Component-Based Groupware Development Based on the 3C Collaboration Model

Marco Aurélio Gerosa[1], Alberto Barbosa Raposo[2],
Hugo Fuks[1], Carlos José Pereira de Lucena[1]

[1]Laboratório de Engenharia de Software (LES), Departamento de Informática
[2]Grupo de Tecnologias em Computação Gráfica (Tecgraf), Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro – PUC-Rio
Rua Marquês de São Vicente, 225, Rio de Janeiro, RJ, 22453-900, Brasil

{gerosa, hugo, lucena}@inf.puc-rio.br, abraposo@tecgraf.puc-rio.br

Abstract. *Groupware is evolutionary and difficult to develop and maintain. Thus, its code becomes unstructured and difficult to evolve. In this paper, a groupware development approach based on components organized according to the 3C collaboration model is proposed. In this model, collaboration is analyzed based on communication, coordination and cooperation. Collaboration requirements of the group, analyzed based on the 3C model, are mapped to software components, also organized according to the model, in order to compose the system. The proposed approach is applied as a case study to the development of the new version of the AulaNet environment. The environment's code currently suffers from the aforementioned problems. In order to instantiate the environment's communication services, 3C based component kits were developed for the case study. The components allow composition, re-composition and customization of services to reflect changes in the collaboration dynamics.*

1. INTRODUCTION

Douglas Engelbart [1968] pointed to the relevance of applications for office automation, hypertext and groups. Today the first two are widely available, used and commercially accepted, while groupware technology is still perceived to be unstable and commercially risky, as well as possessing few products [Greenberg 2006]. In most companies, computational support for collaboration is limited to systems for exchanging messages or filing documents.

Groupware technology has still not attained its use potential. Research at CSCW is now at a fairly advanced stage, yet it lacks a manner of simplifying the programming of collaborative applications and promoting a critical mass of users. Groupware development still requires qualified programmers trained to deal with protocols, connections, resource sharing, distribution, rendering, session management, etc. This limits the number of developers active in the area and dislocates the creativity and efforts of these developers from the creation of solutions to the solving of low-level technical problems, relegating the investigation of interactivity and support for group collaboration to a secondary level.

These kinds of problems in developing groupware are experienced in the development and maintenance of the AulaNet environment. AulaNet is a web-based groupware solution. AulaNet has been under development since 1997 and is widely used. The

AulaNet development group is made up of doctoral, masters and graduate students who, as well as maintaining the software, use it in their theses, dissertations and monographs, implementing and testing the concepts produced in their work. The system has grown through prototyping, while its functions have been implemented in evolving fashion. The constant changes in the support for collaboration and the evolution of the technologies used has made the application's code strongly linked and with a low level of cohesiveness. Technical aspects permeate the entire code, becoming mixed with the collaboration support, diverting the developer's attention. Changes in the environment are reflected in various parts of the code and cause undesirable collateral effects, hindering the evolution of the environment, integration of new members to the development team and integration with the company responsible for distributing and customizing the environment.

This scenario illustrates the need to support groupware development, enabling developers to build an extendable system, in a way more suited to accompanying the evolution of collaboration support and the characteristics of tasks and groups. The low-level complexities should be encapsulated and the investigation of interaction through prototyping should be better supported. Non-specialized developers should be able to adapt and reconfigure the solution for their specific needs – a desirable aim, given that there is no way of foreseeing all the collaboration demands [Pumareja et al. 2004].

This article proposes the use of 3C based components as a means of enabling the development of extendable groupware whose assembly is determined by collaboration needs. By conceiving the problem from the viewpoint of the 3C model and using a component structure designed for this model, changes in collaboration are mapped into the computational support, which is replaced or added as required. The developer is provided with a component-based infrastructure designed specifically for groupware, based on a collaboration model.

Each group using a collaborative environment has specific collaboration requirements. By designing and developing the collaboration tools in the form of software components, the developer is given the means to assemble a specific groupware for the collaboration needs of the group. The tools are selected from a component kit based on the 3C model in support of the established dynamics. The developer selects the components most suited to the situation in question from those with the same purpose. These components encapsulate the technical complexities and collaboration support and help the developer concentrate on the composition of a specific groupware solution. The components encapsulate business implementations and rules on collaboration, provided by specialists from the domain and obtained by experimentation, and reused in a variety of situations. As a result, the modeling is enabled on the basis of predefined models and elements.

The proposed approach is being applied in the re-development of the AulaNet environment as a case study. The new version of AulaNet is being developed with the capacity to recompose the environment, reuse its services in various situations and reconfigure them to accompany the evolution of the work processes and group characteristics. A layered architecture is used containing component frameworks to deal with services and collaboration components deriving from a component kit. The component kit is obtained from domain engineering that allows for reuse and interoperability.

2. COMPONENT BASED GROUPWARE

Szyperski [2003] lists four main reasons for using component software. The first and oldest is related to the idea of a *components market* in which companies search and purchase components. The second reason is related to *product line*. Components are developed with the aim of reusing them in various systems, reducing the total investment and maintenance costs. The third is related to the idea of assembly by the final user (*tailorability*). The fourth reason is related to the use of *dynamic services*, discovered and installed as they become necessary. Some component based groupware are presented bellow.

The LIVE platform [Banavar et al. 1998] provides support for the construction of synchronous groupware. The component model used by LIVE is based on the JavaBeans specification and supplies a high-level interface for developing groupware. DISCIPLINE [Marsic 1999] is a platform designed for the development of synchronous groupware for the educational domain. DISCIPLINE's architecture consists of replicated components and resources centralized on the server.

