CoFFEE : Cooperative Face2Face Educational Environment

Rosario De Chiara, Antonio Di Matteo, Ilaria Manno, Vittorio Scarano
ISISLab, Dipartimento di Informatica ed Applicazioni "R.M. Capocelli",
Università di Salerno, Italy

***Abstract*—Co-located collaboration in classroom is the topic we tackle in this paper. We present** CoFFEE **a tailorable collaborative environment that is designed for interactive, co-located (i.e. Face2Face) collaboration in classroom.**

We present the requirements for tailorability that have driven our design, CoFFEE **architecture, some tools that have been deployed and discuss the latecomer management issue that is offered by the core of our architecture to all the tools that can be developed within the framework.**

I. Introduction

Current research in Computer Supported Collaborative Workgroup (CSCW) has produced many studies and several classifications of the situations where the collaboration takes place. The space-time matrix [20] is a well-known classification that defines the four basic space-time situations obtained by the cross-product of same/different place and same/different time. The focus of our research is on the *same time - same place* situation, i.e. the so called face-to-face (F2F) collaboration [9].

This paper reports some of the technical developments in a research project of the in a VI Framework European Union funded project, "LEAD: Technology-enhanced Learned and Problem-solving Discussions: Networked Learning Environment in the Classroom" [25]. LEAD project is focused on the computer-mediated communication during face-to-face discussions within the existing school setting. The essential situation that is the focus of the project is Collaborative learning [21], in which students at various performance levels work together in small groups toward a common goal. They are responsible for one another's learning as well as their own. Thus, the success of one student helps other students to be successful. The project research partners are defining a model for collaborative problem solving to be embedded in the software [3]. The same model, guides teachers in designing classroom activities supported by the software.

Here, we present the design of a tailorable architecture for F2F collaboration system, named CoFFEE , in an educational setting.

II. The reference technological scenario

In this paper we present the CoFFEE system, with an emphasis on its architecture. From a technical point of view the requirements of computer mediated collaboration in a face to face settings pose some limitations.

In fact, the network infrastructure and the user interactions are particularly influenced by the scenario. By referring to the space-time matrix [20] we consider the collaboration that happens among a group of users simultaneously (same time row in the matrix) connected to the system and that also share the same room:

- Same time: often referred also as synchronous collaboration; users are connected in the same time, this implies that the users do expect a tight interaction with the system, every single user expects a F2F reaction to his/her actions within the system, right when the action is issued (e.g. A laugh corresponding to a joke). Also, the ability of a user to glance at others' monitors offers the challenge to synchronize actions within a time that is felt as "instantaneous" by the user.
- Same place: users shares a room, that, often, is a particularly designed class in school or a meeting room in a workplace. This implies that computer are connected on a local area network. The most important implication of this assumption is that the network is reliable, respect to, for instance, Internet, and the available bandwidth is wide.

Our system has been designed keeping as underlying principle the need, well-known in the community that the software should be easily modifiable ([22]).

In general terms, the user needs evolve over the time and the groupware should be able to adapt as much as possible to the user expectations. This means that the groupware should be able to adapt to different settings already existing, but should be able also to evolve to fit new scenarios [8]. Several terms have been used in this context (such as tailorability [39], malleability [29]), extensibility, composability [42]) with slightly different meanings.

Our definition of user-centered (UC) tailorability is inspired, in general, by the structure provided by Slagter [39], [38] with a user-centered approach that instantiates the stakeholders of tailorability's advantages. In our view, four different (and increasingly complex) forms of UC tailorability can be envisioned:

- **Tailorability by Customisation:** it is the simplest form of UC tailorability; it allows to configure the basic properties of a groupware, in such a way to slightly modify the behavior of a functionality.
- **Tailorability by Integration (or Composability):** it allows the user to select the desired functionalities (tools) from a predefined set that is given within the system.

It requires predefined communication interfaces between the components. In ([42]) three composability levels are defined: coexistence (the simplest one), connecting components with predefined communication interfaces, custom composition of components; in our definition only the first two levels fit within the tailorability by integration.

- **Tailorability by Expansion:** the user is empowered to widen the set of available functionalities by adding new, compatible tools to the system.
- **Tailorability by Extension (or Extensibility):** it allows the user to add new components from third parties in the groupware without changing the existing components. This requires open standards for interoperability.

