Service Orchestration for Collaboration Patterns

ROBERT SLAGTER[1], MARGIT BIEMANS[1] and VAL JONES[2]

1 Telematica Instituut, P.O. Box 589, 7500 AN Enschede, The Netherlands.
2 University of Twente, The Netherlands.

Robert.Slagter@telin.nl

Abstract: This article describes a structuring of groupware services that allows end users to orchestrate the provided services in order to match their collaboration patterns. Our approach is based on the notion that different forms of collaboration require different combinations of groupware services, and that these provided services are the most important aspect of a groupware system for its users. Based on an analysis of a series of high-level collaboration patterns from the healthcare domain we illustrate where flexibility in groupware service design is needed. The resulting structuring has been evaluated by experts using scenarios and has been implemented in a proof-of-concept demonstrator.

Keywords: groupware design, service-oriented design, patterns, tailoring.

1 INTRODUCTION

In order to benefit from a set of high-level collaboration patterns, such as the ones described in this special issue, one needs to design groupware in such a way that collaborating people can easily work according to those patterns. To do so, the services, i.e., the *behaviour*, groupware provides has to match the collaboration patterns. Current groupware systems are not designed for this purpose, which makes it difficult, or even impossible, to adapt them to specific collaboration patterns.

In groupware one size does not fit all: the behaviour provided by a groupware application has to match the requirements of the collaborative setting (Bardram, 1998).

At the same time, one cannot predict exactly how people will collaborate, and what changes may occur in the course of collaboration. However, by investigating *collaboration patterns* we can obtain indications of where flexibility in groupware service design is needed.

In this article we analyse a series of high-level collaboration patterns and derive a *structuring of groupware services*. This structuring allows end users to orchestrate the provided groupware services in order to match their collaboration patterns.

Although this article does not focus on describing collaboration patterns themselves, it focuses on bridging the gap between such patterns and groupware service design.

2 COLLABORATION PATTERNS IN HEALTHCARE

The high-level collaboration patterns stated in this section describe recurring problems in (the support of) collaboration, the forces associated, and related proven solutions. These patterns have been selected as they cover different aspects of collaboration and provide indications where flexibility in groupware service design is needed. We have chosen the healthcare domain since processes around a patient typically involve many

different professionals, with different roles and complementary expertises, where collaboration is essential.

2.1 Pattern format applied

To describe collaboration patterns, we apply a structure similar to the one described in Schümmer & Slagter (2004). Because of space limitations, we have only included the most relevant parts of the patterns.

2.2 Pattern 1: Finding expertise

Intent: To contact people by their role or expertise, rather then by identity.

Problem: In some collaborative settings, you wish to contact people based on specific properties, such as their role or expertise, rather then contacting a specific individual. However, many groupware systems have name-based mechanisms to invite people. In that case, the invitor has to do the matching with the properties, i.e., s/he has to know who has which role or who is good at what.

Symptoms: Questions such as: “Who do you contact for …?”

Solution: Allow users to contact individual based on properties, such as their role or expertise.

Rationale: When the available properties are displayed with the names of people, one can contact individuals based on their identity, or based on specific properties, such as their role. This process may also be

automated, resulting in a redirection mechanism to contact individuals based on their role or expertise.

Scenario: A cancer patient is cared for at home by a virtual care team coordinated by a cancer nurse[1]. The nurse visits the patient daily. In cases of out-of-hours problems (such as pain) the patient or a family member can press an alarm that links them with a medical call centre. At the call centre the duty nurse does a preliminary assessment. Depending on the assessment he can ask the system to search for an on-call nurse or GP to be dispatched immediately to the patient's home, or he can call an ambulance. At the patient's home the visiting GP or nurse can request a video-conference with any oncologist on call (which could be anywhere in the world, including different time zones) for further assessment and advice.

Known uses include: The 911 number, info@…, webmaster@….

2.3 Pattern 2: Finding out availability

Intent: To reduce the number of cases where you try to contact somebody who is currently not available or busy.

Problem: You do not know whether somebody is available for communication until you actually try to contact that person.

[1] Scenario inspired by DITIS project (Pitsillides, Samaras, Dikaiakos, & Christodoulou, 2005)

Symptoms: A large number of “missed calls”; frequently having to call back later.

Solution: Provide presence awareness, in this case covering information about the availability of people for communication.