FreEvolve [Won et al. 2005], previously called EVOLVE [Stiemerling et al. 1999] (before its release under the GPL license), is a component-based system developed in a client-server architecture on the Internet. FreEvolve was designed to enable adaptation and assembly by final users of the application. The component model used in FreEvolve is called FlexiBeans, an extension of JavaBeans.

The DACIA platform (Dynamic Adjustment of Component InterActions) [Litiu & Prakash 2000] is aimed towards the development of groupware for mobile devices. The platform defines its own component model. In this model, a component is called PROC (Processing and ROuting Component).

DreamTeam [Roth & Unger 2000] is a component-based platform for assembling synchronous groupware. The platform provides a development environment, with specific tools, an execution environment and a simulation environment. The components are called TeamComponents and are associated with user interface and data manipulation components. The developer is supplied with groupware components, user interface components and data manipulation components.

The CoCoWare platform [Slagter & Biemans 2000] offers final users the capacity to assemble the application according to their needs and extend it to follow the evolution of work processes. CoCoWare offers components for dealing with work sessions and managing collaborative tools, providing information on what can be modified, how it can be done and the impact of the modifications.

GroupKit [Roseman & Greenberg 1996] is a toolkit containing components and an execution platform. GroupKit is built in Tcl/Tk and is aimed towards the development of synchronous groupware. The toolkit encapsulates various complexities inherent in this type of application, meaning developers can focus their attention on the interaction.

The approaches found in the literature concerning the use of software components in groupware development basically focus on the second and third reasons identified by Szyperski [2003] (*product line* and *tailorability*). Some systems provide tools for building collaborative applications based on interoperable components, while some offer the possibility of assembly by final users. There is no standard component model.

The approach proposed in this article focuses on the second reason (*product line*), aiming to provide groupware developers with the tools to assembly collaborative systems based on the 3C collaboration model. However, some flexibility is offered to final users, who can select and install new collaboration services. This enables them, to a certain extent, to adapt the collaborative application to their specific needs, following the characteristics of the group and the tasks.

The literature contains various proposals for using software components in the construction of groupware. However, none of these uses the 3C collaboration model, or other collaboration model, as a basis for designing and organizing software components and the development process.

3. THE 3C COLLABORATION MODEL

Projecting high quality groupware demands an understanding of collaboration. The purpose of this section is to introduce the 3C model. Collaboration is analyzed as a guide to developing groupware.

The 3C collaboration model is based on the idea that to collaborate, members of a group communicate, coordinate and cooperate. The 3C model derives from the seminal article by Ellis et al. [1991]. The model proposed by Ellis et al. is used to classify computational support for collaboration. In this article, the 3C model is used as a basis for modeling and developing groupware. There is also a difference in terminology; the joint operation in the shared space is called collaboration by Ellis, while it is termed cooperation in the 3C model.

The 3C model is equivalent to the Clover model [Laurillau & Nigay 2002]. This model defines three classes of functionalities: communication, coordination and production. The production modality in the Clover model corresponds to the concept of cooperation in the 3C model. In contrast to the Clover model, in this article, the 3C model guides the development of component groupware and gives rise to a component-based architecture.

The 3C model is widely used in the literature. Bandinelli et al. [1996] use the three dimensions of the 3C model to improve the computational support of software processes, especially communication and cooperation, which, according to them, are not adequately treated by traditional processes, which are designed more for coordination. Bretain et al. [1997] use the three Cs as a basis to analyze and interview groups whose activities are conducted outside of offices, such as firefighters, plumbers, reporters and sales representatives, in order to design a multimedia and mobile computational support adequate to the needs of each group. Sauter et al. [1995] use the three Cs to classify groupware in Swiss companies. Borghoff & Schlichter [2000] use the three Cs to classify collaborative tools. Marsic & Dorohoceanu [2003] use the three Cs to analyze user interface elements.

A diagram of the 3C model is shown in Figure 1. Communication involves the exchange of messages and the negotiation of commitments. Coordination enables people, activities and resources to be managed so as to resolve conflicts and facilitate communication and cooperation. Cooperation is the joint production of members of a group within a shared space, generating and manipulating cooperation objects in order to complete tasks. Despite the separation of these activities for analytic purposes,

communication, coordination and cooperation are not undertaken in a hermetic and isolated fashion; they are undertaken continually and repeatedly during group work [Fuks et al. 2005]. Tasks arise from commitments negotiated during communication, are managed by coordination and are undertaken during cooperation. Through awareness mechanisms, individuals receive feedback from their actions and feedthrough from the actions of their colleagues. Cooperation demands renegotiating and making decisions on unexpected situations, which in turn requires new rounds of communication and coordination.

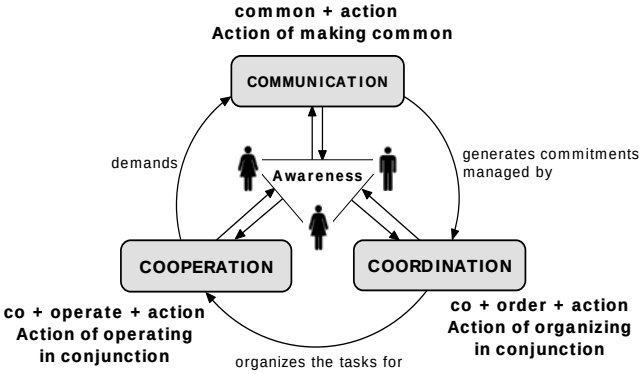

Figure 1. Diagram of the 3C collaboration model

Collaboration can be broken down into activities and each activity broken down into subactivities with their own planning, participants and methodologies. Each of these subactivities possesses distinct communication, coordination and cooperation needs. Before effectively carrying out a task, for example, the group organizes itself. This activity also demands specific collaboration needs, which are distinct from the needs that occur during the execution of the task. The individuals responsible for planning may not be the same as those executing the tasks. For example, in the assembly line, the activities are planned and subsequently each individual carries out his or her tasks without interacting directly with the others. In collaboration, the plan is renegotiated dynamically, making it impossible to separate fully coordination from cooperation. Collaborating, individuals learn and refine the work processes, renegotiating the initial plans and interspersing action and negotiation. Groupware should support this flexibility in renegotiating plans and performing communication, coordination and cooperation in parallel. A specific communication activity, such as chat for example, requires communication (exchange of messages), coordination (access policies) and cooperation (registration and sharing).

