Coview: A Cooperative Architecture for Digital Video Editing

By Benjamim Fonseca and Eurico Carrapatoso

Benjamim Fonseca

Eurico Carrapatoso

Desktop video editing plays an important role in the digital video market, and cooperative applications are important for organizations. Cooperative video editing tools can be an interesting solution for large broadcasters or for remote reporting, and such commercial products are lacking. This paper describes a cooperative video editing tool, Coview, which uses Web services to provide the cooperative functionalities. For this purpose, an overview is given of the basic issues of Web Services and Computer Supported Cooperative Work. Then, the Web Services-based cooperative infrastructure and the Coview prototype is described. Finally, some experimental results and concluding remarks are presented.

One technological development that has revolutionized the professional and leisure activities in the last decade is the internet, particularly the world wide web (WWW). Its ease of utilization and its potential regarding information retrieval and business and leisure activities have led to an exponential growth of the users' community, fostered by the recent availability of mobile access to the Web. The dawn of this century witnessed the emergence of a new technology that enables the exchange of messages between remote applications or services, using Web protocols and data formats widely supported. This technology, known as Web services, has capabilities that ease the interaction between peers in heterogeneous environments.

In the current organizational context of companies and institutions, one success factor is the ability to effectively realize teamwork. This fact has raised the interest of organizations in applications of computer supported cooperative work (CSCW). In this kind of application, usually referred to as groupware, interoperability issues, familiarity with applications, and users' mobility support assume significant importance. A common requirement for cooperative applications is the ability to synchronously notify the occurrence of events produced as the result of cooperative activities.

The main field in which Web services are currently used is business-to-business (B2B) applications and in the vast set of groupware categories, workflow management is the only one that has shown significant developments in the use of Web services. This fact, along with the issues referred to in the previous paragraph, motivated the authors to define a model for the creation of cooperative applications that uses Web services technology to provide the mechanisms that support the cooperative

activities, relying on the asynchronous notification of events to reflect them. The main objective of this model was to define a set of core services that enabled the development of cooperative applications that use these services to provide features such as sharing of cooperative events and shared data consistency.

Earlier video editing activities involved walking into the archive, searching for the desired tapes, and carrying them out to the editing room. Then, the editor would use videotape recorders and TV monitors to go through the tapes, select the desired sequences, and record the result on a new tape. Finally, it was necessary to take the tapes back to the archive. Furthermore, all these video manipulations used analog technology, with the disadvantages it encompasses. The evolution in computer technology has led to the use of digital techniques to handle video material. The first step was to convert the video sequences into digital format and store and manipulate them in a computer-based editing station (nonlinear editing). Nevertheless, the source and the final result were kept in analog format (tape). Recently, the evolution in acquisition equipment, high-speed networks, and compression techniques brought about a novel concept: the digital studio, where video sequences are acquired, stored, transmitted, and manipulated digitally.[1-3]

Cooperation in video editing activities is usually done presentially. However, in large broadcasters, or producers, or in remote reporting activities, it can be useful to have a desktop application that enables the cooperative editing of digital video sequences involving editors or journalists remotely situated.

This paper presents a prototype of a cooperative desktop video editing tool, COooperative VIdeo Editing on the Web (Coview). This application uses the functionalities of a cooperative framework, Services Architecture for Groupware Applications (SAGA),[4] composed by a set of Web services that provide the features required by cooperative applications. Among these features, the asynchronous notification of cooperative events assumes particular importance. For this purpose, some theoretical issues concerning Web services and CSCW are first outlined. Then, the SAGA architecture is presented and the main implementation details of the prototypes of SAGA's services and of Coview are described. Finally, some experimental results concerning the use of SAGA and Coview are exposed and some conclusions are drawn.

Web Services

The concept of service is frequently associated with the idea of an application accessible through interfaces, which tell us how to use the operations they provide. These applications are usually referred to as applications with a Service Oriented Architecture (SOA).[5-7] The Web has a utilization paradigm and a protocol set for communications and data representation that were easily accepted and are widely supported. The ability to build services accessible using Web protocols is very attractive for a significant part of the software industry and the international scientific community. The convergence of SOA and Web protocols, under the guidance of the World Wide Web Consortium (W3C),[8] produced the technology currently known as Web services.[7,9] A simple definition of a Web service is an application that is accessible through an interface, using common Web protocols, such as the HyperText Transfer Protocol (HTTP),[10] and use data representations that follow the de facto standard XML (eXtensible Markup Language).[11] As a component, a Web service represents a functionality that can be reused without knowledge of its implementation details.

