Integrating Collaborative Applications with LEICA

Roberta L. Gomes\textsuperscript{1}, Guillermo J. Hoyos Rivera\textsuperscript{2}, Jean Pierre Courtiat
LAAS - CNRS
Toulouse, France
\{rgomes, ghoyos, courtiat\}@laas.fr

Abstract Due to their unpredictable nature, collaboration activities are rarely supported by a single CSCW technology. Therefore, different types of collaborative applications are usually applied in order to support group work such as collaborative e-Learning. But in spite of being executed to accomplish a common collaboration task, these applications are executed independently without getting any advantage of each other. The integration of such applications would allow them to dynamically combine their different functionalities in a controlled way. In order to achieve integration, we propose LEICA\textsuperscript{3}, a loosely-coupled integration environment which allows collaborative applications to interact without losing their autonomy. LEICA relies on the definition of collaboration policies for controlling the interactions between collaborative applications.

Index Terms— Collaborative systems, CSCW integration

I. INTRODUCTION

Advances in networking and computing technologies, combined with the communication demands of geographically dispersed organizations (e.g., research teams, universities, companies), have given rise to the development of a great variety of collaborative applications, or CSCW (Computer Supported Cooperative Work) systems.

As collaborative activities such as collaborative e-Learning can present different requirements, they are rarely supported by a single technology. Thus some CSCW systems try to combine different functionalities in order to support collaboration into a single environment. The main weakness of this approach is the challenge behind the anticipation of all the requirements of cooperative situations involving several people, with different group tasks and needs. This way a CSCW system is hardly suitable enough for every collaborative activity.

In practice, current collaborative environments consist of a range of applications, working side by side but independently, without really getting advantage of each other. Allowing the integration of these collaborative applications could bring significant benefits to users. An integrated collaboration environment would allow the different functionalities of existing applications to be dynamically combined and controlled (enhancing flexibility and tailoring possibilities).

In order to achieve the integration of existing CSCW systems avoiding dealing with their low-level features, we propose LEICA, a “Loosely-coupled Environment for Integrating Collaborative Applications.” Relying on Web services technology [1] and an event notification system (supporting the publish/subscribe paradigm [2]) different collaborative applications can interoperate by exchanging information within the context of a global SuperSession. The loosely-coupled approach proposed by LEICA overcomes two problems usually related to integration environments: (i) it does not require a true semantic integration of collaborative applications, (ii) once integrated to the environment, collaborative applications keep their autonomy.

The definition of collaboration policies controls the interactions among integrated applications, i.e. how the collaboration activity supported by one application will be affected by information received from other applications. In practice, these applications interact through the notification of events which may lead to performing specific action(s) in some of these applications.

The interaction degree among integrated applications depends obviously on the nature of the events they are able to exchange, and actions they are able to perform. Three main cases may be considered when integrating applications: a) open source applications, b) API-based applications, and c) applications without any API. Integration of open source applications can achieve the tightest interaction degree, since any internal event/action can be exported/performed; it might however imply great development efforts. Integration of API-based applications is straightforward, and interaction is limited to the provided API. Applications providing no API are constrained to interact only through application start and stop actions. LEICA’s integration approach has been mainly driven by the second case (b). We believe that developers are certainly interested in creating specific and performable collaboration tools that can be used either stand-alone or integrated with other applications (through a flexible API), thus being able to get a great share of the market. This is for instance the case of Skype\textsuperscript{TM} [3], a successful example of communication tool that has recently released its API.

The paper is structured as follows: section 2 presents related

\textsuperscript{1} Researcher supported by a scholarship from CNPq, Brazil.
\textsuperscript{2} Researcher supported by scholarship 70360 CONACyT/PROMEP, Mexico.
\textsuperscript{3} This research is partially supported by the European IST project Lab@Future.
work regarding the integration of existing CSCW systems; section 3 overviews the general integration approach of LEICA; section 4 describes how a SuperSession is configured and explains how to specify collaboration policies; section 5 draws some conclusions and directions of future work.

II. RELATED WORK

In [4], Dewan addresses basic issues in interoperating heterogeneous CSCW systems that concurrently manipulate the same artifacts. However, it does not regard the interoperability of CSCW systems that, despite being involved in the same collaborative tasks, do not deal with the same artifacts (e.g. videoconference and shared whiteboard).

