActivateConfMgt, and ActivateEnabling GSMEs.

One can filter the results of the GetAvailableGSMs behaviour by specifying as a parameter one of the GSM types discussed in the next chapter. Using this filter one can for instance obtain a list of available communication / collaboration GSMs.

Our design does not specify how the GetAvailableGSMs behaviour is to be implemented. A well-known mechanism to provide such behaviour is a registry that lists installed software components.

5.8.2 GetActiveGSMs

The GetActiveGSMs behaviour returns the GSMs that are currently activated. The GetActiveGSMs behaviour is part of the SelectPerson, SelectActiveConference, SelectAvailableTool GSME, and DeactivateEnabling GSMEs.

One can filter the results of the GetActiveGSMs behaviour by specifying as a parameter one of the GSM types discussed in the next chapter. Using this filter one can for instance obtain a list of activated communication / collaboration GSMs. Our design does not specify how the GetActiveGSMs behaviour is to be implemented.

5.8.3 ActivateGSM

This behaviour activates the software components that implement a given GSM. The GSM that is specified as a parameter to the behaviour has to be available in order to be activated. This behaviour is part of the Select&ActivateConfMgt, Join, SelectTemplate, AddTool, AddCoord, and ActivateEnabling GSMEs.

5.8.4 ConnectGSMs

This behaviour connects two specified active GSMs. Since the composition of GSMs determine the groupware service that is provided to the co-operating end users, this behaviour allows groupware service users to tailor the provided groupware service. The behaviour is part of the AddTool and AddCoord behaviour. Chapter 6 defines the rules for valid compositions of GSMs in our design.

5.8.5 DisconnectGSMs

This behaviour disconnects two specified connected GSMs. Since the composition of GSMs determine the groupware service that is provided to the co-operating end users, this behaviour allows groupware service users to tailor the provided groupware service. The behaviour is part of the RemoveTool and RemoveCoord behaviour.

5.8.6 GetConnectedGSMs

This behaviour returns the GSM that are connected to a given GSM. One can filter the results by specifying as a parameter one of the GSM types discussed in the next chapter. Using this filter one can for instance obtain a list of communication / collaboration GSMs that are connected to a given conference management GSM. The GetConnectedGSMs behaviour is part of the SelectActiveTool, GetActiveCoord, RemoveTool and RemoveCoord GSMEs.

5.9 Conclusions

This chapter describes the details regarding the individual GSME types, i.e., the types of *elementary units of groupware behaviour*. These GSME types, identified in chapter 4, are described in a structured manner in order to:

1. Describe the abstract interactions that take place between groupware service users and the groupware service provider. These local service interface descriptions state what interactions take place as a result of a GSME, as well as the relations between these interactions and the parameters that are established as part of each interaction.
2. Describe the behaviour provided by the groupware service provider. This description focusses on the information flow between distinct, and possibly geographically distributed, Service Access Points (SAPs) as well as the actions performed by the groupware service provider based on this information. Actions are only described when they influence the externally observable behaviour, as explained in section 3.2.2.

We claim that the identified set of GSME types forms a valid basis to *prescribe* and *describe* the behaviour of groupware applications. This claim is sustained in chapters 7 and 8, when the prescriptive and descriptive properties of the design are evaluated.

5.9.1 Tailoring GSMEs

An innovative aspect of the set of GSME types is that they include *tailoring GSMEs*: GSME types that allow groupware service users to change the composition of Groupware Service Modules (GSMs). As a result, these tailoring GSME types allow for changes to the groupware service provided by the groupware application.

Table 5-32 Tailoring GSME types

Tailoring GSME type	Description
AddTool	Allows groupware service users to associate the behaviour provided by a communication / collaboration GSM with a given conference. As a result, the groupware service is extended with this behaviour.
RemoveTool	Allows groupware service users to remove the behaviour provided by a communication / collaboration GSM from a given conference. As a result, the behaviour is removed from the provided groupware service.
AddCoord	Allows groupware service users to associate the behaviour provided by a co-ordination GSM with a given conference. As a result, the provided groupware service is extended with this behaviour.
RemoveCoord	Allows groupware service users to remove the behaviour provided by a co-ordination GSM from a given conference. As a result, the behaviour is removed from the provided groupware service.
MakeAvailableGSM	Allows groupware service users to make a new GSM available for activation and subsequent use as part of a groupware service.
UnmakeAvailableGSM	Allows groupware service users to make a GSM unavailable for activation, thus preventing the use as part of a groupware service.
Select&ActivateConfMgt	Allows groupware service users to add the behaviour provided by a conference management GSM to the provided groupware service.
ActivateEnabling	Allows groupware service users to add the behaviour provided by a people listing GSM or a conference listing GSM to the provided groupware service.
DeactivateEnabling	Allows groupware service users to remove the behaviour provided by a people listing GSM or a conference listing GSM from the provided groupware service.

Because of the existence of tailoring GSME types, this chapter already introduces the GSM types distinguished in the CooPS groupware reference model: the tailoring GSME types adjust the composition of GSMs that form the groupware service. The next chapter describes the various GSM types in detail, stating the GSME types they consist of and their relations on a service level.

Chapter 6

The CooPS groupware reference model

This chapter describes the CooPS groupware reference model: a service reference model that defines *Groupware Service Module (GSM) types*. As indicated in *Figure 3-8*, GSM types are formed by grouping the more elementary GSME types. GSMs, created based on these types, are service elements: they form the *units of composition of groupware services*.

While the GSME types described in chapter 5 can be applied both to describe and to prescribe groupware services, the CooPS groupware reference model is *prescriptive*: it defines the types of units to compose groupware services.

The chapter describes the criteria applied to derive the GSM types and the design choices involved. Subsequently, the chapter provides detailed description of the various GSM types distinguished in the reference model, as well as their relations on a service level.

6.1 Introduction

Reference models, as the name suggests, are about *reference* and *models*. They provide a reference in the sense that they offer concepts and definitions that are shared by people as a basis for their understanding and discussions and, more importantly, as a basis for their actions. They are about models in the sense that they describe essential aspects of systems in abstract terms, typically by focussing on what systems should do rather than how they physically are constructed and operate.

Reference model

In this context, a reference model is defined as a structure, or organization, of related functional entities, i.e., groupware service modules. A reference model states only the key functions and key relationships of these entities and defines them at a high level of abstraction:

1. Given the high level of abstraction, functions and relationships are defined as implementation independent as possible, leaving freedom for the individual manufacturer to choose an implementation strategy.
2. Only the most relevant functions and relationships are defined, leaving freedom for design teams to complete the functional design by adding functional elements that originally are not considered most relevant.

6.1.1 Purpose of a reference model

A reference model serves several purposes:
- It provides a set of precisely defined basic architectural concepts, as described in chapter 3, that can be used to construct a design model;
- It presents a translation of the essential user requirements in terms of a design at a high level of abstraction;
- It defines an initial structuring of a service;
- It acts as a basis to communicate a high-level design with the designers of (parts of) the real world system that has to be developed;
- It acts as a basis to design (parts of) a real world system. In this process, detail is added to the reference model;
- To a lesser degree, it acts as a basis to discuss a high-level design with prospective users and a design team in order to establish requirements and document agreements. We consider a reference model primarily as a tool for designers; it may be too abstract for communication with prospective users.

6.1.2 Positioning the CooPS groupware reference model

The CooPS groupware reference model is a service-level reference model that defines types of groupware service modules (GSMs). These GSM types are a grouping of GSME types. The reference model defines the relations between GSM types on a service level. The reference model does not prescribe *how* the behaviour associated with a GSM type is to be implemented. Typically, different implementations will exist of the same GSM type. For instance an audio conferencing tool and a shared whiteboard are both implementations of the communication / collaboration GSM type introduced in this chapter.

In relation to the ISO Open System Interconnection (OSI) reference model (ISO, 1987), the CooPS groupware reference model can be seen as a specification of the application layer: the model specifies the service that is provided to human groupware service users. Additionally, the model describes the service elements that are needed to specify this service, and their relations.

6.1.3 Geographical distribution aspects

The CooPS groupware reference model provides a service-level description; it does not provide implementation details regarding the functional entities that provide this service. One functional entity, i.e. a GSM, may be implemented as a series of software components that are distributed over multiple clients and servers. The CooPS groupware reference model does not prescribe how the software that implements a GSM is geographically distributed. The model assumes that in a conference, all groupware service users have access to the specified groupware service. As a result, it assumes that the various conference participants have access to the behaviour provided by the individual GSMs that are composed to provide the groupware service.

As groupware applications are inherently distributed, the service-level specification frequently needs to differentiate between *local* and *remote* behaviour. The local behaviour denotes the behaviour that allows a local groupware service user to initiate a given GSME, and to receive *feedback* about the results of this action. The remote behaviour indicates the behaviour to inform other groupware service users about the results of the action, denoted as *feedthrough* information.

In the CooPS groupware reference model, different groupware service users make use of different Service Access Points (SAPs) to access the groupware service provided. The distinction of different SAPs reflects the geographical distribution on a service level.

6.1.4 Notion of conferences

The CooPS groupware reference model is created based on the notion of *conferences*. In CSCW literature this notion is sometimes also denoted as the *session model*. Given this design choice, our groupware reference model is especially suited to prescribe support for *synchronous* co-operation: settings where people are present at the same time in order to co-operate. Examples of support for synchronous co-operation include chat applications, audio- and videoconferencing applications, group decision support systems, and collaborative workspaces.

It is also possible to apply the CooPS groupware reference model to specify more asynchronous forms of co-operation: the model does not specify the delays between the *request*, *indication* and *feedthrough* interactions associated with a given GSME. As such, one could apply the reference model to specify the behaviour of an e-mail application towards its users.

6.1.5 Relation with independent extensibility

A system is *independently extensible* if it is extensible and if independently developed extensions can be combined (Szyperski, 1998). The CooPS groupware reference model improves the possibilities for independent extensibility by providing service-level descriptions of the units that form groupware services. These service-level descriptions provide a valid starting point to assess whether independently implemented groupware service modules interfere. However, our service-level descriptions cannot guarantee independent extensibility.

On an implementation level, more detailed descriptions are needed to construct independent extensible groupware applications. Typically, precise software component interfaces need to be specified to achieve composability and extensibility of software applications. The CooPS groupware reference model provides a basis to design such software component interfaces for this purpose.

6.2 Criteria for defining Groupware Service Module types

A groupware service is determined by the Groupware Service Module Elements (GSMEs) it consists of. In our groupware service reference model, GSME types are grouped into Groupware Service Module (GSM) types. These GSM types are introduced to facilitate the process of selecting and composing groupware behaviour: concrete GSMs, implemented based on these types, are designed to form meaningful and recognizable *units of composition* of groupware services. This notion of service composition corresponds to the feature composition described by Teege (2000).

A GSM type is defined by the GSME types it consists of. As a result, the relations between GSM types on a service level can be derived from the GSME types they consist of: the causality relations between GSME types that have been allocated to different GSM types determine the causality relations between GSM types, as illustrated in the following figure.

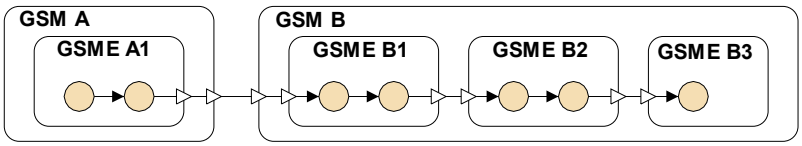


Figure 6-1 Service-level relations between GSM types are derived from relations between GSME types

6.2.1 Underlying architectural criteria

The GSM types defined in the CooPS groupware reference model are based on the criteria outlined in section 3.7.4. Those criteria are translated into the following criteria to cluster GSME types into GSM types:

- *Generality*. The GSM types identified should be sufficient to construct groupware services for a wide range of co-operative settings.
- *Conceptual consistency*. GSM types should provide similar behaviour in similar ways to groupware service users: First, a single GSM type should provide similar behaviour in similar ways. Second, different GSM types should provide similar behaviour in similar ways;
- *Correctness and completeness*. The grouping of GSME types into a GSM type should not change the behaviour associated with the individual GSME types. Additionally, each GSME type should be allocated to at least GSM type. Note that in order to increase the *effectiveness* and *efficiency* of the design, one GSME type may be allocated to multiple GSM types, even though that increases the *complexity* of the design;
- *Orthogonality*. Different GSM types handle different functions: all GSME types related to one aspect of the co-operation should be clustered in one GSM type. Orthogonality allows groupware designers to focus on a single aspect: a groupware designer can for instance create a new communication / collaboration GSM to share high-resolution medical images, without the need to implement conference management behaviour;
- *Propriety and parsimony*. The set of GSM types should be as small as possible: the set should only include GSM types essential to provide the required groupware behaviour. Additionally, the interdependencies between different GSM types should be minimized.

6.2.2 Underlying criteria based on tailorability

In our design, end users can select and compose GSMs. They do so via tailoring GSMEs: GSMEs that allow groupware service users to add GSMs to the groupware composition or to remove GSMs from the groupware composition. Based on this mechanism, end users can tailor the behaviour of their groupware application, i.e., the provided groupware service. This results in the following criteria on the grouping GSME types into GSM types:

1. *Appropriate flexibility*. The set of GSM types should be sufficiently large to allow end users to select and compose the groupware behaviour that is appropriate for their co-operative task. The task-technology fit theory, introduced in chapter 1, states that the behaviour provided by a groupware application should be sufficient for end users to effectively and efficiently perform their co-operative tasks. At the same time, the

provided behaviour should not result in an information overload for the co-operating end users;

2. *Recognizable for end users*. The selected GSM types should represent recognizable groups of GSME types. We assume that when GSM types are in line with the types of units tailors distinguish in a groupware application, the chances of successful tailoring will increase. In this context, successful tailoring is defined as selecting and composing groupware behaviour that is appropriate to support a specific co-operative task;
3. *Substitution of clusters of behaviour*. GSME types that are likely to be replaced as a group should be clustered in a separate GSM type. Different implementations may present the same GSME in fundamentally different ways, for instance to reflect different interaction styles. As an example, co-operating people may wish to substitute the set of GSMEs to manage a conference with a different implementation of the same GSMEs. Such behaviour is an indicator to cluster that set of GSMEs in one GSM type;
4. *Inclusion and exclusion of clusters of behaviour*. A group of GSME types that are needed in some conferences, and left out in other cases should be clustered in a separate GSM type. As indicated in section 4.4, some conferences should include GSMEs to configure and enact a co-ordination policy, while a co-ordination policy may be counterproductive in other conferences. Such an observation is an indicator to cluster the GSMEs related to defining and enacting a co-ordination policy in a separate GSM type. That way, groupware service users can choose to include one type of co-ordination (such as role-based access control) in one conference, include a different type of co-ordination (such as a workflow) in another conference, and leave out co-ordination in a third conference.

As a result of these criteria, GSMs, which are designed based on the GSM types defined in our groupware reference model, have to be *units of selection and composition* by end users: groupware service users have to be able to select and compose GSMs to form groupware services. As a derivative of this, GSMs should be *units of design and implementation* for designers: it has to be possible for groupware designers to design and implement individual GSMs, independent of the implementation details of the GSMs it is to be composed with. As the CooPS groupware reference model is a service-level reference model, it provides a valid basis for this form of independent extensibility. However, a service-level architecture alone is not sufficient to guarantee independent extensibility: this property is also influenced by the way in which one implements GSMs.

6.3 GSM type design choices

This section states and motivates design choices to group GSME types to form more coarse-grained units of composition of tailorable groupware services. These design choices result from applying the previously mentioned criteria.

1. Cluster GSME types associated with one communication tool in a separate GSM type;
2. Cluster GSME types associated with one collaboration tool in a separate GSM type;
3. Cluster GSME types associated with conference management and participation management in one GSM type;
4. Cluster GSME types to configure and enact a co-ordination policy in a separate GSM type;
5. Cluster GSME types to obtain information regarding people to invite in a separate GSM type;
6. Cluster GSME types to obtain information regarding conferences to join in a separate GSM type;
7. Cluster GSME types to bootstrap groupware services in a separate GSM type.

Cluster GSME types associated with one communication tool in a separate GSM type

Section 4.4.1 states that depending on their tasks, preferences and context, people should be able to select the tools to communicate. In order to facilitate this selection process, all behaviour associated with such a tool should be clustered in a separate unit of composition.

In our research, *communication tools* denote tools that support direct communication between groupware service users. The CooPS groupware reference model does not provide a strict distinction between *communication tools*, i.e., tools that support direct communication between conference participants, and *collaboration tools*, i.e., tools that support indirect co-operation via shared information objects. Instead, the CooPS groupware reference model applies the concept of *tools* to denote both extremes, as well as any intermediate forms.

Cluster GSME types associated with one collaboration tool in a separate GSM type

Section 4.4.2 states that depending on their tasks, preferences and context, people should be able to select the tools to collaborate using shared information objects. In order to facilitate this selection process, all

behaviour associated with such a tool should be clustered in a separate unit of composition.

The set of active tools, i.e. the active communication tools as well as the active collaboration tools, determines to a large extent how the people in a conference can work together. Conversely, the task people perform together determines the set of tools they need (Bentley & Dourish, 1995; Zigurs & Buckland, 1998).

As described by Ter Hofte (1998), media lend themselves very well to be used as a unit of composition and extension by groupware users. The GSM type that comes into being if one clusters all interactions associated with a single communication tool or a single collaboration tool correspond to one single medium, as denoted by Ter Hofte. Examples of GSMs of this type include a GSM for audio conferencing, a shared whiteboard, a GSM for text-based chat, and a GSM to access a video database together. This type of GSMs is denoted in the CooPS groupware reference model as a *communication / collaboration GSM*, abbreviated to CC-GSM.

Software engineers can implement individual CC-GSMs separately from other GSMs: CC-GSMs lend themselves as units of design and implementation. This aspect is demonstrated in chapter 7.

One notable exception takes place when different CC-GSMs implementations have to synchronize their states. This is for instance needed to implement lip synchronization between separately developed audio conferencing and video conferencing CC-GSMs. The CooPS groupware reference model does not specify relations between different CC-GSMs. To solve this issue, groupware programmers may for instance specify additional software interfaces between CC-GSM implementations. However, groupware programmers may not rely on such additional interfaces to exist, as that would counter the possibilities for independent extensibility.

Currently, a number of organizations already design and implement separate CC-GSMs that depend on a separate conference management GSM. For instance, the Dutch company Suite75[9] has developed a CAD viewer CC-GSMs that can be used with the Groove Workspace. Another example are the separate tools that are developed for the MBone conferencing application[10]. This indicates the possibility, and the existence of a business model, to design and implement separate CC-GSMs.

[9] For more information, see http://www.suite75.com/about.htm
[10] For more information, see http://www-mice.cs.ucl.ac.uk/multimedia/software/

Cluster GSME types associated with conference management and participation management in one GSM type

Section 4.4.5 states that different types of online conferences are likely to require different conference management styles. In order to facilitate this selection process, all behaviour associated with managing conference existence and conference participation should be clustered in a separate unit of composition.

According to the definition on page 8, a conference denotes a set of people, supported by a groupware service. The groupware service associated with a specific conference is determined by the composition of GSMs associated with that conference[11]. As a result, a conference denotes a set of co-operating people and the set of GSMs that supports them. The type of GSM that comes into being if one clusters all behaviour to manage the set of co-operating people and to manage the set of GSMs that supports them, is denoted as the *conference management GSM*.

The notion of a conference is particularly useful since during a conference the majority of interactions between groupware service users and the groupware service provider involve the same group of conference participants and the same set of GSMs. Another reason for keeping together the behaviour associated with conference management and participation management is that this set of behaviour forms a coherent set of GSME types that can be considered meta-functions to govern the co-operation.

A conference management GSM is a desirable unit of selection and composition in order to handle different types of conferences. Even though different conference management GSMs provide the same abstract behaviour to groupware service users, they may interact with the users in fundamentally different manners. Aspects of a conference such as the number of conference participants, the anticipated dynamics of conference membership, and the anticipated dynamics in the required groupware service determine what makes an effective, efficient and satisfactory conference management GSM for a given conference. As a result, different conferences, which differ in terms of these aspects, may require different conference management GSMs. However, we do not expect that during a conference, groupware service users will need to switch between different conference management GSMs. As a result, our groupware design does not include behaviour to do so.

A conference management GSM is a feasible unit of design and implementation. Existing groupware applications that separate conference management and participation behaviour from the behaviour to actually co-

[11] Only conference manager GSMs, tool GSMs and co-ordinator GSMs can be associated with a specific conference.

operate include the Groove Workspace and NSCA Habanero[12]. This observation is an indicator that the guideline corresponds to implicit software engineering practices.

Cluster GSME types to configure and enact a co-ordination policy in a separate GSM type

Section 4.4.3 states that some types of conferences require explicit co-ordination of the manner in which groupware service users work together. As a result, a groupware application should allow groupware service users to specify whether a co-ordination engine should be associated with the conference, and if so, which co-ordination engine to use. In order to facilitate this process, all behaviour to configure and enact a co-ordination policy should be clustered in a separate unit of composition.

According to this design choice, all groupware service provider behaviour associated with defining and enacting a co-ordination policy are clustered in a separate GSM type. As a result of this guideline, groupware service users are enabled to choose the kind of co-ordination engine that matches their co-operative setting, independent of the CC-GSMs they use to co-operate or the selected conference management GSM. Users may for instance select an engine for role-based access control, a workflow engine, or leave out a co-ordination engine from the groupware service composition. The type of GSM to configure and enact a co-ordination policy is denoted in the CooPS groupware reference model as a *co-ordination GSM*.

As reported in section 4.4.3, depending on the types of co-operation different co-ordination mechanisms may be required. Based on this, we consider a co-ordination GSM a desirable unit of selection and composition in groupware services. Aspects such as the number of conference participants and the desired openness of the conference determine which co-ordination style is beneficial for a conference. However, we do not envision that end-users frequently change between different co-ordination styles during a conference. One example of such a change is a conference that starts with a small number of participants without any co-ordination, and is joined by many more people. In that case, the participants may decide to include a co-ordination GSM and appoint a chairman to regulate the conference.

A co-ordination GSM lends itself as a unit of design and implementation. The co-ordination policies, which are defined and enacted by a co-ordination GSM, can be separated from the GSME types they control. One may for instance specify that a given user is allowed to expel other participants from a conference. Independent of the manner in which

[12] For more information, see http://www.isrl.uiuc.edu/isaac/Habanero/

this Expel GSME is presented to the groupware service user, the associated interactions are enabled or disabled by the co-ordination GSM.

As a prerequisite for this separation, the GSME type descriptions in chapter 5 specify which GSME types may be co-ordinated: they include the CheckEnable behaviour as part of the service provider behaviour.

One aspect that increases the difficulty of designing and implementing a separate co-ordination GSM is independent extensibility: groupware service users can select and compose the communication / collaboration GSMs, conference management GSMs, and co-ordination GSMs that suit their needs. The implementations of these GSMs may even originate from different manufacturers.

As a result, a co-ordination GSM co-ordinates a possibly dynamic set of GSMEs. Moreover, this set is not known at design time. Nevertheless, some mapping is needed from actual actions on the various GSMs onto the rules defined by the co-ordination GSM. As a complicating factor, a CC-GSM may provide multiple UseTool GSME subtypes.

The CooPS groupware reference model does not prescribe how active CC-GSMs declare to other GSMs what UseTool GSME subtypes they provide. The reference model only states that, on a service level, the set of UseTool GSME subtypes provided by a CC-GSM influences the options for the SetAccessPermission GSME.

The COCA architecture (Li & Muntz, 1998) separates co-ordination functions from other functions to co-operate. The COCA architecture has been applied to design and implement a proof-of-concept platform, which illustrates that a co-ordination GSM is a feasible unit of design and implementation.

Cluster GSME types to obtain information regarding people to invite in a separate GSM type

This design choice was made in order to allow groupware service users to make use of multiple, possibly fundamentally different, mechanisms to obtain information about other people. This information can subsequently be used as part of an invite process.

According to this design choice, all groupware service provider behaviour to obtain information about other people is clustered in a separate GSM type. This type of GSM is denoted as a *people listing GSM* in the CooPS groupware reference model. People listing GSMs are one out of two types of Enabling GSMs.

The behaviour provided by a people listing GSM is independent from the actual conference to which the people are to be invited: one people listing GSM may be applied to invite people to different conferences, with different conference management GSMs. Additionally, a people listing GSM

may be active when the groupware service user does not participate in any online conference.

Different people listing GSMs may collect different types of information regarding people. Based on the information collected. Knowledge about the semantics of this information may be exploited in the GUI of a people listing GSM to represent the information to the groupware service user.

Groupware service users may apply different people listing GSMs for different categories of contacts: a corporate directory with all employees and the divisions they work for, a buddy list that shows what friends are available for communication, and a digital version of the phone book.

The CooPS groupware reference model also allows conference management GSMs to present information regarding people to invite. Although on the service level some GSME types are allocated to both the conference management GSM type and the people listing GSM type, this does not imply that on the implementation level the behaviour is implemented multiple times.

These aspects indicate that a people listing GSM lends itself as unit of selection and composition. Groupware service users may activate and use different people listing GSMs depending on the type of contact they wish to invite, and depending on the type of information they wish to obtain about the person.

Since different people listing GSMs may collect, and represent, different information regarding people, a people listing GSM is considered to be a unit of design and implementation. The combination of Microsoft Windows Messenger and the Groove Workspace is an example from industry where a people listing GSM and a conference management GSM have been separated: if both applications are installed, the Windows Messenger can act as a people listing GSM and trigger the Groove Workspace to invite a specific person to a new conference.

Cluster GSME types to obtain information regarding conferences to join in a separate GSM type

This design choice was made in order to allow groupware service users to make use of multiple, possibly fundamentally different, mechanisms to obtain information about ongoing conferences. This information can subsequently be used as part of the process to join existing conferences.

According to this design choice, all groupware service provider behaviour to obtain information about conferences is clustered in a separate GSM type. This type of GSM is denoted as a *conference listing GSM* in the CooPS groupware reference model. Conference listing GSMs are one out of two types of Enabling GSMs.

Conference listing GSMs are applied to launch an appropriate, i.e., interoperable[13], conference management GSM; the newly activated conference management GSM is subsequently triggered to send the request to join the conference.

Different conference listing GSMs may collect different types of information regarding conferences. This information may include details regarding the purpose of the conference, its organizer, and the communication / collaboration GSMs that are active in the conference. Knowledge about the semantics of this information may be exploited in the GUI of a conference listing GSM to represent the information to the groupware service user.

The CooPS groupware reference model does not prescribe a mechanism to publish information regarding a conference: this is considered out of scope for the current research.

Groupware service users may apply different conference listing GSMs for different types of conferences: a corporate directory may list ongoing online meetings within one company, while a professional association, such as the IEEE, may publish a global list of ongoing online conferences around a given topic.

The CooPS groupware reference model also allows conference management GSMs to present groupware service users information regarding conferences to join. Although on the service level some GSME types are allocated to both the conference management GSM type and the conference listing GSM type, this does not imply that on the implementation level the behaviour is implemented multiple times.

These aspects indicate that a conference listing GSM lends itself as unit of selection and composition. Groupware service users may activate and use different conference listing GSMs depending on the type of conference they wish to join, and depending on the type of information they wish to obtain about the conference.

Conference listing GSMs form units of design and implementation. An example of an existing separate conference listing GSM is the UCL Secure Conference Store[14]. The Secure Conference Store provides users an overview of available multicast conferences they can join. Conferences can be joined using a separate Java applet that acts as a conference management GSM.

---

[13] A conference manager GSM implementation is interoperable with a given conference if it adheres to the communication protocol standard applied in that conference. This interoperability principle also holds for any active tool GSMs and co-ordinator GSMs: different conference participants may apply different implementations of these GSM types, as long as they adhere to a common communication protocol standard.

[14] For more information, see https://www-secure.cs.ucl.ac.uk/

Cluster GSME types to bootstrap groupware services in a separate GSM type

This design choice was made in order to provide groupware service users with a single unit to launch groupware services and to make Groupware Service Modules available for activation and use.

According to this design choice, all groupware service provider behaviour needed to bootstrap groupware services is clustered in a separate GSM type. The CooPS groupware reference model distinguishes three types of GSMs from which a user can start or join a conference: the conference management GSMs, people listing GSMs, and conference listing GSMs. Accordingly, the bootstrapping behaviour encompasses the behaviour needed to activate these three types of GSMs.

Additionally, groupware service users may use templates to start groupware services. Such templates, denoted as *groupware patches* in section 8.6, provide valid starting points for groupware use and tailoring: they describe a valid composition of GSMs, typically related to a specific task. As a result, the bootstrapping behaviour also encompasses the behaviour to select templates and trigger the start of a conference based on a selected template.

As illustrated in *Figure 3-9* on page 45, there are various stages in the lifecycle of a GSM: before a user can make use of the behaviour provided by any GSM, that GSM has to be activated. In turn, before a GSM can be activated, that GSM has to be made available using the MakeAvailableGSM GSME. This behaviour to make GSMs available for selection and activation is also considered part of the bootstrapping behaviour.