The use of widely adopted protocols and data representations gives Web services a highly appreciated and desirable feature that other distributed processing architectures had difficulties achieving efficiently: interoperability. Indeed, the use of Web protocols for communication provides platform independence and the use of XML for data representation provides independence at the programming language level. The latter also has the ability to transform legacy applications into services accessible through Web servers, facilitating the interaction between systems. This feature gives organizations the ability to increase the profitability of their investments in information systems and expand business opportunities. Thus, the tendency will be for any kind of application to be offered as a Web service and become accessible anywhere.

Web services use XML to describe service interfaces and encode the messages exchanged in the invocations. The description of the interfaces is contained in a file using the Web Services Description Language (WSDL).[12] That description contains information regarding the available operations, the data types manipulated, the format of the exchanged messages, the protocols that are supported, and at least one access point (an

address), known as a Uniform Resource Identifier (URI). The messages exchanged in the invocations use a packet format and a data encoding mechanism defined by the Simple Object Access Protocol (SOAP).[13] In addition to the message exchange model, SOAP formalizes a remote procedure call model. Another important feature of Web services is the availability of a service, Universal Description Discovery and Integration (UDDI),[14] that enables other services to register and publish their interfaces, making it possible for a user to discover and know how to use them. The classification of services is also possible, facilitating the processes of discovery and utilization. For each Web service registered, UDDI stores its name, its operations, and its access point, that is the information contained in the WSDL description. The description of a service interface can be used to build the service's client applications. Figure 1 shows the generic architecture of a system based on Web services and the sequence of activities, since a service registers until it is used by a client.

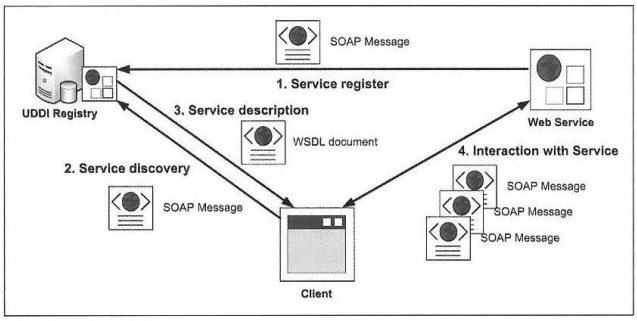

Figure 1. Web services-based system architecture.

CSCW

The scientific field known as CSCW[15,16] investigates how teamwork can be supported by information and communications technologies, in order to improve the performance of a group of persons involved in the execution of common or interrelated tasks. CSCW is an interdisciplinary scientific domain, involving the scientific areas of distributed systems, multimedia communication, telecommunications, information science, and socio-organizational theory. The impact of utilization of CSCW applications (usually referred to as groupware) is not always positive with socio-professional issues being an important consideration. Indeed, the utilization of CSCW applications can substantially modify work practices or dissolve organizational aspects of the team, which can bring negative consequences to their adoption. It his highly recommended to use methodologies that enable understanding of the way people usually work or that enable discovery of ways to improve it.

Some examples of successful groupware applications are workflow applications, such as IBM Lotus Notes[17] and Microsoft Exchange.[18]

An important issue in groupware is the existence of an environment that is shared among the team members. This environment may include documents, shared whiteboards, and shared pointers, among others. Groupware applications must have mechanisms to distribute cooperative events produced by members in the shared environment. Usually, the shared environment coexists with the private environments of each member, imposing the availability of diverse management mechanisms to control the access to information. To ensure consistency of the data being shared, special care must be taken regarding concurrency control, through the implementation of mechanisms such as atomic transactions, locks, versioning, token passing, or voting systems. Potential targets for groupware are software project and engineering teams, coordination of work processes in large organizations, distance learning, telemedicine, and cooperative editing.

SAGA

The diversity and complexity of work methodologies in organizations is increasing, and the participants' responsibilities are not always statically defined. Hence, the execution of tasks by several persons, who may not always play the same role in the execution of a certain type of task, is relatively frequent. Furthermore, during the execution of a task, one person may want to consult others regarding specific issues, or obtain approval from upper levels of the organization.

Usually, groupware tries to support teamwork in the most successful way, providing a means for information sharing, for its joint manipulation and communication among cooperating participants. However, the architecture of cooperative applications is often based on proprietary solutions that do not address issues such as flexi-

bility, interoperability, and support of legacy systems.