Rationale: Before contacting a person, you already have some information about that person’s availability for communication. This information is either provided directly by the person himself or deduced by a system.

Scenario: The medical call centre identifies on-call nurses and GPs in the vicinity of the patient using the presence awareness function of the call centre system. Similarly oncologists available for video consultation can be identified. Some of the on-call oncologists may be currently unavailable, for example because they are at that moment treating another emergency. The system distinguishes “on-call” from “available right now” by means of presence awareness.

Known uses include: Jabber, MSN Messenger, Skype, ICQ

2.4 Pattern 3: Hosting a large-scale online meeting

Intent: To reduce the overhead of managing participation in an online conference.

Problem: As the organiser of a semi-open, large-scale online meeting you may wish to know who is currently present and, e.g., be able to exclude

specific people. At the same time, you should not be interrupted with individual requests to join and leave.

Symptoms: The organiser feels not in control of conference participation, or feels overloaded with participation details.

Solution: Allow for different conference management styles, matching the specific requirements of the setting. In this case: implement a registration mechanism that allows registered participants to join and leave at will, without intervention of the organiser. Provide the organiser with an overview of current participants and the option to exclude specific participants.

Rationale: While in small-scale meetings the organiser may wish to be in control of who is present at which parts of a meeting, in large-scale meetings the organiser may be overloaded with requests to join and leave. A registration mechanism can check who is allowed to join and leave, and reduce the overhead of managing participation.

Scenario: Different teleconference management styles are required e.g., for reasons of security, confidentiality and scalability. The virtual care team holds regular case conferences. Since the care team is not co-located, the case conferences are conducted by videoconference. These are relatively small scale videoconferences but with high confidentiality requirements and are managed with strict admission

control by the chair. In contrast members of the team also participate in regular global consensus forming conferences. At these conferences hundreds of oncologists from all over the world discuss ethical and scientific issues. The conferences are conducted alternately as co-located (face-to-face) conferences and as web-enabled teleconferences. In the teleconference setting the chair does not want to be overwhelmed by the need to explicitly authorize every join request. However, from time to time rogue and nuisance participants join and may need to be evicted by the chair.

Known uses include: WebEx, PlaceWare

2.5 Pattern 4: Co-editing content

Intent: To allow for simultaneous editing and independent viewing of shared content.

Problem: When using typical application sharing tools, only one person can make changes at a time, and participants cannot have individual views of the shared content. This restricts the way people collaborate using shared content.

Symptoms: People feel they have to wait unnecessarily for editing by others; they feel restricted in the way to view shared content.

Solution: Provide a mechanism where collaborating people can, dynamically, select collaboration-aware tools to share and manipulate specialised content.

Rationale: In contrast to a generic sharing mechanism coupled to a locking mechanism, collaboration-aware tools can allow different participants to have different viewpoints on the shared content, allow for simultaneous editing (e.g., via locking on a much more fine-grained scale), and independent access. Furthermore, having syntactic information about the content increases the options for specialised viewing, sharing and editing functions.

Scenario: Different members of the care team need to access and add to the EMR, possibly synchronously. For instance the visiting cancer nurse needs to update the record from the patient‘s home during her visit (e.g., she records measurements such as vital signs and medication information at the time). At the same time the remote oncologist may be adding notes and his secretary is adding laboratory test results to the EMR.

Known uses include: Groove, Habanero, Tango

2.6 Pattern 5: Defining roles

Intent: To allow different people to have different roles, and associated rights, during an online meeting.

Problem: In some cases, it is convenient, or even necessary, to restrict access to specific collaboration services to people with a specific role in the meeting. In other cases, explicit roles and rights may hinder the collaboration.

Symptoms: Participants explicitly articulate roles during a meeting.

Solution: Allow for different co-ordination styles, matching the specific requirements of the setting. When needed, individuals should be able to define and assign roles, and define the associated rights in terms of groupware services, e.g., defining who is allowed to edit shared content or expel other people.

Rationale: While social rules can be very effective to regulate who is allowed (or supposed) to do what, imposing those rules may be convenient or even necessary. E.g., in meetings with many participants one can appoint a "chairman" to coordinate the process.