The type of GSM that provides bootstrapping behaviour is denoted in the CooPS groupware reference model as the *bootstrapping GSM* (B-GSM). Groupware service users have to be able to use the behaviour provided by a B-GSM at any time: without this behaviour the groupware service user cannot start new conferences or join existing ones. The CooPS groupware reference model does not prescribe how a user activates a B-GSM.

The B-GSM is a unit of design and implementation: software engineers are able to design and implement a B-GSM without knowing the internal implementation details of the GSMs it should activate. However, since the component model applied to implement a software component determines the manner in which it should be activated, the software engineers have to know what component model is applied to implement GSMs.

6.4 The groupware service reference model

This section states the details of the CooPS groupware reference model. After providing an overview of the model, the section discusses the various

Groupware Service Module (GSM) types in detail, and describes their relations on a service level.

6.4.1 Overview of the CooPS groupware reference model

The CooPS groupware reference model, depicted in *Figure 6-2* and *Figure 6-3*, provides structuring guidelines to design tailorable groupware services. The reference model is formed based on the criteria described in section 6.2. The CooPS groupware reference model defines different GSM types, which are formed by grouping GSME types. The model states the responsibilities of these types of GSMs in terms of the GSMEs they consist of, and the relations between GSM types on a service level.

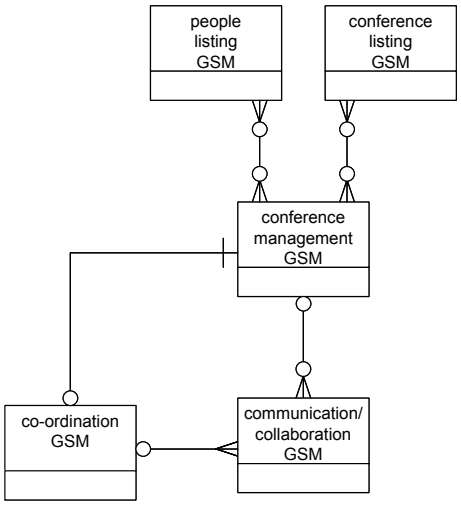


Figure 6-2 ER diagram of the service elements in the CooPS groupware reference model and the cardinalities of the "is associated with" relations

The first GSM type, the *conference management GSM* (CM-GSM), groups all GSMEs related to the management of an online conference. This includes GSMEs to manage conference participation, manage the set of active communication and collaboration tools, and to manage which co-ordination engine is associated with the conference. In the CooPS groupware reference model one active conference management GSM is associated with one active conference: the groupware composition associated with an online conference contains exactly one conference management GSM.

One person may concurrently participate in multiple conferences. In that case, the person has access to the behaviour provided by multiple active conference management GSMs. Note that this relation is not shown in *Figure 6-2*.

To support direct communication as well as collaboration via shared information objects the CooPS groupware reference model includes

communication / collaboration GSMs (CC-GSM). A communication / collaboration GSM provides support for direct communication between conference participants (such as an audio conferencing tool) or support for collaboration via a set of shared information objects (such as a tool to broadcast presentations). Based on the dynamics during co-operation, as described in section 4.4, we expect that changing the set of active communication / collaboration GSMs is an important and frequently applied tailoring operation. During a conference, the conference participants should have at least one means to communicate or collaborate. Nevertheless, before groupware service users have selected the means to co-operate, there may not be an active communication / collaboration GSM included in the groupware composition.

The *co-ordination GSM* (CO-GSM) provides all GSMEs to configure and enact co-ordination rules during a conference. In particular, it controls the use of conference management GSMEs and the use of GSMEs provided by communication / collaboration GSMs. Since activating co-ordination rules is optional during a conference, a groupware service does not need to include an active co-ordination GSM.

Two types of *Enabling GSMs* are included in the design: people listing GSMs and conference listing GSMs. People listing GSMs (PLIST-GSM) provide information about other people in order to facilitate the process of inviting these people to a conference. Conference listing GSMs (CLIST GSM) provide information about other conferences in order to facilitate the process of joining these conferences. This behaviour is optional: not every groupware service needs to include either a people listing GSMs or a conference listing GSM.

Activated Enabling GSMs are not connected with one specific conference: people listing GSMs can be applied to invite people to multiple conferences; conference listing GSMs can be applied to join multiple conferences. As such, Enabling GSMs only have a relation with conference management GSMs.

Figure 6-3 provides an overview of the service-level relations that exist between the various GSM types distinguished in the CooPS groupware reference model. The individual relations will be discussed together with the detailed descriptions of the GSM types.

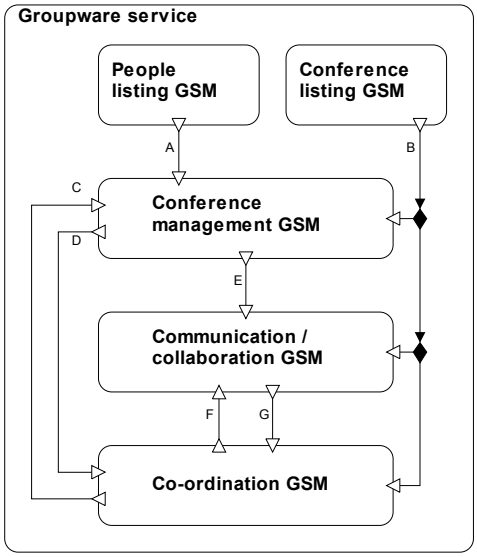


Figure 6-3 Overview of GSM types and their relations on a service level

6.4.2 Overview of GSME allocation

The following table provides an overview of how the GSME types described in chapter 5 have been grouped to form GSM types.

Table 6-1 Allocation of GSME types to GSM types

GSME categories	GSME type	Conf. mgt GSM	Communication / collaboration GSM	Co-ordination GSM	People list. GSM	Conf. list. GSM	Bootstrapping GSM
Conference management GSMEs	StartConf, EndConf, GetConfInfo, ChangeConfInfo	X					
Participation management GSMEs	GetParticipants, Leave, Invite, Expel	X					
	Join (*)	X				X	
Communication and collaboration GSMEs	SelectActiveTool, AddTool, RemoveTool	X					
	SelectAvailableTool (*)	X			X		
	UseTool		X				
Co-ordination GSMEs	GetActiveCoord, AddCoord, RemoveCoord	X					
	SelectAvailableCoord (*)	X			X		
	AssignRole, GetRole, SetAccessPermission, (CheckEnable)			X			
Conference enabling GSMEs	EnterInfoPerson (**)	X			X		
	SelectPerson (*)	X			X		
	EnterInfoActiveConference (**)	X				X	
	SelectActiveConference (*)	X				X	
Bootstrapping GSMEs	(Un-) MakeAvailableGSM						X
	Select&ActivateConfMgt, UpdatePersonalInfo (*)				X		X
	SelectTemplate, (De-) ActivateEnabling						X

(*) These GSME types have been allocated to more than one GSM type. Although this reduces the degree of orthogonality and parsimony in the design, it increases the efficiency and effectiveness of the design. For instance, the allocation of the SelectAvailableTool GSME allows groupware service users to select an available tool *during* a conference via the conference management GSM, as well as in the process of *launching* a new conference via a people listing GSM.

(**) For completeness, these GSME types have also been allocated to the associated Enabling GSM: groupware service users may for instance also enter the information about the person to invite via a people listing GSM.

6.5 Conference management GSM (CM-GSM)

Primary purpose: the conference management GSM is primary responsible for providing groupware service users behaviour to start and end conferences, manage the set of conference participants, manage the set of communication and collaboration tools that is active in the conference and manage which co-ordination engine is active in the conference.

The conference management GSM (CM-GSM) provides users the behaviour to manage online conferences. It provides behaviour to manage the conference lifecycle and participation. Additionally, the CM-GSM includes GSME types to manage the set of CC-GSMs that is active in the conference and GSME types to manage which co-ordination engine, i.e., which CO-GSM, is associated with the conference. A valid composition of active GSMs associated with one conference contains one active CM-GSM.

Section 4.4.5 explains that in different situations, different conference management styles may be preferred. To accommodate this, the CooPS groupware reference model allows groupware users to select and activate a specific CM-GSM implementation. However, groupware service users are only able to do so upon starting a new conference; the model does not include facilities to switch between different CM-GSM implementations during a conference. This option is omitted as we found no evidence in literature that such switches are actually desired by groupware service users.

Since CM-GSMs are also responsible for handling incoming invitations, a CM-GSM of the appropriate type has to be active in order to receive and accept incoming invitations.

6.5.1 Associated GSME types

The CM-GSM type is a grouping of the following GSME types:

- *StartConf*, *EndConf*. Basic GSME types to start and end conferences;
- *GetConfInfo**, *ChangeConfInfo**. When these GSME types are left out, groupware service users cannot set or read conference information. These GSME types should either be included as a group, or be left out as a group;
- *GetParticipants*, *Leave*, *Invite*. Basic GSME types for conference participation management;
- *Join**, *EnterInfoActiveConference**, *SelectActiveConference**. When these GSME types are left out, groupware service users cannot request to join an existing conference: they can only join an existing conference after receiving an invitation. The Join GSME and EnterInfoActiveConference GSME should either be included as a group, or be left out as a group.

The SelectActiveConference GSME should only be implemented if the CM-GSM also provides the Join GSME type;
- *Expel*[*]. When this GSME type is left our, groupware service users cannot expel other groupware service users from a conference;
- *SelectActiveTool*, *SelectAvailableTool*, *AddTool*, *RemoveTool*. Basic GSME types to manage the set of communication / collaboration GSMs that is active in a conference;
- *GetActiveCoord*[*], *SelectAvailableCoord*[*], *AddCoord*[*], *RemoveCoord*[*]. When these GSME types are left out, groupware service users cannot select and include a co-ordination GSM in the groupware composition. As a result, it is not possible to configure and enact a co-ordination policy during the conference. These GSME types should either be included as a group, or be left out as a group;
- *EnterInfoPerson*; This GSME type has to be included to provide a basic means of inviting another person to a conference, in case no people listing GSM has been activated;
- *SelectPerson*[*]. When this GSME type is left out, groupware service users cannot access the information provided by any active people listing GSMs via this CM-GSM.

[*]: Optional GSME type: Not every CM-GSM is required to include this GSME type.

A CM-GSM includes GSME types for conference management, participation management, tool management and co-ordination management. One could allocate these GSME types to separate GSMs. The CooPS groupware reference model combines these GSME types in one GSM type based on the many information dependencies that exist between these GSME types. For instance, as part of the Invite GSME the list of conference participants is updated and the new participant receives information regarding the conference, including the set of active communication / collaboration GSMs and which co-ordination GSM is associated with the conference. Distributing these GSME types over multiple GSM types would oppose the principle of *low coupling*.

By combining all management-related groupware behaviour in one GSM type, we intend to increase the recognizability of this GSM type as the central point to manage a conference. In turn, we assume that this understanding of the purpose of a GSM type increases the likelihood of successful tailoring.

The decision to cluster all management-related GSME types into one GSM type has the disadvantage that conference management, participation management, tool management and co-ordination management GSME types cannot be substituted individually. However, CSCW literature does

not provide evidence that such substitutions are actually desired by groupware service users.

6.5.2 Relations with other GSM types

In the CooPS groupware reference model the CM-GSM type has, on a service level, relations with other GSM types. Three types of relations between GSM types exist:

1. A GSME type allocated to one GSM type may *enable* a GSME type allocated to another GSM type.
2. A GSME type allocated to one GSM type may *disable* a GSME type allocated to another GSM type.
3. A GSME type allocated to one GSM type may *change status information* that is used by a GSME type allocated to another GSM type.

Figure 6-4 summarizes these relations: the connected triangles between GSM types indicate the existence of relations between GSME types allocated to these GSM types.

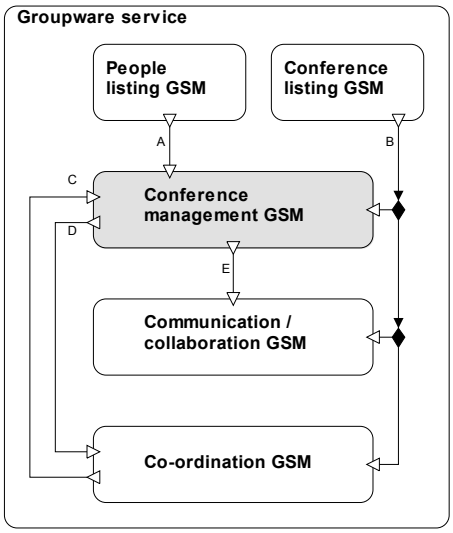


Figure 6-4 Relations between a CM-GSM and other GSM types. The lines may indicate multiple causality relations between GSM types.

The following table states the details regarding the relations between the CM-GSM type and other GSM types, grouped according to the letters in *Figure 6-4*. As an example, the syncConfState interactions described as part of the Invite and Join behaviour cause a service-level relation between the CM-GSM type, the CC-GSM type and the CO-GSM type: this abstract interaction denotes a series of interactions in which a groupware service user is updated regarding the current state of the conference. Since the state of the conference is also determined by the state of the CC-GSMs and

the CO-GSM that may be active in the conference, the state of these GSMs has to be communicated to the new participant. The feedthrough interactions that are part of the AssignRole, SetAccessPermission and UseTool behaviour can be applied for this purpose. As a result, there exists a service level relation between the CM-GSM, which provides the Join and Invite behaviour, and the CO-GSM, which provides the AssignRole and SetAccessPermission behaviour, and the CC-GSM, which provides the UseTool behaviour.

Table 6-2 Relations between a CM-GSM and other GSM types

	source GSME type [allocated to]	relation	affected GSME type [allocated to]
A	Select&ActivateConfMgt (cm) [PLIST-GSM]	enables	StartConf [**CM-GSM cm**]
B	Join [CLIST-GSM]	enables	Leave, EndConf [**CM-GSM**]; <Co-operate> [**CM-GSM**, CC-GSM, CO-GSM]
C	SetAccessPermission [CO-GSM]	enables or disables	<All co-ordinated GSMEs> [**CM-GSM**, CC-GSM]
D	AddTool (t) [**CM-GSM**]	enables	SetAccessPermission (for UseTool subtypes, provided by CC-GSM t) [CO-GSM]
	RemoveTool (t) [**CM-GSM**]	disables	SetAccessPermission (for UseTool subtypes, provided by CC-GSM t) [CO-GSM]
	AddCoord (co) [**CM-GSM**]	enables	GetRole, AssignRole, SetAccessPermission (CheckEnable) [CO-GSM co]
	RemoveCoord (co) [**CM-GSM**]	disables	GetRole, AssignRole, SetAccessPermission (CheckEnable) [CO-GSM co]
	Join, Leave, Invite, Expel [**CM-GSM**]	change status info	AssignRole, SetAccessPermission [CO-GSM]
E	AddTool (t) [**CM-GSM**]	enables	UseTool (t) and any subtypes [CC-GSM]
	RemoveTool (t) [**CM-GSM**]	disables	UseTool (t) and any subtypes [CC-GSM]
	AddCoord (co) [**CM-GSM**]	enables	CheckEnable associated with UseTool (and any subtypes) [CC-GSM co]
	RemoveCoord (co) [**CM-GSM**]	disables	CheckEnable associated with UseTool (and any subtypes) [CC-GSM co]
	Join, Leave, Invite, Expel [**CM-GSM**]	change status info	UseTool (and any subtypes) [CC-GSM]

6.5.3 Relations between GSMs associated with one conference

A CM-GSM provides behaviour to change the groupware service associated with one conference: a user can add CC-GSMs and a CO-GSM to the groupware service. Conversely, the CM-GSM also provides behaviour to remove CC-GSMs and a CO-GSM from the groupware service.

The *AddTool GSME* provided by a CM-GSM allows users to add a specified CC-GSM to a groupware service. After successful completion of this operation, the specified CC-GSM can be used in the conference. This means that any UseTool GSMEs the tool provides, can now be used in a co-operative manner: actions on the tool by one user can result in feedback to that user and feedthrough to the other conference participants. Outside the context of a conference, a CC-GSM may provide UseTool GSMEs, but only in single-user mode: in that case, the actions by a user can only result in feedback to that user, not to any other users.

Conversely, groupware service users may apply the *RemoveTool GSME* to remove a specified CC-GSM from a groupware service. After successful completion of this operation, any UseTool GSMEs the specified tool provides can no longer be used in a co-operative manner. After removal from a conference, a CC-GSM may continue to provide UseTool GSMEs to users. *Figure 6-5* illustrates the relations between the AddTool, RemoveTool and UseTool GSMEs, for co-operative use of a CC-GSM. The behaviour to select an available CC-GSM has been omitted from the figure.

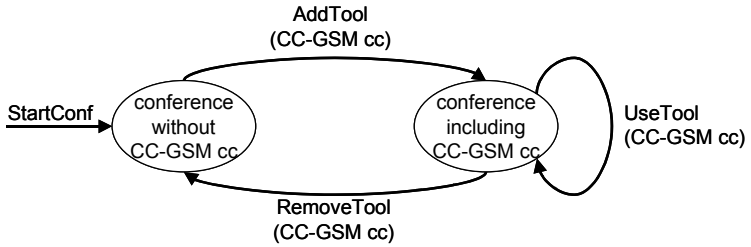


Figure 6-5 Relations between the AddTool, RemoveTool and UseTool GSMEs

The *AddCoord GSME* allows groupware service users to add a specified co-ordination GSM (CO-GSM) to a groupware service. After successful completion of this operation, the specified co-ordination GSM can be used in the conference. This means that the AssignRole, GetRole and SetAccessPermission GSMEs the co-ordinator provides, are now available to the groupware service users.

The co-ordination policy enacted by a co-ordination GSM can co-ordinate access to the behaviour provided by conference management GSMs and communication / collaboration GSMs. For instance, various GSME type descriptions in chapter 5 include an action *CheckEnable*. This action models that, if a co-ordination GSM is included in the groupware service, a query is performed to check whether the specified groupware service user is currently allowed to make use of the given GSME.

Groupware service users may apply the *RemoveCoord GSME* to remove a specified co-ordination GSM from a groupware service. After successful completion of this operation, the AssignRole, GetRole and SetAccessPermission GSMEs are no longer available to be used.

Additionally, since no co-ordination GSM is available to answer the *CheckEnable* queries, the groupware service users are not restricted by the groupware service provider to use any available GSME type. *Figure 6-6* illustrates these relations.

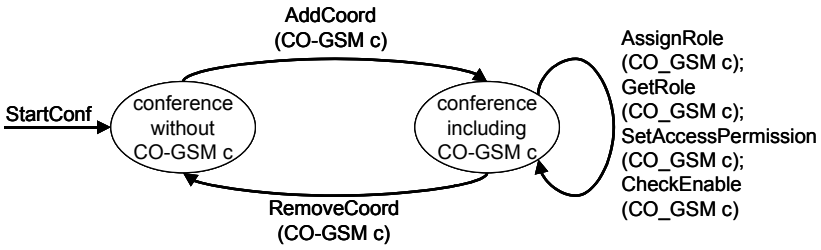


Figure 6-6 Relations between the AddCoord, RemoveCoord, AssignRole, GetRole and SetAccessPermission GSMEs

6.6 Communication / collaboration GSM (CC-GSM)

Primary purpose: a communication / collaboration GSM is primary responsible for providing groupware service users the behaviour associated with one communication tool or one tool to access shared information objects. Access to this behaviour may be subject to the rules defined by a co-ordination policy.

Groupware service users perform actions on tools in order to co-operate with the other people in a conference. Examples include users who apply an audio conferencing tool to speak with each other, and users who use a tool to share a document and be able to observe any changes that are made by other conference participants.

A tool may inform groupware service users about the results of their own actions, typically denoted as *feedback*. Similarly, tools may inform the *other* participants in a conference about the results of user actions, typically denoted as *feedthrough*.

A GSM composition associated with one conference may include multiple active CC-GSMs. Moreover, one conference may include multiple instances of the *same* CC-GSM: a conference may for instance include two instances of a shared whiteboard.

6.6.1 Associated GSME types

The CC-GSM type is a grouping of the following GSME types:

– *UseTool*. This abstract GSME type is the basis for all behaviour to support direct communication between conference participants and collaboration via shared information objects.

Individual CC-GSMs may provide multiple subtypes of the UseTool GSME: a tool to collaboratively edit a database may for instance provide separate functions to create, read, update or delete database entries. These separate functions are modelled as different subtypes of the UseTool GSME. This allows one to specify separate co-ordination rules for the different UseTool GSME subtypes: not every participant in the conference may be allowed to perform each function a CC-GSM provides. Conference participants with the role of guest may for instance be allowed to read database entries, but may not be allowed to delete database entries.

The set of active CC-GSMs in a conference and the set of UseTool GSME subtypes they provide determine the options to set access permissions, using the SetAccessPermission GSME. The CooPS groupware reference model does not specify how a CO-GSM determines what UseTool GSME subtypes a CC-GSM provides. This fact demonstrates that independent extensibility is a property that cannot be achieved only by adhering to the CooPS groupware reference model.

6.6.2 Relations with other GSM types

The behaviour provided by a CC-GSM is affected by several other GSM types. The following figure and table illustrate these relations.

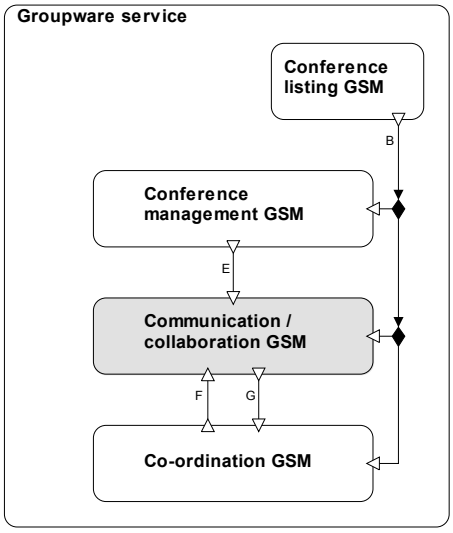


Figure 6-7 Relations between a CC-GSM and other GSM types

Table 6-3 Relations between a CC-GSM and other GSM types

	source GSME type [allocated to]	relation	affected GSME type [allocated to]
B	Join [CLIST-GSM]	enables	<Co-operate> [CM-GSM, **CC-GSM**, CO-GSM]
E	AddTool (t) [CM-GSM]	enables	UseTool (t) and any subtypes [**CC-GSM**]
	RemoveTool (t) [CM-GSM]	disables	UseTool (t) and any subtypes [**CC-GSM**]
	AddCoord (co) [CM-GSM]	enables	CheckEnable associated with UseTool (and any subtypes) [**CC-GSM co**]
	RemoveCoord (co) [CM-GSM]	disables	CheckEnable associated with UseTool (and any subtypes) [**CC-GSM co**]
	Join, Leave, Invite, Expel [CM-GSM]	change status info	UseTool (and any subtypes) [**CC-GSM**]
F	SetAccessPermission [CO-GSM]	enables or disables	<All co-ordinated GSMEs> [CM-GSM, **CC-GSM**]
G	The UseTool subtypes the **CC-GSM** provides	change status info	SetAccessPermission [CO-GSM]

6.7 Co-ordination GSM (CO-GSM)

Primary purpose: a co-ordination GSM is primary responsible for providing groupware service users the behaviour to define, and enact a co-ordination policy: it can be considered a co-ordination engine. A co-ordination policy associates users with access rights to GSMEs, for instance based on user roles.

Co-ordination GSMs are optional modules in a groupware composition: a valid composition of active GSMs does not need to include a co-ordination GSM (CO-GSM). In our design, a CO-GSM represents a generic co-ordination engine: active CO-GSMs may for instance provide a role-based access control mechanism or a workflow engine. As a basic co-ordination mechanism, the CooPS groupware reference model defines the behaviour a CO-GSM has to provide for role-based access control.

A GSM composition associated with one conference may include at most one active CO-GSM.

6.7.1 Associated GSME types

The CO-GSM type is a grouping of the following GSME types:

- *AssignRole*. This GSME type allows groupware service users to assign roles to individual participants in a conference;
- *GetRole*. This GSME type allows groupware service users to obtain the current role of a given participant in a conference;

– *SetAccessPermission*. This GSME type allows groupware service users to associate user roles and access permissions: it allows groupware service users to specify who is allowed to use what groupware behaviour.

The CheckEnable behaviour, described in section 5.5, is provided by the CO-GSM as part of a large range of other GSME types. Since the CheckEnable behaviour has no direct externally observable behaviour, i.e., it does not include interactions with a groupware service user, it is not regarded a GSME type.

6.7.2 Relations with other GSM types

A co-ordination policy defines rules for accessing GSME types. Such a co-ordination policy is configured and enacted by a co-ordination GSM.

Enabling a co-ordinated GSME involves enabling the associated interactions for a specified groupware service user. This behaviour is modelled as the CheckEnable action that enables an interaction with the groupware service user.

CO-GSMs may use the state of the conference to assess whether the specified groupware service user is allowed to use a given GSME. An example of this is a Group Decision Support System (GDSS) that distinguishes a brainstorming phase, a phase to cluster ideas, and a voting phase in a conference. Depending on the phase of the conference, users have access to specific GSMEs.

The state of a conference may be explicitly indicated by a groupware service user, or it may be derived from actions performed by groupware service users on the active GSMs associated with the conference. However, the CooPS groupware reference model does not specify any GSMEs to indicate the state of a conference, and it does not specify relations between GSM types to exchange conference status information. Again, this illustrates that the property of independent extensibility cannot be achieved only by adhering to the CooPS groupware reference model.

The reference model specifies the basic case: determining whether a specific user is allowed to access a given GSME. Groupware designers who wish to include more advanced co-ordination mechanisms may extend the design with additional GSMEs or additional relations between GSM types. However, a designer may not assume that another GSM includes other behaviour than specified in the CooPS groupware reference model.

These relations are summarized in the following figure and table.

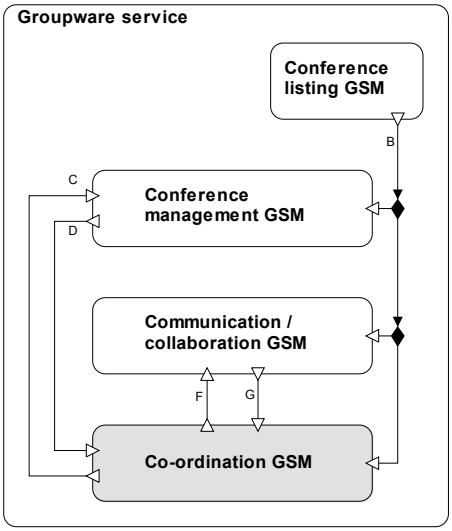


Figure 6-8 Relations between a CO-GSM and other GSM types

Table 6-4 Relations between a CO-GSM and other GSM types

	source GSME type [allocated to]	relation	affected GSME type [allocated to]
B	Join [CLIST-GSM]	enables	<Co-operate> [CM-GSM, CC-GSM, **CO-GSM**]
C	SetAccessPermission [**CO-GSM**]	enables or disables	<All co-ordinated GSMEs> [CM-GSM, CC-GSM]
D	AddTool (t) [CM-GSM]	enables	SetAccessPermission (for UseTool sub-types, provided by CC-GSM t) [**CO-GSM**]
	RemoveTool (t) [CM-GSM]	disables	SetAccessPermission (for UseTool sub-types, provided by CC-GSM t) [**CO-GSM**]
	AddCoord (c) [CM-GSM]	enables	GetRole, AssignRole, SetAccessPermission (CheckEnable) [**CO-GSM c**]
	RemoveCoord (c) [CM-GSM]	disables	GetRole, AssignRole, SetAccessPermission (CheckEnable) [**CO-GSM c**]
	Join, Leave, Invite, Expel [CM-GSM]	change status info	AssignRole, SetAccessPermission [**CO-GSM**]
F	SetAccessPermission [**CO-GSM**]	enables or disables	<All co-ordinated GSMEs> [CM-GSM, CC-GSM]
G	The UseTool subtypes the CC-GSM provides	change status info	SetAccessPermission [**CO-GSM**]

6.8 People listing GSM (PLIST-GSM)

Primary purpose: people listing GSMs provide users awareness information regarding other people, to facilitate the process of inviting these people to a conference.

Active PLIST-GSMs are not associated with one specific conference: an active PLIST-GSM may be applied to obtain awareness information regarding people to invite to a conference, and it may be applied to invite people to multiple conferences. Similarly, multiple PLIST-GSMs may be active at the same time in one groupware service: employees may use a corporate directory of employees and a separate PLIST-GSM that informs them about the presence status of their friends.

6.8.1 Associated GSME types

The PLIST-GSM type is a grouping of the following GSME types:

- *SelectPerson*. This GSME type forms the key function of a PLIST-GSM: providing information regarding other people and allowing a user to select people from this list to invite to a conference;
- *EnterInfoPerson* (optional). This GSME type allows groupware service user to enter all information needed to invite a person. If a PLIST-GSM does not provide this GSME type, groupware service users can only select and invite people who are listed by the people listing GSM;
- *Select&ActivateConfMgt*. This GSME type is required to activate an appropriate CM-GSM in order to start a new conference with the selected person;
- *SelectAvailableTool*. This GSME type allows a user to specify the CC-GSMs to use in the new conference with the selected person;
- *SelectAvailableCoord* (optional). This GSME type allows a user to specify the CO-GSM to associate with the new conference. If a PLIST-GSM does not provide this GSME type, groupware service users cannot select a CO-GSM to associate with the conference; hence, no co-ordination policy can be enacted during the conference;
- *UpdatePersonalInfo* (optional). This GSME allows a groupware service user for instance to update his presence awareness information, to inform others about his availability for communication.

