Peer-to-peer Face-to-face collaboration

Delfina Malandrino and
Ilaria Manno

ISISLab
Dipartimento di Informatica ed Applicazioni "R.M. Capocelli",
Università di Salerno, 84081 Baronissi (Salerno), Italy.
{delmal,manno}@dia.unisa.it

Abstract. In this paper, we present a *proof of concept* application of a technique that is designed explicitly for face to face collaboration software architectures. The objective is to minimize the impact on the installation and deployment of the application, that, while internally keeping a client-server architecture (in order to allow the centralize coordination and monitoring), presents to the user (both teacher and learners) as uniform work environment, integrating client and server components in one piece of software. In order to further limit the impact on the configuration, we define a *start and play* protocol, to start-up the application with no network configuration; the *start and play* protocol takes advantage from the particular conditions of the face to face context i.e. LAN setting.
The application is built on the Eclipse core (Rich Client Platform), and inherits its plug-in based architecture and its advanced tailoring features.

1 Face2face collaboration systems

Current research in Computer Supported Collaborative Learning (CSCL) has produced many studies and several classifications of the situations where the collaboration takes place. The space-time matrix (see Fig.1) is a well-known classification [4] that defines the four basic space-time situations. A lot of works have studied the *different-time* AND/OR *different-place* situations to reduce distances (both in time and space), while there are fewer studies about the *same-time* AND *same-space* situations. Of course, existing synchronous systems for remote situations can also be used in the co-located situations, but the *same-time* AND *same-space* situations is substantially different from the remote ones and the technological support should take in account this difference. Indeed, the tools to support remote collaboration try to achieve a "virtual co-location"

	Same place	Different place
Same time	Co-located collaboration	Remote synchronous collaboration
Different time	Asynchronous collaboration	Remote asynchronous collaboration

Fig. 1. Space Time Matrix

enhancing remote communication by chat, e-mail, file sharing, audio and video conferencing, etc. In f2f situations, this kinds of communications channels are unnecessary because there is no distance to fill up. For these reasons, the systems to support co-located learning could and should focus on collaboration activities rather than on reducing distances, for example, they could provide reviewability and revisability [8], that are important characteristics in particular in the learning process [11].

Our team is involved as technical partner in the European project LEAD, in Sixth framework programme priority IST [5], whose goals are *to develop, implement and evaluate conceptual models, practical scenarios and associated networked-computing technologies for effective face-to-face problem-solving discussions.*

In this project we are focussing on design features and development solutions to produce a *face-to-face* (f2f) collaborative learning tool, in collaboration with others technical partner and according to the conceptual model outlined by the pedagogical partners.

In this paper, we approach the problem of designing an application for face-to-face collaboration that has minimum impact on the installation and management. We present an architecture and a small proof-of-concept prototype that was designed in order to test the effectiveness of our low-cost deployment strategy.

2 Software architectures

Most of the existing systems for CSCL have a client-server architecture. This model, in fact, simplifies data collection process and persistence management; furthermore, the client/server entities support the students/teacher roles, allowing to centralize in the server component the functionalities for the teacher, while the clients components offer the functionalities common to all the students.

The existing systems are Web-based, since the most are designed for remote situations (a survey is presented in [6]). These systems are not always suitable for a f2f didactic context, since they require to communicate with an external server (and therefore they require an Internet connection), and many schools employ restrictive firewalls and access policies. Furthermore, the teachers could not exercise fully control on the external server and is somewhat limited by its availability and configurability. On the other hand, some of the existing systems allow to install a local server, but the installation process is often too much complex for the end users, that may not have the experiences and capabilities to install and configure a Web server.

We aim to design a CSCL system explicitly applied to the f2f context, addressing the particular conditions of such context. In fact, in a co-located situation the system can use only the local area network, so it could and should do without external servers and Internet connection, in this way it can avoid many problems due to restrictive security policies, that are, often, commonplace in educational settings.