The internet is now present in nearly all organizations, through popular applications such as e-mail and Web browsers. Java technology enables the construction of applications that make the most of this situation. However, it does not satisfy several security and privacy issues, or the interoperability between diverse systems, written in different programming languages. Developing cooperative applications based on Web services can be an attractive choice, enabling applications to take advantage of the potential of the technology regarding distribution issues, such as interoperability, security, and legacy systems reuse. Furthermore, the Web services technology is supported by the major actors in the software industry and academic community. Usually, Web services are used mainly in B2B applications. In the CSCW domain, work has been carried out on workflow management, under the workflow management coalition (WfMC).[19] But the features of Web services make them suitable to support other classes of cooperative applications. Thus, the authors defined a model for building cooperative applications based on Web services, which was designated SAGA—Services Architecture for Groupware Applications.

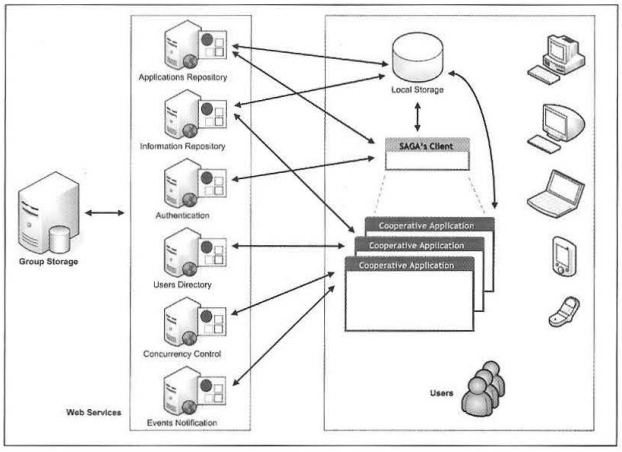

Figure 2. SAGA architecture.

The main goal of SAGA was to constitute a framework that enables the development of cooperative applications through the composition of several core functionalities available through Web services. These must provide a set of operations suitable to fit the requirements of every class of cooperative applications.

Because the tasks assigned to each participant in a cooperative session may vary more or less frequently, it is desirable to have the ability to download applications as needed and execute them immediately. This feature provides the system with a high degree of flexibility and the users with the latest version of the required application. Figure 2 depicts SAGA's overall architecture, showing the generic services that provide support for various kinds of cooperative applications and their interactions with the client-side components. In the SAGA architecture, users can access the services using various kinds of Web-enabled devices. The applications on the client-side act as Web services' clients that invoke the operations available on the services interfaces. The SAGA architecture encompasses a set of core services that provide applications with cooperative functionalities: Applications Repository Service, Information Repository Service, Authentication Service, Users Directory Service, Concurrency Control Service, and Events Notification Service.

The user, after authentication by the Authentication Service, visualizes the list of the applications available in the Applications Repository Service and may then select, download, and execute them immediately. These applications may, in runtime, use the Information Repository Service to access the information resources and download the ones needed. When one user wants to cooperate with another one, the user accesses the Users Directory Service to view the list of available potential partners. Then, the user sends an invitation to the potential partner, who is informed of that fact and has the choice to accept or reject the invitation. Upon acceptance, users register their interest in being notified of cooperative events and can start a cooperative work session. The invitation process and the notification of cooperative events are intermediated by the Events Notification Service. This service is an interface to the functionalities provided by the events notification system, allowing the adoption of the most convenient one (e.g., an organization can have a legacy system that it wants to reuse).

The group storage module is a database that stores the applications and information resources that will be manipulated by cooperative users. Local storage is used

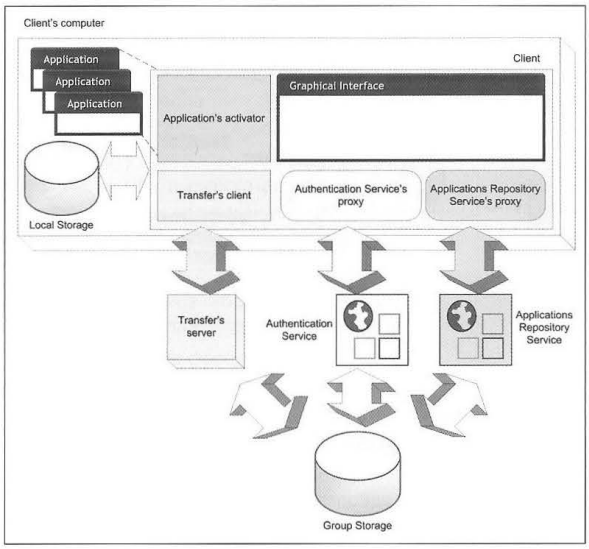

Figure 3. Client's architecture.

to store the resources downloaded from the server and the resources produced by the local user. The resources information contained in the group storage is stored in object databases, which manage data such as the name of the resource, its type, its version, a brief description, and some keywords to simplify queries. For information resources, the identification of their creators are also stored.