6.8.2 Relations with other GSM types

The PLIST-GSM type is applied in the CooPS groupware reference model to model behaviour to obtain information about people and subsequently invite them to a new or an existing conference. The resulting relation is illustrated in the following figure and table.

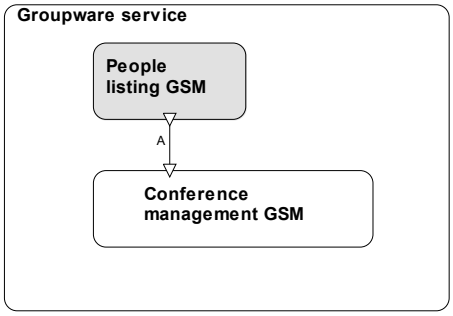


Figure 6-9 Relations between a PLIST-GSM and other GSM types

Table 6-5 Relations between a PLIST-GSM and other GSM types

	source GSME type [allocated to]	relation	affected GSME type [allocated to]
A	Select&ActivateConfMgt (cm) [PLIST-GSM]	enables	StartConf [CM-GSM cm]

6.9 Conference listing GSM (CLIST-GSM)

Primary purpose: conference listing GSMs provide users awareness information regarding other ongoing conferences, to facilitate the process of joining these conferences.

CLIST-GSMs allow groupware service users to select an ongoing conference or enter the information required to join an ongoing conference. Based on this information, an interoperable CM-GSM implementation is activated to join the specified conference.

Active CLIST-GSMs are not associated with one specific conference: an active CLIST-GSM may be applied to obtain awareness information regarding ongoing conferences and may be applied to join multiple conferences. Similarly, multiple CLIST-GSMs may be active at the same time in one groupware service.

6.9.1 Associated GSME types

The CLIST-GSM type is a grouping of the following GSME types:

– *SelectActiveConference*. This GSME type forms the key function of a CLIST-GSM: providing information regarding other conferences and allowing a user to select a conference from this list in order to send a request to join;
– *EnterInfoActiveConference* (optional). This GSME type allows users to enter all information needed to join a conference. If a CLIST-GSM does not provide this GSME type, groupware service users can only select and request to join conferences that are listed by the conference listing GSM;

– *Join*. This GSME type allows groupware service users to join the conference that was specified using the SelectActiveConference GSME or the EnterInfoActiveConference GSME.

6.9.2 Relations with other GSM types

In the CooPS groupware reference model, the CLIST-GSM type is applied to model behaviour to obtain information regarding ongoing conferences. Additionally, the CLIST-GSM type provides users the ability to join one of these conferences. These relations are illustrated in the following figure and table.

Figure 6-10 Relations between a CLIST-GSM and other GSM types

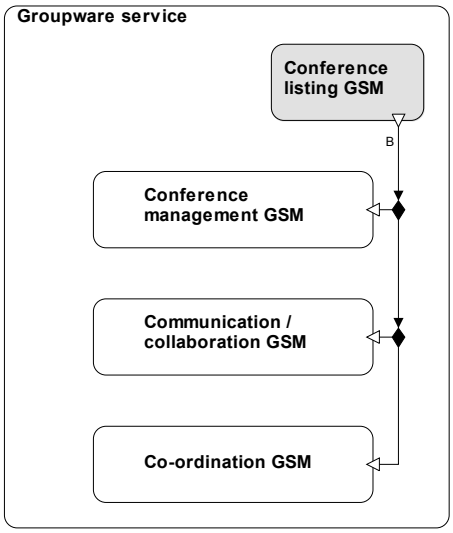


Table 6-6 Relations between a CLIST-GSM and other GSM types

	source GSME type [allocated to]	relation	affected GSME type [allocated to]
B	Join [**CLIST-GSM**]	enables	Leave, EndConf [CM-GSM]; <Co-operate> [CM-GSM, CC-GSM, CO-GSM]

6.10 Bootstrapping GSM (B-GSM)

Primary purpose: the bootstrapping GSM provides users behaviour to make GSMs available for activation, and to activate GSMs to start a conference, receive invitations and join existing conferences.

The B-GSM type provides users the behaviour to select and activate a specific conference management GSM, in order to start new conferences

and to receive and accept incoming invitations. The B-GSM type also provides behaviour to activate people listing GSMs and conference listing GSMs to facilitate the process of inviting people to a new conference and joining existing conferences, respectively. A third manner in which the B-GSM type supports starting new conferences is through the use of groupware templates. Such templates describe a composition of GSMs typically needed for a specific task. More information about groupware templates is given in section 8.6.

Finally, the B-GSM type provides behaviour to make GSMs available for activation: before groupware service users can make use of the behaviour provided by a GSM, that GSM has to be activated. However, only GSMs that have been made available to a groupware service user can be activated by that user. The B-GSM provides the behaviour to make specific GSMs available for activation, and to reverse that process.

The CooPS groupware reference model states that the behaviour provided by the B-GSM is always available to groupware service users. Based on the secondary importance of the behaviour provided by the B-GSM *during a conference*, it is omitted from *Figure 6-2*.

6.10.1 Associated GSME types

The B-GSM type is a grouping of the following GSME types:

- *(Un-) MakeAvailableGSM*. This GSME type allows groupware service users to make GSMs available for activation, and to reverse that process;
- *Select&ActivateConfMgt*. This GSME type allows groupware service users to select a specific conference management GSM and activate it. This behaviour is needed to start a new conference, without the use of a people listing GSM or a template;
- *SelectTemplate* (optional). This GSME type allows groupware service users to select a template and start a new conference based on this template;
- *(De-) ActivateEnabling* (optional). This GSME type allows groupware service users to activate people listing GSMs and conference listing GSMs to facilitate the process of inviting people to a conference and joining existing conferences, respectively;
- *UpdatePersonalInfo* (optional). This GSME type allows groupware service users to store information regarding himself or herself in the groupware service provider.

6.10.2 Relations with other GSM types

The function of the bootstrapping GSM type in the CooPS groupware reference model results in a number of relations with other GSM types, as illustrated in the following table.

Table 6-7 Relations between the B-GSM type and other GSM types

	source GSME type [allocated to]	relation	affected GSME type [allocated to]
	MakeAvailableGSM (CM-GSM cm) [**B-GSM**]	enables	Select&ActivateConfMgt (cm) [PLIST-GSM]
	MakeAvailableGSM (CC-GSM cc) [**B-GSM**]	enables	SelectAvailableTool (cc) [CM-GSM]
	MakeAvailableGSM (CO-GSM co) [**B-GSM**]	enables	SelectAvailableCoord (co) [CM-GSM]
	UnMakeAvailableGSM (CM-GSM cm) [**B-GSM**]	disables	Select&ActivateConfMgt (cm) [PLIST-GSM]
	UnMakeAvailableGSM (CC-GSM cc) [**B-GSM**]	disables	SelectAvailableTool (cc) [CM-GSM]
	UnMakeAvailableGSM (CO-GSM co) [**B-GSM**]	disables	SelectAvailableCoord (co) [CM-GSM]
	Select&ActivateConfMgt (CM-GSM cm) [**B-GSM**]	enables	StartConf [CM-GSM cm]
	SelectTemplate (template includes CM-GSM cm) [**B-GSM**]	enables	StartConf [CM-GSM cm]
	ActivateEnabling (PLIST pl) [**B-GSM**]	enables	EnterInfoPerson, SelectPerson, Select&ActivateConfMgt, SelectAvailableTool [PLIST-GSM pl]
	ActivateEnabling (CLIST cl) [**B-GSM**]	enables	EnterInfoActiveConference, SelectActiveConference [CLIST-GSM cl]
	DeActivateEnabling (PLIST pl) [**B-GSM**]	disables	EnterInfoPerson, SelectPerson, Select&ActivateConfMgt, SelectAvailableTool [PLIST-GSM pl]
	DeActivateEnabling (CLIST cl) [**B-GSM**]	disables	EnterInfoActiveConference, SelectActiveConference [CLIST-GSM cl]

6.11 Conclusions

The CooPS groupware reference model defines structuring guidelines for tailorable groupware services. It advocates to first capture on a service level the key properties and key relations of a groupware design. The reference model defines Groupware Service Module (GSM) types, and the relations between these GSM types on the service level. GSMs, created based on these types, form units of composition of groupware services: end users can select and compose GSMs to tailor the provided groupware service. The groupware reference model states how GSME types, described in chapter 5, have been grouped to form the various GSM types. The CooPS groupware reference model distinguishes six types of GSMs:

1. *Conference management GSMs*. This type of GSM is primary responsible for providing the behaviour to start and end conferences, change the set of conference participants, keep track of the set of tools that are active in

the conference and the co-ordination engine that is associated with the conference;

2. *Communication / collaboration GSMs*. This type of GSM is primary responsible for providing the behaviour associated with one communication tool or one tool to access shared information objects. Access to this behaviour may be subject to the rules defined by a co-ordination policy;
3. *Co-ordination GSMs*. This type of GSM is primary responsible for providing the behaviour to define, and enact a co-ordination policy. A co-ordination policy can for instance associate user roles with access rights to groupware behaviour: it defines what GSMEs users are allowed to use, given the role of the user and the state of the conference.;
4. *People listing GSMs*. This type of GSM provides users awareness about the availability of other people for co-operation. This awareness information is aimed to facilitate the process of inviting other people to a conference;
5. *Conference listing GSMs*. This type of GSM provides users awareness about other ongoing conferences. This awareness information is aimed to facilitate the process of joining other conferences;
6. *Bootstrapping GSMs*. This type of GSM provides users a means to make GSMs available for activation, and activate the GSMs needed to start a conference, receive invitations and join existing conferences.

6.11.1 Composition freedom

Figure 6-2 on page 145 depicts the various GSM types, i.e., the different types of service elements distinguished in the CooPS groupware reference model. Actual GSMs, based on these types, can be composed to form groupware services. The ER-diagram states the cardinalities of the relations between such modules. As such, the cardinalities indicate the composition freedom related to the CooPS groupware reference model: it shows for instance that only one active co-ordination GSM may be composed with one active conference management GSM. The diagram does not include the bootstrapping GSM: a groupware service always includes exactly one active bootstrapping GSM.

In the CooPS groupware reference model, one active conference management GSM is associated with one online conference. One active conference management GSM is associated with all conference participants: it provides a service to all participants in the conference. The provided *groupware service* is determined by the composition of active GSMs. During a conference, active GSMs are composed to provide a coherent service to the groupware service users. A composition of active GSMs *associated with one conference* consists of:

- One active conference management GSM;
- Zero or more active communication / collaboration GSMs. During a conference however, at least one communication / collaboration GSM should be active to support co-operation between the people in the conference;
- Zero or one active co-ordination GSMS.

A groupware service user may simultaneously participate in multiple conferences; as such, a groupware service may consist of multiple active CM-GSMs. In the CooPS groupware reference model, these active CM-GSMs are independent.

Apart from the GSM types mentioned before, one bootstrapping GSM is active, to allow groupware service users to make GSMs available for activation, and to allow them to activate the GSMs needed to start a conference, receive invitations and join existing conferences.

Finally, zero or more people listing GSMs as well as zero or more conference listing GSMs may be active. These GSMs facilitate the process of inviting people to conferences and the process of joining existing conferences, respectively. Individual people listing GSMs and conference listing GSMs are not associated with one specific conference: they can be applied to facilitate groupware service users in multiple conferences.

The provided groupware service, i.e., the externally observable behaviour of a groupware application is determined by the complete set of active GSMs.

Chapter 7

From service to implementation; a proof-of-concept

This chapter illustrates the steps needed to design and implement a groupware application based on the service-level specification of GSM types in the CooPS groupware reference model. The purpose of the demonstrator presented in this chapter is twofold: First, it demonstrates that the CooPS groupware reference model provides adequate support to implement tailorable groupware services. Second, it serves as an example of an application that provides a tailorable groupware service.

7.1 Groupware design based on the CooPS groupware reference model

The prime purpose of the CooPS groupware reference model is to provide insight: what functional modules are needed to compose groupware services and what are the relations between these modules? As such, the groupware reference model prescribes what abstract behaviour a GSM should provide, and the abstract interactions that should take place as part of this behaviour.

Since the CooPS groupware reference model is an abstract design, different implementations can be created based on this model. These implementations may vary significantly in terms of user interaction, the type of support they provide (in abstract terms, an audio conferencing service may provide the same behaviour as a text-based chat service) and various other functional and non-functional aspects. As such, the demonstrator presented in this chapter is one example of an implementation.

7.1.1 From reference model to implementation

The CooPS groupware reference model is an abstract design: it specifies on a service level the main units to form groupware services. To implement a concrete groupware application based on this model, one has to design and create software, i.e. one has to implement GSMs, while maintaining *conformance* with the reference model. A GSM conforms to the CooPS groupware reference model if it implements the abstract externally observable behaviour of at least one GSM type. This implies that a GSM at least implements the GSME types that are mandatory for that GSM type.

In the CooPS groupware reference model a single GSM, designed according to one of the GSM types it defines, may provide a service to multiple users. To implement such a GSM, a groupware designer may for instance create multiple software components that together provide the required behaviour. *Figure 7-1* illustrates this and shows that the various software components share information to provide the (global) behaviour associated with a GSM. This information exchange is denoted as *intra*-GSM information exchange.

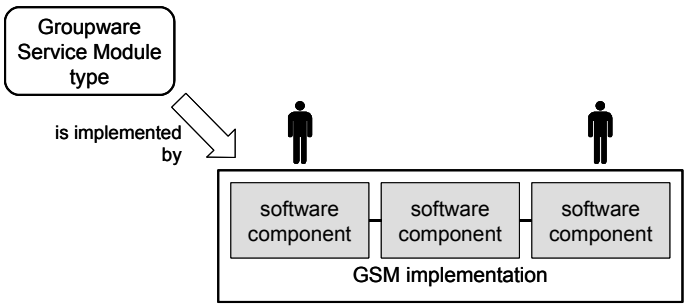


Figure 7-1 Possible implementation of a GSM type via software components

Groupware software component interfaces

In contrast, the CooPS groupware reference model also specifies that multiple GSMs may be composed to provide a groupware service to users. As such, software components that are associated with *different* GSMs may also have to exchange information. This information exchange is denoted as *inter*-GSM information exchange. *Figure 7-2* provides examples of both types of information exchange.

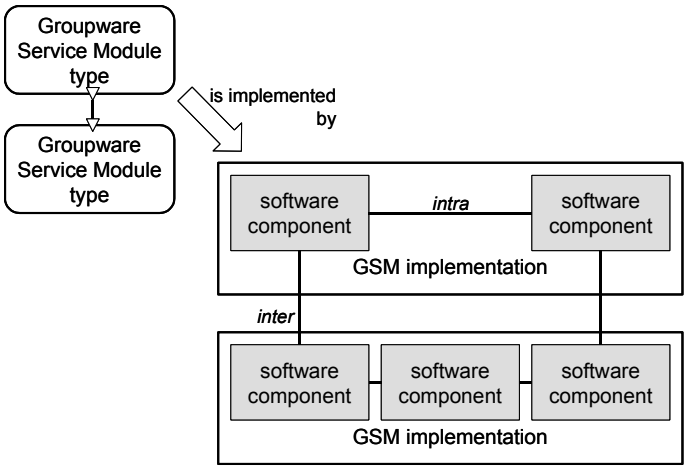


Figure 7-2 Example of intra- and inter-GSM information exchange

The CooPS groupware reference model does not specify how inter-GSM information exchange should take place. For this purpose, one has to add detail to the design. In a design based on software components, one may for instance specify software component interfaces for each GSM type. These software interfaces, together with a specification of the dynamic behaviour of the design, allow different design teams to create a series of GSMs that can be composed to provide a coherent groupware service.

A software component interface can be regarded a *contract* between software components: It states what the client needs to do to use the interface. Simultaneously, it states what the provider has to implement to meet the services promised by the interface. Component interfaces *encapsulate* the internals of software components: other software components can only access a component's behaviour through its interface. In this manner component interfaces shield the internal details of the component implementation. A software component interface specification still leaves a designer freedom to design and implement the internal structure of the software component.

Design decisions regarding groupware software component interfaces strongly influence the degree of composability, extensibility and thus the degree of tailorability of the resulting design. However, the design of such software interfaces depends on the technology applied, such as the selected component model. The CooPS groupware reference model is a service-level reference model: it does not prescribe concrete software interfaces for the various GSM types. So although our reference model stimulates a high degree of composability and extensibility by specifying on a service level the relations between GSM types, it cannot guarantee these desirable properties.

Nevertheless, the CooPS groupware reference model does provide a basis for design decisions regarding groupware software components in four ways:

1. *Initial structuring*. The reference model defines an initial structuring of software components, based on the identified GSM types. For instance, when designing a groupware application for one user one can identify a single software component for each GSM the participant should have access to.
2. *Behaviour allocation*. The reference model prescribes how groupware behaviour is allocated to the various GSM types. As a result, groupware designers know what behaviour the various software components must provide (i.e., the mandatory GSME types), and what behaviour they may optionally provide.
3. *Basis for software interfaces*. The relations between GSM types on a service level typically also result in relations on an implementation level:
   - Service level *enabling* and *disabling* relations between GSM types are an indicator for relations between GSMs: the fact that the source behaviour has been completed needs to be communicated to the affected behaviour.
   - Service level relations between GSM types that are the result of a *status information dependency* are an indicator for relations between GSMs: the fact that some status information has been changed needs to be communicated to the behaviour that uses this status information.
4. *Specifying dynamics*. The groupware service model, illustrated in *Figure 4-1*, *Figure 4-2* and *Figure 4-3*, helps groupware designers in specifying the dynamics of the design: the groupware service model states causal relations between groupware behaviour.

Guidelines for a reference model for groupware software components

Based on the CooPS groupware reference model, one can derive the following generic guidelines for a reference model for groupware software components:

- A reference model for groupware software components should distinguish component types that correspond to the GSM types distinguished in the CooPS groupware reference model. These component types should implement the behaviour that has been allocated to the associated GSM type. Such a component type can be constructed out of multiple sub-components.
- A groupware software component that implements a given GSME type, also has to implement the associated user interactions: for each interaction with a groupware service user in a GSME type specification

the software component has to implement one or more concrete interactions with a user.
- Each *enabling* relation (or *disabling* relation) between GSM types results in communication between software components that implement these GSM types, typically in the form of a method call or an event.
- Each relation between GSM types that is the result of a *status information* dependency results in communication between software components that implement these GSM types, typically in the form of a method call or an event.

Given a design where one GSM has been implemented by multiple groupware software components, one for each participant in a conference:
- Each *feedthrough* interaction in a GSME type description results in *intra*-GSM information exchange: communication between the software components that implement the behaviour of the GSM for the various conference participants.

7.1.2 A GSM and its environment

Concrete GSMs typically need some infrastructure to provide them a run-time environment and to connect them into a coherent application. On the service level, the behaviour provided by this infrastructure is of secondary importance: it does not provide behaviour that is directly observable by groupware service users.

In a concrete groupware application design based on the CooPS groupware reference model, each GSM is implemented by one or more software components. Additionally, the various participants in a conference may have their own replicas of a GSM. These GSMs may use the groupware infrastructure both for inter-GSM information exchange, as well as for intra-GSM information exchange.

Apart from providing a run-time environment and a basis for information exchange, the infrastructure can also provide generic behaviour, to be used by various GSMs. Examples of such generic behaviour include behaviour to secure communication, to manage quality of service levels for transport connections, generic behaviour to connect and disconnect software components, and behaviour to discover what GSMs are available.

Run-time discovery and composition of GSMs

One of the technical challenges related to implementing a groupware application based on the CooPS groupware reference model is the need for run-time discovery and composition of GSMs. To allow end users to select, activate and compose GSMs, the groupware application has to be able to

discover what GSMs are available. Subsequently, the groupware application has to be able to extend the current composition of active GSMs with the selected GSMs.

7.1.3 Physical distribution of GSMs

The CooPS groupware reference model specifies that, on the service level, a single GSM may provide a service to multiple groupware service users. For instance one CC-GSM may provide a shared whiteboard service to all participants in a conference. However, such a GSM may be implemented by multiple software components. Given the shared whiteboard example, the CC-GSM may be implemented in a way that each participant can view and manipulate the contents of the shared whiteboard, while the master copy of the content is stored on a central server, as illustrated in *Figure 7-3*.

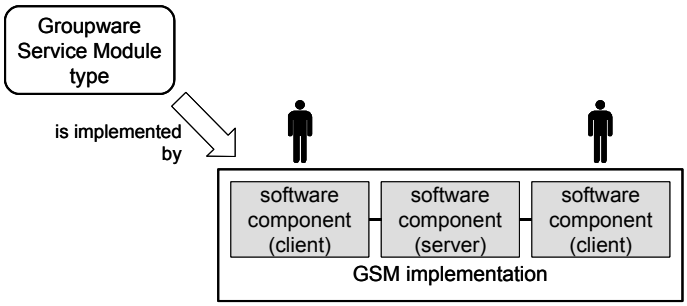


Figure 7-3 Physical distribution of groupware software components: client-server

On the other hand, it is also possible to implement the CC-GSM in a way that each conference participant has a local copy of the contents of the whiteboard, and that modifications are propagated to all other conference participants. This form of physical distribution, denoted as a peer-to-peer distribution, is illustrated in *Figure 7-4*.

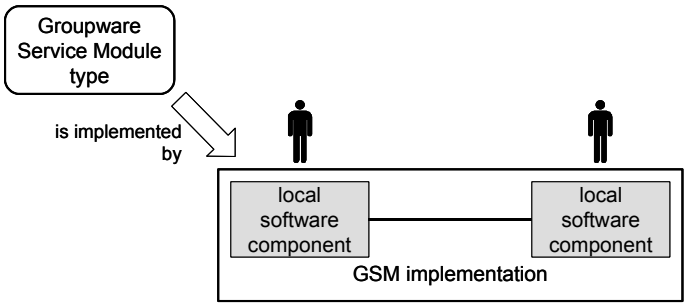
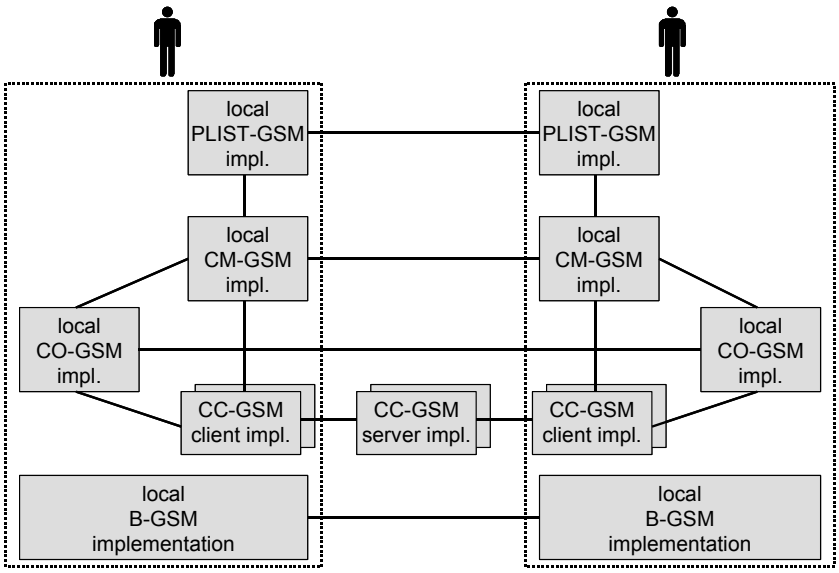


Figure 7-4 Physical distribution of groupware software components: peer to peer

In a groupware implementation, based on the CooPS groupware reference model, different GSMs may apply different physical distributions. *Figure 7-5* shows the physical distribution in a possible groupware implementation.

Each dotted rectangle denotes a part of the groupware implementation that is associated with a single participant in the conference. The horizontal lines indicate intra-GSM information exchanges, while the other lines indicate inter-GSM information exchanges.

Figure 7-5 Example of concrete physical distribution of GSMs



There are no absolute rules that indicate the optimal physical distribution of software components in a groupware application (Dewan, 1998). Design decisions regarding factors such as replication, concurrency and distribution determine important properties of a groupware design, such as scalability, efficiency, performance, and fault tolerance. It is beyond the scope of this research to fully describe the influence of all these factors on the implementation. Instead, we refer to Dewan (1998) for an overview of the advantages and disadvantages of different architectural approaches.

7.1.4 HCI issues

The CooPS groupware reference model describes, in abstract terms, the functional modules to compose groupware services. As such, it specifies in abstract terms the interface with the (human) groupware service users. However, the model does not prescribe design choices for the concrete user interface of a groupware application. The groupware reference model leaves much freedom, on the implementation level, to design the various interactions.

Take for instance the IndGetAvailTools interaction, which is part of the SelectAvailableTool GSME on page 99. In this interaction, the groupware service provider has to present the groupware service user the set of

available CC-GSMs and allow her to select one of them for inclusion in the conference. A concrete implementation of this interaction may represent the various CC-GSMs in a selectable list, or for instance as icons that can be dragged onto a graphical representation of the conference.

Although not meant as a HCI guideline, the grouping of groupware behaviour in GSM types provides a starting point for the design of the GUI of a groupware application: it illustrates a possible grouping of behaviour. Moreover, when the groupware functions provided by separate GSM types are separately presented in a user interface, the recognizability of these GSM types may increase. In turn, an increased recognizability of GSM types may increase the likelihood of successful selection and composition of GSMs.

7.1.5 Embedding in a groupware design approach

Groupware design approaches describe the process of capturing user requirements in a co-operative setting, and translating these requirements into a design that specifies the relevant organizational and technical aspects. Possibly, the approach also supports specifying the trajectory to come from the situation as-is to the situation to-be.

It is beyond the scope of this dissertation to fully describe groupware design approaches. Instead, we refer to two appropriate candidate design approaches: the *Seductive Design Approach* by Agostini et al. (2000) and the *Collaborative Design Approach* by Mulder and Slagter (2002). The latter design approach explicitly pays attention to group dynamics, and proposes tailoring as a mechanism to handle dynamics during the co-operation. The Collaborative Design Approach makes the design process a collaborative effort, in which prospective users, social experts and technical experts collaborate. The design approach helps such design teams to select and compose groupware modules that are appropriate for a given co-operative setting. The CooPS groupware reference model can well be applied to design such groupware modules.

Similarly, the CooPS groupware reference model is complementary to the groupware design methodology by Guareis de Farias (Guareis de Farias, 2002). He has proposed a methodology for the architectural design of component-based groupware systems that tackles the specification of a co-operative work process across multiple abstraction levels. The methodology distinguishes three different concern levels: the enterprise level, the system level, and the component level. At each of these levels, the methodology identifies different perspectives. For instance, at the system level it distinguishes the required service perspective, the decomposed required service perspective, and the internal integrated perspective. At each concern level and each perspective, the methodology defines three different views to

guide the specification of the different sets of concerns: the structural view, the behavioural view, and the interactional view.

The methodology of applying concern levels, perspectives and views helps a groupware designer to focus on the relevant set of concerns at each stage in the design process. As such, the methodology by Guareis de Farias can be applied to identify the relevant aspects for the design of individual GSMs as well as the overall groupware service.

7.2 Development context and design approach applied

A large part of the research in this dissertation has been conducted within the context of the GigaCSCW project: a knowledge acquisition project within GigaPort, the Dutch next generation Internet initiative. The GigaCSCW project focused on the social and technical aspects of supporting electronic teamwork. In particular, it involved research into the introduction of CSCW in organizations, group dynamics, professional and learning communities, and the design of flexible groupware.

The GigaCSCW project applied a multidisciplinary research approach in which social experts and technical experts co-operated on various topics. Based on this approach, user-centred research was conducted, while paying attention to innovative technology potential. The demonstrator, described in this chapter, is one of the results of this project.

The purpose of the demonstrator is to proof the presented concepts: the software was not designed for a specific real life co-operative setting. Consequently, some shortcuts were taken during the design process: mainly since end users only had a limited role in the design process.

To design and implement the demonstrator, an *iterative design approach* has been applied, with multiple design cycles. During these cycles the design was improved and refined based on expert evaluations of the intermediate products. We obtained feedback on the intermediate products from the software developers that worked on the application, from external software developers, and from people we showed the software to at seminars and conferences.

7.3 Overview: The CoCoWare .NET platform

The CoCoWare .NET platform represents a possible implementation that is based on the CooPS groupware reference model. The CoCoWare .NET platform consists of a *groupware infrastructure*, denoted as the CoCoWare .NET framework, and a set of *groupware components* that

implement GSM types. The CoCoWare .NET platform uses the Microsoft .NET component model to implement software components.

The platform allows end users to select and compose at run-time the groupware behaviour they need. Additionally, the platform provides groupware developers a basis to quickly develop flexible groupware applications. To do so, it manages such aspects as run-time discovery, activation and composition of GSMs as well as storing user information and preferences.