Scenario: Different roles in the multidisciplinary medical team (e.g., oncologist, cancer nurse, pain specialist, physiotherapist) have different views of, and access rights to, parts of the administrative and clinical record. In a medical peer-review of treatment plans, only the treating oncologist is allowed to make modifications to the shared content; the consulted physicians and other health professionals are only allowed to view the content. Members of the care team from collaborating

services (e.g., social services, homecare support services) have further restrictions on access to patient information.

Known uses include: Groove, WebEx

2.7 Analysis of the patterns and consequences for design

The patterns described in this section provide only a few examples of high-level collaboration patterns. Nevertheless, these patterns indicate some interesting aspects about collaboration and consequences for groupware design.

First of all, they illustrate the fact that there are many different forms of collaboration, each with different requirements in terms of technology support. Collaboration patterns indicate areas where groupware design needs to be flexible; the solutions indicate how the provided groupware *service* should be changed to solve the problem:

- Patterns 1 and 2 illustrate different mechanisms that may be needed in the initial stages of a meeting, when contacting other people. Consequently, a groupware service design should allow for different mechanisms to initiate collaboration.
- Pattern 3 illustrates the fact that aspects such as group size and membership style (open or closed) determine the appropriate conference management style. As a consequence, conference management behaviour should be adaptable.

- Pattern 4 demonstrates that different collaborative settings require different collaboration tools to share content. Similarly, different settings may also require different communication support. As a result, these aspects of groupware services have to be very flexible, and should even be adjustable in the course of a meeting.
- Finally, pattern 5 illustrates that in some cases it should be possible to assign roles and define associated rights in relation to groupware services. Consequently, a groupware service design should allow for different co-ordination mechanisms.

The previous aspects are not independent of each other: e.g., co-ordination services may influence who is allowed to perform what operations on content sharing tools. Therefore, a groupware service design should also describe the relevant relations between services.

From the foregoing it should be clear that simply describing collaboration patterns is not sufficient; groupware needs to be designed for flexibility to enable individuals to collaborate using their preferred collaboration patterns.

3 SERVICE-ORIENTED GROUPWARE DESIGN

We consider the service a groupware application provides, i.e., its *behaviour* to support co-operating people, to be the most important aspect of the system. The external observable behaviour of a system determines

whether that system is effective and efficient for a specific purpose and also if it is pleasurable to use.

However, our experience is that current groupware design methodologies typically focus on implementation aspects instead of on services: the designer's attention is focussed insufficiently on the *behaviour* that is to be provided. As a result, these methodologies insufficiently help designers to see the "bigger picture", showing the key properties of the groupware application to be designed.

Therefore, we conclude there is a need to apply a service-oriented methodology to the design of groupware. Based on the analysis in section 2.7, we argue that any groupware design should explicitly pay attention to the dynamics of collaboration, in particular to: alternative invitation mechanisms, adaptable conference management functions, mechanisms for concurrent content sharing including different access modalities, and different roles and associated rights for participants.

3.1 Service orchestration by end users

People who are working together know best how they would like to collaborate. Or, to put it differently, the end users of a groupware application know best what collaboration patterns they prefer and when. Moreover, involving system administrators or programmers to perform (small) adjustments to the services provided implies an investment in

terms of time, money and communication needed to explain the issue. While involving system administrators or programmers will remain necessary for more specialised or difficult service adjustments, end users themselves may be enabled to perform some frequently needed adjustments. In this article we propose a mechanism that enables end users to orchestrate the services that support them in their collaboration.

This form of adaptation by end users in the context of application usage is typically termed *tailoring* (Kahler, Mørch, Stiemerling, & Wulf, 2000). We strive to provide end users with appropriate means to perform these adjustments, matching their skills and needs, although both factors differ per individual, are difficult to establish and are likely to change over time.

Real-life, high-level collaboration patterns provide valuable indications of the types of adjustments that may be needed. We aim to match the end users' skills primarily by providing them with suitable units to select and compose services: recognizable groups of collaborative functions of an appropriate granularity, and using the end users' (domain) terminology. Section 4.2 provides more information about the applied design choices. Given this focus, this article does not elaborate on graphical user interface design of tailorable groupware, which is of course an additional important aspect for groupware designers.

4 THE COOPS GROUPWARE REFERENCE MODEL

The CooPS groupware reference model structures groupware services. It does so by prescribing high-level functional building blocks that can be selected and composed to form groupware services. This section introduces the various types of functional building blocks the model distinguishes and the design choices associated, in relation to the analysis of the collaboration patterns.