In [5] authors propose an integrative framework based on a three-level model: ontological, coordination and user interface. An internal knowledge of the collaborative application is needed so that its functionalities can be mapped into the three-level model in order to achieve integration. Accordingly, the integration of third party applications becomes a complex (even impossible) task.

In [6] authors present the CVW, a prototype collaborative computing environment defining a place-based system for integrating document and meeting-centric tools. Basic freeware collaborative applications have already been integrated, and new, special-purpose tools can be integrated, but the integration process is not straightforward (tools must be designed against a place-based API).

Systems like AREA [7] and NESSIE [8] have tried to propose a loosely-coupled integration for supporting cross-application awareness. Like LEICA, these systems are based on the exchange of activity relevant events. However these environments just aim to provide users with a common awareness of the whole collaboration activity. They do not provide any means for defining how an application should react when events are notified by other applications.

Another proposal also based on a loosely-coupled approach is presented in [9]. The authors define a framework where Web services are used to wrap collaborative applications in order to integrate them. Since they leverage open Internet standards, Web services overcome the interoperability issues usually associated with more general integration solutions like CORBA [10], DCOM [11] and EJB [12]. Besides, they are simpler to design, develop, maintain and use. Web services based integration is quite flexible, as it is built on a loose coupling between applications.

One of the drawbacks related to the Web services wrapping approach (in particular to the use of SOAP [13]) is that it represents an additional tier causing some overhead in processing exchanged messages [14,15]. Besides, depending on the architecture of the existing collaborative applications, the complete wrapping of these applications as Web services may imply great development efforts or even applications redesign. Therefore, unlike the approach employed in [9], and following the recommendations of [14] and [15], we decided to use Web services for coarse-grained operations only. Thus, LEICA applies Web services as an initial mechanism for (i) registering newly integrated applications, (ii) setting and (iii) starting up collaborative sessions. Then, a different infrastructure is used to implement the event notification system in charge of interconnecting the collaborative applications during the integrated collaborative session execution. An overview of the proposed integration approach is presented in the following section.

III. LEICA OVERVIEW

LEICA aims at allowing developers to integrate existing collaborative applications. Before explaining the general integration framework defining the main behavior of LEICA, let us present an integration scenario to better illustrate the advantages that such integration can lead to.

A. E-Learning Integration Scenario

An important domain where CSCW systems have been largely employed is e-Learning. In particular, 3D Collaborative Virtual Environments (CVEs) appear as a relevant technology for providing collaborative e-Learning since: (i) virtual reality is really suitable for representing real world metaphors (ii) 3D shared worlds provide an intrinsic awareness of people and objects.

In this scenario, a CVE is used for implementing a virtual reality (VR) world representing a school composed of: an entrance hall, two classrooms (where virtual lectures are hold) and the teachers’ room. Three types of roles can be attributed to users: Teacher, Monitor, and Student. A chat tool is used for communicating Students whose avatars are placed in the entrance hall. Teachers and Monitors can discuss about lectures when they enter into the teachers’ room with the support of a shared whiteboard tool. Lectures are supported by an audioconference tool and a collaborative web browsing tool. To attend a virtual lecture, a Student enters into a classroom, been automatically connected to the respective audioconference and collaborative web browsing sessions. Fig. 1 schematizes this e-Learning integrated session.

With LEICA, all these collaboration tools can be integrated and then make part of a common SuperSession, instead of being executed independently. Users connected to this SuperSession will dynamically collaborate while a certain control can be performed. For example, suppose that both the audioconference and the collaborative web browsing tools define a floor control.
LEICA would allow the implementation of a floor coupling, imposing that the same person who has the floor of the web browsing activity is also the one who has the right to talk in the audioconference.

Our goal here is not to anticipate all possible usage scenarios of CSCW systems. Actually, the unlimited possibilities motivate integration environments to be general enough so as not to put artificial constraints on the range of applications to be integrated.

B. General Integration Framework

In order to integrate a collaborative application to LEICA, a Wrapper must be attached to this application. This Wrapper comprises a Web services interface allowing the collaborative application to register itself with LEICA (registering to a private UDDI registry) as an integrated application. Through its Web services ports, the integrated application can interact with the Session Configuration Service (Fig. 2).