Developers can create GSMs that determine the behaviour as well as the look-and-feel of the groupware application using one of the programming languages supported by the .NET framework. The notion of a groupware platform results from the vision to make simple things simple to program: by offering groupware developers a suitable basis for groupware development and providing solutions to recurring problems, a groupware platform helps developers focus on new and creative possibilities.

7.3.1 Technology applied

The technology selected to implement a groupware application determines some of the properties of the resulting application. For instance, when creating a detailed design, the selected component model prescribes how component interfaces are to be defined and how they software components are connected.

Given the purpose of the CoCoWare .NET platform, the technology applied has to satisfy the following requirements:

- The component model applied must allow for run-time composition of software components;
- An integrated development environment (IDE) should be available to facilitate the creation of software components based on the component model;
- The component model must not impose licensing restrictions on the use of the prototype by users or developers from industry;
- The component model should provide uncomplicated mechanisms for users to install and uninstall software components. This aspect improves the desirable property of allowing one to explore groupware behaviour (Wulf, 2000).

The Microsoft .NET framework fulfils these requirements, and has been selected as the component technology for our proof-of-concept demonstrator. As a consequence, the CoCoWare .NET design applies the concepts and mechanisms defined by the Microsoft .NET framework. For detailed information about the Microsoft .NET framework, see Microsoft corp. (2003) and Slagter et al. (2002).

7.4 CoCoWare .NET framework

The CoCoWare .NET framework provides a run-time environment for CoCoWare .NET components, acts as glue between the various groupware components and provides generic behaviour to these groupware components. The generic behaviour provided by the CoCoWare .NET framework encompasses:

- A component directory, for the discovery of available GSMs;
- Component composition, for run-time connection of activated GSMs;
- User information and preferences management, for storing and retrieving user information and user preferences.

7.4.1 Component directory

In the CoCoWare .NET framework, a component directory is applied to discover, at run-time, what CoCoWare .NET software components are available or have been activated. This component directory obtains its information from the Windows registry, where a list is maintained of all GSMs that are available for activation. This registry is updated when GSMs are made available for activation, and when this process is being reversed, as described in section 7.5.4.

In the CoCoWare .NET architecture, software components express their capabilities by providing specific component interfaces. Based on the component interfaces a software component provides, the component directory can derive the type of the software component, in terms of the GSM types of the CooPS groupware reference model.

The CoCoWare .NET component directory can be queried to obtain references to *available* CoCoWare .NET components as well as references to *activated* CoCoWare .NET software components. Parameters can be provided in the query to specify which types of GSMs the query should return, and whether it should return available GSMs, active GSMs, or both.

Based on the obtained references, it is possible to derive the name of the software component to show in a user interfaces, and a textual description of the purpose and behaviour of the software component to show in a user interface. These descriptions may help to convey the identity and the purpose of the software component in a tailoring interface. In the CoCoWare .NET architecture, these descriptions are stored as component attributes that can be read even if the software component has not been activated yet.

7.4.2 Component composition

In the CoCoWare .NET architecture, the framework is responsible for maintaining the associations between the active software components that

are associated with the same conference. Since the CoCoWare .NET architecture applies a peer-to-peer distribution model, each software component is associated with one participant in a conference: different participants in a conference use replicas of software components to access the behaviour provided. The requirements on associations between software components are derived from the CooPS groupware reference model:

- Each active conference management GSM represents one conference;
- A user can activate a conference management GSM, in this case via the Windows Start menu: the bootstrapping behaviour is provided via this menu;
- Associated to this conference management GSM are zero or more active communication / collaboration GSMs;
- Active communication / collaboration GSMs may exist outside the context of a conference: an active communication / collaboration GSM does not need to be composed with a conference management GSM;
- Each active tool is associated with zero or one active conference management GSMs;
- Associated to the conference management GSM are zero or more active co-ordination GSMs;
- Active co-ordination GSMs cannot exist outside the context of a conference: each active co-ordination GSM has to be composed with one active conference management GSM;
- An active co-ordination GSM is associated with all active communication / collaboration GSMs associated with the same conference. Note that the co-ordination policy enacted by the co-ordination GSM does not need to affect all active communication / collaboration GSMs.

Additionally, zero or more people listing GSMs may be active, as well as zero or more conference listing GSMs. These GSMs however, are not associated with a specific active conference management GSM.

In the CoCoWare .NET architecture the connection, also denoted as composition, of active components is done by the framework. Similarly, the framework is also responsible for removing active software components from the composition. This has the advantage that all bi-directional connections between components are properly set up and torn down again. Additionally, this mechanism has the advantage that the framework is a single source of information regarding the current associations of groupware components for a groupware service user. In the CoCoWare .NET architecture, the software components in a groupware composition receive an event when the composition changes. Based on this event, they may for instance update their representation of the current conference.

7.4.3 User information and preferences management

The CoCoWare .NET framework provides facilities to store, obtain and update information regarding the local user. Note that in the proof-of-concept demonstrator only one local user can be served by one instance of the CoCoWare .NET framework. The user information is stored in named profiles. The use of a set of profiles allows for different preferences when the user is at work, at home or mobile. A profile is identified by its name and contains a set of attributes and their values. Examples of attributes include the user's name to display, e-mail address, phone number, the preferred conference management GSM, and the default communication / collaboration GSM to launch upon starting a new conference with somebody.

The CoCoWare .NET framework provides methods to set these attribute values in all profiles associated with one user, or to update individual profiles. A complete list of pre-defined attributes to store user information and preferences is provided in Slagter et al. (2002).

7.5 CoCoWare .NET components

To demonstrate the possibilities for run-time adaptations to the provided groupware service a number of groupware components have been implemented according to the CooPS groupware reference model. This section introduces a selection of relevant groupware software components. The table below lists the parts of the CoCoWare .NET platform that have been implemented and a mapping onto the CooPS groupware reference model. Note that the CoCoWare .NET platform does not include co-ordination GSMs or conference listing GSMs.

Table 7-1 Parts of the CoCoWare .NET platform and mapping onto CooPS groupware reference model

CoCoWare .NET part	CooPS groupware reference model type
CoCoWare .NET framework	Bootstrapping GSM (+ infrastructure)
CoCoConf .NET	Conference management GSM
CoCoChat .NET	Communication / collaboration GSM
CoCoAudio .NET	Communication / collaboration GSM
CoCoPhone .NET	Communication / collaboration GSM
CoCoMobilePhone .NET	Communication / collaboration GSM
CoCoVideoSearch .NET	Communication / collaboration GSM
TestTool .NET	Communication / collaboration GSM
CoCoMessenger .NET	People listing GSM

7.5.1 Conference management GSM: CoCoConf .NET

For the proof-of-concept demonstrator only one conference management GSM has been implemented: CoCoConf .NET. This conference management GSM implementation, shown in *Figure 7-6*, provides the following GSME types:

- *StartConf*. Groupware service users can start a new conference by activating the CoCoConf .NET component. This results in a new conference with the initiator as the only participant;
- *EndConf*. Groupware service users can leave or end a conference by closing the CoCoConf .NET component. The conference is ended when a user performs this action while being the only conference participant;
- *GetConfInfo*. Groupware service users obtain the conference information through the title bar of the CoCoConf .NET GUI. This title bar contains a textual description of the purpose of the conference, if one has been provided;
- *ChangeConfInfo*. Groupware service users can update the description of the purpose of the conference via a menu option;
- *GetParticipants*. The CoCoConf .NET GUI displays a list of the current participants in the conference;
- *Leave*. Groupware service users can leave a conference by closing the CoCoConf .NET component. If one or more other people remain in the conference this action is interpreted as a leave operation;
- *Invite*. Groupware service users can invite additional people to a conference based on their computer name or IP address;
- *SelectAvailableTool*. The CoCoConf .NET GUI provides users a list of all tools that are available for association with the conference;
- *SelectActiveTool*. The CoCoConf .NET GUI indicates which tools are active in the conference. Checkmarks in front of the available tools indicate which tools are currently active in the conference;
- *AddTool*. Groupware service users can add available tools to a conference by checking the checkbox in front of the name of the tool on the

CoCoConf .NET GUI. A short textual description of the tool appears as a tooltip when the user selects a tool;
- *RemoveTool*. Groupware service users can remove a tool from a conference by unchecking the checkbox in front of the name of the tool;
- *EnterInfoPerson*. Groupware service users can enter the computer name or IP address of a person as a basic means to specify the person to invite to a conference.

Figure 7-6 Screenshot of the CoCoConf .NET implementation: providing support for dynamic conference participation and use of conference tools

The CoCoConf .NET implementation does not provide the following GSME types:
- *Join*, *EnterInfoActiveConference*. The CoCoConf .NET implementation does not support requesting to join an existing conference: groupware service users can only enter an existing conference upon invitation by one of the current conference participants. Based on the purpose of the proof-of-concept demonstrator, this feature was given a low priority;
- *Expel*. The CoCoConf .NET implementation does not support expelling participants from a conference. Based on the purpose of the proof-of-concept demonstrator, this feature was given a low priority;
- *GetActiveCoord*, *SelectAvailableCoord*, *AddCoord*, *RemoveCoord*. The CoCoConf .NET implementation does not support co-ordination: groupware service users cannot select an available co-ordination GSM for association with a conference. Even though these GSMEs help to illustrate the concepts of the CooPS groupware reference model, they were not implemented since run-time adaptation of the groupware service could already be demonstrated using CC-GSM composition.
- *SelectPerson*. The CoCoConf .NET implementation does not show groupware service users information about people to invite, obtained from active people listing GSMs. However, groupware service users can use any active people directory to trigger the CoCoConf .NET GSM to invite a specific person;
- *SelectActiveConference*. Since the Join GSME was not implemented, this GSME was also left out.

The decision not to implement these GSME types was made due to resource limitations and since the mechanism to tailor the provided groupware service at run-time could already be demonstrated using the other GSME types.

7.5.2 Communication / collaboration GSMs

To demonstrate the possibilities for run-time tailoring of the provided groupware service a set of CC-GSM implementations is required. This section states the most relevant CC-GSMs that were implemented according to the CooPS groupware reference model.

CoCoChat .NET

The CoCoChat .NET CC-GSM provides support for text-based communication: it allows the participants in a conference to communicate using text messages. The messages are displayed together with the name of the participant who made the contribution. CoCoChat .NET, shown in *Figure 7-7*, has been implemented using C#.

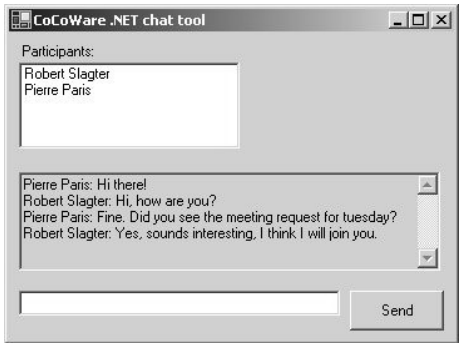


Figure 7-7 Screenshot of the CoCoChat .NET implementation

CoCoAudio .NET

The CoCoAudio .NET CC-GSM provides support for audio communication: it allows the participants in a conference to communicate using speech. The CoCoAudio .NET implementation is based on the MBone Robust Audio Tool (RAT)[15], an open-source audio conferencing and streaming application that allows users to participate in audio conferences over the internet. RAT can create direct connections between two participants, or multicast connections between groups of participants. The CoCoAudio .NET implementation has been developed to a beta stage. CoCoAudio .NET has been implemented using C#.

[15] For more information, see: http://www.netsys.com/mbone/software/rat/index.html

CoCoPhone .NET

CoCoPhone .NET is another CC-GSM that provides support for audio communication: it allows the participants in a conference to communicate using their TAPI desktop telephones. When the CoCoPhone .NET tool is added to a conference, it sets up a phone connection between the participants in the conference using TAPI commands. The phone numbers of the conference participants are obtained from their individual profiles. Depending on the services provided by the switchboard, it is also able to initiate multi-party calls when more than two people participate in the conference. CoCoPhone .NET has been implemented using C#.

CoCoMobilePhone .NET

CoCoMobilePhone .NET is another CC-GSM that provides support for audio communication: it allows the participants in a conference to communicate using their cellular phones. When the CoCoMobilePhone .NET tool is added to a conference, it sets up a phone connection between the participants in the conference using modem commands that are transmitted to the cellular phones using a bluetooth connection. The phone numbers of the conference participants are obtained from their individual profiles. Depending on the services provided by the GSM network operator, it is also able to initiate multi-party calls when more than two people participate in the conference. CoCoMobilePhone .NET has been implemented using C#.

CoCoVideoSearch .NET

The CoCoVideoSearch .NET CC-GSM provides support for co-operative searching through video databases. This CC-GSM has been implemented by an external software engineer, who had no access to the source code of the other CoCoWare .NET component implementations. Solely based on the provided documentation, base classes and interface descriptions this software engineer was able to create a valid CC-GSM implementation.

CoCoVideoSearch .NET is based on the VIP Full Client[16], an existing client-server application that provides facilities to search and retrieve full-screen, high-quality video content from an online video collection. CoCoVideoSearch .NET extends this application with collaborative behaviour: the obtained video content is shown to all conference participants. The participant who started sharing video content can start and stop the video stream or jump to another part of the video content. This participant can indicate on the CoCoVideoSearch .NET GUI whether the other conference participants are also allowed to perform these operations.

[16] For more information, see: http://vip.telin.nl

Figure 7-8 Screenshot of the CoCoVideoSearch .NET implementation

CoCoVideoSearch .NET, shown in *Figure 7-8*, has been implemented using Visual Basic .NET.

Testtool .NET

Testtool .NET acts as an example of how to create a CC-GSM implementation. It demonstrates software engineers how to make use of the CoCoWare .NET baseclass for a CC-GSM implementation, it illustrates what software interfaces and functions need to be implemented and how to obtain participation information from the associated conference management GSM. Testtool .NET has been implemented using C#.

7.5.3 People listing GSM: CoCoMessenger .NET

One people listing GSM has been implemented for the demonstrator: CoCoMessenger .NET. This GSM provides users presence awareness regarding a list of contacts: it shows whether these people are available for communication and allows users to invite these contacts to an online conference.

The implementation is based on the Microsoft Windows Messenger application: it obtains the list of contacts together with their presence information from Windows Messenger. For demonstration purposes CoCoMessenger .NET displays this information in a separate GUI that

allows groupware service users to invite people to a conference, using one of the available conference management GSMs and CC-GSMs.

7.5.4 Bootstrapping GSM

In CoCoWare .NET, the behaviour to select and activate a specific conference management GSM and to activate enabling GSMs is provided to groupware service users via the Windows Start menu. When a user selects a specific conference management GSM or an enabling GSM, the corresponding executable is activated.

The CoCoConf .NET conference management GSM can be activated in two modes: with or without an active GUI. When CoCoConf .NET is activated without an active GUI, it can be applied to receive and accept incoming invitations. When CoCoConf .NET is activated with an active GUI, the groupware service user can access all provided behaviour.

In CoCoWare .NET, making GSMs available for activation is done by executing Windows Installer packages. Executing these packages launches a wizard that helps groupware service users to make GSMs available for activation, and to reverse the process. The set of GSMs that is available for activation by a specific user on a given computer is stored in CoCoWare .NET in the Windows registry.

The current CoCoWare .NET implementation does not support conference templates or patches: no behaviour has been implemented to store templates or to start a new conference based on a template.

The current CoCoWare .NET implementation provides support for storing personal information and preferences. The implementation allows groupware service users to create multiple named profiles that include for instance information regarding the person's display name, telephone number, e-mail address, and preferred CC-GSM.

7.6 Tailorability provided by CoCoWare .NET

The CoCoWare .NET framework and components provide groupware service users the following possibilities to tailor the provided groupware service to their needs and personal preferences:

– *Tailoring the set of active communication / collaboration GSMs*. The CoCoConf .NET GUI shows groupware service users what CC-GSMs are available for inclusion in the conference, and what CC-GSMs are currently active in the conference. By checking and unchecking the checkboxes in front of the names of the CC-GSMs, groupware service users can add CC-GSMs to the groupware service composition and remove them from the groupware service composition. Upon adding a

CC-GSM to a conference, the associated behaviour becomes available for all conference participants: from that point in time, the conference participants can co-operate using the selected CC-GSM. To facilitate groupware service users in selecting appropriate CC-GSMs for their co-operative tasks, the CoCoConf .NET GUI presents a short textual description of the behaviour provided by each CC-GSM. The CoCoWare .NET CC-GSM interface specifies an attribute that contains this textual description. Upon removing a CC-GSM from a conference, the conference participants can no longer co-operate using the specified CC-GSM. Individual users may however continue using the removed CC-GSM in single-user mode.

- *Tailoring the set of active people listing GSMs*. In CoCoWare .NET, users can activate individual PLIST-GSMs directly from their Start menu. Upon installing a PLIST-GSM, the GSM implementation is responsible for placing a shortcut in the user's Start menu. An installed and activated PLIST-GSM is able to provide awareness information about other people, in order to facilitate the process of inviting these people to a conference.
- *Setting defaults*. The CoCoWare .NET platform allows groupware service users to set their default conference management GSM and default CC-GSMs as part of their profile. A PLIST-GSM may use these defaults for instance when a user invites another person: unless specified otherwise, the invitation is sent using the default conference management GSM. If the invitation is accepted, the default CC-GSM is added to the newly established conference.
- *Using different conference management implementations*. As described in section 4.4.5, different co-operative settings may require different conference management styles. The CoCoWare .NET platform supports different conference management styles by allowing groupware service users to choose from the available conference management implementations.
- *Extending the set of available GSMs.* The CoCoWare .NET framework allows groupware service users to install new CoCoWare .NET components. After installation, these components, i.e., GSM implementations, can directly be activated and used as part of the groupware service.

7.7 Conclusions

In this chapter we have shown the steps to map a service-level specification onto an implementation. The resulting proof-of-concept demonstrator, the CoCoWare .NET platform, allows end users to tailor the provided

groupware service, in the way envisioned in our research. The iterative process to design and implement this platform has led to various improvements of the underlying CooPS groupware reference model.

Apart from serving an academic purpose, the CoCoWare .NET platform also provides software engineers a solid basis to create flexible groupware applications. It provides them a framework and a number of software components for frequently needed groupware behaviour. In this way, the CoCoWare .NET platform allows software developers to concentrate on those functions that are important for their situation. The provided components are written in C# and Visual Basic .NET and use the .NET framework for inter-GSM information exchange. The requirements on the software components, in terms of the component interfaces they have to provide, are documented in Slagter et al. (2002).

More information about the proof-of-concept demonstrator can be obtained from http://www.cocoware.net. At this location, it is also possible to download the CoCoWare .NET framework and a selection of CoCoWare .NET components.

Chapter 8

Evaluation

This chapter evaluates of the CooPS groupware reference model in four ways:

1. The GSM types identified in the model are compared to the types of units tailors distinguish when describing a groupware application. As described in section 6.2.2, we assume that when GSM types are in line with the types of units tailors distinguish in a groupware application, the chances of successful tailoring will increase.
2. The model is evaluated using the Software Architecture Analysis Method (SAAM). This method uses scenarios to assess the extent to which an architecture supports a set of abstract criteria.
3. The *prescriptive* qualities of the CooPS groupware reference model are evaluated by creating a proof-of-concept demonstrator. This part of the evaluation is mainly described in chapter 7.
4. The *descriptive* qualities of the GSME types are evaluated by expressing the externally observable behaviour of two existing groupware applications in terms of the GSME types.

8.1 The types of units tailors distinguish

One of the important requirements on GSM types is that these types are in line with the types of units tailors distinguish in a groupware application. Section 6.2.2 assumes that when this is the case, the chances of successful tailoring will increase. In this context, successful tailoring is defined as selecting and composing groupware behaviour that is appropriate to support a given co-operative task.

Additionally, we assume the types of units tailors distinguish are in line with a given set of GSM types when tailors describe groupware applications in terms of units with similar behaviour as these GSM types.

In a design that allows end users to combine service elements, end users are likely to benefit when they recognize the service elements defined in the design. Nevertheless, existing service-oriented design methodologies typically do not evaluate whether this is the case. The following section describes how we applied a questionnaire as a lightweight means to explore the types of units tailors distinguish.

8.1.1 Evaluation approach

To perform the evaluation, we asked six test subjects to complete a questionnaire. As part of this questionnaire, three scenarios of concrete co-operative settings were provided to the test subjects. Based on each scenario the test subjects were asked to describe the units of a groupware application that would support them in the given scenario. The scenarios provided a task description and did not focus on the manner in which technology was applied to complete the task. The scenarios also did not provide hints regarding the units of a groupware application. The scenarios themselves have been included as Appendix B.

The scenarios were constructed to describe different co-operative settings, with different properties. The first scenario represents a basic scenario in which two people co-operate. They use information about the availability of people, so-called presence information, when establishing contact. Details about the way they communicate are left out on purpose. Instead, the scenario just states that one person "notifies" the other, and that they "discuss the most important changes [to the design]". The second scenario is more elaborate: three people communicate and discuss shared information objects. The third scenario describes an online conference in which many people communicate and share information objects. In this scenario, participants can have different roles in the conference.

To aid the test subjects, one example was provided: "being able to see which contacts are online" (translation of the Dutch equivalent). The subjects were told this was just an example, and that this unit did not need to be present in their description. However, all test subjects chose to include this example in their description for the first scenario. This example was chosen since it describes a groupware function at the border of the core functionality to actually co-operate. Moreover, the example represents a unit of reasonable granularity, without restricting the granularity of other units the test subjects identify.

The test subjects were selected to represent potential tailors: groupware users without particular programming skills that frequently use groupware applications. Their experience with various types of groupware applications is summarized in *Table 8-1*. The test subjects had no prior knowledge of the research and no prior knowledge of the CooPS groupware reference model.

Table 8-1 Details regarding the test subjects

Test subjects (n=6)	1	2	3	4	5	6
Age: (31 average)	24	23	27	29	34	49
Gender:	m	f	f	m	m	m
Working domain: (M=management/consultancy, H=healthcare, S=student)	S	H	H	M	M	H
Experience using chat (-=never, +=sometimes, ++=regularly)	++	++	++	-	++	++
Experience using shared workspaces	+	-	-	-	-	-
Experience using videoconferencing	+	+	-	++	++	++
Experience using weblogs or discussion websites	++	+	+	+	+	-
Experience using websites to share pictures	-	+	++	+	+	-
Experience using group decision support systems	+	-	-	-	-	-
Experience using online team agendas	++	++	++	++	++	-
Experience using applications to collaboratively edit a document or work on a design	+	++	++[17]	-	++	+
Experience using other groupware applications, namely:	n/a	n/a	n/a	n/a	n/a	n/a

8.1.2 Results

The six test subjects described for each of the three scenarios what units they distinguish in a groupware application in order to support the scenario. *Table 8-2* summarizes the results, as analyzed by the author.

Table 8-2 Results of the questionnaire

Subject	Scenario	Results
1	Overall	Describes concrete communication tools and concrete collaboration tools, sometimes including specific properties. Uses "being able to" to describe functions.
	1	Describes functions to communicate. Describes functions and tools to collaborate, and a function to see who else is online.
	2	Distinguishes an invite function coupled to a list of available people. Describes tools to communicate and functions to collaborate.
	3	Describes functions and tools to communicate, functions to collaborate, a function to see who is currently in the conference, and a function to co-ordinate who has the floor.
2	Overall	Describes concrete communication tools and concrete collaboration tools. Uses "being able to" to describe functions.
	1	Describes functions to communicate, functions to collaborate, and a function to see who else is online.
	2	Distinguishes functions and tools to communicate, functions and tools to collaborate, and a function to see who else is online.

[17] Collaboratively editing a document using Microsoft Word with the option to track changes switched on.

Subject	Scenario	Results
	3	Describes functions and tools to communicate, functions and tools to collaborate. Describes a function to see who is currently in the conference. Describes required peripheral equipment.
3	Overall	Uses "being able to" to describe functions.
	1	Distinguishes functions to communicate, functions to collaborate, and a function to see who else is online.
	2	Distinguishes a function to see who is currently in the conference, a function to invite people to the conference, functions to communicate, and functions to collaborate.
	3	Distinguishes functions to communicate and functions to collaborate.
4	Overall	Distinguishes parts of well-known applications. Sometimes uses "being able to" to describe functions.
	1	Describes parts of well-known applications to communicate and parts to collaborate. Describes a function to see who is currently in the conference.
	2	Describes parts of well-known applications to communicate and parts to collaborate. Describes a function to see who is currently in the conference. Describes a function to invite people.
	3	Describes parts of well-known applications to communicate and parts to collaborate. Describes a function to co-ordinate who has the floor.
5	Overall	Describes concrete communication tools and concrete collaboration tools, sometimes including specific properties. Explicitly mentions multi-party and Quality of Service requirements.
	1	Distinguishes functions and tools to communicate, functions and tools to collaborate, and a function to see who else is online. Describes required peripheral equipment.
	2	Distinguishes function to manage multi-user conferences. Describes functions and tools to communicate and functions and tools to collaborate. Distinguishes "a phonebook function" to invite other people.
	3	Distinguishes one tool to communicate.
6	Overall	Describes concrete communication tools and collaboration tools, sometimes including specific properties. Explicitly mentions practical issues, Quality of Service requirements and requirements on the environment.
	1	Describes functions and tools to communicate, functions and tools to collaborate, and a function to see who else is online and offline.
	2	Describes functions and tools to communicate and functions and tools to collaborate.
	3	Describes functions for co-ordination, functions and tools to communicate and tools to collaborate.

8.1.3 Discussion and conclusions

Given the small number of test subjects, it is not possible to generalize our findings. However, the results indicate some interesting similarities and reveal what units our test subjects distinguish when describing a groupware

application. The following bullets summarize observations based on the results.

- All test subjects distinguish multiple concrete communication & collaboration tools to support the scenarios, sometimes specifying specific properties of these tools. On average, 3.3 communication & collaboration tools were applied to support a single scenario.
- Four out of six test subjects use the Dutch equivalent of "being able to" to describe units of a groupware application.
- Three test subjects identify separate co-ordination functions.
- Five test subjects identify separate conference management functions, such as functions to invite additional people or to get an overview of the active conference participants.
- Two test subjects make a distinction between functions to manipulate data and to share that data with other people.
- Two test subjects, who frequently use video conferencing, explicitly mention Quality of Service (QoS) requirements on the groupware application.
- One test subject (with a management/consultancy background) expresses the groupware application in terms of units of existing applications.
- None of the test subjects distinguish a separate unit to tailor the behaviour of a groupware application.

These observations illustrate that our test subjects focus on describing *what the system should do for the participants*, rather than how the system should do this. This is in line with the service-oriented approach advocated in this dissertation: the methodology and CooPS groupware reference model focus on the service needed to support co-operating people.

These observations also seem to support the choice in the design for a tailoring mechanism that allows end users to select and compose *groupware service modules*, i.e., GSMs. Nevertheless, we do not have conclusive proof that this tailoring mechanism is an appropriate one: additional research is needed to compare various tailoring mechanisms.

All test subjects describe units that provide support for communication and units that provide support for collaboration: such units are modeled in the CooPS groupware reference model by the CC-GSM type. Additionally, all test subjects identify separate functions to see who else is online – the main purpose of the PLIST-GSM type. Five test subjects identify separate functions to manage conference participation – an important purpose of the CM-GSM type. Three test subjects identify separate functions to co-ordinate the co-operation – the main purpose of the CO-GSM type.

The test subjects did not describe units or functions that correspond to the CLIST-GSM type. An explanation for this may be that only the third

scenario includes a description that might have been supported by a unit of the CLIST-GSM type. Similarly, the test subjects did not describe units that correspond to the B-GSM type. Apart from the need to launch a groupware application, the scenarios did not describe situations were a unit of the B-GSM type is needed.

The fact that all test subjects identify separate functions to see who else is online may be biased by the provided example, which corresponds to such a function.

The fact that the test subjects did not distinguish a separate unit to tailor the behaviour of a groupware application does not imply that this function is not required: they all distinguished different communication & collaboration tools to support the three scenarios. One explanation for this may be that the test subjects regarded the option to select and compose appropriate communication & collaboration tools an inherent part of a groupware application.

8.2 Evaluation using the Software Architecture Analysis Method (SAAM)

The Software Architecture Analysis Method (SAAM) by Gregory Abowd (1998) is designed to help articulate the purposes of a *software architecture* and then determine the degree to which a given architecture meets them. In our research, the SAAM is applied to analyse the degree to which the CooPS groupware reference model meets the criteria formulated in section 3.7. The CooPS groupware reference model is a *service architecture* instead of a software architecture. It is also possible to apply SAAM to analyse a service architecture based on concrete scenarios that incorporate the various important criteria. However, one should realize that a SAAM analysis of a service architecture results in more generic outcomes than that of a software architecture, since additional design decisions are made in the process to create a software architecture based on a service architecture.

8.2.1 Stages in SAAM

SAAM distinguishes six stages in the analysis of software architectures:

1. Scenario development. In SAAM, scenarios should illustrate the kinds of activities the system must support and the kind of anticipated changes that will be made to the system. In the context of our research, this means that the set of scenarios should describe typical cases of co-operation via a groupware application as well as typical cases of groupware tailoring. Appendix A lists the set of scenarios applied.

