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
R. 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 has specific difficulties of development and maintenance. Its code normally 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, analyzed based on the 3C model, are mapped onto software components, also organized according to the model. The proposed approach is investigated 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.*

Resumo. *Groupware é evolucionário e possui dificuldades específicas de desenvolvimento e manutenção. Seu código normalmente se torna desestruturado e difícil de evoluir. Neste artigo, é proposta uma abordagem de desenvolvimento de groupware baseado em componentes organizados em função do modelo 3C de colaboração. Neste modelo, a colaboração é analisada a partir da comunicação, coordenação e cooperação. Os requisitos de colaboração do grupo, analisados em função do modelo 3C, são mapeados em componentes de software, também organizados em função do modelo. A abordagem proposta é investigada num estudo de caso no desenvolvimento da nova versão do ambiente AulaNet. O código do ambiente atualmente sofre dos problemas mencionados. Para instanciar os serviços de comunicação do AulaNet, kits de componentes foram desenvolvidos. Os componentes possibilitam a composição e customização dos serviços para refletir mudanças na dinâmica da colaboração.*

1. INTRODUCTION

Douglas Engelbart [1968] pointed out 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, generating few products [Greenberg 2006]. In most companies, computational support for collaboration is limited to systems for exchanging messages or filing documents.

Groupware technology has not attained its potential yet. Research on CSCW is now at a fairly advanced stage, although it lacks a manner of simplifying the programming of collaborative applications and of 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 misplaces the creativity and efforts of these developers, taking their attention out from the creation of solutions to the solving of low-level technical problems, disrupting the investigation of collaboration support.

This kind of problems in developing groupware are experienced in the development and maintenance of the AulaNet environment. AulaNet is a web-based environment for teaching and learning. 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 functionalities have been implemented in an evolving fashion. The constant changes in the collaboration support 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, getting 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, allowing 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 collaboration demands [Pumareja et al. 2004].

This article proposes the use of 3C based components as a means of developing extendable groupware whose assembly is determined by collaboration needs. By analysing the problem from the viewpoint of the 3C model and using a component structure designed for this model, changes in the collaboration dynamics are mapped onto the computational support. This way, the developer has a workbench with a component-based infrastructure designed specifically for groupware, based on a collaboration model.

By designing and developing collaboration tools in the form of software components, the developer is given the means to assemble a specific groupware for the collaboration needs of each group. Tools are selected from a component kit based on the 3C model for the support of the established dynamics. The developer selects the most suited components from those of the same family.

The proposed approach is being applied to the re-development of the AulaNet environment. A layered architecture is defined comprising component frameworks and collaboration components.

2. COMPONENT BASED GROUPWARE

Szyperski [2003] lists four main reasons for using component software. The first and oldest one 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. Its architecture comprises 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 on 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 capability 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 uses 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 previously identified by Szyperski [2003] (*product line* and *tailorability*). Some systems provide tools for building collaborative applications based on interoperable components, while some others offer final users assembly. There is no standard component model.

The proposed approach focuses on the second reason (*product line*), aiming to provide groupware developers with the means to assemble collaborative systems based on the 3C collaboration model. The literature presents many proposals for using software components for groupware development. However, none of these use the 3C collaboration model as a basis for designing and organizing software components and the development process.

3. THE 3C COLLABORATION MODEL

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 workspace is denominated collaboration by Ellis, while it is denominated cooperation in the 3C model.

The 3C model is equivalent to the Clover model [Laurillau & Nigay 2002], with a slight denomination difference: communication, coordination and production. Differently from the Clover model, in this article the 3C model guides the development of component groupware and serves as a basis for 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 privilege coordination. Bretain et al. [1997] use the three Cs as a basis to analyze and interview groups whose activities are conducted outdoors, such as firefighters, plumbers, reporters and sales representatives. Sauter et al. [1995] and 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 [Fuks et al. 2005]. Despite their separation for analytic purposes, communication, coordination and cooperation should not be seen in an isolated fashion; there is a constant interplay between them.