4.1 Groupware service composition

We apply the term *groupware service* to denote the (distributed) behaviour that a groupware application provides to a group of co-operating individuals. In reasoning about groupware services, one can distinguish units to form such a service, which we denote as Groupware Service Modules (GSMs). GSMs form *units of composition* of groupware *behaviour*: one can adapt a groupware service by selecting and composing GSMs.

It is important to note that this notion of behaviour building blocks is different from the more commonly used implementation building blocks, i.e., software components. Although a behaviour building block may be implemented by a software component, there is no implied one-to-one

relation. As stated in section 3, we consider the resulting behaviour to be more important than the way in which that behaviour is implemented.

While GSMs are groups of (related) groupware functions designed for selection and composition by end users, one can even distinguish units on a smaller scale. As illustrated in figure 1, one GSM consists of multiple Groupware Service Module Elements (GSMEs). A GSME is a *unit of groupware behaviour*, one atomic groupware function. These GSME types allow one to *describe* and *compare* the external observable behaviour of groupware applications. However, GSMEs are typically too fine-grained for end user selection and composition of groupware behaviour.

While this article focuses on defining GSM types and their relation with collaboration patterns, it is worthwhile to mention that the set of atomic groupware functions, i.e., the GSME types, should be sufficient to express the external behaviour of a wide range of groupware applications. Slagter (2004) describes the process to derive these GSME types and the details of all individual GSME types.

As, in theory, an infinite number of concrete GSM and GSME *implementations* may exist, our approach focusses on GSM *types* and GSME *types*.

In contrast to the *descriptive* GSME types, the GSM types in this article are *prescriptive*: they define behavioural building blocks that can be composed to form adaptable groupware services. Figure 2 illustrates the

most relevant stages in the lifecycle of a GSM. The names on the right-hand side denote associated GSME types, that is functions identified in our groupware service model which cause a transition between stages in the GSM lifecycle.

4.2 Design choices

As GSMs are service building blocks, the most important design choice involves the choice of groupware behaviour (i.e., GSME types) to bundle. When deciding on appropriate GSM types, we applied the following heuristics:

- Design from the point of view of the end user who selects and composes GSMs: what bundles of services are recognisable and usable for end users?
- GSMs should be units of selection and composition. GSMs should be as independent from each other as possible.
- Decide on granularity based on the required flexibility: the GSM types should neither be too fine-grained, as that complicates the selection and composition process, nor too coarse-grained, in order to provide sufficient flexibility to accommodate a wide range of collaboration patterns.
- Define different GSM types for different concerns.

- Define GSM types to be as generic as possible. The definition should allow for a wide range of groupware implementations.
- Avoid unnecessary behaviour. A GSM definition should only include required behaviour.
- Group behaviour that is always used together in one GSM type.
- Avoid overlap in provided behaviour. Overlap in the behaviour of GSM types is an indicator that concerns are not properly separated.
- Keep GSM designs consistent: make sure that similar behaviour is defined in a similar way over multiple GSM types.

As is clear from the previous points, these heuristics may conflict. For instance, a generic GSM design may include unnecessary behaviour. It is the designer's task to consider the various conflicting forces and maintain the overview. In this process, the overriding principle is that end users should be able to recognise, select and combine the resulting GSMs.

4.3 Overview

Application of this design process resulted in the CooPS groupware reference model. This reference model, depicted in figure 3, provides structuring guidelines to design tailorable groupware services. It prescribes GSM types, states the responsibilities of these building blocks in terms of the behaviour they provide (i.e., the GSME types they consist of) and describes their relations on a service level.

Sections 4.4 to 4.9 provide more details regarding the various GSM types illustrated in figure 3, as well as the sixth GSM type, the *bootstrapping GSM*.

4.4 Conference Management GSM

The term conference management refers to the collection of processes relating to starting, stopping, joining and leaving online conferences. Pattern 3 illustrates the fact that different aspects determine the appropriate conference management style. These aspects include the duration of the conference, the number of participants and the nature of conference membership (static or dynamic, open or closed). By allocating conference management behaviour to a separate type of building block one can select the appropriate conference management ‘flavour’, independent of, for instance, the communication modalities used.