4. Assembly of Groupware and Collaborative Services

A groupware environment normally offers the participant a set of collaborative tools used in the different moments of collaboration. Table 1 shows the tools found in the following groupware systems: AulaNet (http://www.eduweb.com.br), TelEduc (http://teleduc.nied.unicamp.br), AVA (http://ava.unisinos.br), WebCT (http://www.webct.com) and Moodle (http://www.moodle.org), all from the educational domain, and GroupSystems (http://www.groupsystems.com), YahooGroups (http://groups.yahoo.com), OpenGroupware (http://www.opengroupware.org) and BSCW (http://bscw.fit.fraunhofer.de), designed for group work.

	Communication Services							Coordination Services								Cooperation Services												
	Mail	Discussion List	Forum	Mural	Brainstorming	Chat	Messenger	Agenda	Activities Report	Participation Monitoring	Questionnaire	Tasks	SubGroups	Resource	Guidance	Voting	Repositories	White Board	Search	Glossary	Links	Cooperative Journal	Classifier	Wiki	Contact Manager	Peer Review	FAQ	Notes	RSS
AulaNet	X	X	X			X	X		X	X	X	X	X			X	X				X							X	
TelEduc	X		X	X		X		X	X	X	X	X	X				X				X						X	X	
AVA	X		X	X		X		X	X	X	X	X			X		X			X	X						X	X	
WebCT	X		X			X		X	X		X	X				X	X	X	X	X							X		
Moodle			X			X	X	X	X	X	X	X	X			X	X		X	X	X	X		X		X	X		X
GroupSystems			X		X			X		X	X	X	X			X							X					X	
YahooGroups		X				X		X	X		X					X	X		X	X	X								X
OpenGroupware	X			X				X				X		X											X				
BSCW			X					X	X			X	X			X	X		X		X				X				

Table 1. Collaborative tools found in groupware

As the table shows, a variety of similar tools are used in groupware. For example, most of the analyzed systems provide forums, chat, agenda, activity reports, questionnaires, task management, voting, repository and links. Each tool can be seen in a relatively isolated form within the environment. These characteristics are well suited to the application of component-based development techniques, where the collaborative tools are the groupware components. The environments would provide a set of components instantiated and customized for each group and collaboration dynamics.

In this article, collaboration tools are called services. Services can be classified according to their purposes and characteristics using the 3C model. The services in Table 1 are organized into communication, coordination and cooperation. In a component-based environment, the developer selects the services most suited to the group's collaboration needs. A unified components model would enable the developer to select from the most suitable service from all those performing the same purpose. As can be noted in Table 1, there are services that are provided by one groupware solution only, but which in principle could be made available for use in all the other solutions. For users, it is interesting to interchange services in order to complement environments and reuse experiences.

As well as the services provided by the analyzed groupware solutions being similar, these services also possess similar functionalities. Almost every chat possesses a shared area where messages are displayed, a list of connected participants and an area for writing messages. By using a component-based architecture, these characteristics can be reused.

For example, Moodle's chat service displays the time the participant remains inactive, a beep for attracting a participant's attention and each participant's photo. WebCT provides support for private messages and notifies when someone enters the chat room. Encapsulating these functionalities in components would also allow the application developer to make the various aspects of collaboration support available for reuse, meaning other developers can use them to select the functionalities best suited to the groups and activities in question. Encapsulating recurrent functionalities into components enables the reuse of each service's computational collaboration support, increasing the reuse of code. It also becomes possible to evolve, adjust and build services by varying and reconfiguring the collaboration components.

These scenarios indicate the need to adopt software componentization at two levels, as illustrated in Figure 2. The first level contains the components that provide the collaborative services, used to offer computational support to the collaboration dynamics as a whole. The second level contains the components used to assemble the services, providing support to specific aspects of the collaboration within the dynamics of a particular activity. In the proposed approach, the components that implement the collaborative tools are called *services* and the components used to implement the computational support for service collaboration are called *collaboration components*.