An existing cooperative system providing a LAN-based approach is MeetingWorks [7]: chauffeur and participant components are local applications and every participant links up with the chauffeur automatically, but the system[1] needs a shared directory to

[1] We have tested only the free version of the program, that has only LAN participants.

which every participant needs to gain access. Using a shared directory is a critical choice in the context of a classroom because the standard hardware and software equipment may not support many concurrent accesses (e.g. limitations on the number of simultaneous remote accesses in standard operating systems that are intended for desktop and not for servers). Therefore, the network use and configuration must be carefully designed not only to avoid problems due to security policies but to assure effectiveness and efficiency as well.

Beyond the network architecture, we are interested particularly in enhancing friendliness and deployment easiness: the system must be simple to configure, to start-up and to use, in order to encourage its usage and spreading.

In an overall view, we are designing a LAN-based system, providing a uniform work environment and a *start and play* protocol, to offer an application simple to install and to start up, in other terms, an application that exhibits a *low cost deployment*.

3 Our architecture

Several studies [9] suggest component-based architectures to address architectural requirements for collaboration systems. In particular, we are studying the Eclipse Platform [3] architecture. Eclipse is a component-based Integrated Development Environment that provides a framework (Rich Client Platform, RCP) to build general purpose applications using the Platform architecture. In the following we introduce briefly the Eclipse architecture (sec. 3.1), and then our approach to use RCP to build a face to face collaborative application (sec. 3.2).

Since the reasons (simplifying data management and matching teacher and students roles) to use the client-server model are well-grounded for the f2f system too, we do not set aside the client-server model, but we are studying how to use it in a *LAN-based* architecture, so that it can be independent of both Internet connection and external servers.

In order to simplify the system usage, we propose here an architecture with server and client components embedded in the same application, so that the system could provide a uniform work environment between teacher and students, and without requiring the management of a separate server. The idea is that the application looks *peer-to-peer* to the users even if their internal structure makes one of the peers (tipically, the teacher's one) to be the server. Of course, this architecture leverage on the growing availability of CPU cycles on low-end desktops, and on the inherent limitation on the size of the classroom, which makes acceptable the workload on a server placed on a desktop machine.

Another aspect affecting user-friendliness is the start-up phase: to simplify the start-up phase we are defining a low cost deployment approach to allows the end users to *start and play* the collaborative application, with no network configuration. We describe these design features in the section 3.3.

3.1 Eclipse architecture

Eclipse is a component-based Integrated Development Environment grounded on three key concepts: plug-ins, extension-point and lazy activation.

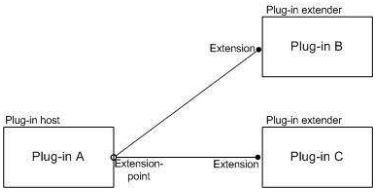

Fig. 2. The extender plug-ins B and C provide extensions to the host plug-in A.

A plug-in is the smallest independent software unit; even if a tool could be composed by more than one plug-in, the term plug-in is often used as "tool" or "component". Every plug-in declares its identity and properties in a file *manifest*[2], so these information are available without activating the plug-in.

The extension-points define the rules of plug-ins composition: an extension-point is the point exposed by a plug-in to allow extensions from other plug-ins. The plug-in that exposes the extension-point is the plug-in host, while the plug-in that provides the extension is the plug-in extender (see fig. 2).

The plug-in host declares the extension-point in its file manifest, and the plug-in extender declares the extension in its file manifest, so that the information about extension relation between the two plug-ins are available without activating them.

The lazy activation is the property that allows to activate a plug-in on demand, so that there can be a lot of plug-ins installed but only few active.

Beyond the flexibility and scalability, the Eclipse architecture assures the *extreme tailorability* [2, 9, 10], allowing customization, integration and extension.

3.2 Building on Rich Client Platform.

Rich Client Platform (RCP) is the "core" of Eclipse: it is composed by the fundamental plug-ins, mainly to manage graphic interface and plug-ins life cicle, without any specific feature of the development environment. The RCP is a framework to build general purpose applications based on the Eclipse architecture (see fig. 3). The applications built on RCP inherit the tailorability provided by the Eclipse architecture.

To build the system on the RCP framework, we have to define the components of the application. We can distinguish two types of building blocks: the Core and the collaboration tools. Each component, the Core and the tools, is a plug-in. The Core provides fundamental functions, that are, at least, user awareness (presence and activity), installed collaboration tools discovery, start-up of tools (on demand, if possible), definition of the rules for composing the building blocks in the system.