Our user-centered approach allows to clearly identify the categories of users which each different tailorability is aiming to. In fact, the user/learner needs only the configurability of the system (Tailorability by Customisation), the teacher/facilitator needs the Tailorability by Integration, the researcher requires to expand the fixed set of tool with new additional services/tools (Tailorability by Expansion) and, finally, the developer is entitled to the full tailorability that can be offered, by extending the system with newly designed tools (Tailorability by Extension).

This (skewed) "pyramid" of tailorability with the relationship to the users categories is depicted in Figure 1.

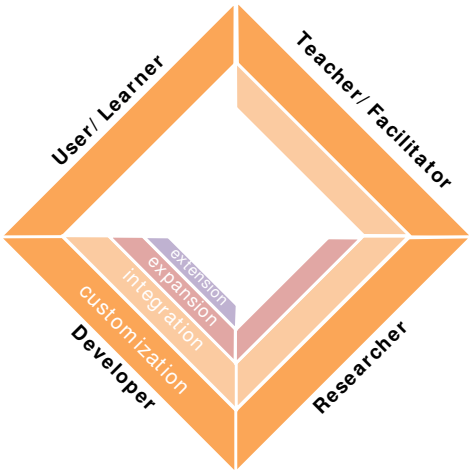

Fig. 1. Tailorability pyramid: it is obtained by mapping different User-Centered Tailorability levels to different kinds of users.

A. Related works

The research in F2F collaboration is often aimed to implement it in contexts where different typologies of hardware are available, for instance in [35] they have investigated the use and the support to collaboration of large interactive displays; in [36] it is showed how the use of an interactive surface, a table-top display, can help the sharing of documents. Another approach that can be found in literature is to exploit existing synchronous systems designed for remote cooperation in the co-located scenario, for instance instant messaging systems used in different collaboration scenarios [24], [19], [16]. In such contexts the systems have to support remote collaboration trying to achieve a "virtual co-location" enhancing remote communication by shared calendars, file sharing, audio and video conferencing. In a F2F context some of this communication channels are unnecessary because there is no distance, due to different location, to fill up. Considering that the same-time/same-place scenario is deeply different from the same-time/different-places, it is our opinion that such differences must drive the design of a system for F2F collaboration. The systems to support co-located workgroup and/or learning could and should focus on collaboration activities rather than on reducing distances, for example, they could provide reviewability [32] that are important characteristics in particular in the learning process [37].

Among the recent academic research that is relevant, we cite Omega+, "an environment built around a chat oriented kernel providing the usual functionalities found in regular chat tools with multiple rooms and private channels" ([29]). It is designed according to a "model based genericity". The learning context is defined in four models customizable by users. Omega+ provides advanced tailoring by customization, but does not provide tailoring by integration or by expansion. Indeed, it defines as "developmental malleability" the possibility to modify the implementation, but this feature concerns explicitly the tools' developers, so it does not address tailorability by integration or expansion.

Drew (Dialogical Reasoning Educational Web tool) [10] provides a set of tools to support collaboration. It provides a form of tailoring by integration (it allows the users to choose the tools to use from an initial set) but does not provide customization. It can be extended by *Drewlets*, but this possibility is reserved to developers, so the users cannot extend it.

On the other side, among the commercial products, interesting is MeetingWorks ([31]), a group decision support system (in different versions for same-place/same-time, different-place/same-time, different -place/different-time). It is not highly customizable and does not provide tailoring by integration or by extension for end users. Not much news is available since it is commercial product.

III. CoFFEE system: an overview

A. Design guidelines

While designing the system, some explicit guidelines were drawn in order to serve as non functional requirements for the system. Besides tailorability, three are the main other non functional requirements. We describe them briefly and discuss the impact on the technical design.

Generality is the first, most important, guideline for our design. The system that we designed was not to be devoted to a specific scenario of application and should be able to support a wide variety of applications of F2F computer-mediated communication. Be it used in classroom, in laboratories, or in

professional settings, the infrastructure we developed is able to support a wide variety of tools, providing them general services. Scenarios varies from supporting cooperation of learners in class, or in laboratories, but also in supporting impromptu collaboration (maybe via a wireless ad-hoc connection) or via a private network for a business meeting. Impromptu collaboration springs up where a number of users intend to engage in collaborative activities (such as collaboratively edit shared documents, brainstorm about a topic, contribute to a decision) wherever the collaboration cannot rely on a strong preexistent infrastructure and may be mediated by the support of networked computers. This requirement influenced the design mainly in the choice of the technology for the development of the system. In fact, with a variety of scenarios such as those previously described, the underlying technology cannot be based on neither heavily-loaded application servers, accessible via Internet, nor on WWW based server-push technology (such as AJAX and the hyped Web 2.0). Moreover, the technology must be able to follow the evolution of the users' expectations about the complexity and elaborateness of the services provided by a CSCW system.