The Client module refers to an application that is an access point to the system and corresponds conceptually to the client of both the Authentication Service and the Applications Repository Service. Indeed, clients of Web services interact with service proxies instead of directly with service instances. These proxies run on the client computer and intermediate communications between clients and services. Hence, the Client module in the SAGA architecture uses proxies for the Authentication and Applications Repository services. Figure 3 shows the block diagram of the Client module.

Cooperative activities usually incorporate at least one of the following functionalities: communication among team elements, information sharing, and joint visualization of activities or work environments. For the latter two, it may be highly relevant to ensure exclusive access to shared resources, using concurrency control mechanisms. These mechanisms include free manipulation of resources by several simultaneous processes and restriction to a few or even to a single process (e.g., locks). For this purpose, the SAGA architecture contemplates a Concurrency Control Service, which has methods to obtain and release locks and tokens, to assign version numbers, and to manage voting systems.

The Events Notification Service interface provides operations to register interest in receiving cooperative events and to remove that interest, as well as to be notified of the events produced during a cooperative session. For each user joining a cooperative session, an instance of the Events Notification Service is created. This instance is registered in the Events Notification Service as an event consumer. Each cooperative event fires the invocation of the notification operation, which delivers it to the Events Notification system, which in turn will propagate it to all registered consumers.

Since SAGA adopts generic architectural solutions, it has features that make it suitable for a vast set of cooperative applications, namely:

- The features provided by Web services offer a high degree of interoperability and the ability to integrate legacy systems.
- The Events Notification Service enables the creation of multiple cooperative sessions and the propagation of any type of events.
- The Concurrency Control Service provides diverse mechanisms that enable the adoption of the concurrency control policy that best suites the application needs.

Coview

An important component of a video production chain is video editing, which allows users to modify a video clip by changing the order of its sequences; cut some parts or the combination of several clips into a new one; and add new audio tracks, subtitles, or special effects. Current commercial products for nonlinear editing of digital video allow only a single user at a time. When a user requires the involvement of someone else, users must be physically co-located or must establish a conversational communication (e.g., phone call). The idea of having a digital video editing system that allows various users to share some parts of an editing environment, staying in their rooms and using their computers, can be

both operationally and financially attractive.

Coview was created to test the viability of an innovative cooperative video editing application. This prototype is a simplified application, in terms of the functionality it offers, because the aim was mainly to determine if cooperative video editing is viable and if Web services can support it. Therefore, Coview implements only the definition of editing points (In and Out) and the reproduction of both the original clips and the clips produced as a result of the editing process. These operations fire events to be propagated to cooperators, who can then properly manipulate them. The implementation of a complete cooperative video editing tool is a quite complex task and would require the involvement of a large multidisciplinary team. Indeed, the choice of events to be propagated and what to do with the events that are received is a subject that deserves appropriate attention by a team with professional expertise in video production and also in the area of social sciences, which was not the case. Nevertheless, for the purposes mentioned above, the authors believe it was sufficient.

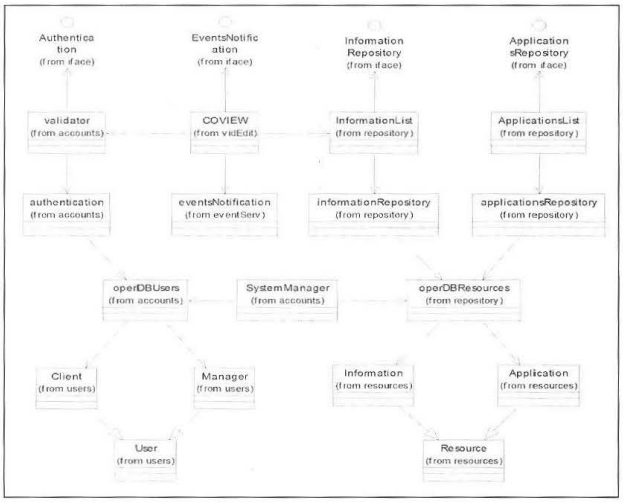

Figure 4. SAGA generic class diagram.