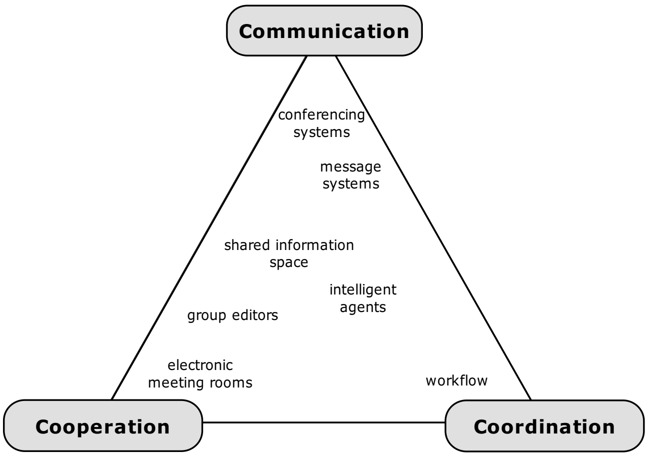

Figure 1. Diagram of the 3C collaboration model [Borghoff & Schlichter 2000]

Groupware such as chat, for example, which is a communication service, requires communication (exchange of messages), coordination (access policies) and cooperation (registration and sharing).

4. ASSEMBLY OF GROUPWARE AND COLLABORATIVE SERVICES

A collaboration environment normally offers its participants a set of collaborative services to be used in the different moments of collaboration. Table 1 shows the collaboration services found in the following environments: 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							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 services

As the table shows, there are many similar services normally found in a variety of environments. For example, most of the analyzed systems provide forums, chat, agenda, activity reports, questionnaires, task management, voting, repository and links. Additionally, each service may be seen as an independent unit. These characteristics propitiate the application of component-based development techniques, where the collaborative services are the groupware components.

Services may be classified according to their purposes and characteristics according to the 3C model. The services in Table 1 are organized into communication, coordination and cooperation. A unified component model would let the developer to select from the most suitable service from all those belonging the same family. Moreover, as can be noted in Table 1, there are services that are provided by a groupware solution, but given that they are not in a component form they cannot be made available for use in other solutions. For users, it is interesting to interchange services in order to complement environments and reuse experiences.

The same rationale that was used for environment and its services, may be used for services and their functionalities. For example, almost every chat has 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 functionalities may be reused.

Moodle's chat service displays the time that the participant remained 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 into components would also allow other developers to use them in their projects. It also becomes possible to evolve, adjust and build services by varying and reconfiguring the collaboration components.

This analysis leads to the adoption of software components at two levels, as illustrated in Figure 2. The first level comprises the components that implement the communication, coordination and cooperation services, used to offer computational support to the collaboration dynamics as a whole. The second level comprises the components used to assemble the aforementioned services, providing specific support to communication, coordination and cooperation within the dynamics of a particular service. The components that implement the collaborative services are called *services* and the components used to implement the computational support for service collaboration are called *collaboration components*.

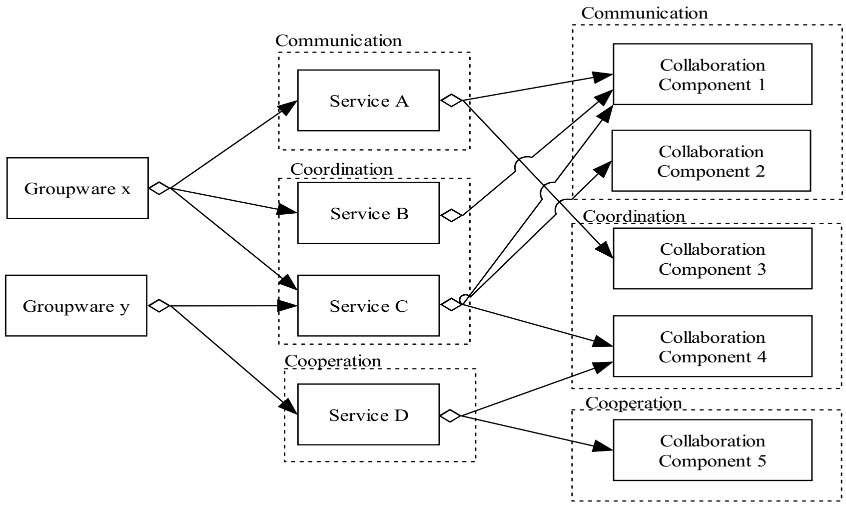

Figure 2. Groupware composition

A component-based groupware environment comprises communication, coordination and cooperation services, which may be reused in many other environments. 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. The collaboration components of a C are reused in services of the other Cs.