The conference management GSM (CM-GSM) provides end users with the essential behaviour needed to manage on-line conferences. It is primarily responsible for providing conference management groupware services: behaviour to start and end conferences, to manage the set of conference participants, to manage the communication and collaboration tools that are active in the conference and to manage which co-ordination engine is active in the conference.

To perform these last two functions, the CM-GSM provides behaviour (i.e., GSME types) to manage which communication / collaboration GSMs (described in the next section) are associated with the conference, as well as the associated co-ordination GSM.

In this way, the CM-GSM can enable as well as disable behaviour provided by those GSM types. In turn, the behaviour provided by a CM-GSM can be subject to the rules defined by a co-ordination policy. Such a policy is defined and enacted by a co-ordination GSM, which is described in section 4.6. See section 4.10 for more information about the service-level relations between GSM types in the reference model.

According to the reference model, every composition of GSMs associated with one online conference contains exactly one CM-GSM instance. Since the reference model specifies the *distributed* behaviour of GSMs this one CM-GSM instance may in fact be implemented by a series of software component that are physically distributed over the various participants in the online conference.

4.5 Communication / Collaboration GSM

Communication / collaboration GSMs (CC-GSMs) provide end users with the behaviour to actually collaborate: a CC-GSM provides the behaviour associated with one communication tool or one tool to share content. An example of a CC-GSM is a building block to communicate via speech, or

a building block to share and annotate digital x-ray images. Access to the behaviour provided by a CC-GSM can be subject to the rules defined by a co-ordination policy, enacted by a co-ordination GSM as described in the next section.

As illustrated by pattern 4, different collaboration patterns are likely to require different support in terms of the modalities to communicate and the types of content to share. By allocating the associated behaviour to a separate GSM type, the reference model allows end users to select on a fine-grained level the support that matches their collaboration independent of, for instance, the conference management ‘flavour’. A configuration of GSM instances associated with a single conference may include multiple instances of CC-GSMs.

4.6 Co-ordination GSM

A co-ordination GSM (CO-GSM) is primarily responsible for providing end users with the behaviour needed to define and enact a co-ordination policy; it can be considered to be a co-ordination engine. A co-ordination policy associates participants with access rights to groupware behaviour, for instance based on user roles.

The reference model prescribes what types of groupware behaviour can be subject to co-ordination as well as the service-level dependencies between the GSM types involved. In this way, the reference model

increases the degree of independent extensibility: it allows one, for instance, to compose CC-GSM implementations (i.e., collaboration tools) from one vendor with a CO-GSM implementation from another vendor, while the behaviour provided by the CC-GSMs can still be co-ordinated by the CO-GSM. As such, the service-level specifications form abstract interfaces that can be considered contracts between GSM implementations.

By allocating co-ordination behaviour to a separate GSM type, the reference model allows for a single co-ordination policy that covers, for instance, all active content sharing tools. By separating this behaviour from conference management behaviour the reference model allows one to independently select both aspects. Pattern 5 illustrates that one should be able to select an appropriate co-ordination mechanism. In fact, in many cases, no co-ordination is needed at all. As a result, one or zero instances of CO-GSMs are active in a single online conference.

4.7 People Listing GSM

People listing GSMs (PLIST-GSMs) provide end users with awareness information about other people, in order to facilitate the process of initiating communication with these people.

Pattern 1 and 2 illustrate different issues during the initiation of online conferences. The PLIST-GSM bundles the behaviour to facilitate this

process in general, and the behaviour to convey awareness information about people to invite in particular. By allocating this behaviour to a separate GSM type the reference model allows one to select the mechanism to initiate the collaboration independent from the way the collaboration itself is actually supported. Multiple instances of PLIST-GSMs may be active in a single online conference.

4.8 Conference Listing GSM

Conference listing GSMs (CLIST-GSMs) provide end users with awareness information about other ongoing conferences, in order to facilitate the process of joining these conferences.

One particular form of starting communication is to join an existing online conference. The CLIST-GSM bundles the behaviour to facilitate this process. Since the behaviour provided is, especially from the end users' point of view, different from the behaviour by the PLIST-GSM and because of the different dependencies with the CM-GSM the reference model includes these two different types of building blocks to initiate collaboration. Multiple instances of CLIST-GSMs may be active in a single online conference.