The collaboration tools can provide any kind of functionality (free chat, structured chat, graphic shared editor, mix of previous, games, etc.); they must only observe the

[2] As a matter of fact, the *manifest* is a couple of files: plugin.xml and manifest.mf, that contain respectively information about relations with other plug-ins and about the runtime. They are often referenced as a single file, first for historical reasons and then because they can be edited with a single advanced editor.

composition rules fixed by the Core. The Core depends on RCP (see fig. 4) and is the main plug-in, that is, the plug-in defining the application. The Core provides an extension-point named `tools` defining the API that any collaborative tool must implement to be integrated in the system. This extension-point (like all extension-points) may have zero or more extensions. A plug-in extender has to declare in the file manifest an extension to the extension-point `tools` and has to implement the API specified by the extension point. The Core analyzes the extensions to the extension-point at runtime, so it is possible to add a tool to the system without changing the Core.

The plug-in based architecture allows to build each tool component with its own server embedded. The idea of a server for each tool has two reasons; first, in this way the Core ignores completely the tools details (and the tools servers details), so that whatever tool will be needed, it could be added without modifying the Core, since the tool embeds its own specific server functionalities; second, having a server for each tool and thanks to the lazy activation property, in each moment only the required tool servers are running. So, the strongly component oriented architecture of Eclipse assures fully tailorability, thanks to plug-ins and extension-point concepts. Furthermore, the *lazy activation* assure scalability: each collaborative tool will be activated only when required.

The flexibility and the extendibility of RCP would allows to extend the system as the collaboration needs arise, achieving a richer system, placed at the top of the classification framework presented in [10], where at the bottom there are basic collaboration functions, while at the top there are “comfortable” collaboration functions.

3.3 Low cost deployment: uniform work environment and start and play protocol

Part of our studies concerns the problem of the start-up: we would enhance start-up transparence so that the users could start the application and could use it with no configuration (i.e. *start and play*). Furthermore, we aim to provide a uniform work environment to make semi-transparent the difference between server and clients: they are integrated in the same application so that the application server instance is not perceived

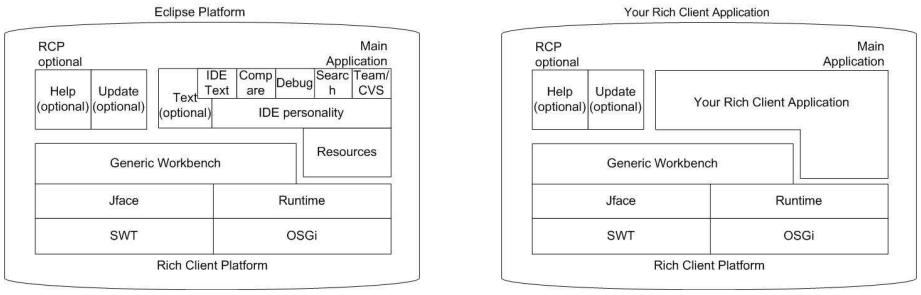

Fig. 3. *(a)* The Eclipse architecture and *(b)* a general purpose application on RCP

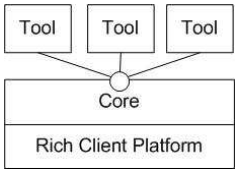

Fig. 4. The Core based on RCP can be extended by many tools.

by users as “the server” but as “a powered peer” (“powered” because it has more functionalities than standard clients). From technical point of view, this is merely a seeming difference, but from user’s point of view there is no external application to install, to configure, to start up and to manage. For these reasons, each component contains both the server side and the client side, even if, in each moment, only one instance of the application over the net runs in the server mode.

To achieve a *start and play* application, we developed a UDP-based server localization protocol, using only the local network. When the application starts, it is in client mode, and the Core client sends a “server lookup” message in broadcast; if in a timeout it does not receive the server reply, it instances and runs the Core server. Every subsequent application sending the “server lookup” message will find the server (see fig.5). Furthermore, the Core manages the start-up of the tools, so when a user (see fig. 6) requires to start a tool, the core client of user 2 sends a “start tool” message (specifying the id of the tool) to the Core server. When the Core server receives the “start tool” message, it instances and runs the tool server and forwards to all users the “start tool” message; each users that receives the start message runs the tool client. Each tool client sends a “tool server lookup” in broadcast, and will receives the reply of the tool server.