In general, the design will be careful to achieve generality with **simplicity and usability** in all the contexts, by separating, for example, (potentially complex) configuration by the (simpler) execution, as well as providing a consistent hierarchy of complexity in the level of configuration that is needed. The trade-off between generality and simplicity will be exploited by the design, by addressing different needs in different scenario, realized by users with different capabilities and diverse background (a teacher, a researcher, an informal meeting facilitator, etc.).

Low cost deployment, that is, easy start up, management and low network requirements, is another important argument for CSCW F2F systems, because they are applied to end users that may have no specific abilities and experiences to configure complex systems in the meeting rooms. As a matter of fact, we consider the low cost deployment as a fundamental point to encourage real use of co-located systems: such systems should be easy to install, to configure, to use and to manage. Therefore, a significant aspect of our work will be in achieving the best trade-off of effective collaboration features coupled with a low cost deployment approach.

B. Software Technology

As previously described, tailorability is a crucial issue for CSCW software architectures. Our architecture is based, as a consequence, on a foundational component-based framework Eclipse Rich Client Platform ([12]). The framework is based on the core functionalities of Eclipse, a component-based Integrated Development Environment that, besides being an open development platform for building extensible development frameworks, also offers RCP to build general purpose applications using the Eclipse architecture.

In Eclipse architecture there are three groups of plug-ins: the core, named Rich Client Platform, the main application (that in Eclipse is the development environment) and the optional plug-ins. The RCP and the optional plug-ins can be used to build general purpose applications. The Eclipse architecture is based on the concepts of plug-ins, extension-points and lazy activation. Briefly, the plug-ins are the components of the system, the extension-points are the rules of plug-ins composition and lazy activation is the property of activating a plug-in only on demand.

The network communication between the distributed components is based on the Eclipse Communication Framework ([11]); it is a subproject of Eclipse community and provides framework for supporting the development of distributed Eclipse-based tools and applications by using asynchronous point-to-point or publish-and-subscribe messaging functionalities.

Our choice of the reference software frameworks on two well-established products, Eclipse and ECF, that are open source and Java-based, is meant to ensure sustainability of the project, by providing the availability to new scenarios and services (for example, in ECF it is already supported Voice over IP, as well as standard communication protocols (bitTorrent, XMPP (Jabber), Skype, etc.)).

C. System architecture at a glance

COFFEE is a suite of Eclipse RCP-based applications: the Roster Editor, the Session Designer, the Session Player and the Session Client (see Fig. 2). The Session Player (launched by the teacher) and the Session Clients are the applications developed to support the face to face collaboration in the classroom. The teacher organizes the session by using the SessionDesigner, leveraging on a set of pre-assembled template sessions (coming from the experiences and research from the pedagogical partners of the project) and template configurations of the tools. Then the session is played at runtime by the server (Session Player) that allows to choose also the authentication mode (no nickname, or nickname/password as specified in the roster of the class).

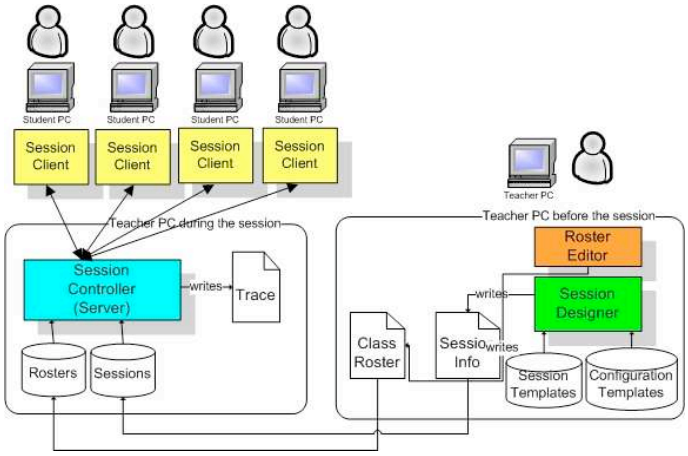

Fig. 2. The architecture of COFFEE

The Session Player and the Session Client have a component based architecture. We have defined a core component