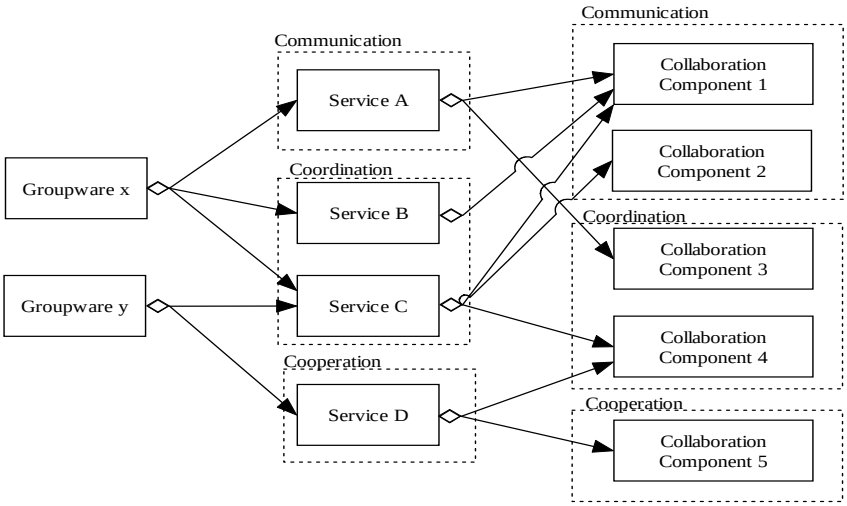

Figure 2. Groupware composition

A groupware solution is composed of services, which are reusable in various other groupware solutions. The services share collaboration components that implement collaboration support, modeled in this article based on the 3C model. Based on component kits organized according to the 3C model, the developer assembles an application to provide support to the collaboration dynamics. As well as allowing reuse, this approach also increases the solution's capacity to be extended by allowing the inclusion of new components. In this approach, even a communication service, as a discussion forum, besides the communication components, also uses coordination and cooperation components. The collaboration components of one C are reused in the services of the other Cs.

4.1. The Collaboration Component Kit

The proposed approach aims to provide the developer with component kits to be used in assembling groupware solutions and collaborative services. Domain engineering is needed to obtain the set of components. Domain engineering aims to provide components that implement the concepts of one software domain in particular and can be reused to implement new applications from this domain. By mapping the domain concepts, the chance of being able to reuse components in all the development phases from analysis to implementation increases, taking into consideration not a single application but a family. This approach increases the level of reuse of a given component, as well as its understanding, since it will be directly mapped within the application's domain (meaning less semantic gap) [D'Souza & Wills 1998]. The component is coded (space of the solution) according to the needs of the domain (space

of the problem). Domain engineering enables the uniformity of the concepts used by those involved in the project and represented in the various development stages and products deriving from the process.

After the domain has been modeled, various distinct applications are developed based on the same model. The modeling enables the creation of tools, techniques and components to provide support to the range of software development activities, enabling the development process as a whole. Analysis of the domain is made by consulting domain specialists and the literature on the area, or based on the knowledge acquired by the developers during the process of development or use of various applications from the same domain. Domain engineering is well suited to use in domains presenting complex processes and characteristics, where there are difficulties in modeling using traditional processes, which is the case of CSCW and groupware.

In this work, the domain analysis, the first step of domain engineering, was based on the literature and on the knowledge accumulated by the AulaNet developer group, which has eight years of experience in developing tools for collaboration. The domain analysis was restricted to communication tools, which in addition to their communication elements present a representative cross-section of coordination and cooperation elements. As we discussed earlier, even a communication service uses coordination and cooperation components.

A communication tool possesses messages, which use textual, video, audio or pictorial media [Daft & Lengel 1986]. The media sometimes present a degree of variability, such as limits on text size or available vocabulary, in the case of textual media, and the rate of data capture and transmission, in the case of audio and video. Some tools send email to recipients, enable files to be attached to messages and provide a spell-checker to help write text. Some tools, such as the AulaNet Conferences, offer message categorization. Communication tools may also offer support for commitment stores, as in the case of ACCORD [Laufer & Fuks 1995], and conversation paths, available in Coordinator [Winograd & Flores 1987]. A tool's messages are organized in a linear or hierarchical dialogue structure or in a network [Stahl 2001]. The messages are transmitted in blocks or continuously.

Support for coordination in a communication tool is related to channel access policies, task and participant management and participation monitoring. Communication tools present management of access permissions associated with participants or roles. The roles are associated to tasks within the scope of an activity. Sometimes the tasks are monitored via a workflow engine. The participation of individuals, especially in communication tools linked to teaching-learning, is assessed so as to provide data for qualitative monitoring of their participation. Message assessment provides information for assessing the competency of participants, used to define the dynamics of the activities, the association of roles and the definition of subgroups. Participation takes place in the context of a session and is based on awareness information, such as the information that a particular participant is writing a message. Some tools provide information on the presence and availability of participants in order to enable a better synchronization of the participation of the interlocutors. One coordination mechanism implemented by software found in some communication tools, such as AulaNet's Debate, is floor control, which allows conversation techniques and channel access policies to be implemented.

Support for cooperation in a communication tool is related to the registration and manipulation of information. The messages or sessions of the communication tools are registered in repositories in the form of cooperation objects. These objects are associated with a version manager, access registration, statistical analyses, trash bin, recommendation system, search mechanism or ranking mechanism. The operations on objects are registered in the form of a log, enabling future auditing and restoration of earlier states.

A component kit is a collection of components designed to work as a set [D'Souza & Wills 1998]. A family of applications can be generated from a component kit, using different combinations and sometimes developing other components on demand. A component kit does not need to be exhaustive. Component kits are extendable, allowing new components to be absorbed as necessary. Software components that are truly reusable are refined repeatedly until they reach the desired maturity, reliability and adaptability.

With the aim of providing tools for the groupware developer, this article offers the Collaboration Component Kit, using collaboration components to assemble services implementing the collaboration aspects. The collaboration components were obtained from the domain analysis. The components are shown in the following table.