2. Architecture description. The candidate architecture has to be described in an architectural notation that is well understood by the parties involved in the analysis. In the context of our research, these descriptions have to state the service modules that together provide the groupware service. The description has to state the behaviour provided by these service modules. Accompanying this static description of the architecture should be a more dynamic representation that specifies the order in which the behaviour can be used. Chapter 6 states the Groupware Service Module (GSM) types distinguished in our design, while chapter 5 describes the details of the Groupware Service Module Element (GSME) types: elementary units of groupware behaviour. The relations between GSME types are depicted in *Figure 4-1*, *Figure 4-2* and *Figure 4-3*.
3. Classification of scenarios. SAAM applies a walk-through simulation of each scenario in terms of the architectural constructs to determine whether it is a *direct scenario* or an *indirect scenario*. The direct scenarios are those scenarios that the architecture directly supports. Indirect scenarios require an adaptation of the architecture to successfully complete the walk-through. This adaptation could for instance be an adaptation to the behaviour provided, the addition of new behaviour to a GSM type, and the addition of a causal relation between GSM types. In section 8.2.3 the parts of the evaluation scenario are classified.
4. Individual evaluation of indirect scenarios. For each indirect scenario, the adaptations to the architecture that are necessary for it to support the scenario must be determined, as well as the complexity of these adaptations.
5. Assessment of scenario interaction. When two or more indirect scenarios require adaptations to a single GSM type, they are said to *interact* in that GSM type. The amount of scenario interaction is related to metrics such as structural complexity, coupling and cohesion (Abowd, 1998). High interaction among scenarios that are fundamentally different is an indicator for low cohesion and high structural complexity. High interaction among fundamentally similar scenarios is an indicator for high cohesion.
6. Overall evaluation. This last stage in a SAAM analysis allows for a comparison of alternative architectures, based on the relative importance of each scenario and scenario interaction. Since SAAM is, in this research, not applied to compare alternative architectures, this last stage has not been performed.

8.2.2 Regarding the selected scenarios

Scenarios of groupware use and tailoring have been applied to create the CooPS groupware reference model: they are a means to gather and articulate design requirements. To evaluate the CooPS groupware reference model, a different set of scenarios has been used. The evaluation scenarios are obtained from Tietze (2001). They consist of eleven related scenarios of groupware use and tailoring, denoted as ES1 to ES11, which are included in the next section for reference.

Appendix A states the other seven scenarios of groupware use and tailoring that were applied during the design of the CooPS groupware reference model: four scenarios that represent basic cases of groupware use and tailoring, denoted as BS1 to BS4, and three more comprehensive scenarios that reveal some of the complexity that may occur in real life settings, denoted as CS1 to CS3.

8.2.3 Classification of scenarios

As the third stage in the SAAM analysis, the eleven evaluation scenarios are classified as *direct scenarios* or *indirect scenarios*.

Introduction to the evaluation scenarios

A company with branches in several distributed locations uses the corporate network to deploy a collaboration environment providing the users with the ability to form distributed teams. Using the environment, the employees can conduct their activities, closely collaborating on common document bases with their team partners (including synchronous and asynchronous collaboration as well as multi-party multimedia conferencing), exchanging documents and document drafts and leaving notes and annotations for others. The collaboration environment accesses common databases used to persistently store the data objects on which the users are working. We will observe a number of different collaboration situations encountered by a distributed team of office workers, named Andrew, Barbara, Charles and David for easier reference.

Scenario ES1: Access to shared artifacts and collaborative tools.

After coming into the office in the morning, Andrew logs into the system at his workstation and is presented with his desktop environment, including:

- *a set of icons depicting shared document folders to which he has access and which he uses to collaborate with his various project partners,*
- *a calendar tool depicting his appointments,*
- *a mail in/out-box used for sending and receiving email messages,*

- *a set of tools which he can use to perform his tasks,*
- *a user list: a list of collaboration partners who are currently available.*

Andrew now begins working on his tasks. From the set of icons depicting shared document folders, he selects the folder for "Report 1" opens it. He is presented with a "Folder Browser" component showing the contents of this repository: A hierarchy of folders and sub-folders containing documents, annotations, etc. Using the component palette (or menu), he accesses a group calendar component and checks which appointments are entered for the groups of which he is a member.

SAAM analysis of ES1

Direct scenario. In scenario ES1, Andrew and his project partners can collaborate using documents that are accessible via a document sharing tool. In the CooPS groupware reference model, this can be modelled as a long conference, for the duration of the project, where the various project partners are the conference participants. In the conference, a CC-GSM is active to share documents. The calendar tool also be modelled as CC-GSM in this conference. To implement such CC-GSMs that support asynchronous forms of collaboration, a central server is typically applied. Additionally, the groupware service may be implemented in such a manner that the conference remains active, even when no people are logged in. The mail in/out-box can be modelled as a separate conference for each e-mail that is sent, where the e-mail function is provided by a CC-GSM. The various ways to model asynchronous forms of co-operation are discussed in section 8.3.4.

As such, the various tools Andrew can use to perform his tasks correspond to the notion of CC-GSMs in the CooPS groupware reference model. The user list behaviour is provided by a people listing GSM implementation.

Scenario ES2: Computer guidance in selecting appropriate tools

The goal of the work group of which Andrew is a part is to collaboratively write a complex technical report. Each member of the group is responsible for one chapter of the report and also contributes his specific expertise in certain technical areas to other group members when required. Andrew continues working on his chapter of the document. He has already written an initial version of the chapter that is available as an icon on his desktop. To do so, he uses a function of the desktop to indicate his desire to open the document. The system knows, which is the correct application for this kind of document and opens the document with the correct tool. Andrew now proceeds to work on the document.

SAAM analysis of ES2

Indirect scenario. The CooPS groupware reference model does not prescribe how to guide a user in selecting appropriate tools for a given type of shared information object. Instead, groupware implementations may for instance use the file type associations as maintained by the operating system.

Scenario ES3: Provision of Group Awareness

> *Barbara is not in her office, but instead uses her laptop computer and wireless network connection (through cellular phone) to access the system. She logs into the system and is also presented with a view of her collaboration desktop, similar to the one Andrew got after logging in. An icon representing Barbara now automatically appears on the user list on Andrew's desktop, indicating to him that Barbara is now available for collaboration. Likewise, an icon representing Andrew is displayed in the user list on Barbara's desktop.*
>
> *On her desktop, Barbara has a folder representing the group's technical report, containing icons for each chapter. Upon opening this folder, she sees a small nametag next to the chapter currently being edited by Andrew, indicating to her that Andrew is in fact currently working on his chapter.*

SAAM analysis of ES3

Direct scenario. In the CooPS groupware reference model, the information about who is online can be provided by the SelectPerson GSME, which is provided by people listing GSMs.

In order to be notified about the fact that Andrew is currently working on his chapter, the document sharing tool will need to register which groupware service user is currently accessing a document via the tool. In the CooPS groupware reference model, each interaction between a groupware service user and the groupware service provider takes place at a unique service access point. The information regarding the service access point is one means for the groupware service provider to establish which groupware service user has performed a given action. This information can be communicated to the other conference participants as part of the feedthrough information of the UseTool GSME.

Scenario ES4: Support for synchronous and asynchronous collaboration

> *Looking at the contents of one of the shared folders on his desktop, Andrew discovers that a chapter of the shared report has recently been edited. This chapter has been added to by Barbara while he was out of the office. Since it is relevant to the chapter he is writing, Andrew accesses the new chapter version and looks up a certain technical definition. Again,*

the system provides support by invoking the appropriate tool. Andrew is not satisfied with the explanation given in the chapter and can now do a number of things:

- *He can attach a note to the chapter and leave it for Barbara to see,*
- *he can directly contact Barbara to discuss the chapter's contents,*
- *he can directly rewrite the definition (potentially leaving a note about the changes he made), or*
- *he can contact Barbara using e-mail or the telephone to discuss more directly with her.*

Since the user list on his desktop shows the availability of Barbara, Andrew chooses to contact her in order to make the changes together. In addition to opening the appropriate communication channel(s), e.g., using a voice conference or simply making a telephone call, Andrew invites Barbara into the editing session in which he is currently viewing her chapter. Upon accepting the invitation, Barbara receives an editor tool that is shared with Andrew, the two can now co-operatively work on the document, making changes, discussing the contents, etc. Andrew also invites Barbara into the editing session on his own chapter in order to explain to her where the need for referring to her chapter arises. The group awareness information attached to the two chapters is updated to reflect that both Andrew and Barbara are now working on this chapter.

SAAM analysis of ES4

Direct scenario. The behaviour to share documents and obtain notifications about updates is modelled by the CC-GSM type in the CooPS groupware reference model. In order to maintain a central copy of the shared document, even when all people have left the conference, the CC-GSM implementation will need to include a central server. Such a central server is typically needed to implement asynchronous groupware services.

The behaviour to see which contacts are currently online is provided by the SelectPerson GSME, allocated to the PLIST-GSM. Andrew may for instance invite Barbara to a new conference directly from this PLIST-GSM, and select the voice conferencing tool and the already activated editing tool using the SelectAvailableTool GSME. As part of the StartConf GSME Barbara receives an invitation to join the conference.

Andrew includes two versions of the editing tool in the conference: one with his chapter and one with the chapter of Barbara. The CooPS groupware reference model allows multiple versions of the same CC-GSM to be concurrently active in a conference.

To update the awareness information, the document sharing tool will need to register which groupware service users are currently accessing

documents via the tool. As described with ES3 this is possible in CooPS groupware reference model.

Scenario ES5: Ubiquitous access to collaboration environment

After working with Barbara on the common problem for some time, Andrew needs more in-depth information, which is not accessible in his office, but for which he needs to go to the lab, which is also equipped with PCs. He therefore logs off from the office system, goes to the lab and logs in to the lab PC. After starting the general collaboration desktop application and identifying himself to the system, he is presented with his desktop contents as he left them in his office environment and can directly resume the collaboration sessions with Barbara. The two can now continue discussing and collaborating on the technical reports.

SAAM analysis of ES5

Direct scenario. In the CooPS groupware reference model, the information regarding groupware service users is stored by the groupware service provider. As a result, a groupware service user can access the behaviour provided by the groupware service provider, independent of the service access point applied.

Note that the CooPS groupware reference model does not include any specific interactions to log in, in order to establish and verify the identity of the groupware service user. In a groupware implementation according to the CooPS groupware reference model this behaviour can for instance be allocated to the bootstrapping GSM, and considered part of the trigger, indicated as the starting point in *Figure 4-1*.

Scenario ES6: Multiple simultaneous collaboration modes and transitions between them

While working together with Andrew, Barbara uses an editing tool in a separate session, currently visible only to herself, to take additional notes. She intends to incorporate these notes into her chapter at a later point in time, but does not yet wish to share the notes taken with Andrew. She opens an editing tool in a separate session and begins individual work in this tool, allowing her to concentrate on the technical details of a certain paragraph. Having completed the paragraph, she invites Andrew into her editing session in order to show him the result. Together, they now integrate the new paragraph into their common document.

SAAM analysis of ES6

Direct scenario. In scenario ES6, Barbara activates a second version of the editing tool. Later on, she decides to invite Andrew into her editing session. In the CooPS groupware reference model, this can be modelled as Barbara

activating a CC-GSM outside the context of a conference. Later on, Barbara invites Andrew using the SelectPerson GSME, selects the already activated CC-GSM as part of the SelectAvailableTool GSME, and starts a new conference with Andrew using the StartConf GSME. The CooPS groupware reference model allows groupware service users to participate in multiple, concurrent conferences.

Scenario ES7: Dynamic extensions of the collaboration environment

> *In the course of collaboratively working on the shared technical report, Andrew and Barbara need to incorporate more elaborate figures into the document. There is currently no suitable diagram editing tool available in the collaboration environment, but Andrew has recently downloaded one onto the lab PC. In order to use this tool in the collaborative work, Andrew loads the external tool into the collaboration environment. It automatically appears on all users' tool palettes and is directly available for collaborative use. It is not necessary for Andrew or Barbara to exit or restart the system in order for the new tool to be available to them.*
>
> *Barbara now creates a diagram as provided by the new diagram editing toolkit and introduces it into the common document. Using the new shared diagramming tool, the two can now jointly create the new figures in their report.*

SAAM analysis of ES7

Direct scenario. In scenario ES7, Andrew extends the set of available tools with a new diagram editing tool. In the CooPS groupware reference model this is modelled as the MakeAvailableGSM GSME: this GSME allows groupware service users to make additional GSMs available for activation in conferences. Although the CooPS groupware reference model does not guarantee that one groupware service user can make a GSM available for all other users in a conference, the MakeAvailableGSM GSME can be implemented with these semantics. After successfully completing the MakeAvailableGSM GSME, the new diagram editing tool is directly available as a CC-GSM for inclusion in conferences.

Scenario ES8: Coupling of different tools

> *In order to get an overview over their current document structure, Barbara opens a hierarchical outline viewing and editing tool on their report. She activates the tool in a separate, non-shared mode but uses it to access the report she is writing together with Andrew. The outline viewer therefore displays to her a graphical overview of the current structure of their report. While the two continue their work on the report, the overview display is continuously updated and synchronized.*

Using the hierarchy viewer, Barbara detects a problem with the chapter sequence of the report. She uses the hierarchy editing capability to fix this problem by rearranging the chapter sequence. Since she is working directly on the structure of their shared report, these changes are also immediately reflected in the tools she is sharing with Andrew. After discussing the changes she made, their joint work can continue.

SAAM analysis of ES8

Indirect scenario. The CooPS groupware reference model does not prescribe how different (local) CC-GSM implementations can be coupled: there is no direct relation specified on a service level between different CC-GSMs. Typically, coupling of different (local) CC-GSM implementations requires mutual knowledge about implementation details. When different CC-GSM implementations may originate from different manufacturers, it is unlikely that such knowledge is available.

It is possible to implement one CC-GSM that incorporates both the hierarchical outline viewing and editing functions, but that counters the main purpose of this scenario: namely to couple *different* tools.

Scenario ES9: Support for mobile work

Charles, a project manager and the third member of the distributed team, currently spends a lot of time on one of the company's campuses, walking between meetings, labs and different users' offices. In order to check on the progress of the report, he uses his hand-held wireless PDA (Personal Digital Assistant) to log into the collaborative system.

He is presented with a "stripped-down" version of the collaboration desktop, adapted to suit the restricted screen estate of the mobile device. Still, he has access to the same shared objects as Andrew and Barbara and can also perceive the group-awareness information about what is currently going on. He now uses an outline viewing tool on the common report in order to get an overview over the structure of the report and see what has changed since he last checked. He receives an indented table-of-contents view showing the current structure of the report. This structure display is a "live" display of report's table of contents. As the work on the report progresses (e.g., as additional subsections are added by Andrew and Barbara), Charles's view of the structure is always kept up-to-date.

SAAM analysis of ES9

Direct scenario. In scenario ES9, Charles accesses the behaviour provided by the collaboration desktop in a manner that suits the limitations of the mobile device he uses. The CooPS groupware reference model assumes that

all participants in a conference have access to the same conference management GSM, the same set of CC-GSMs and the same co-ordination GSM. But although the abstract interactions are the same, this does not imply that the various GSMs are presented to the conference participants in the same manner. In the CooPS groupware reference model it is well possible to implement a document editing CC-GSM that can present an elaborate GUI to display at a desktop as well as an elementary GUI for use on mobile devices.

Scenario ES10: Combination of existing tools (End-User customizability)

> *At several points throughout her collaboration sessions, Barbara observes the need for a number of users to combine textual and graphical data in their discussions. They often use a shared drawing tool (shared whiteboard) and a multi-user chat tool together in order to allow textual interaction (chat) to be enhanced by quick scribbles, notes and rough illustrations. The collaboration system provides individual tools for chatting and for drawing. The recurring act of setting up these two individual tools for a group of users is perceived to be arduous and repetitive.*
>
> *Using facilities provided to all end-users, Barbara sets up a new tool, incorporating the Chat and Whiteboard tools and deploys this new combination of components via the server. Now, whenever the need to do such communication arises, she only needs to open the one new tool and invite her co-workers into the collaboration session. All other users will automatically receive the new component configuration, which appears to them to be a single tool providing complex compound functionality.*

SAAM analysis of ES10

Indirect scenario. The CooPS groupware reference model does not allow groupware service users to combine existing CC-GSMs into a new type of CC-GSM, which subsequently appears as a single tool. When specific combinations of CC-GSMs are frequently applied, groupware service users can define a template for this combination. The SelectTemplate GSME allows groupware service users to apply such a template to start a new conference. However, the combination will not appear as a single tool providing complex compound functionality. As such, we derive that scenario is not directly supported.

In the CooPS groupware reference model, templates can also be applied to specify which conference management GSM and co-ordination GSM to use in the conference. Advanced templates may also include a list of people to invite to the conference, possibly including their roles.

Scenario ES11: High system performance

While Barbara and Andrew work together in the shared whiteboard and on their shared document, they are discussing the changes they are making to the document contents. Both expect the system to behave in a way reflecting the interactive and direct nature of their collaboration. They expect changes to the document to be reflected at their partner's site instantaneously and they expect the ability for their own work to proceed as naturally as possible throughout the collaboration.

Technically speaking, the users expect low latency, fast turn-around and expect the performance loss due to synchronization of their shared work to be as low as possible. Different collaboration settings have different requirements in terms of feedback and feedthrough. The highest demands are imposed in real time distributed collaboration settings. Here, the direct nature of the interaction between the participants requires fast system response. The better the feedback and feedthrough performance of the CSCW system is, the more direct and synchronous situations the tools built on the system can support.

SAAM analysis of ES11

Not applicable. The CooPS groupware reference model provides an abstract design, and as such it does not specify how to achieve high system performance. In a SAAM analysis of a service architecture such performance issues cannot be properly evaluated.

8.2.4 Individual evaluation of indirect scenarios

As indicated in the previous section, the CooPS groupware reference model does not directly support scenarios ES2, ES8 and ES10. The fact that the CooPS groupware reference model directly supports seven other scenarios is an indicator that the architecture is appropriate to support the presented cases of groupware use and tailoring, and the classes of similar situations they represent.

This section states the adaptations that are necessary in the CooPS groupware reference model for it to support these scenarios, as well as the complexity of these adaptations.

ES2: Computer guidance in selecting appropriate tools

The issue in scenario ES2 is that the CooPS groupware reference model does not include behaviour to activate the appropriate CC-GSM when the groupware service user selects a shared information object. Typically, the operating system maintains such associations between object types and applications to access the object. To accommodate scenario ES2, the

CooPS groupware reference model could be extended with a GSME type that allows groupware service users to influence the association between types of information objects and CC-GSMs. A valid implementation of this behaviour can be made relying on the facilities that modern operating systems typically provide for this. Such an adaptation impacts the bootstrapping GSM type, as this GSM type is responsible for providing the behaviour to make new GSMs available for activation. The complexity of this adaptation is limited: the addition does not influence other GSM types on a service level.

GSM involved: bootstrapping GSM
Complexity: low

Another manner in which groupware service users can be supported in selecting appropriate groupware behaviour is to present them information regarding the consequences, i.e., the impact, of activating or deactivating specific groupware behaviour. Such descriptions state the result of the operation in terms of the impact on the provided groupware behaviour and ultimately, the impact on the co-operation. The CoCoWare .NET demonstrator uses such a mechanism of textual descriptions to inform groupware service users during the process of selecting appropriate CC-GSMs during a conference.

ES8: Coupling of different tools

The issue in scenario ES8 is that the CooPS groupware reference model does not prescribe how different CC-GSM implementations can be coupled, for instance to synchronize their states. To accommodate the scenario a service-level relation would have to be introduced between CC-GSM types. This relation would have to specify how behaviour provided by one CC-GSM implementation affects the behaviour of other (local) CC-GSM implementations. Using such a relation, it is for instance possible to implement lip-synchronization between a local audio conferencing CC-GSM and a video conferencing CC-GSM or, as mentioned in scenario ES8, to synchronize an outline viewing tool and an editing tool.

GSM involved: communication / collaboration GSM
Complexity: high

As the examples above illustrate, one requires knowledge about the semantics of the synchronization information in order to couple different tools. Defining a standard to exchange information regarding the coupling of tools is very complex, especially if CC-GSM implementations may originate from different manufacturers.

ES10: Combination of existing tools (End-User customizability)

The issue in scenario ES10 is that the CooPS groupware reference model does not allow groupware service users to define new CC-GSMs, based on a combination of existing CC-GSMs. In order to accommodate this scenario, the bootstrapping GSM will have to provide behaviour to compose a new CC-GSM out of existing CC-GSMs, and to make this new GSM available for activation.

GSM involved: bootstrapping GSM
Complexity: medium

The complexity of this addition of behaviour is moderate: the other GSM types do not need to be adapted: given the conformance rules, a CC-GSM may be a façade for a composition of CC-GSMs.

8.2.5 Assessment of scenario interaction

The indirect scenarios ES2 and ES10 interact in the bootstrapping GSM: both scenarios require an adaptation of the bootstrapping GSM in order to accommodate the scenario. Since scenarios ES2 and ES10 represent semantically unrelated scenarios, this interaction is an indication that the bootstrapping GSM combines semantically unrelated functions.

The description of the purpose of the bootstrapping GSM, in section 6.10, states that it is applied to make new GSMs available for activation as well as to select and activate GSMs. Although these operations handle different stages in the lifecycle of a GSM, they are in fact not semantically unrelated. As such, the scenario interaction does not imply a design flaw.

8.2.6 Conclusion

The SAAM evaluation indicates that the CooPS groupware reference model is appropriate to support a wide range of cases of groupware use and tailoring. Three evaluation scenarios are not directly supported: ES2, ES8, and ES10. To accommodate these scenarios, the CooPS groupware reference model has to be adapted. The required adaptations range from adding behaviour to an existing GSM type (an adaptation of low to medium complexity) to adding a service-level relation between GSM types (an adaptation of high complexity).

The fact that only one case of scenario interaction occurs in this evaluation is an indicator that, in general, the CooPS groupware reference model exhibits the favourable properties of high cohesion, low coupling and a limited structural complexity.

8.3 Evaluation of the descriptive properties

The descriptive properties of the CooPS groupware reference model denote the degree to which the model supports people is expressing the service a groupware application provides. In this section, the service provided by the Groove Workspace application and Microsoft Windows Messenger is expressed in terms of the GSME types. These two applications have been chosen as they represent different families of groupware applications: while one is based on the notion of shared virtual workspaces where people typically co-operate in an asynchronous manner, using shared information

objects, the other is based on the notion of conferences where people make contact to communicate in a synchronous manner.

The CooPS groupware reference model is primarily a *prescriptive* model: it states how groupware services should be designed. The model includes causal relations. Since causal relations cannot be observed from external behaviour, a *descriptive* model should not include causal relations. Hence, to describe existing groupware services only the various GSME types are applied, not the causal relations between them.

8.3.1 Groove Workspace

The Groove Workspace application[18], from Groove Networks Inc., is a peer-to-peer groupware application to support small teams by means of shared virtual workspaces. The product applies a peer-to-peer approach, which minimizes the need for central servers and allocates as much functionality as possible at the systems of the individual users. This approach allows for more flexibility, as it reduced the dependency on other, economic, parties and as such empowers groupware service users to maintain autonomy and control over the behaviour of their groupware application.

The Groove Workspace is created out of software components. These components, based on Microsoft's Component Object Model specification, are connected by means of the Groove framework, which provides basic services to software components. Such basic services include the discovery of other groupware components, naming, and a service to transmit information, described in XML, to the other workspace members.

Groove distinguishes two main types of groupware components: those that form the Groove framework, which is denoted as the Groove transceiver, and those that represent tools. The Groove transceiver provides functions to manage workspaces: it allows users to start and end workspaces, invite people to a workspace and select a set of tools to use in a workspace. It is possible to assign user roles to workspace members, and assign access rights accordingly. Moreover, the Groove transceiver shows the user a list of his contacts, together with presence information. From this list, a user can start a chat session, modelled as a special type of workspace, or invite that user to a new or existing workspace.

The second type of Groove components represents tools. A tool can be used as part of a Groove workspace, and provides a means to communicate or collaborate using shared information objects. The Groove transceiver allows end users to modify the set of active tools in a workspace, both at the creation of a workspace and during cooperation. The set of tools to choose

[18] For more information, see http://www.groove.net

from is determined by the set of Groove tools that is available at the user's computer. This set can be extended by downloading Groove tools from the Internet and by accepting an invitation to join a workspace with Groove tools that are not yet available at the user's computer.

Groupware behaviour and mapping onto GSME types

Table 8-3 states the groupware behaviour provided by the Groove Workspace and how they can be expressed in terms of the GSME types. The described behaviour forms a subset of the complete behaviour that is provided by the application: only the externally observable behaviour that falls within the definition of a groupware service, as provided in section 4.5, qualifies as groupware behaviour.

Groupware behaviour may be composed of multiple GSMEs. A semicolon indicates a sequence: for instance, StartConf; EndConf denotes that, in our observations of the behaviour, the EndConf GSME occurred after the StartConf GSME had finished.

Table 8-3 Groove Workspace groupware behaviour and equivalent GSME types

Groove Workspace groupware behaviour	Equivalent GSME types
Launch the Groove application	Select&ActivateConfMgt; UpdatePersonalInfo (status -> online)
Shut down the Groove application	UpdatePersonalInfo (status -> offline)
Create a Shared Space (Select individual tools)	SelectAvailableTool; StartConf; AddTool (audio conferencing, chat, selected tools); AddCoord (role-based access control)
Create a Shared Space (Standard toolsets)	SelectTemplate; StartConf; AddTool (audio conferencing, chat, tools as specified in template); AddCoord (role-based access control)
Rename a Shared Space	ChangeConfInfo
Pause a Shared Space	<no equivalent GSMEs>
Delete a Shared Space	EndConf
Save as Template	<no equivalent GSMEs>
Invite: select a contact	SelectPerson; Invite
Invite: enter contact info	EnterPersonInfo; Invite
In the My Identity GUI: Edit my Identity	UpdatePersonalInfo
In the My Spaces GUI: View Spaces	GetConfInfo (for all current conferences)
In the My Spaces GUI: View Space Members & their online status	GetParticipants (incl. participant status information)
In the My Spaces GUI: Go to Space	UpdatePersonalInfo (status -> Active in selected Space)
In the My Contacts GUI: View Contacts	SelectPerson

Groove Workspace groupware behaviour	Equivalent GSME types
In the My Contacts GUI: Send Message	SelectPerson; SelectAvailableTool; StartConf; UseTool; EndConf (see the different ways to model asynchronous co-operation in section 8.3.4)
In the My Contacts GUI: Invite to Chat	SelectPerson; SelectAvailableTool; StartConf
In the My Contacts GUI: Launch Microsoft NetMeeting (for instance)	Select&ActivateConfMgt; SelectPerson; StartConf
In the My Contacts GUI: Invite to Space	SelectPerson; Invite
In a Shared Space: Invite: select a contact	SelectPerson; Invite
In a Shared Space: Invite: enter contact info	EnterPersonInfo; Invite
In a Shared Space: Uninvite	Expel;
In a Shared Space: View members & their online status	GetParticipants (incl. participant status information)
In a Shared Space: Hold-to-talk	UseTool (audio conferencing)
In a Shared Space: Send chat message	UseTool (chat tool)
In a Shared Space: Add tool	SelectAvailableTool; AddTool
In a Shared Space: Rename tool	<no equivalent GSMEs>
In a Shared Space: Delete tool	SelectActiveTool; RemoveTool
In a Shared Space: Roles	GetRole & AssignRole
In a Shared Space: Shared Space Permissions	SetAccessPermission
In a Shared Space: Tool specific Permissions	SetAccessPermission
In a Shared Space: Navigate Together	<no equivalent GSMEs>
In a Tool GUI: Action on a tool	UseTool
In the Add Tool GUI: More tools on Groove.net…	MakeAvailableGSM

Additional behaviour provided by the application

The Groove Workspace application provides mechanisms for secure communication, secure storage of workspaces, communication through firewalls, and maintaining replicated workspaces consistent. Although being important aspects of a groupware service, the CooPS groupware reference model does not prescribe how these aspects should be designed. Instead, the model stimulates groupware designers to first focus on the key functions and key relations in a groupware design.

As can be observed in *Table 8-3*, the Groove Workspace application offers groupware behaviour that cannot properly be expressed using the GSME types. This behaviour is described below, as well as the motivation why our design does not include GSME types to express it.

– *Pause a Shared Space*. Pausing a Groove space stops Groove from sharing updates with other participants. A Groove Space may be paused to delay sending or receiving a large amount of data. The GSMEs identified in our design only allow groupware service users to change the settings of

individual tools. Nevertheless, preventing sending or receiving a large amount of data typically involves only an individual tool. In those cases one does not need to pause an entire shared workspace.