Since in CSCL f2f systems it is desirable that some operations are reserved to the teacher, the servers should be hosted by the teacher. To match this requirement, we have defined a *client running mode* and a *server running mode*. The running mode can be explicitly enabled by specifying a command line parameter. The teacher application

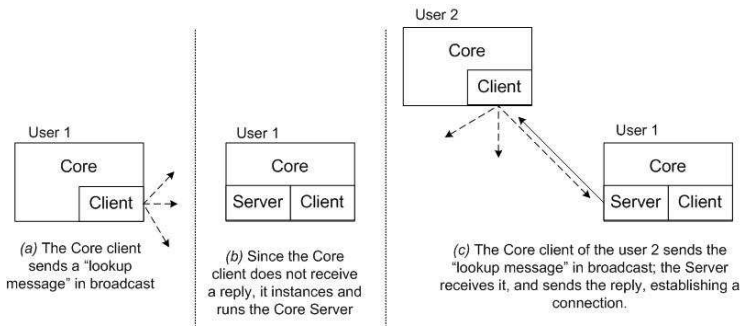

Fig. 5. The Core activation sequence.

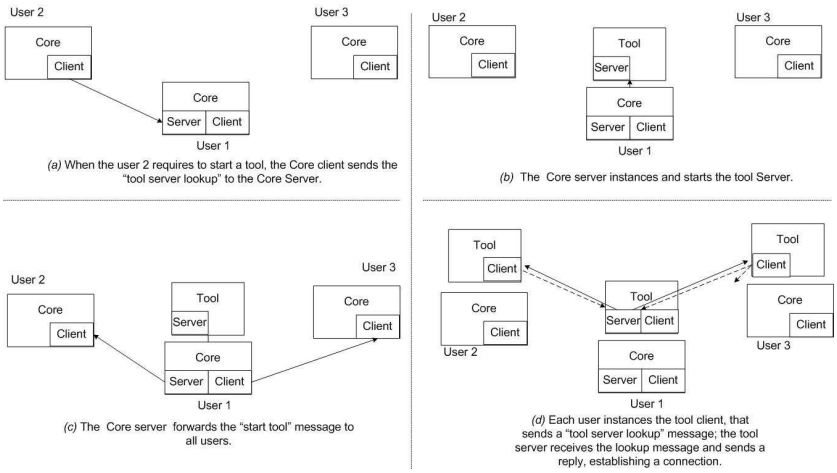

Fig. 6. The tool activation sequence.

instance runs in the server mode and directly creates and starts the Core server, skipping the lookup message broadcast; the students application instances run in the client mode and look for the server, but if do not receive reply, do not instance nor start the server. Obviously, if there is an application instance in server mode, all the other must be in client mode (or, however, they must fail if try to instance a server). This solution keeps the uniform work environment and the start and play phase, but reserves the access to the servers functionalities for the teacher. As matter of fact, we have forced (by the running modes) the protocol to achieve a powered application instance for the teacher, because the original protocol does not impose conditions about the user that hosts the server. Indeed, with the original protocol, the first user starting a component, instances and runs the server of the component (*p2p running mode*), so that servers of different components could be hosted by different users, moving the system toward dynamic architectures [1]. This solution did not seem suitable for educational settings, where it is preferable to instance and run all the servers at the teacher application, to provide servers functionalities access only to the teacher. The extreme tailorable architecture of RCP enhances the start-up transparency allowing to design the system with a set of servers: a server for the core and a server for each provided tool [3]. With the plug-in based architecture and the lazy activation it is possible to design the system so that in each moment a chosen tool can be activated and then, silently, the server of the tool is started and then the clients of the tool find it.

These features together (transparent start-up and uniform work environment) provide the end user with the perception of a peer-to-peer system, although the system is instead a client server one.