4.9 Bootstrapping GSM

The bootstrapping GSM (B-GSM) provides end users with behaviour to make GSMs available for activation and to actually activate and connect GSMs. Additionally, the B-GSM provides basic behaviour to store user information and preferences, and to activate and compose a series of GSMs based on a template. Such a template describes a configuration of GSMs, for instance corresponding to a specific collaboration pattern. As such, a B-GSM facilitates the process of launching a groupware service to support a collaboration pattern. An end user requires one instance of a B-GSM to launch other GSMs, which may be associated with different online conferences.

4.10 Relations between GSM types

The CooPS groupware reference model describes the relations between GSM types on a service level. Three types of relations between GSM types exist:

- A GSME type allocated to one GSM type may enable a GSME type allocated to another GSM type. For example, the CM-GSM behaviour to add a CC-GSM to the groupware service composition enables the collaboration behaviour associated with that CC-GSM.
- A GSME type allocated to one GSM type may disable a GSME type allocated to another GSM type. For example, the co-ordination

behaviour provided by the CO-GSM type may disable collaboration behaviour provided by a CC-GSM.

- A GSME type allocated to one GSM type may change status information that is used by a GSME type allocated to another GSM type. For example, the CM-GSM behaviour that allows one to join a conference changes status information about the set of conference participants. This information, which is maintained by the CM-GSM, is used in the behaviour to assign roles, which is allocated to the CO-GSM.

A more elaborate description of the service-level relations between GSM types is provided in Slagter (2004).

4.11 Conclusions

The CooPS groupware reference model defines structuring guidelines for tailorable groupware. It advocates that designers first capture on a service level the key properties and key relations of a groupware design. The reference model defines six types of Groupware Service Modules (GSMs), 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 groupware service provided.

GSMs form functional building blocks of groupware applications: a GSM can, for example, be realized using one or more software components. The details about the various GSM types and associated design choices are described in Slagter (2004).

5 ORCHESTRATING GROUPWARE SERVICES

In a design based on the CooPS groupware reference model, orchestrating groupware services is possible by adding or removing GSMs to the composition associated with an online conference. The resulting groupware service should match the way the users actually collaborate, i.e., their collaboration patterns. While end users should be able to orchestrate groupware service, groupware designers can accelerate this process by preparing a series of default GSM configurations for recurring collaboration patterns.

For example, a designer can study how teams in an organisation currently collaborate online and how they prefer to work online. In this process, a designer should explicitly pay attention to the aspects of collaboration mentioned in section 2.7; these aspects help to discover and describe the variations between the collaboration patterns that should be supported.

For each of the identified recurring collaboration patterns a configuration of GSMs should be defined that properly supports the

pattern. The expected result is a set of default configurations that forms a starting point for service orchestration: by selecting the appropriate configuration end users can select and compose groupware services to support them.

We advocate increasing the recognizability of default configurations by describing their purpose using the terminology of the end users' domain, for instance by describing the collaboration pattern they are designed to support. We expect that an increased recognizability of configurations will facilitate the orchestration of groupware services by end users.

In cases where the predefined patterns do not match, end users should be able to fine-tune the default configurations, or create new configurations, appropriate for their specific way of collaborating.

The option to create new configurations is likely to be used less frequently than the other types of adaptations. Nevertheless, end users should be able to discover:

- What GSMs are available;
- Which behaviour (service) these GSMs can provide, i.e., how they support collaboration. Note that such service descriptions are subjective and context-dependent;
- How to include the GSM in the groupware service configuration.

It should be clear that designing an appropriate graphical user interface for tailoring is not trivial. We refer to Wulf (1999) for an example of a tailoring interface for a shared search tool: the presented tailoring interface addresses the issue of showing the different types of building blocks and how they can be connected.

As the reference model concerns the *distributed behaviour* of a groupware application, orchestrating groupware services typically has an impact on other participants in an online conference as well; adapting the composition of GSMs associated with an online conference typically affects the service provided to the other participants as well. Groupware designers should be aware of this fact and should, for instance, implement appropriate awareness mechanisms to convey that the provided service has been changed and who or what caused the change.

6 EXPERIENCES

Based on the CooPS groupware reference model a proof-of-concept demonstrator has been implemented. While this demonstrator has yielded some initial experiences with selecting and composing groupware behaviour, it has not been applied in large-scale settings yet.