(respectively, the SessionPlayer and the Session Client) and the tools components. Each tool has a server side extending the Session Player and a client side extending the Session Client. The integration of the tools on the cores is managed with the extension-point mechanism: we have defined on the cores an extension point specifying which information and API must be provided by the tools. Any tool wishing to extend CoFFEE has to provide a server component and a client component and both the components have to extend the extension point, providing the required information and API.

The communication between the Session Player and the Session Client is based on the Eclipse Communication Framework (ECF); it is a framework developed within the Eclipse foundation to support the creation of plug-ins (extending Eclipse or Eclipse-based applications) requiring messaging and communications functionalities. We use two kinds of communication objects provided by ECF: the containers and the shared objects; the containers provide access to a communication protocol; the shared objects manage the messaging; the shared objects are hosted by a container where they are uniquely identified. A shared object can send/receive messages only to/from other shared objects with the same ID. We have used an ECF container in the core of the Session Player and of the Session Client, while each tool uses the shared objects (Figure 3). In detail, each tool defines a Service as a pair (GUI, SharedObject), where the GUI provides the tool functionalities to the user while the shared object provides the communication functionalities; fundamentally, the Service represents the tool functionalities; potentially, a tool can have several independent Service instances running at the same moment. Each tool has

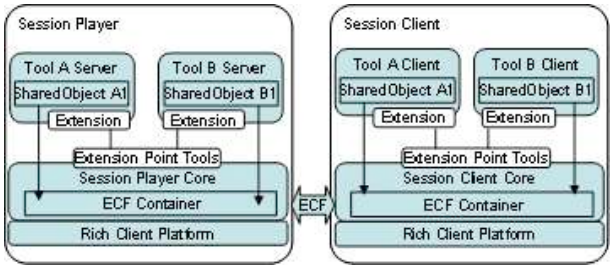

Fig. 3. Session Player and Session Client Architecture with the ECF objects

a set of properties that allows to configure the Service; some basic properties are *anonymous* and *private* (that is, a tool can be available for single users without collaboration), but further properties can be used by specific tools. More details are available in the next section.

IV. CoFFEE Sessions and Tools

Several pedagogical studies introduce Scripts to scaffold by structuring the collaborative learning to lead the interactions in a productive way ([23], [44]). Dillenbourg in [6] defines a script as a sequence of phases, then define a phase as a set of four attributes: the task that the students have to perform, the group composition, the interaction mode and the timing of the phase, and then defines each of these attributes. Similarly, in the area of commercial systems for meeting facilitations, systems such as MeetingWorks [31] and GroupSystem [1] provide "agendas" for the meeting, whose essence is similar to CoFFEE sessions.

Following a similar structure, the usage scenario of CoFFEE is structured in several phases; in each phase there may be a classroom discussion or a groups discussion and the interactions mode is defined by the set of tools chosen among those provided by the system.

The activities sequence is assembled, beforehand, as a *Session* defined as a *sequence of steps*, through the Session Designer component of CoFFEE . Of course, a set of predefined Sessions as well as a set of Session templates, will be provided to the teacher so that his/her task is facilitated since he/she can use an existing Session, modify an existing Session, create a Session from an existing template, or create a Session from scratch. Once it has been created and saved, the session is used as input by the Session Player (see 2) that executes the steps and drives the Session Clients.

We emphasize here that by allowing to structure the collaborative process in a Session CoFFEE provides the teacher with the capability of composing the desired set of functionalities in each step, thereby providing what we called *Tailorability by Integration*.

A. A CoFFEE Session

The collaborative script is modeled as a *Session*. A session consists of a *sequence of steps* that fit with certain pedagogical objectives, tasks and instructions, and include a working method with rules and techniques.

A step can be a *classroom step* or a *group step*, that is, can have just one group (i.e., the whole classroom) or more groups. For sake of clarity, we will refer in the following to group both if there is only one group (i.e. the classroom) or several groups. Several *services* can be enabled and configured differently, for each group. For example, one of two groups may be using tool A and tool B, while the second group is using tool A and tool C. A service is a *tool* with a specific configuration that modifies some characteristics of its functionalities.