- *Save as Template*. This option allows groupware service users to derive a template from the current conference and store this template. Although our design specifies that templates may be used to launch new conferences, the design does not specify how such templates are to be created. As described above, this behaviour is not considered to be core groupware functionality.
- *Rename tool*. This option allows groupware service users to assign a new name to a tool that is active in a Groove space. Even though this behaviour affects the other participants in a conference as well, we do not consider this behaviour to be relevant at a service level: groupware designers may choose a suitable mechanism to provide this behaviour. Similarly, the CooPS groupware reference model does not prescribe tool-specific details regarding the implementation of the UseTool behaviour.
- *Navigate together*. This option allows the members of a Groove space to move in unison throughout the tools in a shared space. Our design does not include GSME types to navigate to a specific CC-GSM. However, our design allows individual CC-GSMs to apply the UseTool GSME to share whether a given groupware service user has focussed on that CC-GSM.

Differences with the CooPS groupware reference model

The CooPS groupware reference model is prescriptive. As such, groupware applications that have not been developed based on this reference model may apply a different grouping of behaviour: although their behaviour can be expressed using GSME types, it cannot be expressed in terms of GSM types. The Groove Workspace application differs from the CooPS groupware reference model is the following ways:

- The co-ordination functions have not been clustered in a separate module. As a result, co-ordination of the use of Groove Workspaces can only be regulated using role-based access control. Tailors can for instance not exchange the default role-based co-ordination mechanism in a workspace for a workflow-based co-ordination mechanism;
- The Groove Workspace does not provide the functions associated in the CooPS groupware reference model with conference listing GSMs. In Groove, users can join a workspace only upon invitation.
- The Groove Workspace does not allow groupware service users to install or activate alternative units that provide conference management functions, similar to the CM-GSM in our model.

8.3.2 Microsoft Windows Messenger

The Microsoft Windows Messenger application[19] is primarily an application to obtain presence awareness about a set of contacts and to communicate with these contacts via text-based chat. Presence awareness encompasses awareness regarding the availability of a person for communication. In the Windows Messenger application the availability for communication of contacts is indicated by their status, which can be set to: *online*, *offline*, *busy*, *away*, *be right back*, *on the phone*, or *out to lunch*.

Although groupware service users can manually set their presence status, the application can also automatically update the user's status based on her actions. For instance, when a user launches the application her status is set to *online*. When a user does not use the mouse and keyboard for a given amount of time, the status can automatically be changed to *away*.

Based on this presence awareness, the Windows Messenger application offers groupware service users various means to co-operate with these contacts: when a person is offline it is only possible to contact that person by e-mail. On the other hand, a person who is online can be contacted using a wide range of tools: ranging from text-based chat, which is the default, to video conferencing. The application also supports audio conferencing, a shared whiteboard and application sharing. Other applications, such as games, may register themselves with the Windows Messenger application, after which they are included as a co-operation tool. As such, the set of tools to co-operate is extensible in the Windows Messenger application.

Groupware behaviour and mapping onto GSME types

Table 8-4 states the groupware behaviour provided by the Microsoft Windows Messenger application and how they can be expressed in terms of the GSME types.

Table 8-4 Windows Messenger groupware behaviour and equivalent GSME types

Windows Messenger groupware behaviour	Equivalent GSME types
Start the Windows Messenger application	Select&ActivateConfMgt
Change your own presence status, Automatically update presence status	UpdatePersonalInfo

[19] For more information, see http://messenger.microsoft.com

Windows Messenger groupware behaviour	Equivalent GSME types
Menu option: Send an Instant Message…, Start a Voice Conversation… , Start a Video Conversation…, Send a File or Photo…, Ask for Remote Assistance…, Start Application Sharing…, Start Whiteboard…, Start a Microsoft DirectPlay application	SelectAvailableTool; SelectPerson or EnterInfoPerson; StartConf
Menu option: Send E-mail…	SelectAvailableTool; SelectPerson or EnterInfoPerson; StartConf; UseTool; EndConf (see the different ways to model asynchronous co-operation in section 8.3.4)
Menu option: Start Groove… (for instance)	Select&ActivateConfMgt; SelectPerson or EnterInfoPerson; StartConf
Context menu @ a contact: Send an Instant Message, Start a Voice Conversation, Start a Video Conversation, Send a File or Photo…, Send E-mail, Ask for Remote Assistance, Start Application Sharing, Start Whiteboard, Start a Microsoft DirectPlay application	SelectPerson; SelectAvailableTool; StartConf
Context menu @ a contact: Start Groove	Select&ActivateConfMgt; SelectPerson; StartConf
Conversation window: Display list of conference participants in title bar and at top of the conversation window	GetParticipants
Conversation window: Send an Instant Message	UseTool
Conversation window: Invite Someone to this Conversation…	EnterInfoPerson or SelectPerson; Invite
Conversation window: Start Talking, Start Camera, Ask for Remote Assistance Start Application Sharing Start Whiteboard Start a Microsoft DirectPlay application	SelectAvailableTool; AddTool (since Windows Messenger currently only supports multi-party text-based chat, it was not possible to test whether a separate conference was created)
Separate GUI: Use Voice Conversation GUI, Use Video Conversation GUI, Use Remote Assistance GUI, Use Application Sharing GUI, Use Whiteboard GUI, Use a Microsoft DirectPlay application	UseTool

Windows Messenger groupware behaviour	Equivalent GSME types
Close the conversation window	Leave or EndConf (when the last participant closes the window, the conference is ended)
Select and close Voice Conversation GUI, Select and close Video Conversation GUI, Select and close Remote Assistance GUI, Select and close Application Sharing GUI, Select and close Whiteboard GUI, Select and close a Microsoft DirectPlay application	SelectActiveTool; RemoveTool (since Windows Messenger currently only supports multi-party text-based chat, it was not possible to test whether a separate conference was created)

Differences with the CooPS groupware reference model

The Microsoft Windows Messenger application focusses on providing groupware service users presence awareness regarding their contacts, and allowing groupware service users to communicate with their contacts, by default via text-based chat. The Windows Messenger application provides the behaviour associated in the CooPS groupware reference model with:

- a people listing GSM: the behaviour to obtain and represent presence awareness information;
- a conference management GSM: the behaviour to start and end conferences, and to invite people to a conference;
- communication / collaboration GSMs: the various means to communicate or work together via shared information objects;
- a bootstrapping GSM: the behaviour to launch the Windows Messenger application to distribute presence information and to handle incoming invitations.

The Windows Messenger application does not provide the behaviour associated with conference listing GSMs or co-ordination GSMs.

The Windows Messenger application does not separate the behaviour to obtain information regarding people to invite from conference management behaviour. As a result, groupware service users cannot use other people directory implementations to obtain presence information regarding their contacts. The benefits of combining this behaviour include an improved performance and a close tuning of capabilities and the representation of presence information.

8.3.3 Comparing the degree of tailorability

To compare an aspect of the externally observable behaviour of applications, their behaviour should be expressed in a common language. This section applies GSME types to express and compare the degree of tailorability of two existing groupware applications.

Provided tailoring GSME types

Both the Groove Workspace application and the Microsoft Windows Messenger application provide externally observable behaviour that can be expressed using the following tailoring GSME types: Select&ActivateConfMgt, AddTool, RemoveTool.

The Groove Workspace provides a co-ordination mechanism, which cannot be substituted. Users can assign roles and associated access permissions. The Microsoft Windows Messenger application does not provide a co-ordination mechanism: any co-ordination has to be implemented by the individual tools.

Both applications have a fixed set of people listing GSMs that can be used to invite additional people. Both applications do not include behaviour associated with conference listing GSMs.

The Groove Workspace application provides behaviour that allows users to make additional tools available for activation and inclusion in groupware services. As such, it provides the MakeAvailableGSM GSME type. The Windows Messenger application provides a more limited form of extensibility: external applications can register themselves as valid tools, for instance using the Microsoft DirectPlay technology. After completing this procedure, these applications can be selected and included in conferences.

Conclusion

Both the Groove Workspace application and the Microsoft Windows Messenger application provide a similar degree of tailorability during online co-operation, but the Groove Workspace application provides a slightly higher degree of *extensibility*: the application itself provides behaviour to install new tools, and as such allows end users to extend the groupware behaviour to choose from.

8.3.4 Describing asynchronous forms of co-operation

Asynchronous forms of co-operation, such as using e-mail, can be modelled in two ways using the GSME types: First, one can model asynchronous co-operation as one long conference that spans the period of time in which the groupware service users interact with each other. In this model, multiple e-mail messages between conference participants can be sent in one conference. Second, one can model each interaction between conference participants as a separate conference. In the example of e-mail messages, this corresponds to creating a separate conference for each e-mail message that is sent.

8.3.5 Conclusions

The GSME types described in chapter 5 provide a valid basis to describe and compare the externally observable behaviour provided by the Groove Workspace application and the Microsoft Windows Messenger application. Some behaviour provided by the Groove Workspace application cannot be fully expressed using the set of GSME types. As motivated in section 8.3.1, that behaviour is either considered to be outside the core behaviour of groupware, or the most relevant part of the behaviour can be described using the set of GSME types.

8.4 Evaluation of the prescriptive properties

The prescriptive properties of the CooPS groupware reference model denote the degree to which it supports programmers in creating groupware services that incorporate the previously identified favourable properties, outlined in section 3.7.

Ideally, one assesses this property based on a large range of implementations. However, in the context of our research only one implementation has been created based on the reference model.

During the design of the CoCoWare .NET platform the CooPS groupware reference model proved to be a helpful vehicle to communicate key properties of the design between designers and implementers. Moreover, as described in section 7.1.1, the CooPS groupware reference model provides adequate support for designing groupware software component interfaces, associated with the GSM types it defines.

8.4.1 Process outcomes and lessons learned

The process of creating a groupware design based on the CooPS groupware reference model, and subsequently implementing the CoCoWare .NET platform based on this design has yielded a number of results:

1. *Demonstration of prescriptive properties*. The process demonstrates the feasibility to create a detailed design and implement a groupware application based on the abstract CooPS groupware reference model.
2. *Proof-of-concept*. Another objective of the CoCoWare .NET platform is to demonstrate the concept of tailoring a groupware service. The implementation of the CoCoWare .NET platform allows us to test and demonstrate this aspect by selection and composition of functional modules.
3. *Independent extensibility*. The fact that a CC-GSM that was implemented by an external software engineer could successfully be composed with other groupware components demonstrates that the CoCoWare .NET

design allows for independent extensibility. The CoCoWare .NET design promotes composability and extensibility by considering software component interfaces as contracts: software components communicate via well-defined interfaces only, and the interfaces a software component provides states what other software components need to do to use the interface. The CoCoWare .NET software interfaces have been derived from the service-level relations that exist between GSM types.

The process of implementing the CoCoWare .NET platform has been performed in an iterative manner. Each iteration yielded improved insights, while some iterations even disclosed inconsistencies and shortcomings of the detailed design. Based on these findings the detailed design has been updated, typically resulting in updated software component interfaces for the various GSM types.

An example of a shortcoming in the original design was the fact that individual software components were responsible for setting up and tearing down the connections with other software components. However, many software components in our design have bi-directional connections. As a result, inconsistencies occurred when, for some reason, one component closed the connection, while the other component assumed it still existed. This issue has been resolved by making the groupware infrastructure responsible for setting up and tearing down such connections: this way, all associations between software components are managed from a single location.

During the implementation the programmers discovered that the original model of groupware behaviour lacked a description of how to receive and accept incoming invitations. Similarly, the original model did not include the bootstrapping behaviour, although that behaviour proved to be needed as part of the demonstrator.

8.5 Support for tailoring in the CooPS groupware reference model

The GSM types, defined in the CooPS groupware reference model, support tailoring in six ways:

1. During a conference, groupware service users can add additional communication / collaboration GSMs to a GSM composition. This allows groupware service users to extend the provided groupware service with the behaviour of the new CC-GSM. Similarly, by removing a CC-GSM from the groupware composition, groupware service users can remove the associated behaviour from the groupware service. The

AddTool GSME specifies that groupware service users can add both activated CC-GSMs as well as CC-GSMs that have not been activated yet. The option to add activated CC-GSMs allows users to share ongoing work, such as documents they are currently editing.

2. During a conference, groupware service users can add a co-ordination GSM to a GSM composition. In this manner, groupware service users can determine which co-ordination engine to associate with a conference. The selected co-ordination GSM may for instance be an engine for role-based access control, a workflow engine or an engine for any other type of co-ordination.
3. Groupware service users may activate one or more people listing GSMs. A people listing GSM provides information regarding other people, in order to facilitate the process of inviting people to a conference.
4. Groupware service users may activate one or more conference listing GSMs. A conference listing GSM provides information about other conferences, in order to facilitate the process of joining these conferences.
5. Groupware service users may select an appropriate conference management GSM to start new conferences. Aspects of a conference, such as the number of conference participants, the anticipated dynamics of conference membership, and the anticipated dynamics in the required groupware service determine what makes an effective, efficient and satisfactory conference management GSM for a given conference. Additionally, different conference management GSM implementations may apply different communication protocol standards for communication between conference participants. During a conference, the CooPS groupware reference model does not allow groupware service users to substitute the applied conference management GSM.
6. At any time, groupware service users may make additional GSMs available. After installation, these GSMs are available to be activated and used as part of a groupware service.

These six ways of supporting tailoring cover the anticipated dynamics in GSME use, as described in section 4.4.

8.6 How to present tailoring options in groupware

In any tailorable groupware service that allows end users to select and compose groupware behaviour, these users should have information about the tailoring options. The tailoring interface should provide answers to questions such as: *what* are the tailoring options: what groupware behaviour can be selected and composed, *how* to select and compose this behaviour

into a coherent groupware application, and what will be the *impact* of a tailoring operation in terms of the system functioning and on the co-operation itself (Slagter & Biemans, 2003).

On the groupware design level, this implies for instance that any GSM should be able to state, in a human-readable form, what behaviour it can provide to co-operating people.

8.6.1 Groupware patches

Corresponding to this, Slagter & Biemans (2003) propose to apply *task-oriented groupware patches*. Such patches are concrete, recognizable, domain-specific groupware templates for a given task. The concept of patches has been adopted from the Information Foraging Theory by Pirolli and Card (1999). A task-oriented groupware patch defines a set of GSMs that are typically needed to perform a specific task. When groupware patches are named after concrete and recognizable tasks, end users can select and compose groupware behaviour for a given task by selecting and activating the corresponding groupware patch.

A patch determines the set of GSMs that will be activated by default (denoted as *tailoring by selection*) and the GSMs that can optionally be added during a conference (denoted as *tailoring by variation*). Advanced patches can also define a default set of people to invite, specify user roles and associated access rights.

Groupware patches are domain-specific: it requires knowledge of a domain and the user tasks to create recognizable, appropriate patches. Since groupware patches reflect good practices, end users should be able to create new patches to cope with new co-operative tasks. Additionally, people who have specialized their groupware application by adding or fine-tuning GSMs should be able to store the new configuration as a patch. This allows users to create new patches for specialized tasks that require specific groupware behaviour. An example of a specialized patch is the *review stroke treatment* that has been derived from the generic *peer review* patch.

As such, groupware patches form valid starting points for groupware use, as well as for groupware tailoring.

8.7 Limitations of the CooPS groupware reference model

The presented methodology, including the CooPS groupware reference model, focusses on specifying *groupware behaviour* and mechanisms to tailor this behaviour. As a result, our work does not elaborate on some other aspects of a groupware design. Notably, the presented work does not provide details about:

- *User identification*. The reference model does not prescribe how users should be identified: it models abstract interactions between the groupware service provider and groupware service users. The model assumes that in a concrete secure groupware implementation, the identity of the groupware service user is established as part of these interactions.
- *Coupling of communication / collaboration GSMs*. The reference model does not specify how different CC-GSMs can be coupled, for instance to synchronize interactions. As such, the current design does not allow for lip-synchronization between a video conferencing GSM and a audio conferencing GSM that originate from different manufacturers. Defining such a coupling requires knowledge of the semantics of the information processed by individual CC-GSMs.
- *GUI composition*. The reference model does not specify how different GSM implementations may share one GUI: the service-level specification does not prescribe how GSMs should obtain information about, or even manipulate, the way other GSMs are presented to groupware service users.
- *Data transport services*. Given the distributed nature of groupware systems, data frequently needs to be transported between various geographically dispersed system parts. A data transport service offers a means to set up communication channels between various (remote) system parts and exchange data. An important aspect of data transport services is *scalability*: how is the performance of the service influenced when the data is to be sent to additional parties? In a groupware application that allows large groups of people to co-operate at the same time, scalability is crucial.
- *Persistence services*. In asynchronous settings, in which the various collaborating people are not necessarily active at the same time, persistent storage of contributions is frequently needed. The reference model does not prescribe how shared information should be stored in a persistent manner. As illustrated in section 7.1.3, individual CC-GSM implementations may for instance apply a client-server distribution as part of the solution to achieve persistency.
- *Object replication services*. When various participants collaborate using shared objects, it may frequently be needed to make consistent copies, replicas, of those shared objects. One of the main reasons for using local replicas is that it takes less time to receive feedback about operations on local objects, which may be important from a usability point-of-view.
- *Consistency management services*. When various participants have local replicas of a shared object and they perform operations on this shared object, the various copies have to stay consistent. Consistency management services try to maintain this consistency of replicated

objects, even if concurrent operations on the shared object by different participants are conflicting.
- *Encryption services*. When sensitive data needs to be communicated over insecure networks, such as the Internet, security by encryption plays an important role. Also when sensitive data is persistently stored security is important. Security in groupware has many aspects, including encryption of data, user authentication, but also examining the origin of downloaded groupware components, and checking whether downloaded groupware components behave correctly.
- *Fault tolerance*. Our methodology prescribes a groupware *service* design. Real life implementations may not provide this service since the implementation does not adhere to the specification or, for instance, since some network error occurs or a system stops responding. The design does not prescribe how concrete implementations should handle such faults, and minimize their influence on the system functioning as a whole.

8.8 Adherence to design criteria

The degree to which the CooPS groupware reference model adheres to the design criteria, described in section 3.7.4, is indicated by the various evaluation methods described in this chapter. Since this groupware reference model is an abstract design, it is not possible to determine strict values, denoting how well the design adheres to the individual abstract criteria. The SAAM analysis is targeted at reducing this issue, by analyzing a design based on concrete scenarios that reflect these criteria. Combining the results of the SAAM analysis and observations regarding the design, it is possible to derive *indications* of how well the design adheres to the various criteria.
- *Functionality*. The high percentage of scenarios that were directly supported during the SAAM analysis is an indicator that the design provides the functions for which it was intended. The three indirect scenarios reveal cases of groupware use and tailoring for which the design does not provide appropriate support. Similarly, this high percentage of direct scenarios is an indicator for a high degree of *effectiveness*. By shaping the design around the lifecycle of a conference and by providing alternatives to access the provided behaviour, the design aims to achieve a high degree of *efficiency*, at the expense of a reduced degree of parsimony.
- *Tailorability*. The presented design provides users a wide range of tailoring options, matching the anticipated dynamics of real life co-operation, as described in section 4.4. The fact that the design includes

GSMEs to change the composition of GSMs allows groupware service users to tailor the provided groupware behaviour.

- *Usability of tailoring*. Although we did not evaluate the usability of the selected tailoring mechanism in an experimental setting, there are indications that the selected tailoring mechanism is appropriate for tailors. The results of the questionnaire indicate that the types of units tailors distinguish in a groupware application correspond to our GSM types in terms of the functionality they provide as well as their granularity. Additionally, by defining only six types of GSMs, we strive for a low degree of complexity in the CooPS groupware reference model.
- *Conceptual consistency*. The presented design gives evidence of a high degree of consistency. For instance, groupware service users make use of similar behaviour in similar ways: all GSME types start with a *request* interaction, followed by an *indication* interaction, or, if another groupware service user responded, by a *response* interaction. Another example is the fact that associating communication / collaboration GSMs with a conference is performed in a similar manner as associating a co-ordination GSM with a conference.
- *Correctness and completeness*. The structured approach that has been applied as part of the methodology contributes to achieve correctness and completeness in the resulting design. However, no formal checks have been performed to verify the *correctness* of the model, for instance by checking the absence of deadlocks in the causal relations. This remains an opportunity for future work. Only the correct mapping of causal relations between GSME types onto relations between GSM types has been checked. An indicator of the *completeness* of the design is the fact that all identified GSME types have been allocated to at least one GSM type.
- *Orthogonality*. The design choices for groupware services, described in section 6.3, illustrate the degree of orthogonality of the design. These design choices have been derived from various design criteria, including orthogonality. Additionally, the small amount of scenario interaction during the SAAM analysis is also an indicator for a high degree of orthogonality in the design.
- *Propriety and parsimony*. To increase the degree of propriety, the presented design avoids unneeded restrictions regarding the order in which groupware service users access groupware behaviour. Additionally, the design only includes functionality that is essential given our definition of a groupware service. The small number of GSM types identified in the CooPS groupware reference model is an indicator for a high degree of parsimony. The fact that a small number of GSME types

may be provided by multiple GSM types reduces the parsimony of the design.

– *Interoperability within applications.* This form of interoperability is achieved in our design by specifying the relations between GSM types and describing the semantics of the exchanged information. The CooPS groupware reference model specifies the service-level relations between GSM types, while the descriptions of the individual GSME types state what the semantics of the information that is exchanged.

8.9 Relations with other groupware reference models

Existing groupware reference models are typically aimed to facilitate the groupware *implementation* process: they state how design decisions regarding the internal structure of groupware applications impact the behaviour of the application, for instance in terms of scalability and availability. Additionally, they may also state how these decisions affect more implementation-oriented aspects, such as the reusability of system parts. Instead, our reference model focusses on the *service* that is provided to co-operating people. The current section relates our methodology and groupware service reference model to three existing groupware reference models.

CoMeCo groupware service architecture

The CoMeCo groupware service architecture by Ter Hofte (1998) distinguishes three grouplet service types to compose a groupware service. The concept of grouplet services in CoMeCo corresponds to the concept of GSMs in our design: both are units of composition in a decomposed prescriptive functional model of groupware services.

The CooPS groupware reference model differs from the CoMeCo Groupware Service Architecture in four important aspects:

1. Our methodology to design tailorable groupware services departs from the individual *elementary units of groupware behaviour* (GSMEs) and clusters these to form *units of composition of groupware services*. The CoMeCo model identifies one abstract model for a collaborative interaction, and constructs a groupware service architecture based on this model;
2. Our methodology expresses groupware behaviour in terms of actions, interactions and causality relations, applying the AMBER language. The CoMeCo model applies natural language and a first-order predicate logic to express the behaviour associated with a groupware service;
3. Our methodology states the causal relations between GSME types, expressed using the AMBER language.
4. The CooPS groupware reference model explicitly distinguishes *enabling behaviour*: behaviour to make people aware of the availability of others for

communication and to make people aware of the existence of other conferences. The CoMeCo model includes the concept of superconferences and conference hierarchies to model the existence of enabling behaviour.

Given these differences, the CooPS groupware reference model and methodology extend and complement the CoMeCo Groupware Service Architecture and the corresponding methodology. The CooPS groupware reference model is a constructive model that explicitly states causal relations between units of groupware behaviour. The AMBER language allows one to express the actions and interactions that take place as part of groupware behaviour, as well as the associated causality relations. Even though we did not exploit the possibilities for functional analysis of the design[20], the AMBER language proved to be a suitable means to express the key concepts in our design.

Dewan's generic collaborative architecture

Dewan (1998) describes how collaborative applications can be characterised by the modules, functional layers, replicas, threads and processes into which the application is decomposed; the awareness in these components of collaborative functions; and the interaction among these components. In his research, Dewan explains how design choices regarding these components influence the function, fairness, fault tolerance, ease of modification, and performance of the application.

Dewan based his model on the notion of generalized editing: an application is considered an editor of semantic objects defined by it. As a consequence, he states that at an abstract level, any collaborative application can be considered a generalized editor (Dewan, 1998). The semantics of a collaborative application are divided into single-user semantics, which define the feedback users receive in response to commands entered by them or actions taken by the application autonomously (in response to internal state changes or messages from other applications); and collaboration semantics, which define the feedthrough users receive in response to commands entered by others.

Moreover, the model assumes that a user's input/output is processed by a hierarchy of layers. A lower-level layer, in his research denoting a layer closer to the user, manages objects that are interactors of objects in the immediately higher-level layer: the latter is an abstraction of the former. Layers communicate with each other using events.

---

[20] Testbed Studio, a tool to create AMBER models (see: http://www.bizzdesign.com/), for instance offers functional analysis options.

The generic architecture by Dewan specifies that a collaborative application can be expressed in terms of such layers, where some layers are shared while others are replicated, as shown in *Figure 2-7* on page 26. The replicas of a replicated layer are denoted as *private layers*: each replica processes the input and output for one participant in the session. A shared layer processes the input and output of the complete set of participants in a conference. An object in a private layer is private, while objects in a shared layer are shared by all conference participants. The private layers at a certain level are referred to as peers.

The architecture identifies a session manager as a special component, not included in the layered architecture. Instead, this external component manages the participation during a session. The component is therefore responsible for creating the tree of shared and private layers, and for connecting the private branches with the shared stem.

Dewan's generic architecture and associated terminology can be used for understanding, comparing and classifying existing groupware application designs. It can also be applied to design new applications. Dewan states that ideally one would like to identify universal principles that should be followed in the design of all groupware architectures. However, as he explains, there are no absolute rules in the design of these architectures. Instead, Dewan offers a set of qualified rules summarizing the advantages and disadvantages of different architectural approaches (Dewan, 1998).

Similar to our design, Dewan defines a decomposition of groupware applications. However, the decomposition by Dewan focusses on internal aspects: he defines a hierarchy of functional layers and describes the impact of design choices on aspects, such as concurrency and distribution. Instead, the CooPS groupware reference model is directed at capturing the key behaviour groupware applications should provide, including the foreseen dynamics in use of this behaviour. Subsequently, our reference model helps designers to define on a service level the units to form this groupware behaviour, as well as the relations between these units. As such, Dewan's generic architecture for collaborative applications can be considered complementary to our approach: Dewan's architecture is well suited for creating a detailed design of the individual GSMs, after they have been identified and described on a service level using the CooPS groupware reference model.

The Clover architecture

Similar to Dewan's generic architecture for collaborative applications, the Clover architecture by Laurillau & Nigay (2002) also focusses on the internal structure of groupware applications. It decomposes collaborative applications based on functional layers. The Clover architecture specifies at

each layer a partitioning into production, communication and co-ordination functions.

- The Clover production functions denote functions to access shared information objects, denoted in Clover as *caddies*. Access to caddies can be subject to access control, although the Clover architecture does not explicitly state where the functions for access control should be allocated;
- The Clover communication functions denote functions for direct communication between the participants in a conference;
- The Clover coordination functions denote all support for joining or leaving conferences, and to change the set of participants in a conference.

As Clover defines these types of functions on all functional layers, they cannot be mapped directly onto CooPS GSME types. Moreover, the Clover layers are layers in an implementation architecture, not a service architecture. Although Clover helps designers structure groupware implementations, it does not help in capturing key behaviour of such applications on a service level. Clover also does not describe how a service-level specification should be mapped onto the various layers.

Based on this we conclude that the Clover architectural model helps designers structure a groupware implementation. By distinguishing production, communication and coordination functions in each layer, system design issues within the software architecture are made explicit on a more fine-grained level than using Dewan's generic architecture. However, Clover does not help designers focus on the key aspect of groupware: what support has to be provided to the co-operating people. Such service-level specifications provide a "bigger picture": the frame around which the rest of the design is created.

8.10 Conclusions

We have shown that the CooPS groupware reference model conforms to the design criteria outlined in section 3.7.4 in four steps:

First, a questionnaire was applied to assess whether the GSM types identified in the CooPS groupware reference model are in line with the mental model tailors have of a groupware application. This method yielded valuable insights in the units people distinguish when describing a groupware application. In particular, the test subjects focus on what the various units of a groupware application should do, rather than how they should do it. Moreover, the types of units the test subjects described closely

resemble the GSM types identified in the CooPS groupware reference model.

Second, the presented design was evaluated using a SAAM analysis. This analysis revealed the extent to which the CooPS groupware reference model adheres to the previously described design criteria. During the SAAM analysis, a set of concrete scenarios of groupware use and tailoring were applied to evaluate the design. Combining the results of the SAAM analysis and observations regarding the design, it was possible to derive *indications* of how well the presented design adheres to the various design criteria. The SAAM analysis indicates that the CooPS groupware reference model is well suited to support a wide range of cases of groupware use and tailoring. The analysis revealed one location in the design where different concerns are not separated over multiple GSM types. However, the analysis indicates that, in general, the CooPS groupware reference model exhibits the favourable properties of high cohesion, low coupling and a limited structural complexity.

Third, the descriptive properties of the model were evaluated by expressing the externally observable behaviour of two existing groupware applications in terms of GSME types. This process demonstrates that the identified set of GSME types is sufficient to express and compare relevant aspects of the externally observable behaviour of groupware applications.

Finally, the prescriptive qualities of the CooPS groupware reference model were evaluated in chapter 7. That chapter describes the steps needed to derive a concrete groupware service implementation based on a service-level design. Section 8.4 evaluates this process.