6.1 Interviews

To have an indication whether the GSM types defined in the CooPS groupware reference model are suitable units of selection and composition

we held a series of interviews with six experienced groupware users (from the healthcare domain and from business/consultancy) who had no knowledge of the reference model. In these interviews, the test subjects were provided with a series of scenarios. For each of these scenarios they 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 results showed that the test subjects identified groupware units of a similar granularity as the ones defined by the CooPS groupware reference model. The test subjects all distinguished units corresponding to CC-GSMs, while five out of six distinguished units corresponding to the CM-GSM. Three test subjects identified separate co-ordination functions. Most interestingly, the majority of test subjects used the term “being able to” to describe the units of a groupware application. The latter supports our belief that the *service* a groupware unit provides, i.e., what it can do for end users, is its most important aspect.

6.2 SAAM analysis

We also evaluated the CooPS groupware reference model using the Software Architecture Analysis Method (SAAM) (Abowd, 1998).

Although this method is primarily used to compare alternative (software) architectures, it can also be applied to assess a single architecture.

SAAM is designed to help articulate the purposes of an architecture and then determine the degree to which a given architecture meets them. The method uses scenarios of use. To assess our design we applied the scenarios of groupware use and adaptation by Tietze (2001) as benchmark scenarios.

The results of the analysis indicate that the CooPS groupware reference model is appropriate to support a wide range of cases of groupware use and adaptation: it supported the vast majority of Tietze's scenarios. Three of the scenarios were not directly supported, and required adaptations to the model. The SAAM analysis also provided indications that the reference model exhibits the favourable properties of *high cohesion*, *low coupling*, and a limited structural complexity.

7 DISCUSSION

A service design specifies the external observable behaviour that has to be provided; it does not specify *how* that behaviour should be implemented. Section 7.1 discusses issues around implementing the prescribed behaviour and physically distributing software implementations. Section 7.2 positions the reference model in relation to other approaches.

7.1 Implementation and physical distribution

Given the reference model, groupware programmers are free to decide on how to implement GSM behaviour and which technologies to use. However, the fact that various GSM implementations have to be able to make use of each other's services imposes constraints with regard to interoperability.

One way to promote interoperability between GSM implementations is to define a groupware framework. Such a framework provides a runtime environment in which GSM implementations can be instantiated, discover and connect to each other. These connections can both be local (between GSM implementations that provide services for one particular individual) as well as remote (between GSM implementations of different participants in an online conference).

As such, the CooPS groupware reference model does not impose a particular physical distribution: GSM implementations may use a client-server distribution, a peer-to-peer distribution, or some hybrid form.

7.2 Relations to other approaches

Existing groupware reference models are typically aimed to facilitate the *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.

For example, two well-known groupware reference models, Dewan's generic collaborative architecture (Dewan, 1998) and the Clover architecture (Laurillau & Nigay, 2002), both focus on the internal structure of groupware applications. They decompose collaborative applications based on functional layers. Although both help designers to structure groupware implementations, they do not help in capturing key behaviour of such applications on a service level. Additionally, these approaches also do not describe how a service-level specification should be mapped on the various layers.

8 CONCLUSIONS

The basic motivation for our research is that the *behaviour* provided by groupware applications should seamlessly match the requirements of the collaborative setting they should support. Our approach is based on the notion that different forms of collaboration, i.e., different collaboration patterns, require different combinations of groupware services. However, current groupware systems are not designed for this purpose, which makes it difficult, or even impossible, to adapt them to specific collaboration patterns.

This article presents a structuring of groupware services that allows end users to orchestrate the provided services in order to match their collaboration patterns.

Based on the analysis of a series of high-level collaboration patterns from the healthcare domain we have illustrated where flexibility in groupware service design is needed. Subsequently, we have introduced the CooPS groupware reference model: a structuring of groupware services in terms of Groupware Service Module (GSM) types. This structuring enables end users to select and compose groupware services, i.e., groupware *behaviour*, to match their collaboration patterns.

Section 5 illustrates how this structuring helps end users to select and compose groupware services in relation to their collaboration patterns. The fact that end users should be able to select and compose groupware services has had a considerable influence on the design: the identified GSM types have to be recognisable and usable bundles of groupware behaviour for end users.

While end users should be able to orchestrate groupware service, groupware designers can accelerate this process by preparing a series of default GSM configurations for recurring collaboration patterns. To increase the recognizability of these default configurations, we advocate using the end users' terminology to describe the purpose of these

configurations, for instance via a description of the collaboration pattern they are designed to support.