During the playing of the sequence (in class), before a group step begins, the teacher populates the group by assigning each student to a group and, then, let the group step begin: each group will use the services as defined.

During the execution of a group step, the students can see only the artifacts created within their group, while, at the end of the step, the artifacts of all groups are revealed to everybody.

Furthermore, at the end of each step, the services are *frozen*, so that all the artifacts are readable, but no other changes can be done within that step.

The Session leading the collaborative process must be planned in advance, and this allows to design it very carefully, structuring the whole process and customising the tools exactly as desired. While useful in certain settings, as noticed by some of the pedagogical partners of the project, such a "rigidity" of the session doe not fit naturally in some scenarios where more flexibility is needed in order to follow the deflections of the

collaborative process. Current research about *over-scripting* [6] is introducing the need of flexibility into the enacting of the scripts [7]. So, currently, we are realizing also a dynamic environment where steps can be added into an (existing) session at run-time. As suggested by earlier experiences with CoFFEE prototypes, and by the simplicity non-functional requirement described before, it is important that *extra steps* can be added easily and quickly, during the collaborative process in the classroom, so a set of predefined extra steps will be provided (and possibly augmented by configuring the SessionPlayer) so that the teacher only has to select the kind of step to include.

By offering the possibility to define the configuration of a tool (within the service) and by allowing both off-line flexibility (during the design) and on-line usability and flexibility at once (by adding extra steps selected among a set of predefined steps)CoFFEE offers an environment that can be easily made by the teacher to fit his/her classroom specific needs, thereby providing the *Tailorability by Customisation*.

B. The tools

CoFFEE provides a high level of Tailorability by Integration (or composability) since it allows the user (during the design of the session but also later on, during the execution) to select for each step the set of tools that are more suitable for the step, chosen among a set of tools.

All the tools that we developed follow an important important guideline called malleability [28], i.e., they have many configurable options, so that they can be made to fit different users needs. Some of the configurable options are common to all the tools, for example the anonymity or the possibility to use a tool as a private workspace instead of a collaborative workspace. Many other options are tool specific.

The project is focusing, currently, on two main tools, a Threaded Discussion tool and a Graphical Discussion tool.

The Threaded Discussion tool allows synchronous messaging between the users, structuring the contribution in threads. As reported in literature (see, e.g. [40] for a detailed description) the standard chats have limitations at managing the discussion flow and organizing turn taking, making sometimes the whole discussion comprehension difficult. The usage of a threaded chat aims to address the lack of control over discussion structure in the standard chat. It must be said, that the threaded chat shows also some limitations due mainly to the lack of awareness about the location of new contribution. We addressed this issue by providing a simple (and configurable) awareness mechanism that highlights the most recently added nodes (or branches) in the threaded view.

The Graphical Discussion tool allows synchronous messaging between the users, representing the contributions as boxes in a graphical space, eventually linked by several kinds of arrows. This tool is designed to support brainstorming processes and conceptual maps creation, but it is enough generic and malleable to satisfy other usage scenarios.

CoFFEE provides also other tools, like a Group Presence tool to provide presence and group membership awareness within the groups, a Co-editor tool to allow cooperative writing with strict turn taking (just one user at a time), and a Private Notes tool to provide a personal workspace to write some textual notes.

1) The Threaded Discussion tool: Several studies report that chat tools introduce a certain ambiguity in the interactions, by not offering the (natural) correlation between contributions and replies. In literature, the shortcomings of a standard chat are well known [17], [40], [33], [27], [34] and particular attention is payed to the difficulties in following the discussion flow, that is often beyond the time ordering offered by the chat. The threaded chat mechanism is recognized as on of the natural solutions to address the standard chat lacks of coherence (other proposals are presented in [45], [15], [43]).

As noticed in literature, e.g. [40], one of the shortcomings of a threaded chat is the lack over the awareness of new contributions. To address this issue, the Threaded Discussion tool provide the possibility to highlight the last n contributions (and their forefathers if the tree containing the new contribution is closed). It is possible to customize this possibility choosing the number n of contribution to highlight.