Chapter 9

Conclusions

The basic motivation for our research is that the *behaviour* provided by groupware applications should seamlessly match the requirements of the co-operative setting they should support. The fact that real life co-operative settings are dynamic and have changing requirements has led us to investigate the following main research question:

> *How can groupware be designed in order to achieve tailorability of groupware services?*

9.1 Research results

To answer the main research question, we have created a structuring of groupware services. This structuring is provided for two target groups:

1. *Co-operating end users*. To allow co-operating end users to select and compose the groupware behaviour they require, coarse-grained Groupware Service Module (GSM) types are defined. GSM types structure a groupware service: GSMs, created based on these types, are units of groupware service composition. As the *service* a groupware system provides is the most relevant aspect for the users of that system, our approach offers tailoring options at the service level.
2. *Groupware designers*. To support groupware designers, coarse-grained Groupware Service Module (GSM) types are defined. Groupware designers can select and combine GSMs to match on the requirements of a specific co-operative setting. The GSM types as well as their relations on the service level are described in a groupware service reference model. The various GSM types are constructed by grouping more fine-grained Groupware Service Module Element (GSME) types. GSMEs are elementary units of groupware behaviour. As such, GSME types structure groupware services on a more fine-grained level. The

structuring on the level of GSME types helps groupware designers to map a service-level design onto an implementation by specifying, in abstract terms, the actions and interactions that take place as part of the GSME types, and the causal relations between GSME types.

The foundation for the structuring of groupware services is a generic model of the services a groupware application has to provide to co-operating end users. This model, described in chapter 4, forms the basis to describe the types of *elementary units of groupware behaviour*, denoted as Groupware Service Module Element (GSME) types. Chapter 4 also describes the answer to the second research question, as it describes the anticipated dynamics in use of groupware services.

An important aspect of our groupware design approach is the focus on *tailorability*: the co-operating people themselves are empowered to select and compose the groupware behaviour to match their dynamic requirements. For this purpose, the set of GSME types includes behaviour to change the composition of Groupware Service Modules (GSMs). GSMs form units of groupware service composition; changing the composition of GSMs results in changes to the provided service. This mechanism, which allows tailors to select and compose groupware behaviour on a service level, forms the answer to the third research question.

The CooPS groupware reference model, described in chapter 6, forms the answer to research questions 4a, 4b and 4c. The model describes GSM types, their responsibilities in terms of the GSME types they consist of and their relations on a service level. The design choices underlying the applied grouping of GSME types in this service reference model is described in detail in section 6.3.

9.1.1 Benefits for groupware designers

This dissertation advocates a *service-oriented approach* to groupware design. The reason for this is that the provided *service*, i.e., the externally observable behaviour, is considered to be the most important aspect of a groupware application for its users: the provided service determines whether a given groupware application properly supports a co-operative setting. How a service is *implemented* is typically less relevant for its users. Nevertheless, many existing groupware design methodologies focus on implementation aspects.

By focussing on the service level first, it is possible to first specify the key functions of a system and the key relationships. These key functions and relationships then form the boundaries for the rest of the design. By making the key functions and relationships explicit in the early stages of the design process, our methodology tries to prevent designers to delve into

implementation details while loosing track of the "bigger picture". Additionally, the methodology acknowledges the importance of properties such as conceptual consistency of the design, completeness, orthogonality, propriety and parsimony. Chapters 4 and 6 illustrate how these properties influence groupware design decisions.

One of the complicating factors in groupware design is the inherent dynamics of co-operation: designers are faced with the task to design systems that can cope with these dynamics and provide matching support for changing co-operative settings. The presented methodology illustrates how a service-oriented approach can be applied to design tailorable groupware services out of functional modules. In this manner, service elements can be selected and combined, depending on the requirements of the co-operative setting.

9.1.2 Benefits for co-operating end users

The presented methodology and service reference model are also beneficial for the co-operating end users themselves. Although they cannot guarantee that the resulting groupware application matches the users' requirements, it does stimulate designers to design groupware based on the service that has to be provided to co-operating end users.

Co-operation is inherently dynamic: as a result, co-operating end users should be supported by groupware that is flexible enough to cope with these dynamics. Chapter 2 illustrates that current groupware applications either lack the required flexibility or are difficult to tailor. The presented methodology and groupware reference model empower end users to select and compose the groupware service modules that match their changing needs and personal preferences: the CooPS groupware reference model explicitly includes groupware behaviour to tailor the provided groupware service. End users are provided with behaviour to select and compose *groupware service modules*, i.e., GSMs. Additionally, end users are provided with behaviour to make new service elements available for selection and composition. In this manner, the CooPS groupware reference model incorporates the property of composability as well as extensibility by end users, conform the definitions in section 3.2.6.

By allowing end users to select and compose groupware service modules the design aims for an intuitive appealing tailoring mechanism. The observations during the evaluation indicate that the test subjects indeed expressed the capabilities of a groupware application in terms of service elements: what the system should do for the participants. Moreover, the service elements our test subjects identified resemble the GSM types in our model in terms of the functions they provide and their granularity.

However, more research is needed to confirm that tailoring via service elements is an appropriate tailoring mechanism.

Finally, the presented groupware reference model stimulates conceptual consistency across the services provided by groupware applications that originate from different manufacturers. Such conceptual consistency makes it easier for groupware users to switch between different applications.

Informing end users about the service reference model

Although the CooPS groupware reference model provides a structuring of tailorable groupware services, this does not imply that tailors need to be informed about the details of the reference model. As described in section 8.6, tailors have to be informed about *what* can be tailored about a groupware service, *how* that can be done, and what the *impact* of a tailoring operation will be on the provided behaviour and the co-operation. A tailoring interface, i.e., a user interface that allows end users to perform tailoring operations, should convey this information. To do so, the tailoring interface may distinguish the various GSM types identified in the CooPS groupware reference model and allow the tailor to select and compose GSMs.

One possibility to denote in a tailoring interface how various types of groupware modules can be connected is to create graphical representations of the various GSMs that include connection points. For instance the tailorable search tool described by Wulf (1999a) applies coloured circles to denote various types of connections: when a groupware module representation includes a full red circle, this indicates it can be connected to modules that have an empty red circle. In this manner the tailor is informed about the various GSM types and the possibilities for composition, without explicitly introducing the underlying reference model.

9.2 Contributions to the state of the art

The research presented in this dissertation advances the state of the art in the following areas:

- *Structuring of groupware services*. The main contribution of our research is a structuring of groupware services, as described in section 9.1.
- *Methodology to design tailorable groupware services*. Another contribution of the research is a methodology to design tailorable groupware services: the methodology guides designers through the process of specifying the key groupware behaviour. Additionally, it helps designers to allocate this behaviour to coarse-grained units that can be composed to form groupware services.

- *Service-oriented design*. The methodology advocates a service-oriented design approach based on precisely defined architectural concepts. By focussing on the service level, the methodology helps designers to first concentrate on high-level, key functions of the design: the *behaviour* that has to be provided to the users of the system. Designers should first focus on these key functions and key relations before switching to other levels of abstraction, such as the implementation level. While the architectural concepts are applied to groupware design in this dissertation, the concepts are generic, and can be applied to design a wide range of interactive services.
- *Groupware service specification*. The set of Groupware Service Module Element (GSME) types, described in detail in chapter 5, can be applied to describe and compare the behaviour of a wide range of groupware applications.
- *Component-based groupware design*. The service-level groupware reference model, presented in chapter 6, provides a valid basis to design and implement component-based groupware. It groups functions in different Groupware Service Module (GSM) types and describes the relations between GSM types on a service level. The presented groupware reference model promotes conceptual consistency across groupware applications from different manufacturers. The latter makes it easier for end users to switch between different applications.
- *Tailorable groupware*. The CooPS groupware reference model allows groupware service users to select and compose the provided groupware *behaviour* to match their changing needs and personal preferences. As such, our approach allows for tailoring operations at the service level, since the service a groupware application provides is the most important aspect for its users.

9.3 Reflections on the evaluation

Design criteria, such as tailorability, generality and parsimony do not exist in isolation but rather have a meaning within a context (Abowd, 1998). A system is tailorable (or not) with respect to certain classes of changes. This notion of context-based evaluation has led us to adopt scenarios to evaluate the CooPS groupware reference model with repect to the design criteria, defined in section 3.7. In line with this, we applied the Software Architecture Analysis Method (SAAM) to evaluate the design based on concrete scenarios of groupware use and tailoring.

Although the SAAM was designed to analyze software architectures, the method is also applicable to analyze service architectures. One can evaluate whether a service architecture defines appropriate services to support a

given scenario. The SAAM analysis reveals that the CooPS groupware reference model is well suited to support a wide range of cases of groupware use and tailoring. Additionally, the analysis reveals some cases that are not directly supported by our design, and indicates a location in the design where different concerns may not have been separated over multiple GSM types.

Although the evidence produced during the evaluation is insufficient to draw firm conclusions about the degree to which our design meets the design criteria, the SAAM analysis provides strong indications our design does meet the design criteria, as defined in section 3.7.4.

9.3.1 Reflections on the questionnaire

A questionnaire has been applied to obtain insight in the units tailors distinguish in a groupware application. In this questionnaire the tailors were asked to describe the units they distinguish in a groupware application that should support people in a given concrete scenario. The result of the questionnaire is an indicator that the GSM types identified in the CooPS groupware reference model are in line with the types of units tailors distinguish in a groupware application. In our research, we assume that such a match increases the likelihood of successful tailoring: since tailors have to select and compose the GSMs that suit their needs, it is important that tailors regard these GSMs as "logical" units.

To avoid a bias, the constructed scenarios focus on the co-operative setting, instead of how the application supports the co-operating people. Similarly, we selected potential tailors as test subjects: people who frequently use groupware applications. The set of test subjects did not include people with knowledge of the research, people with a computer science background or system administrators. The results of the questionnaire were coded and analyzed by the author.

The results of the questionnaire also indicate an important condition for successful tailoring: our test subjects frequently describe groupware units in terms of *what these units allow them to do*. Based on this we conclude that a tailoring user interface should state this information to potential tailors.

9.4 Directions for future research and development

Our research provides a thorough basis to design tailorable groupware services. However, there are many areas left to be explored. The following directions illustrate interesting areas for further research and development:

- Investigate how well the concepts and groupware reference model presented in this book can be applied to describe and design purely asynchronous groupware services.
- Improve the CooPS groupware reference model by including facilities for inter-CC-GSM synchronization: this would allow different CC-GSMs, which may originate from different manufacturers, to synchronize their content and representation of that content.
- Validate in an experimental setting whether service module selection and composition is an appropriate mechanism to tailor groupware services.
- Measure the extent to which co-operating end users apply tailoring in real life co-operative settings.
- Create design guidelines for how to present tailoring options to groupware users.
- Investigate how to determine at run-time what the impact of a tailoring operation will be on the functioning of the groupware application and ultimately, on the co-operation itself.
- Recently, there has been much progress regarding the state of the art of service-oriented design. The presented design and structuring of groupware services can be evaluated in the light of these developments.
- Use the methodology presented in this book to develop a mature groupware application, including a large set of GSMs.

9.4.1 Recommendations regarding the AMBER language

In our research, the AMBER language has been applied to describe groupware behaviour. Even though this language allows one to express the relevant concepts of actions, interactions and causality relations in an appropriate manner, we recommend the following extensions to the language:

- *Disabling a complete behaviour*. While AMBER allows one to disable a single action or interaction, we frequently needed to disable a complete behaviour and introduced a shorthand notation for this purpose: a triangle, named *disabling*, pointing inward to the affected behaviour. This shorthand notation does not add to the expressiveness of AMBER, but increases the ease of use of the language.
- *Software component interface generation*. Based on AMBER service specifications, the skeleton for software component interfaces could be generated. These software components should realize the specified behaviour.

Appendix A

Scenarios of Groupware Use and Tailoring

In this section eight scenarios of groupware use and tailoring are presented. These scenarios capture specific instances of groupware use, not generic cases, although each scenario represents a class of similar situations. These scenarios have three purposes:

1. They serve as examples, clarifying typical cases of groupware use and tailoring.
2. They form a source of functional requirements, as they describe typical cases of how people use and tailor the behaviour provided by groupware applications.
3. They form a basis for evaluation. As described in section 8.1, the Software Architecture Analysis Method (SAAM) (Abowd, 1998) has been applied to evaluate our design. This method uses scenarios to determine the degree to which an architecture meets some articulated objectives. The evaluation of the CooPS groupware reference model has been performed using a different set of scenarios than the ones applied to establish requirements on the design.

The first four presented scenarios, denoted as BS1 to BS4, represent basic cases of groupware use and tailoring. The next three scenarios, denoted as CS1 to CS3, represent more comprehensive cases, revealing some of the complexity that may occur in real life co-operation. Each of these scenarios starts with a brief description of the key issue, followed by a narrative that states which actors perform what actions, as well as any relevant triggers, and process outcomes.

The last scenario is an external scenario: it has been formulated by an external researcher. This scenario has been added to evaluate our design.

BS1: A two-person chat session

This first scenario describes a basic case of synchronous co-operation between two people. They apply a straightforward mechanism to contact each other, and only apply one means of communication.

Karen and Richard work together in a project team. One morning, Karen starts her groupware application. Later that morning, Richard starts his groupware application as well. Richard's groupware application informs the groupware applications of his contacts that he is now available for communication. Karen, who is one of Richard's contacts, observes this and decides to contact him about a design they are both working on.

Karen clicks on Richard's name in her groupware application after which a list appears with the available means of communication. Karen selects the chat tool. Richard's groupware application now presents him a dialog box stating that Karen has invited him for a chat session. Richard accepts the invitation. The chat tool is activated at both their systems. Karen and Richard briefly discuss the changes they made to the design. Richard explains that he has to attend another meeting, so he leaves the chat session. Karen ends the conference.

BS2: Adding a collaboration tool

This scenario describes a basic case of co-operation between two people, in which the people introduce a tool to share an information object, namely a document. They apply a straightforward mechanism to contact each other.

Sam and Jack work at different locations of the same hospital. They use a groupware application to discuss patient treatment plans, on a fixed moment each week. To have these meetings, Sam clicks on Jack's name and selects the audio conferencing tool. They use the audio conferencing tool as the main means of communication throughout the session.

To discuss the patient treatment, they also share documents containing the patient treatment plans. To do so, Sam selects the checkbox in front of "document sharing" on the interface of her groupware application. As a result, the document sharing tool is activated at both their systems, and they can share documents by dragging them onto the sharing tool. After they have finished discussing their patients, Sam leaves the conference, after which Jack closes the conference.

BS3: Inviting an additional person

This scenario describes a basic case of co-operation, starting with two participants, but extended during the conference with a third one. The participants use only one means of communication, and a straightforward mechanism is applied to contact each other.

Michael and Rachel are working together on a design for a villa. They use a groupware application to co-ordinate their work. One morning Michael contacts Rachel, to discuss the implications for the construction if they include an indoor pool. Rachel recalls that one of their colleagues, Simon, has recently designed an indoor pool for another villa. During their chat conference, Rachel presses the button labelled "Invite", and selects Simon from the list of available people. Simon accepts the invitation on his screen, after which the chat tool is activated at his system as well. The three of them discuss the implications for the construction, and Simon tells them how the foundations have to be adapted to cope with the extra weight. After their discussion Rachel and Simon leave the conference, and Michael ends the conference.

BS4: Role-based access rights

This scenario describes a basic case of co-operation between multiple participants, while a role-based co-ordination policy is in place. Marc, a teacher at a distance learning university, uses a groupware application to lead a class of students to relevant information on the Internet. Only the teacher is allowed to change the website, and while all can use the chat and audio conferencing, the students are not allowed to add additional tools to the conference.

Marc starts his groupware application. He first prepares the co-operative session by selecting the tools that will be available: text-based chat, audio conferencing and a shared web browser. Marc specifies that only people with the role of "administrator" will be allowed to lead the group to a new URL using the shared web browser. Then he invites his students to the co-operative session, based on their school e-mail addresses, and specifies that these students will have the role of "participant". He also states in the invitation when the co-operative session will take place.

By the time the class starts, the students accept the invitation for the co-operative session. As a result, the tools associated with the session are activated on their computers. Marc explains the purpose of the co-operative session using the audio conferencing. He and his students use the chat and

audio conferencing to communicate during the session. Meanwhile, Marc leads the students to some relevant web pages that they discuss. While the students can explore the web sites on their own, Marc is able to lead them as a group to a new URL.

At the end of the class, all students leave the co-operative session, and Marc ends the conference.

CS1: Peer review of treatment plans

This scenario describes a comprehensive case of co-operation between multiple participants in the healthcare sector. On a scheduled moment each week a group of healthcare professionals from two locations of the same hospital discuss their patient treatment plans. To avoid travelling, they use dedicated discussion rooms at both locations, connected by a groupware application.

Each Monday at nine o'clock the technical assistants in both locations of the hospital prepare the co-operative session. They both start their local groupware application and log in. The technician from location North activates the audio and videoconferencing, and the application sharing. He then invites location South to the conference by selecting it from a list of recently contacted computers.

Once the technician from location South has accepted the invitation, the selected tools are activated at his computer as well. They now have a functioning audio and video connection, so the peer review can start. During the review of treatment plans the physicians typically share documents containing the treatment plan, and the patient's medical history, sometimes accompanied with high-detail medical images, such as x-ray scans. A co-ordination policy is in place, enforcing that only a participant with the role of "treating doctor" can modify a shared document, while the other participants can view the document. This is done since in their hospital only the treating doctor is allowed to update a treatment plan. Since multiple patients are discussed during the meeting, the role of "treating doctor" is dynamic. When needed, a separate high-resolution image viewer is activated to share the detailed medical images. After the physicians have discussed their patients, the technicians end the conference.

CS2: Chance encounters on a website

This scenario describes a comprehensive case of co-operation between two people in a research setting. They have a chance encounter on a virtual location, and co-operate using communication and collaboration tools.

Maria and Bill are researchers in the area of Human Computer Interaction (HCI) who work at different universities. One morning, Maria visits the website of a HCI conference she plans to visit. That website includes a service to see who else is currently on the website. Maria sees that Bill is also present on that website, and since she knows him from one of his publications, she uses this opportunity to ask him whether he will attend the conference.

Maria clicks on Bill's name on the website and a menu appears with a set of communication tools. She selects "chat" and types a message to Bill. Bill is notified about the incoming message from Maria, and chooses to open it. As a result, his groupware application establishes a co-operative session and activates the chat tool. Bill reads the message and replies that he will indeed attend the conference.

Bill also asks what topic in HCI Maria is investigating. She answers that she is interested in emotional computing, which is also Bill's focal area. Bill explains that he has recently done some experiments regarding emotional computing, and is presenting the findings at the conference. Maria is interested in the findings, so Bill selects the PDF version of his paper, and drops it on the conversation window. As a result, the groupware application launches the corresponding viewer and shares that viewer using application sharing. Bill points Maria to some of the interesting findings, and they finish by deciding to talk some more at the conference. Maria closes the co-operative session.

CS3: Task-oriented tailoring

This scenario describes a comprehensive case of co-operation between two people in the healthcare sector. They are physiotherapists who work at different locations. Sam works in the local hospital, while Catherine works in the rehabilitation centre. They frequently contact each other about various issues: to peer review treatment plans, to refer a patient to each other, or to request additional information regarding a patient who has been referred.

One morning, Catherine starts her groupware application to contact Sam in order to review the treatment plans for John Smith, a patient who recently

has been transferred from the hospital to the rehabilitation centre. When Catherine has started her groupware application, it presents her with a selection of communication templates, named after the task they are designed to support. These templates are domain specific, and in her case include “peer review”, “refer a patient”, and “request additional information”. Catherine clicks on “peer review” and subsequently selects Sam from her list of contacts. As a result, the groupware application activates the groupware behaviour that fits the selected co-operative task. In this case, the groupware application activates the audio conferencing service and document sharing. Additionally, a co-ordination policy is activated that controls that only the person who introduces a document (i.e., a treatment plan) is allowed to modify it. This latter requirement is needed, since only the treating healthcare professional is allowed to adapt the treatment plan. If Catherine would have selected a different communication template, a different set of groupware services would have been activated.
Sam receives the invitation for the online peer review session with Catherine. Sam accepts this invitation, after which her groupware application activates the corresponding services at her computer as well. They can now communicate using audio conferencing, and share documents. Together they discuss the treatment of John, after Catherine has shared the document with the treatment plan. When they are finished Sam leaves the online meeting, and Catherine closes the conference.

Appendix B

Groupware questionnaire

This appendix includes the questionnaire (in Dutch) that was given to the test subjects.

Voor dit interview willen we graag weten welke groupware applicaties u in het verleden gebruikt heeft, en hoe vaak: (kruis een vakje aan)

	nooit	wel eens	regelmatig
chat (MSN, ICQ, …)			
shared workspace (Groove, BSCW, …)			
videoconferencing applicatie (NetMeeting, …)			
weblog of website om discussies te voeren			
website om foto's te delen			
group decision support system(Ventana GroupSystems, …)			
online team agenda (Schedule+, Outlook calendar, …)			
applicatie om samen aan een document / ontwerp te werken			
anders, nl.:……			

We zullen u nu drie scenario's voorleggen waarin een aantal mensen op afstand samenwerken. We vragen u het eerste scenario aandachtig door te lezen.

Scenario 1

Personen: Karin lid van projectteam
Richard lid van projectteam

Karin en Richard werken samen in een projectteam aan het ontwerp van een mobiele telefoon. Op een ochtend komt Karin als eerste op kantoor en start haar computer.

Haar groupware applicatie start ook meteen op. Wanneer Richard wat later ook zijn computer aan zet, ziet Karin via haar groupware applicatie dat hij ook aanwezig is. Ze maakt van de situatie gebruik om Richard via haar groupware applicatie even te melden dat ze een verandering aan het ontwerp heeft aangebracht en dat ze wat later die dag de nieuwe versie van het ontwerp zal komen brengen. Ze bespreken de belangrijkste wijzigingen en spreken af dat Karin later die ochtend met het nieuwe ontwerp langs zal komen.

Opdracht: Beschrijf de onderdelen van Karin's groupware applicatie die nodig zijn om haar in dit scenario te ondersteunen.

Scenario 2

Personen:	Karin	fysiotherapeut
	mevr. Jansen	patiënt
	Mark	huisarts mevrouw Jansen
	Peter	arts in het ziekenhuis

Karin, een fysiotherapeut, behandelt mevrouw Jansen die aan het revalideren is na een beroerte gehad te hebben. Omdat de revalidatie van mevrouw Jansen niet zo snel verloopt als Karin verwachtte, neemt ze via haar groupware applicatie contact op met Mark, de huisarts van mevrouw Jansen. Karin en Mark bespreken de situatie en Mark herinnert zich dat mevrouw Jansen bij haar beroerte gevallen is. Ze besluiten de arts die mevrouw Jansen in het ziekenhuis behandeld heeft bij het gesprek uit te nodigen. Deze arts, Peter, ontvangt via zijn groupware applicatie de uitnodiging van Karin.

Peter gaat in op de uitnodiging en komt als derde persoon in het gesprek. Na de situatie besproken te hebben, besluiten ze gezamenlijk nog eens naar de röntgen foto's van mevrouw Jansen te kijken. Peter deelt via de groupware applicatie de röntgen foto's van mevrouw Jansen die hij in het ziekenhuis gemaakt heeft. Samen vinden ze een botbreuk, die eerder niet opgemerkt was.

Opdracht: Beschrijf de onderdelen van de groupware applicatie die nodig zijn om Karin, Mark en Peter in dit scenario te ondersteunen.

Scenario 3

Personen:	Karin	bezoeker online congres
	Monique	sessie-leider op het congres

Karin heeft een aankondiging gezien van een online congres over watermanagement. Aangezien Karin deskundige is op dat gebied, besluit ze deel te nemen aan het congres. Op de dag van het congres bezoekt Karin de website van de organisatie die het online congres organiseert: hier vindt ze een overzicht van alle huidige en geplande congressen. Karin klikt op de link naar het watermanagement congres, waardoor zij als bezoeker het online congres binnenkomt.

Als bezoeker kan Karin horen wat de huidige spreker vertelt. Verder ziet zij op haar scherm de sheets van de presentatie en kan ze via haar groupware applicatie aangeven dat ze een vraag heeft. Monique, de sessie-leider op het congres, kan hierop besluiten Karin het woord te geven. Als Karin het woord heeft, is ze in staat om via spraak haar vraag te stellen aan de spreker; anders is dit niet mogelijk. Ook de overige bezoekers van het online congres kunnen de vraag en het antwoord horen.

Opdracht: Beschrijf de onderdelen van de groupware applicatie die nodig zijn om Karin, Monique en de overige deelnemers aan het online congres te ondersteunen.

Summary

Modern information and communication technology enables us to work together and share knowledge in an advanced manner, while bridging differences in location and time. The general term to denote technology that allows this is *groupware*. We consider the provided *service*, i.e., the behaviour co-operating people observe while using a system, to be the most important aspect of a system. As such, we apply a service-oriented approach to design groupware in this dissertation. The approach explicitly pays attention to the dynamics of co-operation: our design enables co-operating people, the end users of groupware, to tailor the *behaviour* of their application.

The main research question dealt with in this dissertation is: How can groupware be designed in order to achieve tailorability of groupware services?

This question is relevant since the state of the art exploration reveals that the groupware landscape is fragmented: there exists no standard to design groupware services. Currently, groupware services are designed based on different paradigms, such as shared databases, communication channels, or shared virtual workspaces. As a result, different groupware services are not presented to groupware users in a similar manner.

The state of the art exploration also reveals that tailorability of groupware services is an important success factor for groupware services. Nevertheless, current groupware applications are typically difficult to tailor. One reason for this may be that current groupware design methodologies typically focus on implementation aspects. However, *what* the system does for them is more important for end users than how this service is implemented: the service a groupware application provides is the most relevant aspect for the users of groupware.

Existing groupware design methodologies insufficiently focus the designer's attention on this aspect; instead, they typically focus on implementation aspects. As a result, these methodologies do not help

designers to obtain the "bigger picture", showing the key properties of the groupware application to design. Therefore, we conclude there is a need for a service-oriented methodology to design tailorable groupware services. When such a methodology includes a structuring of groupware services, for instance in a service reference model, that structuring can become a standard to design groupware services.

Moreover, a service-oriented design approach allows us to define service modules that can be selected and composed to form groupware services. In this manner, end users can be empowered to select and compose the groupware behaviour they need. As the *service* a system provides is the most important aspect of the system for its users, end users should be presented with tailoring options on a service level.

To answer the main research question, we have created a structuring of groupware services. This structuring is provided for two target groups:

1. *Co-operating end users*. To allow co-operating end users to select and compose the groupware behaviour they require, coarse-grained Groupware Service Module (GSM) types are defined. These GSM types structure a groupware service: GSMs, created based on these types, are *units of groupware service composition*. End users can tailor the provided groupware service by changing the composition of GSMs.
2. *Groupware designers*. To support groupware designers, coarse-grained GSM types are defined. Groupware designers can select and combine GSMs to match on the requirements of a specific co-operative setting. The GSM types, as well as their relations on the service level, are described in a groupware service reference model. The various GSM types are constructed by grouping more fine-grained Groupware Service Module Element (GSME) types. GSMEs are *elementary units of groupware behaviour*. As such, GSME types structure groupware services on a more detailed level. The structuring on the level of GSME types also helps groupware designers to map a service-level design onto an implementation: the structuring specifies, in abstract terms, the actions and interactions that take place as part of the GSME types, and the causal relations between GSME types.

The foundation for the structuring of groupware services is formed by a set of precisely defined architectural concepts, design criteria and a generic model of groupware services. This model describes the service a groupware application has to provide to co-operating end users: it identifies the GSME types. Chapter 4 describes the anticipated dynamics in use of groupware services.

An important aspect of our groupware design approach is the focus on *tailorability*: the co-operating people themselves are empowered to select

and compose groupware behaviour to match their dynamic requirements. For this purpose, the set of GSME types includes behaviour to change the composition of GSMs. Changing the composition of GSMs results in changes to the provided groupware service. In this manner, tailors are empowered to select and compose groupware behaviour on a service level.

Our service-oriented groupware design approach, including the structuring defined by the CooPS groupware reference model, have been evaluated in four ways:

1. By designing and implementing a groupware application based on the CooPS groupware reference model. This product, the CoCoWare .NET platform, is a proof-of-concept demonstrator. Chapter 7 illustrates the steps needed to map a service-level specification onto an implementation.
2. A questionnaire was applied to evaluate whether the defined GSM types match to the units people apply when describing a groupware service. The fact that our test subjects described units with similar functions and a similar granularity is an indicator for this match. In our research we assume that such a match is beneficial for successful selection and composition of groupware behaviour.
3. The externally observable behaviour of two existing groupware applications has been expressed in terms of the GSME types. This illustrates the possibility to use the GSME types to express relevant aspects of groupware behaviour, as well as the possibility to compare the behaviour of applications based on the descriptions.
4. The Software Architecture Analysis Method (SAAM) has been applied to evaluate that the CooPS groupware reference model adheres to the design criteria that have previously been identified as being most relevant for tailorable groupware services.