COMMUNICATION	**COORDINATION**	**COOPERATION**
MessageMgr	AssessmentMgr	CooperationObjMgr
TextualMediaMgr	RoleMgr	SearchMgr
VideoMediaMgr	PermissionMgr	VersionMgr
AudioMediaMgr	ParticipantMgr	StatisticalAnalysisMgr
PictorialMediaMgr	GroupMgr	RankingMgr
DiscreteChannelMgr	SessionMgr	RecommendationMgr
ContinuousChannelMgr	FloorControlMgr	LogMgr
MetaInformationMgr	TaskMgr	AccessRegistrationMgr
CategorizationMgr	AwarenessMgr	TrashBinMgr
DialogStructureMgr	CompetencyMgr	
ConversationPathsMgr	AvailabilityMgr	
CommitmentMgr	NotificationMgr	

Table 2. Collaboration Component Kit

Component frameworks [Syzperski 1997] are used to provide support to the management and execution of the components. In the proposed architecture, a component framework is used for each proposed component type, allowing the peculiarities of each one to be met. Services are coupled in the Service Component Framework, providing support to assembly of the groupware solution, and collaboration components are coupled in the Collaboration Component Framework used to assemble the services. The component frameworks are responsible for handling the installation, removal, updating, deactivation, localization, configuration, monitoring, and import and export of components. The Service Component Framework manages the instances of the services and the links with the corresponding collaboration components. The Collaboration Component Framework manages the instances of collaboration components derived from the Collaboration Component Kit.

Most of the functionalities of the component frameworks are recurrent and reusable. A framework can be used for the instantiation of a family of systems. In this article, a framework is used to instantiate the component frameworks. This type of framework is

called a *component framework framework* (CFF) [Szyperski 1997, p.277]. A component framework framework is conceived as a second-order component framework whose components are component frameworks. Just as a component interacts with others directly or indirectly via the component framework, the same applies to component frameworks, whose highest level support is the component framework framework. Figure 3, extending the notion used by Szyperski [1997], illustrates the application architecture, including the Groupware Component Framework Framework, as a second-order component framework.

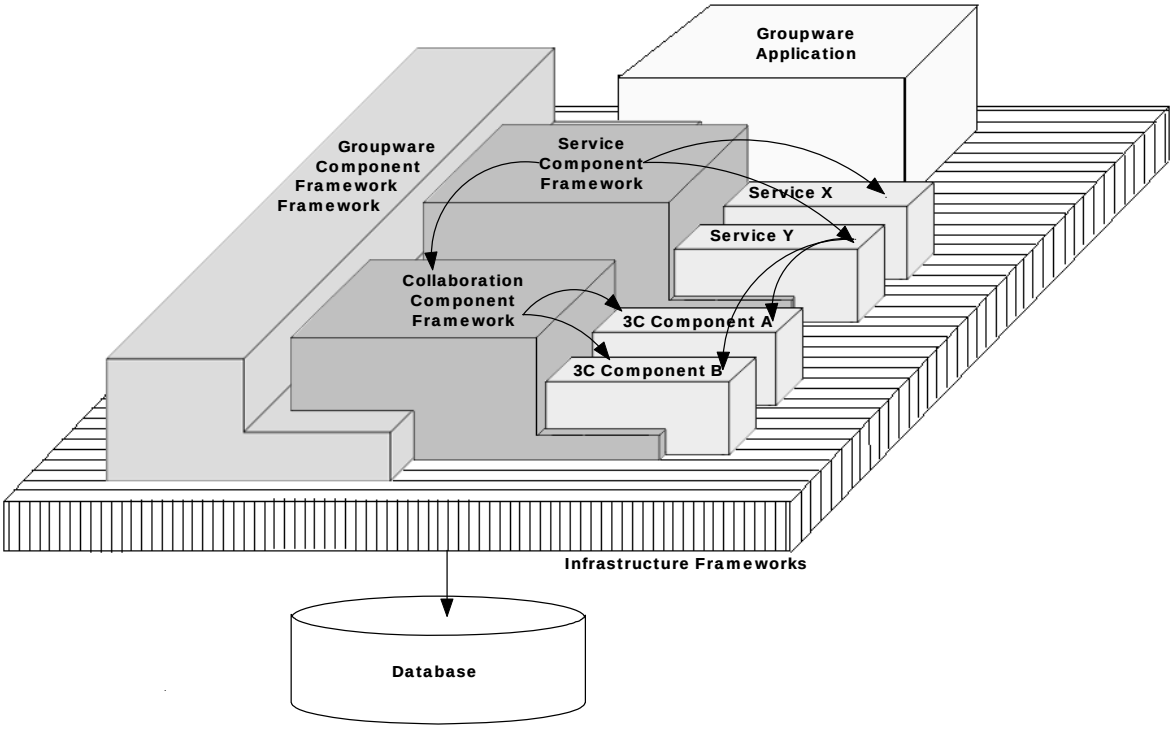

Figure 3. The proposed architecture

The proposed architecture includes a division in layers, comprising the presentation layer (not represented in Figure 3), responsible for the capture and presentation of data and for interaction with the user; the business layer, which captures the model of the business logic of the application's domain; and the infrastructure layer, which implements the low-level technical services. The division in layers is important in responding to the complexity of component-based systems [Szyperski 1997].

The same infrastructure developed for the business layer can be used for more than one presentation. When the business layer services need remote access to a PDA client, for example, web services are made available that encapsulate the façade of the business layer. In other cases, the presentation directly accesses the business façade.

The application's architecture reflects the structure of the domain's components, representing a high level logical project independent of the support technology [D'Souza & Wills 1998]. The components plugged in the business layer implement the concepts of the 3C collaboration model.

The same service can possess various instances independent of each other. For example, in the case of the AulaNet environment, a component instance is created for each course that uses the service. The Service Component Framework manages the service instances and keeps the current state of each of them, enabling restoration at a later date. When a new instance is created, the standard values defined in the descriptor file are used.