4.1. The Collaboration Component Kit

Domain engineering aims to provide components that implement the concepts of a software domain and may be reused to implement new applications on this domain. By mapping domain concepts onto components, the chances of reusing components in all development phases increases. Using this approach, the components are coded (space of the solution) according to the needs of the domain (space of the problem) [D'Souza & Wills 1998]. Domain engineering is well suited to use in domains presenting complex processes and characteristics, where there are modeling difficulties when 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 development group, which has nine 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.

A communication service deals with 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 vocabulary, in the case of textual media, and the rate of data capture and transmission, in the case of audio and video. Some services send email to recipients, enable attachment of files and also provide a spell-checker to help write text. Some functionalities such as message categorization, conversation paths and commitment stores appear in systems, namely AulaNet Conferences [Fuks et al. 2002], ACCORD [Laufer & Fuks 1995] and Coordinator [Winograd & Flores 1987].

Messages are organized in a linear, hierarchical or network dialogue structure [Stahl 2001] and may be transmitted in blocks or continuously.

Support for coordination in a communication service is related to channel access policies, task and participant management and participation monitoring. Communication services present management of access permissions associated with participants or roles. Roles are associated to tasks within the scope of an activity. Sometimes, tasks are enacted by a workflow engine. Message assessment provides information for evaluating 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 blinking that informs that a participant is writing a message. Some services provide information on the presence and availability of participants.

Support for cooperation in a communication service is related to the archiving and handling of information. The content of communication sessions is registered in repositories in the form of cooperation objects. These objects are associated with a version manager, access registration, statistical analysis, trash bin, recommendation system, search mechanism and ranking mechanism.

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 included as necessary. To be reusable, software components should be refined repeatedly until they reach the desired maturity, reliability and adaptability [D'Souza & Wills 1998].

Collaboration components are used to assemble services implementing the collaboration aspects. The collaboration component kit that was obtained from the domain analysis is shown in Table 2.

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 (service, collaboration), allowing the peculiarities of each one to be met. Services are plugged into the Service Component Framework for the assembling of the groupware environment, and collaboration components are plugged into the Collaboration Component Framework for the assembling of the services. 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 their links to 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.. Extending the notion used by Szyperski [1997], Figure 3 illustrates the application architecture, including the Groupware Component Framework Framework, as a second-order component framework.