Prototypes of the services specified by SAGA were also implemented. The intention was not to have an implementation covering the entire system, but to introduce some simplifications that would make the implementation feasible and still address the main features required by Coview. All services and applications were developed in Java, since it is an object-oriented language with good semantics and syntax capabilities and it has a set of frameworks in areas crucial to this application, namely those related to events notification, Java Shared Data Toolkit (JSDT);[20] media playback, Java Media Framework (JMF);[21] and object database management, Java Data Objects (JDO).[22] For the Web services infrastructure, Systinet WASP Developer[23] was chosen because it has a mature implementation and a vast programmers' community and can be integrated with major Java development environments, such as Sun Microsystems NetBeans[24] and IBM Eclipse,[25] enabling automatic generation of several useful code fragments. Figure 4 shows the generic class diagram of the SAGA prototype implementation that was created; it shows:

- Client applications—implemented by the classes validator, Coview, InformationList, ApplicationsList.
- Proxies to the Web services—implemented by the classes Authentication, EventsNotification, InformationRepository, ApplicationsRepository.
- Web services—implemented by the classes authentication, EventsNotification, InformationRepository, ApplicationsRepository.
- Management applications—implemented by the classes SystemManager, operDBUsers, operDBResources.
- Databases' classes—implemented by the classes User, Client, Manager, Resources, Information, Application.

To manage the data required by services and applications, there are two main object databases: one for storing users' information and other to store both information and application resources. The information stored in the users' database is used in the authentication process. The resources database stores resource objects, which can be applications or multimedia information resources, as well as the associated metadata. There are two types of users: the common user (Client) and the system manager (Manager). Both have specific attributes and operations and inherit others from the superclass user. A similar situation occurs for the resources database, which has a superclass (Resource) and two subclasses, Information and Application, as can be seen in Fig. 5.

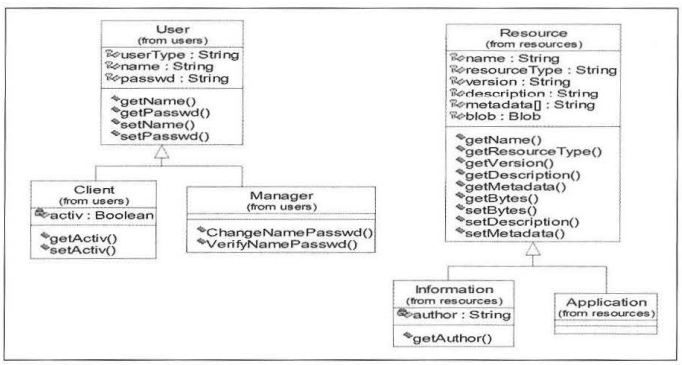

Figure 5. Databases' class diagram.

Figure 6. List of informational resources.

The repositories provide operations to list, add, remove, and search for resources in the databases. The prototype implementation of SAGA contemplates search operations by name and keywords, which are performed using Object Query Language (OQL)[26] statements.

A management application allows system managers to create, destroy, or change application and informational resources, as well as user accounts.

Cooperative applications, such as Coview, can be downloaded as JAR files (because graphical interfaces usually produce several class files) and instantly activated on the client side, using a class loader for this purpose. Figure 6 shows the window containing the list of informational resources available to the users. Clicking the resource entry and then the "OK" button opens a window showing the characteristics of the selected resource.

The Concurrency Control Service and the Events Notification Service are particularly important for cooperative activities. The prototype of the Concurrency Control Service implements a versioning mechanism, because it was sufficient for the application that was chosen to validate SAGA. Moreover, the main interest was in testing the performance of the system and the Events Notification Service is crucial to this issue. This service follows a publish-subscribe model, in which applications (event producers) publish their interest in propagating their events and other applications manifest their intention (subscribe) of receiving these events. When an application starts its execution, it registers as a consumer of system events, so that its user may be invited for a cooperative session. Upon acceptance, it is also registered as a consumer of the specific events of that session. The registration of one application creates an instance of the Events Notification Service that is used to asynchronously notify it of the events in which it is interested. This instance mediates the registration and notification processes between the applications and the events notification system (supported by JSDT). Each user may be engaged with several other users in one cooperative session or in several ones. Indeed, user A may have a cooperative session with user B and another cooperative session with users C and D, without mixing cooperative events from distinct sessions. The notifications are asynchronous to avoid the need for applications to be Web services with public operations. All cooperative events are distributed as strings, making the system suitable for any kind of event produced by any kind of cooperative application.