The instantiation of a new service implies the instantiation of the collaboration components used in assembling the service. The Service Component Framework interacts with the Collaboration Component Framework to enable the instantiation and the association between the instances of the components. In order to reduce the coupling between the two component frameworks, an interface is used to provide a utilization contract between them.

Installation and management of the collaboration components follows a procedure similar to that described for the services. The collaboration components possess descriptor files that define standard configurations, used in the instantiation of the components. The Collaboration Component Framework manages the configuration of the collaboration components.

5. Case Study

This section presents some case studies conducted in order to evaluate the proposed approach.

5.1. The AulaNet environment

The new version of AulaNet is being completely rewritten, using the approach proposed in this article. This version makes use of components based on the 3C model that encapsulate a cohesive set of data and functions. The component frameworks encapsulate and provide low-level services, providing support towards the development and maintenance of groupware.

In designing the computational support for collaboration, services are selected for each of the activities. Based on the feedback obtained from the use of the services, the computational support for collaboration is continually adjusted. If there is a change to the course's dynamics, the environment is reassembled by adding, replacing or removing services. As well as the inclusion of new services, the feedback obtained from the use of the environment may lead to the replacement of existing services. For example, if the coordinators of a course conclude that the learners are having difficulties with the Debate service, they can replace it with Chat, which presents a simpler interface with fewer functionalities. One problem with this substitution is that the sessions, assessments, participations, etc. remain registered for the previous service. If the services are compatible, the environment's import/export functionality can be used to transfer data.

Encapsulating services in the form of components enables the same service to be deployed with different configurations and characteristics to handle distinct tasks. For example, in one of the environment's courses, the Conference is used for the argumentation on the weekly themes and for peer evaluation. Each installation is named and configured in a specific form. Each activity's sessions are separated, enabling more detailed reports and statistics, precision in searches and adoption of different categories,

roles, permissions and evaluation criteria. The Conferences service is duplicated by deploying it twice (duplicating the corresponding file in the directory structure).

In order to encapsulate a tool not originally developed for the environment, it is necessary to create a package that enables it to be installed. It is necessary to create the describer file, the scripts and the directory structure defined in the AulaNet component model.

Some modifications are resolved by customizing rather than replacing the services. For example, to enable the possibility of deactivating the Conferences service, the corresponding property in the component's describer file just needs to be altered.

5.1.1 The Debate Service

A previous version of the AulaNet Debate service was implemented with a communication component, which implements synchronous communication protocols, and a cooperation component, which implements the shared space. This version of Debate is a plain chat tool, containing an expression element, where learners type their messages, and awareness elements, where messages from learners taking part in the chat session are displayed, as shown in Figure 4.

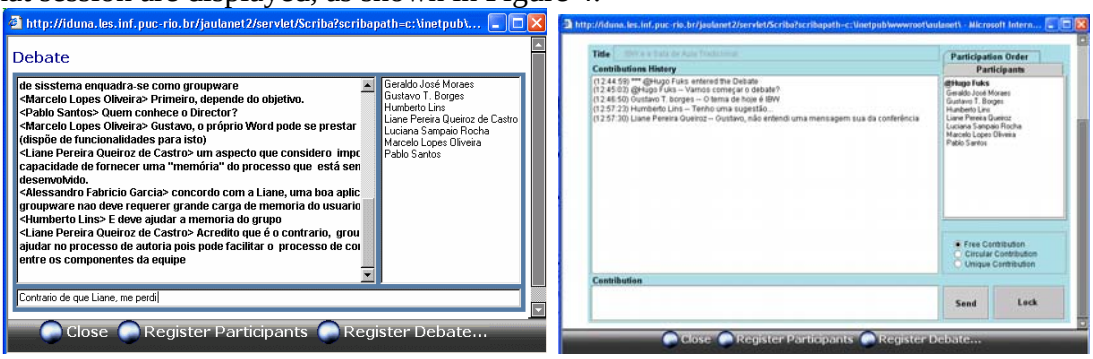

Figure 4. Previous Debate interface (left) and new Debate interface (right)

This version of the Debate service provides no support for coordination, leaving it to the standing social protocol. However, some courses use a well-defined procedure for the debate activity, such as the one shown in Figure 5, which represents the procedure adopted in a course.

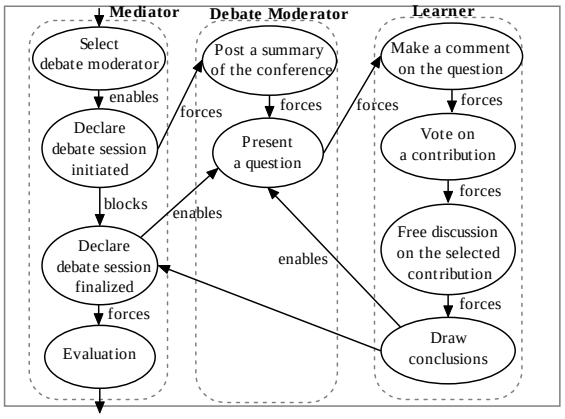

Figure 5. Expanded workflow of a debate

In this procedure, for each debate, the course mediator selects a learner to be the moderator of the session. The debate moderator posts a summary of the discussion that took place in the weekly conference and then presents questions. For each question, each learner posts a comment; when every learner has posted their comment, they vote on which question will be discussed. A free discussion then takes place. The learners have to reach a conclusion before a new question is tackled.

In order to provide better support for closely integrated activities like the example above, coordination mechanisms were implemented in the current version of the Debate service (presented in Figure 4). Floor control, participation order and shared space blocking ability were added to the service. The shared space was also enhanced with new awareness elements, like session title, timestamp and identification of mediators.