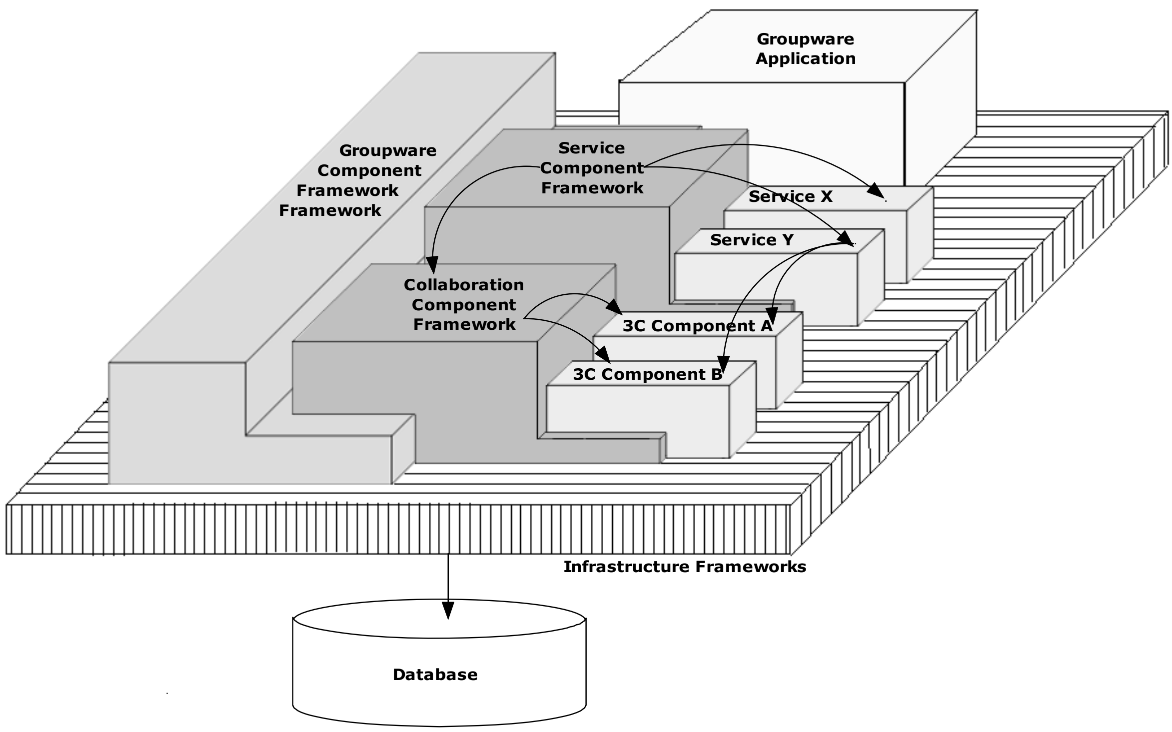

Figure 3. The proposed architecture

The proposed architecture is divided in layers, comprising the presentation layer (not shown in Figure 3), for the capture and presentation of data and for user interaction; 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 to 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 may have many independent instances. For example, in the case of the AulaNet environment, a component instance is created for each course that uses the same service. The Service Component Framework manages the component instances and keeps the current state of each of them, enabling restoration at a later date. Whenever a new instance is created, the standard values defined in the descriptor file are used.

The instantiation of the collaboration components used in assembling the service follows the service instantiation. 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, a contract interface is used.

Installation and management of the collaboration components follows a procedure similar to that described for the services. The collaboration components have 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 to 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 learners are having difficulties with the Debate service, they may replace it with a regular Chat, which presents a simpler interface with less functionality. A problem that may arise with this substitution is that sessions, assessments, participations, etc. remain registered from the use of the previous service. If the two services use the same components for message recording, then the environment's import/export functionality can be used to transfer data.

Encapsulating services into components enables the same service to be deployed with different configurations and characteristics for handling distinct tasks. For example, in one of the environment's courses, the Conference is used for the argumentation on the weekly argumentation and for peer evaluation. Each installed component is specifically named and configured. Thus, each service's sessions are independent, enabling more detailed reports and statistics, more precision in searches and the adoption of different categories, roles, permissions and evaluation criteria. Any service may be duplicated by deploying it twice (duplicating the corresponding file in the directory structure).

In order to encapsulate a service not originally developed for the environment, it is necessary to create a package that supports its installation. It is necessary to create the descriptor file, the scripts and the directory structure defined in the AulaNet component model.

Some modifications may be solved by customizing rather than replacement of services. For example, to make it possible the deactivation of the Conferences service, the corresponding property in the component's descriptor file must be available as a parameter.

5.1.1 The Debate Service

An early version of the Debate service was implemented using a communication component, tailored for synchronous communication protocols, and a cooperation component, which implements a plain shared space. This version of Debate is a typical chat service, 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.