The unstructured nature of standard chat affects also the kind and the length of the contributions, because the users aim to make short contributions in order not to lose the conceptual link with the reference contribution. Because of the threads, in a threaded discussion tool contributions can be more articulated and long. Therefore, the Threaded Discussion tool provides multiline contributions, i.e. contributions that can be longer than one line, allowing the user to compose it.

Furthermore, the Threaded Discussion tool can be configured so that it uses Categories, i.e. separate threaded chats with a name, that are accessible (and active) at the same time. This allows to structure the discussion along separate topics each one with its own separated threaded chats.

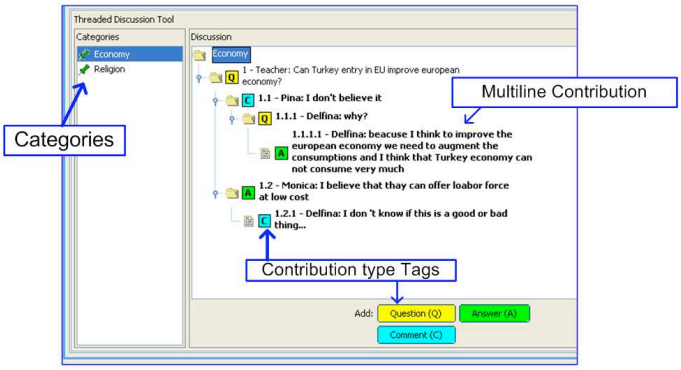

Fig. 4. A snapshot of the Threaded Discussion tool.

The categories are created at runtime and can be configured so that they can be created only by the teacher or by all the users. It should be noticed that Threaded Discussion tool can be configured to work with or without Categories.

The Threaded Discussion tool can be configured so that it offers the possibility to tag each contribution with a label identifying the contribution type (for example Q:Question,

A:Answers, C:Comment, and so on). As said, this functionality is configurable, and the teacher can choose to use or not the tagging mechanism, and can define the desired types (and their labels). The possibility to define new contribution types is important to assure the generality of the tool, since whatever closed set of predefined types would be insufficient, sooner or later.

Another interesting feature provided by the Threaded Discussion Tool is the possibility to add *private* contributions in a *public* threaded chat: given a threaded chat used by a group of users, anyone can add a private contribution that will not be seen by all other users. This functionality is also configurable.

Other configurable options of the Threaded Discussion tool are about the structure of the tree underlying the threaded chat discussion. First, the number of contributions allowed to any user can be limited to achieve a brief discussion. This functionality is useful in classroom, when the available time slot could be short and is believed to be fostering better though-out contributions. Obviously, the number of allowed contribution can be configured also as unlimited. Then, the maximum tree depth can be limited to avoid to obtain trees too large, uncomfortable to read. Also this option can be configured as unlimited. Finally, the numbers labeling the contributions in most threaded chats can be configured to be shown or hidden.

As shown so far in the tool description, the malleability has been a fundamental guideline in designing and developing the Threaded Discussion Tool, so all the key features are been made configurable. The high configurability makes the Threaded Discussion tool achieve the requirements of Tailorability by Customization. This property assure the generality required to make the tool adaptable to many different scenarios, following the principle that the groupware should evolve together with the user needs [38].

2) The Graphical Discussion tool: Given the chat limitations in the educational environment, several studies introduce argumentation and discussion with structured tools, not necessarily tree-structured. In some cases the 2-dimensional plane is used (instead of a hierarchical thread in a threaded discussion) to structure the contributions in such a way that the learners can shape the space as they "embed" their argumentation within it. With regard to this aspect, the graphical tool, while germane to the threaded chat in objectives, seems to be offering more expressiveness to the learners than hierarchies. Part of the research of the pedagogical partners in our project are devoted into exploring peculiarities and shortcomings of the two tools.

CoFFEE Graphical Discussion tool provides a graphical workspace where the user can structure a discussion through boxes containing the textual contributions. The boxes can be linked with arrows and lines to represent relations between the contributions.

The boxes can contain maximum 100 characters, and they have all the same size, so that any box can not "dominate" graphically the others on the screen. They can be configured to represent the contribution type through a label and a color. The set and the representation of the contribution types is not predefined, and new contribution types can be configured defining their representation. Similarly, new connectors types can be configured defining their style (solid, dash, dashdot, etc.), color, semantic and if they should be arrowhead or not.