The same communication component was used in the new version of Debate, since the synchronous communication protocols and the message characteristics remained unaltered. The cooperation component, which implements the shared space, was also enhanced with new awareness elements. Finally, a new coordination component, responsible for implementing the coordination mechanisms, was implemented and plugged into the Debate architecture.

This example illustrates the need for a component-based architecture capable of dealing with the three Cs of the collaboration model. Table 3 presents the composition of both versions of the tool.

Chat	**Debate**
MessageMgr	MessageMgr
DiscreteChannelMgr	DiscreteChannelMgr
AssessmentMgr	AssessmentMgr
RoleMgr	RoleMgr
PermissionMgr	PermissionMgr
ParticipantMgr	ParticipantMgr
SessionMgr	SessionMgr
CooperationObjMgr	AwarenessMgr
	CooperationObjMgr
	FloorControlMgr

Table 3. 3C components used in the Chat and Debate services

The collaborative service was extended to follow the evolution of the work dynamics. The use of the 3C model allowed an isolated analysis of the necessities and difficulties of each collaboration aspect. Based on this analysis a more suitable tool was assembled. The 3C components allow testing the collaboration support.

5.2 Groupware Engineering Case Study

The proposed approach was also used on the Groupware Engineering course for undergraduates and postgraduates at PUC-Rio's Computer Science Department. The case study was conducted with the class from the second semester of 2005, with two undergraduate, 3 masters and 2 doctoral students. The result of the case study was evaluated via direct observation by course lecturers and the application of individual questionnaires.

Each student selects an application and analyzes its functionalities, classifying them as communication, coordination or cooperation. The student also presents an architecture

and a prototype that offers support to a extension of the system, using the infrastructure and the 3C components proposed in this article. The students succeeded in using the components in building the groupware.

The students received and replied to a questionnaire on the tool in question. Most of the students evaluated the difficulty level of using the 3C model for the analysis of the chosen application as moderate, on a scale from very difficult to very easy. The understanding of the 3C model received the same evaluation. These results were considered satisfactory, given that the students had their first contact with the 3C model and the approach during the course and are not specialists in groupware. In relation to the reach of the 3C model in modeling the system, 5 students evaluated it as sufficient and 2 as fair, indicating that the majority of the functionalities identified were classified. In relation to the use of 3C components, 5 students identified the solution as complex and 2 as normal, in a scale ranging from very simple to very complex. This result was also considered satisfactory, given that, in addition to not being specialists in groupware, the students are not specialists in software components. Although the majority classified the solution as complex, all evaluated the utilization of 3C components in the assembly of the groupware solution as good or very good. In relation to the encapsulation of low-level complexities, 3 students evaluated the solution as neutral, 2 as good and 2 as very good. Finally, in evaluating the computational support for groupware using 3C components, 2 students evaluated the solution as neutral and 5 as good. The results obtained in the questionnaire were in general positive.

6. CONCLUSION

The composition and characteristics of work groups evolve over time, as well as the tasks executed. The group learns, affinities and conflicts emerge among the members, and people enter and leave, meaning the group changes continually. A component-based architecture provides the capacity to assemble and evolve a specific work environment, selecting and configuring a set of specific collaborative tools adapted to its needs.

This research proposes the structuring of a collaborative system using components that encapsulate the technical difficulties of distributed and multi-user systems and reflect the concepts of collaboration modeled by the 3C model. In the context of this work, domain engineering is based on the 3C collaboration model. The transition between development activities and the mapping of analytic concepts to the structures of the code is narrowed, facilitating iterative development and future maintenance of the application.

In developing groupware, the requirements are rarely clear enough to allow a precise specification of the system’s behavior in advance. It is difficult to predict how a particular group will collaborate and each group has highly distinct characteristics and objectives [Gutwin & Greenberg 2000]. By involving a group, the possibilities of interactions multiply and the demand for synchronism and access competition increases, which poses problems in the construction of suitable interaction mechanisms and conducting tests. Using a set of components that are reused in diverse situations increases the system’s reliability and stability, as well as allows the replacement of components with limited impact. The developer can also experiment and prototype various configurations so as to refine the system’s requirements and the support for collaboration. The use of components also offers the capacity for customization and extension. Providing a set of components makes anticipating and providing support for

all the potential uses unnecessary. Medium granularity blocks are provided, which the developer uses to assemble the application [Szyperski 1997]. The reuse provided by the components reduces the code of the services. For example, the AulaNet conference service has 4279 exclusive lines of code (not used in other services). In the new version, it has 2905 lines of code, where just 317 are related to the business layer. The other part is related to the presentation layer, which was not changed from the previous version.

"Without an adequate architecture, the construction of groupware and interactive systems in general is difficult to maintain and iterative refinement is hindered" [Calvary et al. 1997]. A component-based architecture allows components to be selected to assemble a groupware solution meeting a group's specific interests. The components are customized and combined as required, keeping in mind future maintenance. The use of this approach enables prototyping and experimentation, which are fundamental in CSCW, given that the success cases are still few and far between and largely undocumented. Prototyping must be quick since it occurs when the group is working, in order to obtain timely feedback and carry out adjustments. The use of components improves the dynamics adaptation of the environment and the support for collaboration through the system's reassembly and reconfiguration.

However, its worth stressing that the proposed solution does not eliminate the need for an aware developer who is knowledgeable about the subject in question, since it is not enough to link the components randomly to produce an effective collaborative system.