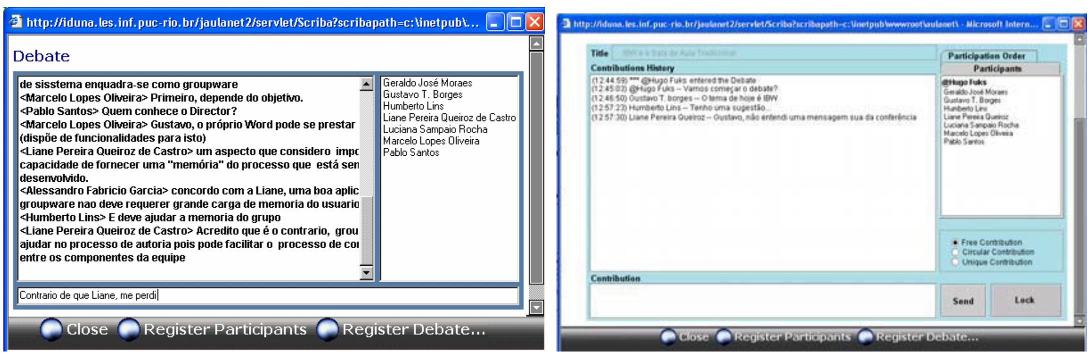

Figure 4. Early Debate interface (left) and current Debate interface (right)

The early version provided no support for coordination, leaving it to the standing social protocol. However, some courses that use a well-defined procedure for the debate activity, such as the one shown in Figure 5, need effective coordination support. 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.

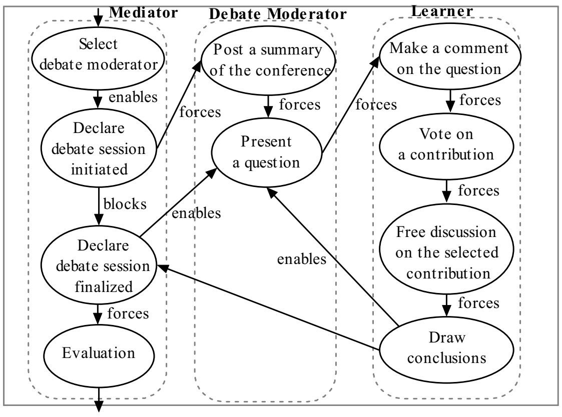

Figure 5. Expanded debate workflow

The same communication component was used for the new version of Debate, given that the synchronous communication protocols and the message characteristics remained the same. The cooperation component, which implements the shared space, was also enhanced with new awareness elements.

This example illustrates the usage 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.

Early version	Current version
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 service was assembled, mapping collaboration necessities onto software components, both of them organized according to the 3C collaboration model.

5.2 Case Study on a Course

The proposed approach was also used on the Groupware Engineering course at PUC-Rio's Computer Science Department. The case study was conducted on 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 selected an application and analyzed its functionalities, classifying them as communication, coordination or cooperation. The student also presented an architecture and a prototype that offers support to a extension of the system, using the infrastructure and the 3C components. They succeeded in using the components for designing groupware.

The students also received and replied to a questionnaire. Most of them evaluated as moderate the level of difficult of using the 3C model for the analysis of the chosen application, on a scale from very difficult to very easy. The understanding of the 3C model was also considered moderate. These results were considered satisfactory, given that the students had their first contact with the 3C model during the course and that they are not specialists in groupware. Regarding the reach of the 3C model in system modeling, 5 students evaluated it as sufficient and 2 as fair. 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 either. Although the majority classified the solution as complex, all of them evaluated the utilization of 3C components in the assembly of groupware 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

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 onto the structures of the code is narrowed, facilitating iterative development and future maintenance of the application. Component based development together with specific domain concepts has shown itself as a feasible approach to groupware development.

In developing groupware, the requirements are rarely clear enough to allow for 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 solving deadlocks increases, posing 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 prototype different configurations in order to refine system’s requirements and collaboration support. Providing a component kit mitigates the need to anticipate and provide support for all the potential uses. Medium granularity blocks are provided, which the developer uses to assemble the application [Szyperski 1997]. The reuse provided by the components also reduces the amount of lines of code. 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 rest is related to the presentation layer, which remained the same.

"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 very few and poorly documented. The use of components improves the dynamics adaptation of the environment and the support for collaboration through the system's reassembly and reconfiguration.

However, it is worth stressing that the proposed solution does not eliminate the need for an aware developer who is knowledgeable about the subject in question. 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., Gerosa, M.A. & Lucena, C.J.P. (2002), "The Development and Application of Distance Learning on the Internet", Open Learning Journal, V. 17, No. 1, February 2002, ISSN 0268-0513, Cartafax Pub, pp. 23-38.

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.