As already noticed for the Threaded Discussion tool, the open set of contribution types and connectors ensures the generality of the tool, allowing the user to define the contributions types (and connectors types) best fitting his/her needs.

The Graphical Discussion tool can be also configured so that the contribution type can be applied (or changed) on the boxes after their creation. The hypothesis behind this choice suggests that the free of thoughts is not interrupted and that no blocking happens if the contribution can be first written and, then, assigned a contribution type to. Furthermore, the connectors can be configured so that they can have bend points (maximum 4) in order to make the diagrams more graphically pleasant and well-ordered.

The Graphical Discussion tool, like the Threaded Discussion tool, has been designed and developed to be highly configurable and to satisfy the Tailorability by Customization. As already noticed for the Threaded Discussion tool, generality that is achieved is needed to make the tool adaptable to new scenarios and users expectations.

3) Other tools: The Threaded Discussion tool and the Graphical Discussion tool are the main tools provided currently by CoFFEE within the Lead project; nevertheless, other tools have been designed that can be used in the Session steps. These tools provides supplementary functionalities with respect structured discussion, and they can be composed to work with the discussion tools or between them without the discussion tools.

The Group Presence tool can be enabled for each group and provides the (group) users with presence awareness and group membership awareness. It can be configured to be colored so that each user is represented with a color depending on the group to which the user belonged in the previous step.

The Private Note tool provides a private textual workspace to each user; it can be used to write personal notes or to prepare some contribution in advance.

The Co-editor tool provides a shared editor with strict turn-taking: in each group, just one person can write, while the artifact is shown to everybody. The teacher can move the write right to another member of the group. This tool can be used, for example, to write reports of a session, where the writer represents the spokesperson for the group.

The tools just described, together with the discussion tools, form a predefined set of modular tools provided by CoFFEE to design the Session. This is the set of tools which the teacher can choose from to configure them in each step and group, thereby offering the Tailorability by Composition.

4) Developing new tools: Our main aim in designing CoFFEE was to provide an environment that could be simple to extend by integrating new or third party tools. The Eclipse RCP component-based architecture already allows to develop

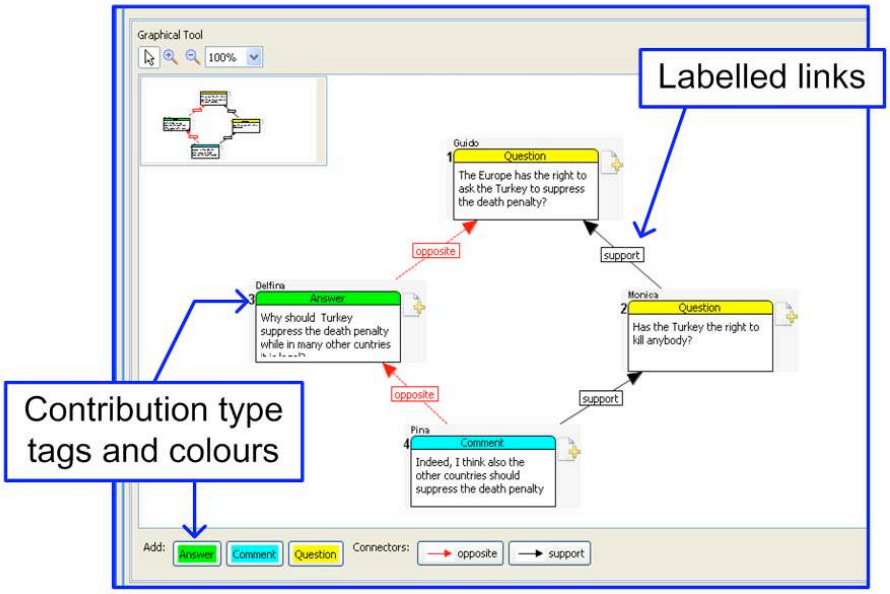

Fig. 5. A snapshot of the Graphical Discussion tool.

and to integrate new tools as bundles, but we have focused on designing the system so that new tools can be added without requiring knowledge about the core or the communication framework: CoFFEE provides the superclasses to implement a tool with minimum effort.