We conclude that in order to benefit from a series of high-level collaboration patterns, one needs to design groupware in such a way that collaborating people can easily work according to those patterns. Therefore, the collaborative pattern community needs a structuring mechanism for groupware services, such as the one presented in this article. A groupware design based on our structuring allows end users to orchestrate groupware services to match their collaboration patterns.

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.

Bardram, J. E. (1998). Designing for the dynamics of cooperative work activities. In *Proceedings of the ACM 1998 conference on Computer Supported Cooperative Work* (CSCW '98) (pp. 89-98). New York: ACM Press.

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.

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.

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.

Pitsillides, A., Samaras, G., Dikaiakos, M., & Christodoulou, E. (2005). DITIS: Collaborative Virtual Medical team for home healthcare of

cancer patients. In: *Conference on the Information Society and Telematics Applications*, Catania, Italy, 16-18 April 1999.

Schuemmer, T. & Slagter, R. J. (2004). The Oregon Software Development Process. In J.Eckstein & H. Baumeister (Eds.), *Extreme programming and agile processes in software engineering: 5th International Conference, XP2004* (pp. 148-156). Berlin: Springer-Verlag.

Slagter, R. J. (2004). *Dynamic Groupware Services: Modular design of tailorable groupware*. Enschede, The Netherlands: Telematica Instituut.

Tietze, D. A. (2001). *A Framework for Developing Component-based Co-operative Applications*. Sankt Augustin, Germany: GMD.

Wulf, V. (1999). "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.

FIGURES

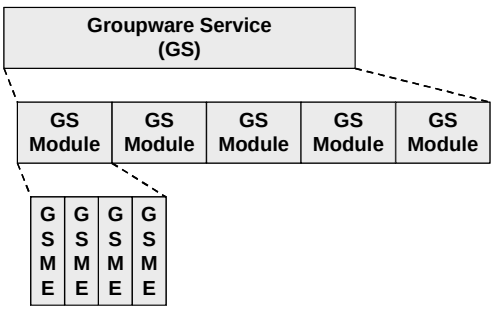

Figure 1 Consists-of relations between Groupware Service, Groupware Service Modules and Groupware Service Module Elements

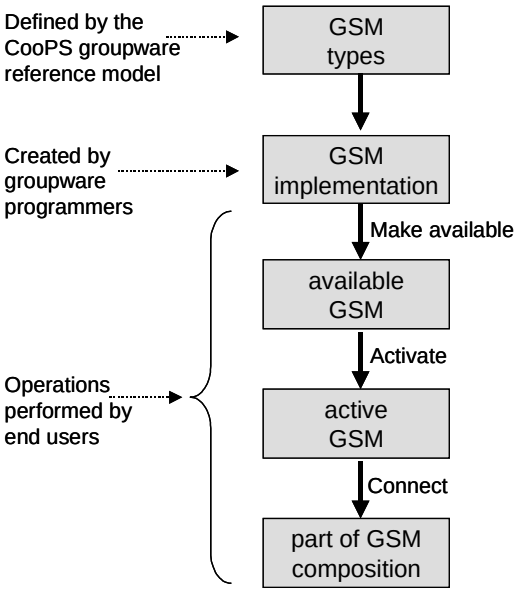

Figure 2 Most relevant stages in the lifecycle of a GSM

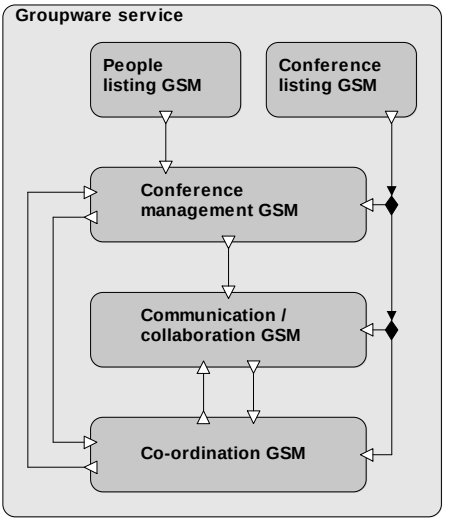

Figure 3 GSM types in the CooPS groupware reference model