Figure 7 shows the main classes involved in the event notification process, namely the Events Notification Service (eventServ) and its client-side proxy (I_Event), some JSDT classes and interfaces that helped to implement the notification mechanism, the test application (Coview), and the classes related with the asynchronous notification feature of the Web services infrastructure (GenericAsyncCallback and AsyncConversation). Beside the operations to register in sessions, to quit them, to invite/accept cooperation and to send/receive events, there is an operation that shows the list of active users. This operation is used to search for potential partners to invite for cooperation. It corresponds to the Users Directory Service, which was not implemented separately, but integrated with the Events Notification Service, because their functionality is closely related.

Figure 8 shows the main window of the Coview appli-

cation, where the various menus and buttons available and a video clip being played can be observed.

Figure 9 shows the options available in two of the Coview menus. The "File" menu has options for loading or saving video clips, locally or in the Information Repository, as well as a search by keyword facility—the window is shown in Fig. 10. The "Cooperation" menu (Fig. 9) has operations to control the cooperation process.

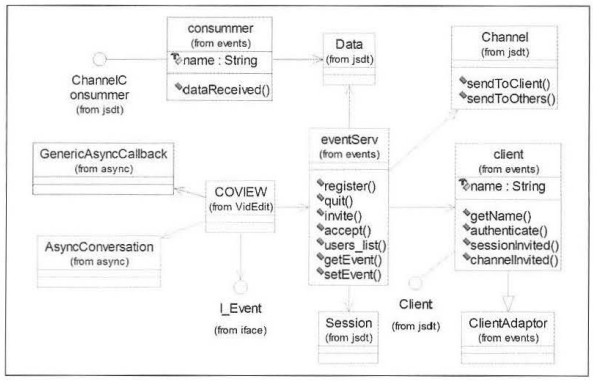

Figure 7. Events Notification's class diagram.

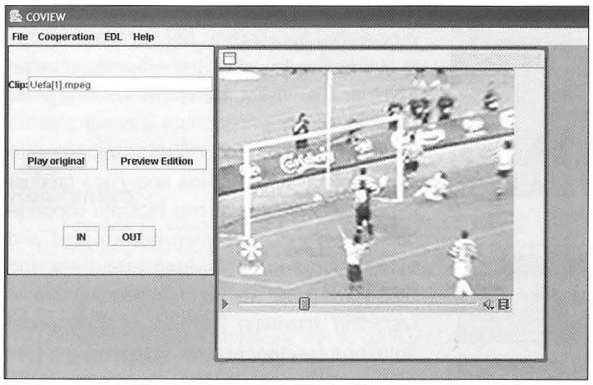

Figure 8. Coview: main window.

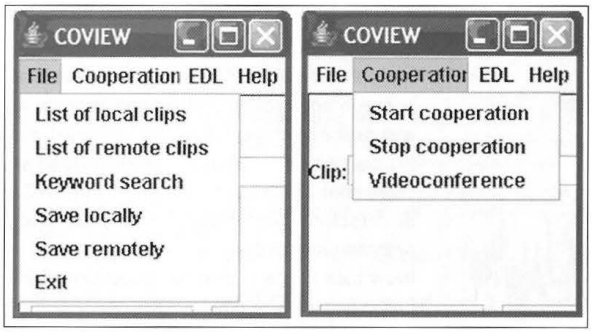

Figure 9. "File" and "Cooperation" menus.

The keyword introduced in the window shown in Fig. 10 is used to perform a query in the Information Repository, and the result is presented in a window similar to that of Fig. 6, showing only the resources that match the specified keyword. If an informational resource is selected in the list, a window is shown displaying the resource's features, as can be seen in Fig. 11. This window enables any client to download the selected resource and the resource's author to change the description and metadata.

The manipulation of video clips is achieved with the help of the JMF framework. Cooperation can be started by choosing the corresponding entry in the Cooperation menu. This action fires the invitation process described previously. Figure 12 shows a sequence of messages that are displayed in a cooperative session: invitation, acceptance, notification of a cooperative event, and abandon.

Experimental Results

The operation of the system was subjected to both qualitative evaluation and quantitative measurements. The qualitative evaluation of Coview, when used by several cooperators in a near real-world scenario, was very positive, because it performed quite well as far as response time and did not show any errors or locks related to cooperative activities. Measurements were also made to register the delay introduced by the propagation of cooperative events provided by SAGA. These measurements were carried out by building small applications that simulate the massive production of cooperative events at predefined time intervals. The results of this performance test are summarized in Table 1, where it can be observed that the system performed rather well in almost all situations, except for the case of a production of events at time intervals of only 0.1 sec, which is

Figure 10. Search by keyword.