The Session Configuration Service is a Web service (accessed through a Web portal) used for (i) configuring new global SuperSessions and (ii) starting up SuperSessions. A SuperSession is an integrated collaborative session holding the whole collaboration activity. Within the context of a global SuperSession, different specificSessions can exist. A specificSession is then a conventional collaborative session defined within the context of one collaborative application (e.g. a videoconference session, a whiteboard session, etc.).

During the SuperSession configuration process, the Session Configuration Service dynamically contacts each integrated collaborative application (Fig. 3) in order to request: (i) which specific data is required to create specificSessions for this respective application (e.g. a videoconference tool might need a multicast address); and (ii) which kind of events it can notify, and action requests it can receive. This second information will be used during the collaboration policies definition process.

Collaboration policies are a set of rules following a condition/action model. These rules define how collaborative applications might react when events coming from other collaborative applications are notified. In other words, collaboration policies are the mechanism allowing to determine how an application should react when receiving information (events) notified by other applications.

At the end of the SuperSession configuration process, a configuration file is generated. Based on this file, the Session Configuration Service can start the respective SuperSession. To do so, firstly it contacts each integrated collaborative application in order to create the specificSessions defined in this SuperSession (Fig. 4). Then, these collaborative applications are interconnected through an event notification service (as previously explained, from this point Web services are not used anymore).

During collaboration activities, the Wrappers are in charge of managing the collaboration policies. As collaboration activity progresses, collaborative applications exchange event notifications in a peer-to-peer fashion. When the Wrapper of a collaborative application receives an event notification, it verifies if the notified event enables any policy rule concerning this collaborative application. If so, the Wrapper sends action requests to the respective application. Note that LEICA is not intended to support low-level physical events (e.g. mouse click/scrolling) or high frequency synchronization events (e.g. current position of moving objects). It aims at supporting meaningful events that carry some semantics.

IV. SUPERSESSION CONFIGURATION

The configuration of a new SuperSession is accomplished in two steps: (i) Session Management configuration and (ii) Collaboration Policies configuration.

A. Session Management Configuration

In the first configuration step, all data necessary to define the main elements of a SuperSession are provided. Two groups of information have to be specified:

- General Session Management information (GSMInfo). It carries management information like scheduling, membership and general user roles.

---

1 Wrappers are attached to servers of client/server and multi-server applications, and to the peers of peer-to-peer (P2P) applications.

2 Except for P2P applications, which use a P2P Proxy in order to register themselves (since a P2P application is not executed in advance - as servers of client/server applications - they might not be available as a Web service).
• Integrated Applications information (IAinfo). It defines the list of integrated applications to be used during this SuperSession. For each collaborative application, a list of specificSessions is defined. Necessary information for creating specificSessions is provided (e.g., a videoconference application will be provided with an IP multicast address). In the case of role-based applications, specific roles (associated to the general roles) are defined.

B. Collaboration Policies Configuration

The second step of the SuperSession configuration process deals with the specification of collaboration policies. As briefly described in subsection 3.B, these policies are responsible for linking the collaboration activities supported by different specificSessions in the context of the global SuperSession. This is carried out through a set of policy rules, basically allowing the association of event notifications to the execution of actions (under certain conditions).

In order to specify the policy rules, a collaboration policies editor is used. This editor provides a user-friendly interface for creating policy rules through the composition of GUI components, called policy widgets. Once the collaboration policies are created, the editor generates the respective XML data which is appended to the SuperSession configuration file. The rules’ semantics associated with the XML syntax has been defined using the RT-Lotos formal description technique [16]. A Policy Manager module (present in the Wrapper) implements this semantics.

Fig. 5 illustrates the policy widgets used to create policy rules. These widgets can be connected through their connection points. The basic composition rules are: (i) policy rules are read from left to right; (ii) only widgets without any connection point on their left can appear on the left end of a policy rule; (iii) only widgets without any connection point on their right can appear on the right end of a policy rule.

The Event widget represents an event notification. Each Event is associated with a collaborative application (field “From”) and has a type (field “Type”). In the “Event Parameters” area it is possible to define matching patterns (filters) for parameters’ values. The Action widget represents an action execution request. Each Action is associated with a collaborative application (field “To”) and has a type (field “Type”). In the “Action Parameters” area, all the required parameters for this action type are specified. Note that all event and action types (and their parameters) are well-known since they are provided by each integrated application.