Summarizing, our research has yielded the following:

– *Structuring of groupware services*. The main contribution of the research is a structuring of groupware services, as described above.
– *A methodology to design tailorable groupware services*. This methodology is based on precisely formulated architectural concepts, such as actions, interactions and causality relations. The methodology identifies important groupware service design criteria and stimulates designers to first capture on a service level the key properties of the design, before switching to other abstraction levels.
– A description of the *elementary units of groupware behaviour* and their interrelations. This description enables designers to express the externally observable behaviour of current and future groupware applications in a univocal manner. This allows a comparison of behaviour of such applications regarding specific aspects.

– *The CooPS groupware reference model*. This reference model describes GSM types, the associated groupware behaviour, as well as the relations on a service level between GSM types. GSMs, based on these types, form *units of composition of groupware services*: they allow end users as well as designers to select and compose groupware behaviour. In this manner, groupware behaviour can be tailored to match the dynamics that is inherent in co-operative settings.

Samenvatting

Moderne informatie- en communicatietechnologie stelt ons in staat om op geavanceerde manieren samen te werken en kennis te delen, waarbij we verschillen in plaats en tijd overbruggen. De algemene term voor de technologie die dit mogelijk maakt is *groupware*. Wij beschouwen de geleverde *dienst*, d.w.z., het gedrag dat samenwerkende mensen observeren als ze een systeem gebruiken, als het belangrijkste aspect van een systeem. Daarom hanteren we in deze dissertatie een dienst-georiënteerde aanpak om groupware te ontwerpen. De aanpak houdt expliciet rekening met de dynamiek van samenwerking: ons ontwerp stelt samenwerkende mensen, de eindgebruikers van groupware, in staat om het *gedrag* van hun applicatie zelf aan te passen.

De belangrijkste onderzoeksvraag waar in dit proefschrift een antwoord op gezocht wordt, is: Hoe kan groupware ontworpen worden om aanpasbaarheid van groupware gedrag door eindgebruikers mogelijk te maken?

Deze vraag is relevant omdat een verkenning van de stand van zaken laat zien dat het groupware landschap gefragmenteerd is: er bestaat geen standaard om groupware diensten te ontwerpen. Op dit moment worden groupware diensten ontworpen op basis van verschillende paradigma's, zoals gedeelde databases, communicatie kanalen of gedeelde virtuele werkruimtes. Een resultaat hiervan is dat groupware diensten niet op een soortgelijke manier naar eindgebruikers gepresenteerd worden.

De verkenning van de stand van zaken laat ook zien dat aanpasbaarheid van groupware diensten door eindgebruikers een belangrijke succesfactor is voor groupware diensten. Desondanks zijn huidige groupware systemen vaak moeilijk aan te passen door eindgebruikers. Eén van de redenen hiervoor kan zijn dat huidige ontwerpmethoden vaak de aandacht richten om implementatie aspecten. Echter, *wat* een systeem voor ze doet, is voor eindgebruikers belangrijker dan hoe die dienst geïmplementeerd is: welke

dienst geleverd wordt, is het meest relevante aspect voor gebruikers van groupware.

Bestaande groupware ontwerp methodologieën richten de aandacht van de ontwerper onvoldoende op dit aspect; in plaats daarvan richten ze de aandacht gewoonlijk op implementatie aspecten. Een resultaat hiervan is dat deze methodoligieën ontwerpers niet helpen om op het "grotere geheel" te letten; ze laten de kern eigenschappen van de te ontwerpen groupware applicatie niet zien. Daarom concluderen we dat er behoefte is aan een dienst-georiënteerde methodologie om groupware diensten te ontwerpen die door eindgebruikers aangepast kunnen worden. Wanneer zo'n methodologie een structurering van groupware diensten omvat, bij voorbeeld in een diensten referentie model, kan die structurering een standaard vormen om groupware diensten te ontwerpen.

Bovendien, een dienst-georiënteerde aanpak maakt het mogelijk om service modulen te definiëren die geselecteerd en gecomponeerd kunnen worden om groupware diensten te vormen. Op deze manier kunnen eindgebruikers in staat gesteld worden om het groupware gedrag dat ze nodig hebben te selecteren en te componeren. Aangezien de dienst die een systeem levert het meest belangrijke aspect van het systeem is voor zijn gebruikers, moeten eindgebruikers aanpassingen ook op dienst-niveau uit kunnen voeren.

Om de belangrijkste onderzoeksvraag te beantwoorden, hebben we een structurering van groupware services gecreëerd. Deze structurering is gericht op twee doelgroepen:

1. *Samenwerkende eindgebruikers*. Om samenwerkende eindgebruikers in staat te stellen om het groupware gedrag dat ze nodig hebben te kunnen selecteren en componeren zijn grofkorrelige Groupware Service Module (GSM) typen gedefinieerd. Deze GSM typen structureren een groupware dienst, omdat ze de typen vormen van eenheden van groupware compositie. Eindgebruikers kunnen de geleverde groupware dienst aanpassen door de compositie van GSMs te veranderen.
2. *Groupware ontwerpers*. Om groupware ontwerpers te ondersteunen zijn grofkorrelige GSM typen gedefinieerd. Groupware ontwerpers kunnen GSMs selecteren en combineren afhankelijk van de eisen van de setting waarin samengewerkt wordt. De GSM typen, net als hun relaties op dienst-niveau, worden beschreven in een groupware dienst referentie model. De verschillende GSM typen zijn gevormd door meer fijnkorrelige Groupware Service Module Element (GSME) typen te groeperen. GSME typen zijn de elementaire eenheden van groupware gedrag. Als zodanig structureren GSME typen een groupware dienst op een meer gedetailleerd niveau. De structurering op het niveau van GSME typen helpt groupware ontwerpers ook om een ontwerp op

dienst niveau af te beelden op een implementatie: de structurering specificeert, in abstracte termen, welke acties en interacties plaats vinden als onderdeel van GSME typen en de causale afhankelijkheden tussen GSME typen.

De basis voor de structurering van groupware diensten wordt gevormd door een set van precies gedefinieerde architectuur concepten, ontwerp criteria en een generiek model van groupware diensten. Dit model beschrijft de dienst die een groupware applicatie moet leveren aan samenwerkende eindgebruikers: het identificeert de GSME typen. Hoofdstuk 4 beschrijft de geanticipeerde dynamiek in het gebruik van groupware diensten.

Een belangrijk aspect van onze groupware ontwerp aanpak is de focus op aanpasbaarheid door eindgebruikers: de samenwerkende mensen worden zelf in staat gesteld om groupware gedrag te selecteren en te componeren, passend bij hun veranderende eisen. Voor dit doel omvat de set van GSME typen gedrag om de compositie van GSMs aan te passen. Het aanpassen van de compositie van GSMs resulteert in veranderingen in de geleverde groupware dienst. Op deze manier worden eindgebruikers in staat gesteld om groupware gedrag te selecteren en te componeren op het niveau van de dienst.

Onze dienst-georiënteerd aanpak om groupware te ontwerpen, inclusief de structurering die het CooPS groupware referentie model definieert, zijn op vier manieren geëvalueerd:

1. Door het ontwerpen en bouwen van een groupware applicatie op basis van het CooPS referentie model. Dit product, het CoCoWare .NET platform, laat zien hoe de beschreven concepten in de praktijk ingezet kunnen worden. Hoofdstuk 7 illustreert welke stappen gezet moeten worden om een concrete applicatie te ontwerpen op basis van het referentie model.
2. Via een vragenlijst is onderzocht of de gedefinieerde GSM typen aansluiten bij de eenheden die gebruikers zelf noemen wanneer ze een groupware dienst beschrijven. Het feit dat onze proefpersonen eenheden beschreven met vergelijkbare functies en een vergelijkbare granulariteit is een indicator voor deze overeenkomst. In ons onderzoek nemen we aan dat zo'n overeenkomst bevorderlijk is voor het succesvol selecteren en componeren van groupware gedrag.
3. Het extern observeerbare gedrag van twee bestaande groupware applicaties is uitgedrukt in termen van de GSME typen. Dit illustreert de mogelijkheid om middels de GSME typen relevante aspecten van groupware gedrag uit te drukken. Tevens illustreert dit de mogelijkheid om het gedrag van applicaties te vergelijken op basis van deze beschrijvingen.

4. Door toepassing van de Software Architecture Analysis Method (SAAM) is vastgesteld dat het CooPS groupware referentie model voldoet aan de kwaliteitsprincipes die eerder aangemerkt waren als zijnde meest relevant voor aanpasbare groupware.

Samenvattend heeft ons onderzoek het volgende opgeleverd:

- *Een structurering van groupware diensten*. De belangrijkste bijdrage van het onderzoek is, zoals eerder beschreven, een structurering van groupware diensten.
- *Een methodologie om aanpasbare groupware diensten te ontwerpen*. Deze methodologie is gebaseerd op precies geformuleerde architectuur concepten, zoals acties, interacties en causaliteit relaties. De methodologie identificeert belangrijke criteria voor het ontwerpen van groupware diensten en stimuleert ontwerpers om eerst op het dienstniveau de belangrijkste eigenschappen van het ontwerp vast te leggen, voordat men andere abstractieniveaus beschouwt.
- Een beschrijving van *eenheden van groupware gedrag* en hun onderlinge samenhang. Deze beschrijving stelt ontwerpers in staat om het extern observeerbare gedrag van bestaande en nieuwe groupware applicaties op eenduidige manier uit te drukken. Dit maakt het mogelijk om gedrag van zulke applicaties op specifieke aspecten te vergelijken.
- Het *CooPS groupware referentie model*. Dit referentie model beschrijft GSM typen, het geassocieerde groupware gedrag en de relaties tussen GSM typen op dienstniveau. GSMs, gebaseerd op deze typen, vormen *eenheden van compositie van groupware diensten*: zij stellen zowel eindgebruikers als ontwerpers in staat om groupware gedrag te selecteren en te componeren. Op deze manier kan het gedrag van groupware aangepast worden aan de dynamiek die inherent is aan samenwerkingsprocessen.

References

Abowd, G. D. (1998). Analyzing Development Qualities at the Architectural Level: The Software Architecture Analysis Method. In L.Bass, P. Clements, & R. Kazman (Eds.), *Software Architecture in Practice* (pp. 189-220). Boston: Addison-Wesley.

Ackerman, M. (2000). The Intellectual Challenge of CSCW: The Gap Between Social Requirements and Technical Feasibility. *Human-Computer Interaction, 15,* 181-205.

Agostini, A., De Michelis, G., & Susani, M. (2000). From user participation to user seduction in the design of innovative user-centered systems. In R.Dieng, A. Giboin, L. Karsenty, & G. De Michelis (Eds.), *Designing Cooperative Systems - The use of theories and models* (pp. 225-240). Amsterdam: IOS Press.

Baker, K., Greenberg, S., & Gutwin, C. (2002). Empirical development of a heuristic evaluation methodology for shared workspace groupware. In *Proceedings of the 2002 ACM conference on Computer supported cooperative work* (pp. 96-105). New Orleans, Louisiana, USA: ACM Press.

Bardram, J. E. (1998). Designing for the dynamics of cooperative work activities. In *Proceedings of the ACM 1996 conference on Computer Supported Cooperative Work (CSCW '98)* (pp. 89-98). New York: ACM Press.

Bass, L., Clements, P., & Kazman, R. (1999). *Software Architecture in Practice*. Reading,MA.: Addison-Wesley.

Beck, E. E. & Bellotti, V. M. E. (1993). Informed Opportunism as Strategy: Supporting Coordination in Distributed Collaborative Writing. In G.De Michelis, C. Simone, & K. Schmidt (Eds.), *Proc. of the Third European Conference on Computer-Supported Cooperative Work (ECSCW'93)* (pp. 233-248). Dordrecht: Kluwer Academic Publishers.

Bentley, R. & Dourish, P. (1995). Medium versus Mechanism: Supporting Collaboration through Customisation. In H.Marmolin, Y. Sundblad, & K. Schmidt (Eds.), *Proceedings of the Fourth European Conference on Computer Supported Cooperative Work - ECSCW '95* (pp. 133-148). Dordrecht, The Netherlands: Kluwer.

Biemans, M. C. M. & ter Hofte, G. H. (1999). *Tailorability: the state of the art* (Rep. No. TI/RS/99020). Enschede, The Netherlands: Telematica Instituut.

Bock, G. E. & Marca, D. A. (1995). *Designing Groupware*. New York: McGraw-Hill.

Brooks, F. (1975). *The Mythical Man-Month - Essays on Software Engineering*. Reading, MA.: Addison-Wesley.

Calvary, G., Coutaz, J., & Nigay, L. (1997). From single-user architectural design to PAC*: a generic software architecture model for CSCW. In *Proc. of the SIGCHI conference on Human factors in computing systems* (pp. 242-249). Atlanta, USA: ACM Press.

Cockburn, A. & Greenberg, S. (1993). Making contact: Getting the group communicating with groupware. In *Proceedings of the ACM COCS'93 Conference on Organizational Computing Systems* (pp. 31-41). Milpitas, California: ACM Press.

Daft, R. L. & Lengel, R. H. (1984). Information richness: A new approach to managerial behavior and organizational design. *Research in Organizational Behavior, 6,* 191-233.

Daft, R. L., Lengel, R. H., & Trevino, L. K. (1987). The relationship among message equivocality, media selection, and manager performance. *MIS Quarterly, 11,* 355-366.

Dewan, P. (1998). Architectures for Collaborative Applications. In M.Beaudouin-Lafon (Ed.), *Computer Supported Cooperative Work (CSCW)* (7 ed., pp. 169-194). John Wiley & Sons Ltd.

Dewan, P. (2001). An Integrated Approach to Designing and Evaluating Collaborative Applications and Infrastructures. *Computer Supported Cooperative Work: The Journal of Collaborative Computing, 10,* 75-111.

Edwards, W. K. (1994). Session management for collaborative applications. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work* (pp. 323-330). Chapel Hill, NC, United States: ACM Press.

Edwards, W. K. (1996). Policies and Roles in Collaborative Applications. In *ACM Conference on Computer Supported Cooperative Work (CSCW'96)* (pp. 11-20). New York: ACM Press.

Eertink, H., Janssen, W. P. M., Oude Luttighuis, P. H. W. M., Teeuw, W. B., & Vissers, C. A. (1999). A Business Process Design Language. In J.M.Wing, J. Woodcock, & J. Davies (Eds.), *FM'99 - Formal methods* (pp. 76-95). Heidelberg: Springer.

Ellis, C. A., Gibbs, S. J., & Rein, G. L. (1991). Groupware: Some issues and experiences. *Communications of the ACM, 34,* 38-58.

Ellis, C. A. & Wainer, J. (1994). A conceptual model of groupware. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work* (pp. 79-88). Chapel Hill, North Carolina, United States: ACM Press.

Erickson, T. & Kellogg, W. (2000). Social translucence: an approach to designing systems that support social presence. *Transactions on Computer Human Interaction (TOCHI), 7,* 59-83.

Ferreira Pires, L., Vissers, C. A., & Van Sinderen, M. J. (1993). Advances in architectural concepts to support distributed systems design. In *11o Simpósio Brasileiro de Redes de Computadores: Tutoriais e Minicursos*. Campinas, Brazil: Universidade Estadual de Campinas.

Fish, R., Kraut, R., Root, R., & Rice, R. (1992). Evaluating Video as a Technology for Informal Communication. In *Conference proceedings on Human Factors in Computing Systems (CHI'92)* (pp. 37-48). New York,NY.: ACM Press.

Fulk, J., Schmitz, J., & Steinfield, C. W. (1990). A social influence model of technology use. In J.Fulk & C. W. Steinfield (Eds.), *Organizations and communication technology* (pp. 117-140). Newbury Park: Sage.

Greenberg, S. & Roseman, M. (1999). Groupware Toolkits for Synchronous Work. In M.Beaudouin-Lafon (Ed.), *Computer Supported Co-operative Work* (pp. 135-168). Chichester, England: John Wiley & sons.

Grundy, J. C. & Hosking, J. G. (2002). Engineering plug-in software components to support collaborative work. *Software - Practice and Experience (SPE), 32,* 983-1013.

Grundy, J. C., Mugridge, W. B., & Hosking, J. G. (1997). A Java-based Componentware Toolkit for Constructing Multi-view Editing Systems. In *Proc. of the 2nd Component Users' Conference (CUC'97)*. Munich, Germany: SIGS Books.

Grundy, J. C., Mugridge, W. B., Hosking, J. G., & Apperley, M. D. (1998). Tool integration, collaboration and user interaction issues in component-based software architectures. In C.Mingins & B. Meyer (Eds.), *Proc. of TOOLS Pacific '98* (pp. 289-301). Melbourne, Australia: IEEE CS Press.

Guareis de Farias, C. R. (2002). *Architecural Design of Groupware Systems: a Component-Based Approach*. Enschede, The Netherlands: CTIT.

Gutwin, C. & Greenberg, S. (2000). The mechanics of collaboration: Developing low cost usability evaluation methods for shared workspaces. In *Proc. 9th IEEE WETICE: Infrastructures for Collaborative Enterprises* (pp. 98-103).

Hettinga, M. (2002). *Understanding evolutionary use of groupware*. Enschede, The Netherlands: Telematica Instituut.

Hettinga, M. & Heeren, E. (1998). Evolution: The undervalued source of design guidelines for cooperative technology. In F.Darses & P. Zaraté (Eds.), *Proceedings of COOP'98* (pp. 29-31). Rocquencourt, France: INRIA Press.

Hummes, J. & Merialdo, B. (2000). Design of Extensible Component-Based Groupware. *Computer Supported Cooperative Work: The Journal of Collaborative Computing, 9,* 53-74.

IMTC (2003). H.320 Overview. International Multimedia Telecommunications Consortium (IMTC) [On-line]. Available: http://www.imtc.org/h320.htm

ISO (1987). *Basic Reference Model for Open Systems Interconnection, Int Standard ISO 7498*. International Organization for Standardization.

Johansen, R. (1988). Current user approaches to groupware. In R.Johansen (Ed.), *Groupware : Computer support for business teams* (pp. 12-44). New York: Free Press.

Kahler, H., Mørch, A. I., Stiemerling, O., & Wulf, V. (2000). Introduction to special issue "Tailorable Systems and Cooperative Work". *Computer Supported Cooperative Work: The Journal of Collaborative Computing, 9,* 1-4.

Kausar, N. & Crowcroft, J. (1999). An architecture of Conference Control Functions. In *Proceedings of Photonics East*. Boston, MA.: SPIE.

Kellogg, W. (1987). Conceptual consistency in the user interface: Effects on user performance. In H.J.Bullinger & B. Shackel (Eds.), *Human-Computer Interaction - INTERACT '87* (pp. 389-394). Amsterdam, The Netherlands: Elsevier Science.

Koch, M. & Teege, G. (1999). Support for Tailoring CSCW systems: adaptation by composition. In *Proceedings of the 7th Euromicro workshop on parallel and distributed processing* (pp. 146-152). Funchal, Portugal: IEEE Press.

Kristoffersen, S. & Ljungberg, F. (1999). An Empirical Study of How People Establish Interaction: Implications for CSCW Session Management Models. In *Proceedings of the CHI 99 Conference on Human Factors in Computing Systems* (pp. 1-8). Pittsburgh,PA.: ACM Press.

Kuhn, S. & Muller, M. J. (1993). Participatory Design. *Communications of the ACM, 36,* 24-28.

Laurillau, Y. & Nigay, L. (2002). Clover Architecture for Groupware. In *Proceedings of the 2002 ACM conference on Computer supported cooperative work* (pp. 236-246). New York, NY, USA: ACM Press.

Li, D. & Muntz, R. (1998). COCA: collaborative objects coordination architecture. In S.Poltrock & J. Grudin (Eds.), *Proc. of the 1998 ACM conference on Computer supported cooperative work* (pp. 179-188). New York, NY, USA: ACM Press.

Mackay, W. E. (1991). Triggers and barriers to customizing software. In *Proceedings of ACM CHI '91 Human Factors in Computing Systems* (pp. 153-160). New Orleans.

Mandviwalla, M. & Olfman, L. (1994). What do groups need? A proposed set of generic groupware requirements. *Transactions on Computer Human Interaction (TOCHI), 1,* 245-268.

Markus, M. L. (1990). Toward A "Critical Mass" Theory of Interactive Media. In J.Fulk & C. W. Steinfield (Eds.), *Organizations and Communication Technology* (pp. 194-218). Newbury Park, CA: Sage Publications.

Marsic, I. (1999). DISCIPLE: A Framework for Multimodal Collaboration in Heterogeneous Environments. *ACM Computing Surveys, 31.*

McGrath, J. E. & Hollingshead, A. B. (1993). Putting the "group" back in group support systems: some theoretical issues about dynamic processes in groups with technological enhancements. In L.M.Jessup & J. S. Valacich (Eds.), *Group Support Systems: New Perspectives* (pp. 78-96). New York: Macmillan.

Microsoft corp. (2003). Microsoft .NET. Microsoft website [On-line]. Available: http://www.microsoft.com/net/

Mørch, A. I. (1994). Designing for radical tailorability: coupling artifact and rationale. *Knowledge-based Systems, 7,* 253-264.

Mørch, A. I. (1997a). *Methods and tools for tailoring of object-oriented applications: An evolving artifacts approach.* University of Oslo, Norway, Department of Informatics, University of Oslo, Norway.

Mørch, A. I. (1997b). Three Levels of End-User Tailoring: Customization, Integration, and Extension. In M.Kyng & L. Mathiassen (Eds.), *Computers and design in context* (pp. 51-76). Cambridge,MA.: The MIT Press.

Mulder, I. J. & Slagter, R. J. (2002). Collaborative Design, Collaborative Technology: Enhancing virtual team collaboration. In N.Callaos, T. Leng, & B. Sanchez (Eds.), *Proc. of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002). Vol. V* (pp. 74-79). Orlando, USA: IIIS.

O'Day, V. L., Bobrow, D. G., & Shirley, M. (1996). The Social-Technical Design Cycle. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW'96)* (pp. 160-169). New York: ACM Press.

Okamura, K., Orlikowski, W. J., Fujimoto, M., & Yates, J. (1994). Helping CSCW applications succeed: the role of mediators in the context of use. In *Proceedings of the conference on Computer supported cooperative work (CSCW'94)* (pp. 55-65). New York: ACM Press.

Oppermann, R. (1994). Adaptively supported adaptability. *International Journal of Human-Computer Studies, 40,* 455-472.

Oppermann, R. & Simm, H. (1994). Adaptability: user-initiated individualisation. In R.Oppermann (Ed.), *Adaptive user support, Ergonomic design of manually and automatically adaptable software*. Lawrence Erlbaum Assoc.

Orlikowski, W. J. (1992). The Duality of Technology: Rethinking the Concept of Technology in Organizations. *Organization Science, 3,* 398-427.

Palen, L. A. (1998). *Calendars on the New Frontier: Challenges of Groupware Technology*. Irvine: Information & Computer Science, University of California.

Pirolli, P. & Card, S. (1999). Information Foraging. *Psychological review, 106,* 643-675.

Quartel, D. A. C., Ferreira Pires, L., Van Sinderen, M. J., Franken, H. M., & Vissers, C. A. (1997). On the role of basic design concepts in behaviour structuring. *Computer Networks and ISDN Systems, 29,* 413-436.

Roseman, M. & Greenberg, S. (1996). Building Real Time Groupware with GroupKit, A Groupware Toolkit. *Transactions on Computer Human Interaction (TOCHI), 3,* 66-106.

Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R. et al. (2002). SIP: Session Initiation Protocol. IETF Internet Draft [On-line]. Available: http://www.ietf.org/internet-drafts/draft-ietf-sip-rfc2543bis-09.txt

Schuckmann, C., Kirchner, L., Schuemmer, J., & Haake, J. M. (1996). Designing object-oriented synchronous groupware with COAST. In M.Ackerman (Ed.), *Proceedings of the 1996 ACM conference on Computer supported cooperative work* (pp. 30-38). New York, NY, USA: ACM Press.

Slagter, R. J. & Biemans, M. C. M. (2003). Designing Tailorable Groupware for the Healthcare Domain. In J.Favela & D. Decouchant (Eds.), *Groupware: Design, Implementation, and Use: Proceedings of CRIWG 2003, Autrans, France, September 28 - October 2, 2003.* (pp. 58-73). Heidelberg, Germany: Springer-Verlag.

Slagter, R. J., ter Hofte, G. H., & Kruse, H. C. J. (2002). *CoCoWare .NET reference architecture: version 2.0*. Enschede, The Netherlands: Telematica Instituut.

Snyder, F. W. (1971). *Travel patterns: Implication for new communication facilities* Bell Laboratories Memorandum.

Steinfield, C., Huysman, M., David, K., Jang, C. Y., Poot, J., Huis in 't Veld, M. et al. (2001). New Methods for Studying Global Virtual Teams: Towards a Multi-Faceted Approach. In *Proceedings of the 34th Hawaii International Conference on Systems Sciences (HICSS-34)*. IEEE.

Stiemerling, O. (2000). *Component-based tailorability.* University of Bonn, Bonn, Germany.

Strauss, A. L. (1993). *Continual Permutations of Action*. New York: Aldine de Gruyter.

Suchman, L. A. (1987). *Plans and situated actions; the problem of human machine communication*. Cambridge, UK: Cambridge University Press.

Suchman, L. A. & Wynn, E. (1984). Procedures and Problems in the Office. *Office: Technology and People, 2,* 133-154.

Szyperski, C. (1998). *Component Software - Beyond Object-Oriented Programming*. New York: ACM Press.

Teege, G. (2000). Users as Composers: Parts and Features as a Basis for Tailorability in CSCW Systems. *Computer Supported Cooperative Work: The Journal of Collaborative Computing, 9,* 101-122.

ter Hofte, G. H. (1998). *Working Apart Together: Foundations for Component Groupware*. Enschede, the Netherlands: Telematica Instituut.

ter Hofte, G. H., Mulder, I. J., Grootveld, M., & Slagter, R. J. (2002). Exploring a design space for place-based presence. In C.Greenhalgh, E. Churchill, & W. Broll (Eds.), *Proceedings of the 4th International Conference on Collaborative Virtual Environments (CVE 2002)* (pp. 151-152). Bonn, Germany: ACM Press.

Tietze, D. A. (2001). *A Framework for Developing Component-based Co-operative Applications*. Sankt Augustin, Germany: GMD.

Van Sinderen, M. J. (1995). *On the design of application protocols*. Enschede, The Netherlands: University of Twente.

Vissers, C. A., Ferreira Pires, L., Quartel, D. A. C., & Van Sinderen, M. J. (2002). *The architectural design of distributed systems: Reader for the design of Telematics Systems*. Enschede, The Netherlands: University of Twente.

Vissers, C. A., Lankhorst, M. M., & Slagter, R. J. (2003). Reference Models for Advanced E-Services. In *Proc. Third IFIP Conference on e-Commerce, e-Business, and e-Government (I3E 2003), 21-24 September 2003, São Paulo, Brazil* (pp. 25-40).

Wulf, V. (1999a). "Let's see your Search-Tool!" - Collaborative Use of Tailored Artifacts in Groupware. In S.Hayne (Ed.), *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work (GROUP'99)* (pp. 50-59). New York: ACM Press.

Wulf, V. (1999b). Evolving cooperation when introducing groupware: a self-organization perspective. *Cybernetics & Human Knowing, 6,* 55-75.

Wulf, V. (2000). Exploration Environments: Supporting Users to Learn Groupware Functions. *Interacting with Computers, 13,* 265-299.

Wulf, V. & Golombek, B. (2001). Direct Activation: A Concept to Encourage Tailoring Activities. *Behaviour & Information Technology, 20,* 249-263.

Zigurs, I. & Buckland, B. K. (1998). A Theory of Task/Technology Fit and Group Support Systems Effectiveness. *MIS Quarterly, 22,* 313-334.

Index

The entries refer to the pages where the term is either introduced or discussed.

About the author

Robert J. Slagter has been a researcher at the Telematica Instituut since 1998. His research activities include CSCW, groupware, user-centred design and service-oriented design. He received his master's degree in computer science from the University of Twente in 1997, where he graduated on research into visual attention in multiparty communication.

The author contributed to the design and implementation of groupware applications in various projects including GigaPort, the Dutch national initiative for next generation Internet. He authored several scientific publications on groupware design and on involving end users throughout the design process. Furthermore, he organized CBG2000: an ACM workshop on the design of component-based groupware.

Dynamic Groupware Services

Modular design of tailorable groupware

Robert Slagter

Modern groupware allows us to work together, while bridging differences in space and time. However, there exists no standard to design groupware, which has resulted in a lack of conceptual consistency and interoperability. Additionally, current groupware design methodologies typically focus on implementation details. In contrast, we consider the *service* a groupware application provides to be the most important aspect of the system.

Dynamic Groupware Services provides a structuring of groupware services. This structuring allows co-operating end users to select and compose groupware modules in order to form groupware services that match their needs. At the same time, this structuring helps designers design tailorable groupware services. Two important results of this structuring are a description of the elementary units of groupware behaviour and a description of the units to compose groupware services. The latter units, as well as the relations between them, are defined in a reference model for tailorable groupware services.

ISBN 90-75176-37-6

Telematica Instituut