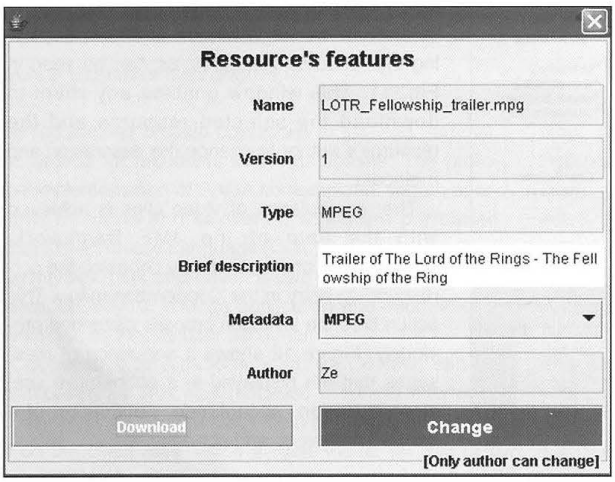

Figure 11. Resources features window.

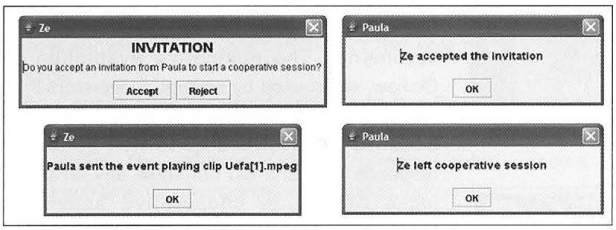

Figure 12. Windows showing cooperation messages.

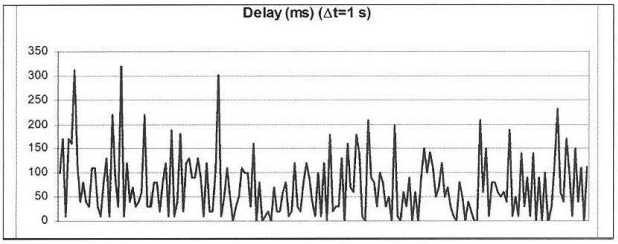

Figure 13. Performance results for a time interval of 1 sec.

a most unlikely situation in any realistic utilization scenario. Furthermore, for almost all situations, the average delay was less than 100 ms and the minimum delay was below 1 ms; the test applications were not able to record it.

Figure 13 shows a chart with the performance results produced for a time interval of 1 sec between consecutive cooperative events. Other time intervals, except for 0.1 sec, exhibit a similar chart.

Conclusion

This paper described Coview, a prototype of a cooperative desktop video editing application that uses a set of Web services to achieve the cooperative functionalities. These Web services constitute a framework, SAGA, and provide a set of core functionalities that can be composed to build several classes of cooperative applications. Indeed, Web services have features that make them suitable to support cooperative applications, namely those related to interoperability; also, the message-oriented approach is adequate to the exchange of event notifications.

Cooperative video editing can be very important for the television and video producing markets. Indeed, it can facilitate cooperation among journalists, management staff, and video editors in large video producers and broadcasters, as well as for supporting remote reporting activities. The lack of commercial solutions for cooperative video editing has made possible, the opportunity to develop the innovative Coview prototype.

SAGA is an open, distributed, interoperable, modular, and evolutionary architecture that proved to be viable, allowing the interaction between applications and support services as well as the exchange of events produced during cooperative sessions. The prototype services that were built, based on the SAGA architecture, performed robustly and their architectural solutions are generic enough to allow their usage in diverse cooperative application scenarios.

The Coview prototype performed quite well

Table 1—Results of the Performance Tests

	Δt—Time interval between consecutive events (s)					
	0.1	0.5	1	2	5	10
Average	49.498	0.326	0.073	0.059	0.055	0.061
Maximum	111.701	2.433	0.320	0.281	0.241	0.331
Minimum	1.713	0	0	0	0	0

and has capabilities that can turn it into a powerful tool, because remote and collaborative video editing is an activity that could be envisaged in many situations in which a reporter sends the raw material to the TV headquarter but wants to be involved in the final editing.

Several improvements of the services and applications described in this article are being planned, namely the addition of metadata to the resources stored in databases and a more accurate specification of the user requirements, resulting from the involvement of multidisciplinary teams (with programmers, human interface designers, sociologists, and audiovisual professionals) in the development process and testing of the final product in real-world situations. The unavailability of commercial products similar to Coview offers the potential to develop the prototype. In this context, the hope is to find partners in defining the real application requirements and in developing a commercially viable product.