Fig. 6 shows a simple collaboration policy rule associating directly one action (or possibly many) with one event. This policy rule is enabled when the specified event is notified. It specifies that: if the application “CA1” notifies an Event of type “T1” with parameter “a:M*” (a string starting by “M”), then an action request of type “T2” must be sent to application “CA2.” The “%” character is a reference operator, indicating that the Action’s parameters “d” and “y” have their values copied from the event parameters “b” and “c.”

A Predicate widget allows the association of conditions to enable policy rules. A Predicate can appear alone or attached to every policy widget but an Action. It contains a predicate that is specified in Java™ language syntax. Predicates can impose time constraints, as well as conditions based on the current SuperSession state. When a Predicate is attached to an Event (or to a Latest) it can also reference the parameters of the respective Event (or the parameters of the Events connected to the Latest).

The Earliest and Latest widgets allow the composition of different Events for the specification of a policy rule. When Events are grouped through an Earliest, the policy rule is enabled when the first event among all the events is notified. When Events are grouped through a Latest, the policy rule is enabled after all events have been notified.

Fig. 7 shows two policy rules examples using Latest and Earliest widgets. In the left example, the policy rule is enabled when both specified events are notified (for the first one, the attached Predicate must be evaluated to true). Then the Predicate associated with the Latest widget is also evaluated. If it is true, then the two specified action requests are sent. In the right example, there is an Event and a Predicate grouped through an Earliest. Policy rules like this one aim at waiting the fulfillment of certain conditions to be enabled (e.g., regarding the SuperSession state); however if a particular event is notified before this condition is satisfied, then the policy rule is also enabled.

An upcoming problem is related to the fact of having more than one Event in the same policy rule. Thus, an automatic sequence number is attributed to each Event in order to be used as identifier while referencing event parameters (as illustrated by Fig. 7). Another constraint related to event parameters referencing appears when Events are grouped through an Earliest. As it defines a non-deterministic behavior (there is no
way of knowing which of the \textit{Events} will enable the policy rule) parameters from \textit{Events} of different types grouped through an \textit{Earliest} can not be referenced by a \textit{Predicate} attached to this \textit{Earliest}.

As illustrated in Fig. 8, different \textit{Earliest} and \textit{Latest} widgets can be combined in order to create composed rules.

![Figure 8. A composed collaboration policy rule](image)

For illustrating the use of collaboration policies, consider the e-Learning scenario described in subsection 2.A. Suppose that just users playing the Teacher or Monitor roles should be connected to a whiteboard \textit{specificSession}. Fig. 9 illustrates the collaboration policy implementing this behavior.

![Figure 9. A policy rule for connecting users to the whiteboard](image)

Fig. 10 illustrates the policy rules implementing the floor coupling between the collaborative web browsing tool and the audioconference tool.

![Figure 10. Policy rules implementing the floor coupling](image)

V. CONCLUSIONS

This paper has presented LEICA, a loosely-coupled environment for integrating collaborative applications. Existing collaborative (and non-collaborative) applications can be loosely integrated using Web Services as integration technology. In the context of a \textit{SuperSession}, a global collaboration activity is supported where different integrated applications are used in a parallel and coordinated way. Based on the specification of collaboration policies, LEICA defines applications’ behavior in response to event notifications.

A first prototype has been developed in order to verify the feasibility of the proposed integration approach. Java™ has been chosen as underlying technology. Web services interfaces are implemented using Tomcat 5.0 [17], Apache SOAP 2.3.1 [18] and UDDI4J [19]. The current prototype confirms the fact that open source applications achieve richer interaction levels since we have all the needed flexibility for binding Wrappers to them. However, new software applications are increasingly coming out of the box with API specifications (sometimes based on Web services), their integration tends to be straightforward. Some performance evaluations are to be carried out on the prototype in order to test its responsiveness (targeting the event notification system and the collaboration policies processing) and scalability (regarding the number of collaborative applications taking part of the same \textit{SuperSession}).

Concerning the definition of collaboration policies, no verification of policies’ consistency conscious has yet been performed. Possible solutions based on the use of formal techniques description to guarantee policies’ consistency will be studied in a near future.

REFERENCES