References

Banavar, G., Doddapaneti, S., Miller, K. & Mukherjee, B. (1998) Rapidly Building Synchronous Collaborative Applications by Direct Manipulation. In Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work (CSCW'98), pp. 139-148.

Bandinelli, S., Nitto, E.D. & Fuggetta, A. (1996) "Supporting cooperation in the SPADE-1 Environment", IEEE Transactions on Software Engineering, V 22, N 12, pp. 841-865

Borghoff, U.M. & Schlichter, J.H. (2000) Computer-Supported Cooperative Work: Introduction to Distributed Applications. Springer, USA.

Bretain, I., Fredin, L., Frost, W., Hedman, L.R., Kroon, P., McGlashan, S., Sallnas, E.L. & Virtanen, M. (1997) Leave the Office, Bring Your Colleagues: Design Solutions for Mobile Teamworkers. Proc. CHI'97, ACM Press, pp.335-336

Calvary, G., Coutaz, J. & Nigay, L. (1997) From Single-User Architectural Design to PAC*: a Generic Software Architectural Model for CSCW. Conference on Human Factors in Computing Systems (CHI'97), pp 242-249.

D'Souza, D.F. & Wills, A.C. (1998) Objects, Components and Frameworks with UML: The Catalysis Approach. Addison Wesley, ISBN 0-201-31012-0, 1998.

Daft, R.L. & Lengel, R.H. (1986). Organizational information requirements, media richness and structural design. Management Science 32(5), 554-571.

Ellis, C.A., Gibbs, S.J. & Rein, G.L. (1991) Groupware - Some Issues and Experiences. Communications of the ACM, Vol. 34, No. 1, pp. 38-58.

Engelbart, D. & English, W. (1968) Research Center for Augmenting Human Intellect, Proc. Fall Joint Computing Conference, AFIPS Press, 395-410

Fuks, H., Raposo, A.B., Gerosa, M.A. & Lucena, C.J.P. (2005) Applying the 3C Model to Groupware Development. International Journal of Cooperative Information Systems (IJCIS), v.14, n.2-3, Jun-Sep 2005, World Scientific, ISSN 0218-8430, pp. 299-328.

Greenberg, S. (2006) “Toolkits and Interface Creativity”, Journal of Multimedia Tools and Applications, Special Issue on Groupware, Kluwer. In Press. Disponível em http://grouplab.cpsc.ucalgary.ca/papers

Gutwin, C. & Greenberg, S. (2000) The Mechanics of Collaboration: Developing Low Cost Usability Evaluation Methods for Shared Workspaces. IEEE 9th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises -WETICE (2000), p. 98-103.

Laufer, C. & Fuks, H. (1995) “ACCORD: Conversation Clichés for Cooperation”, Proceedings of The International Workshop on the Design of Cooperative Systems, France, pp 351-369.

Laurillau, Y. & Nigay, L. (2002) “Clover architecture for groupware”, Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW 2002), pp. 236 - 245

Litiu, R. & Prakash, A. (2000) “Developing Adaptive Groupware Applications Using a Mobile Computing Framework”, Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW'00), pp. 107-116.

Marsic, I. & Dorohonceanu, B. (2003) “Flexible User Interfaces for Group Collaboration”. International Journal of Human-Computer Interaction, Vol.15, No.3, pp. 337-360

Marsic, I. (1999) DISCIPLE: a framework for multimodal collaboration in heterogeneous environments. ACM Computing Surveys, 31 (2es), Article No. 4.

Pumareja, D., Sikkel, K. & Wieringa, R. (2004) “Understanding the dynamics of requirements evolution: a comparative case study of groupware implementation”, REFSQ 2004, Essener Informatik Beiträge 9, pp. 177-194.

Roseman, M. & Greenberg, S. (1996) “Building real time groupware with GroupKit, a groupware toolkit”. ACM Transactions on Computer-Human Interaction, 3, 1, p. 66-106.

Roth, J. & Unger, C. (2000) Developing synchronous collaborative applications with TeamComponents. In Designing Cooperative Systems: the Use of Theories and Models, 5th International Conference on the Design of Cooperative Systems (COOP’00), pp. 353-368.

Sauter, C., Morger, O., Muhlherr, M., Thutchytson, A. & Teusel, S. (1995) CSCW for Strategic Management in Swiss Enterprises: an Empirical Study. Proceedings of the 4th European Conference on Computer Supported Cooperative Work (ECSCW’95), Sweden, 117-132

Slagter, R.J. & Biemans, M.C.M. (2000) “Component Groupware: A Basis for Tailorable Solutions that Can Evolve with the Supported Task”, in Proceedings of the International ICSC Conference on Intelligent Systems and Applications (ISA 2000), Australia.

Stahl, G. (2001) WebGuide: Guiding collaborative learning on the Web with perspectives, Journal of Interactive Media in Education.

Stiemerling, O., Hinken, R. & Cremers, A.B. (1999) The EVOLVE Tailoring Platform: Supporting the Evolution of Component-Based Groupware. In Proceedings of the 3rd International Enterprise Distributed Object Computing Conference (EDOC’99), pp. 106-115.

Szyperski, C. (1997) Component Software: Beyond Object-Oriented Programming, Addison-Wesley, ISBN 0-201-17888-5

Szyperski, C. (2003) Component technology – what, where, and how? Procedings of the 25th International Conference on Software Engineering (ICSE’03), IEEE, pp 684-693.

Winograd, T. & Flores, F. (1987) Understanding Computers and Cognition. Addison-Wesley, USA, 1987.

Won, M., Stiemerling, O. & Wulf, V. (2005) “Component-Based Approaches to Tailorable Systems”, End User Development, Kluwer, pp. 1-27.