References

1. B. Fonseca, P. Oliveira, and E. Carrapatoso, "Non Linear Editing in an MPEG2 Studio, XV Simpósio Brasileiro de Telecomunicações," *Recife, Brasil: Sociedade Brasileira de Telecomunicações,* 1997.
2. P. Oliveira, B. Fonseca, and E. Carrapatoso, "An MPEG-2 Distributed Studio Architecture Based on ATM," MELECOM'98, Tel Aviv, Israel: *IEEE,* 1998.
3. T. A. Ohanian, *Digital Nonlinear Editing,* Second Edition, Focal Press: Boston, 1998.
4. B. Fonseca and E. Carrapatoso, "SAGA: A Web Services Architecture for Groupware Applications," in *Groupware: Design, Implementation and Use,* eds. Y. Dimitriadis, I. Zigurs, and E. Gómez-Sánchez, Springer-Verlag: Medina del Campo, Spain. p. 246-261, 2006.
5. R. Nagappan, R. Skoczylas, and R. P. Sriganesh, *Developing Java Web Services,* Wiley: Indianapolis, 2003.
6. B. Associates, *Service-Oriented Architecture* (SOA) Definition 2005 [accessed 2005]; www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html.
7. W3C, *Web Services Architecture*, 2004 [accessed 2005]; www.w3.org/TR/ws-arch/.
8. W3C, W3C, 2004 [accessed 2004]; www.w3c.org/.
9. E. Newcomer, "Understanding Web Services: XML, WSDL, SOAP, and UDDI," First Edition, Independent Technology Guides, ed. D. Chappell, Addison-Wesley Professional: Boston, 2002.
10. W3C, *HTTP*, 1999 [accessed 2004]; www.w3.org/Protocols/.
11. W3C, *XML,* 2004 [accessed 2004]; www.xml.org/.
12. W3C, *WSDL,* 2001 [accessed 2004]; www.w3.org/TR/wsdl.
13. W3C, *SOAP,* 2003 [accessed 2004]; www.w3.org/TR/soap/.
14. OASIS, *UDDI,* 2004 [accessed 2004]; www.uddi.org/.
15. U. M. Borghoff and J. H. Schlichter, *Computer-Supported Cooperative Work,* Springer-Verlag: Berlin Heidelberg, 1998.
16. M. Beaudouin-Lafon, et al., *Computer Supported Co-operative Work.* Trends in Software, ed. B. Krishnamurthy, John Wiley & Sons: Chichester, 1999.
17. IBM, *Lotus Notes,* 2004 [accessed 2004]; www.lotus.com/.
18. Microsoft, *Microsoft Exchange*, 2003 [accessed 2004]; www.microsoft.com/exchange/default.asp.
19. W3C, *WfMC*, 2004 [accessed 2004]; http://www.wfmc.org/.
20. Sun, *Java Shared Data Toolkit,* 2002 [accessed 2003]; http://java.sun.com/products/java-media/jsdt/.
21. Sun, *Java Media Framework,* 2002 [accessed 2003]; http://java.sun.com/products/java-media/jmf/.
22. Sun, *Java Data Objects (JDO),* 2004 [accessed 2004]; http://java.sun.com/products/jdo/.
23. Systinet, *Systinet WASP,* 2003 [accessed 2003]; www.systinet.com/.
24. Sun, *NetBeans,* 2003 [accessed 2003]; www.netbeans.org/.
25. IBM, *Eclipse,* 2003 [accessed 2003]; www.eclipse.org/.
26. ODMG. *ODMG OQL User Manual* [PDF], 2004 [accessed 2004]; www.odmg.org/oqlg.zip.

A contribution received January 2006.

THE AUTHORS

Benjamim Fonseca is an assistant professor in the engineering department at Universidade de Trás-os-Montes e Alto Douro (UTAD), Vila Real, Portugal. His current research interests include computer-supported cooperative work (CSCW), collaborative virtual environments (CVE), middleware, Web services, and digital video processing. He received a Licenciatura in electrical engineering from UTAD, an MSc in computer and electrical engineering from the engineering faculty of the University of Porto (FEUP), and a PhD in electrical engineering from UTAD. Fonseca can be contacted at benjaf@utad.pt.

Eurico Carrapatoso is an assistant professor at Faculdade de Engenharia da Universidade do Porto (FEUP), Porto, Portugal. His current research interests include distributed multimedia applications, service creation methodologies, digital libraries, E-Learning, and simulation. He received a Licenciatura in electrical engineering from FEUP and a PhD in information systems engineering from the University of Bradford, U.K. Carrapatoso can be contacted at emc@fe.up.pt.