[3] In the educational setting all the server are hosted on the teacher application instance, but even so, it is preferable having a server for each component because this layout enhances extensibility (see sec. 3.2)

4 Conclusions and future work

Here we have presented our studies about the architecture of a system designed and developed explicitly for face to face collaborative learning. Our system provides a uniform work environment and allows the users to *start and play* the application. Compared with existing systems[4], our system is simpler to install, to start-up and to use, because it has neither separate server to manage (uniform work environment) nor network configuration to execute (*start and play*). Furthermore, it inherits from Eclipse advanced tailoring properties.

Since our system has to address specifically face to face collaboration, we can utilize the particular conditions of such context to achieve a more friendly application. The *start and play* protocol takes advantage of the LAN-based context, and really, it is workable only on wired-LAN, because the UDP broadcast is often disabled out of the LAN. Furthermore, the local network often offers low variance delay and this helps to prevent (but it is not the best solution, of course) race conditions in the server start-up phase. Vice versa, the high delay variance of wireless LAN may cause anomalous behaviors of the protocol, due to, for example, expiring timeouts. To use a similar protocol on wireless LAN, it must be specifically designed to address WiFi peculiarity.

In the context of face to face collaborative learning, the server functionalities should be managed by the teacher, so that all the servers (Core server and tools servers) are hosted on the same application instance (the teacher's one). Actually, the described architecture forces this behavior, but interesting studies concern the p2p running mode, that provides the opportunity of hosting the servers in a distributed way over different application instances (for example, the first user starting a tool can host the tool server), migrating the system toward dynamic architectures [1]. Even if the p2p running mode may be unsuitable for educational settings, we wish briefly describe some interesting features and problems related to the p2p running mode. The opportunity to have a distributed servers set allows to share the workload between all the users; furthermore, this allows to relax the roles strictness, matching situations more dynamic and flexible (such as a work group where different members have different roles and competences) than the educational one, where there are the well defined student and teacher roles. A problem related to the distributed servers set concerns the shutdown of a single application instance hosting one server: it should be a *controlled* shutdown, to allow the server migration toward another application instance (i.e. another application instance creates and runs the server). An even more complex problem concerns the *crash* of an application instance hosting one server, and this requires further studies, as well as the data management protocol. Obviously, the p2p running mode and the distributed servers set are based on the idea that each component embeds its own server. Maybe it is too early to make statement about the user capabilities required by a system with a distributed server set, although we expect that the distributed architecture has no consequences on the user level.

[4] i.e. existing remote systems used in a face to face context

References

1. G. Chung and P. Dewan. Towards Dynamic Collaboration Architectures. In *Proc. of CSCW'04*, pages 1–10.
2. A. Dimitracopoulou. Designing Collaborative Learning Systems: Current Trends and Future Research Agenda. In *Proc. of Computer Supported Collaborative Learning 2005: The next Ten years!*, 2005.
3. Eclipse. http://www.eclipse.org.
4. Robert Johansen. *GroupWare: Computer Support for Business Teams*. The Free Press, New York, NY, USA, 1988.
5. Lead - technology-enhanced learning and problem-solving discussions: Networked learning environments in the classroom, 6th Framework Programme Priority IST. http://lead2learning.org/.
6. J. Lonchamp. Supporting synchronous collaborative learning: a generic, multidimensional model. *International Journal of CSCL*, 1(2), to appear in June 2006.
7. MeetingWorks Software. http://www.entsol.com/html/meetingworks_software.html.
8. G.M. Olson and J.S. Olson. Distance Matters. *Human Computer Interaction*, 15(2-3):139–178, 1994.
9. H. ter Hofte R.Slagter, M.Biemans. Evolution in Use of Groupware: Facilitating Tailoring to the Extreme. In *Proc. of Seventh International Workshop on Groupware*, 2001.
10. A. Sarma, A. van der Hoek, and L.T. Cheng. A Need-Based Collaboration Classification Framework. In *Proc. on Eclipse as Vehicle for CSCW Research, Workshop at CSCW 2004*.
11. R.J Simons. Three ways to get content based and in-depth conversation in on-line learning: revisability,focussing and peer feedback, 2006. Seminario Tecnologia Cultura e Formazione.