A tool has a server side extending the SessionPlayer and a client side extending the SessionClient. Most tools only need a simple server that mediates the communication: the clients send the messages to the server that broadcast them. Some special tools may require a server which is able to elaborate the message before forwarding it. Since the implementation of a tool is very simple, we can describe the required four basic classes (for both client and server side):

- **The Activator:** this class manages the plug-in life cycle; it extends a superclass provided by the CoFFEE core, and needs only to implement the constructor that invokes the constructor of the superclass passing as parameters the ids of the tool and a reference to a Tool.
- **The Tool:** this class manages the Service life cycle, where a Service is a pair <GUI, SharedObject>; it extends a superclass provided by the core and needs to implement the abstract method createToolService(); within this method the Tool must create the Composite and the SharedObject of the Service.
- **The Composite:** this is the GUI of the tool; it has a reference to the Shared Object of the Service which both belong to.
- **The SharedObject:** the (ECF) communication object; it extends the super-class provided by the core and needs to implement:
  - a method to handle the received messages by passing them to the GUI and (the server side) forward it to the clients.
  - a method to send the messages received by the GUI by using a method of the super class.

Because of this simple structure, the implementation of new tools is extremely easy for a medium-skilled programmer, but also the integration of tools from third party is facilitated. As demonstrative example, we have integrated in CoFFEE the Chat tool from another (past) european project Drew ([10]); during the integration we had just to provide the classes for the life cycle management both of the plug-in and of the Services, and then we replaced the Drew communication level with our SharedObject. With the support from a programmer from the Drew project (which we gratefully thank for), the task was accomplished by 1 person in 3 hours.

Similarly, we have integrated a tool, provided by ECF, named ScribbleShare implementing a shared whiteboard. The required integration process is about the same as for the Drew chat: we have provided the classes for the life cycle management both of the plug-in and of the Services, and then we replaced the communication level with our SharedObject.

V. Latecomer Users Management

CoFFEE provides native support for managing latecomers. A latecomer is a user that connect to the system after the cooperative session has started and some work has been carried out by users.

Handling the latecomers is an important problem whose efficient solution strongly influences the interactivity and the usability of the system. In fact, managing latecomers in a synchronous session, while difficult, it is an important requirements in a real setting, where latecomers or accidental disconnection and reconnection are possible.

The efficiency of the solution is compared with the settings of the problem, since a latecomer needs a snapshot of the

whole system state to start collaboration with other users, and the state size is influenced by several factors like the number of connected users, the frequency of contribution and the average memory occupation of the contributions. Of course, the "later" is the latecomer, the larger is the state.

A. Related works

The problem of the latecomer in a distributed cooperative setting has attracted some attention from the researchers in very different technological setting; some very early work was presented in ([4]), where a solution for supporting latecomers in X-Windows-based applications was presented.

In general, solutions to the latecomer problem fall into two categories: solutions based on the transport protocol (i.e. collect all the packets and send them to the latecomer) or at the application level. While in the first category we find only one solution (Scalable Reliable Multicast protocol) ([14]), whose efficiency seems limited by the fact that also non-interesting packets are stored and sent to the latecomer, the rest of solutions fall in the second category.

In ([18]) the authors describe the mechanism in a Web-based setting for Java applets, by sending the entire model and interface (i.e. the whole object) over the network. The solution requires explicit locking on all the participants to the cooperative system and, while a general approach, its performances are acceptable in a geographical setting but not in a F2F scenario (like ours) where even the smallest delay or asynchrony is immediately felt by the user (since they can sneak the monitors of their neighbors).

The same algorithm has been recently used in ([13]) with the additional capabilities of multiple sessions and a focus on distributing the state among the clients. The mechanism incurs in additional overhead since, first a distributed locking algorithms is run, then, the selected client sends the state to the server which forwards it to the latecomer. The advantage is on sharing the workload of storing the state of each service among all the clients.

In ([30]) a solution is presented that handles a latecomer through the cooperation of all the clients connected to the system; the objective is to study a mechanism that is tolerant to server unavailability, a scenario that in our (local, synchronous) setting is not covered.

B. Our technique

Compared to the results available in literature, our technique can be described as hybrid. We define, in fact, the state of a CoFFEE service as the set of all the (application-level) messages exchanged by all the connected client services to the CoFFEE server (Session Player). In this case, our hybrid solution, while deployed at the application level, thereby with the knowledge of the system and its semantics, is general and can be used in two different cases:

- Latecomer client's management: when a latecomer client is connected to the server, it receives the session messages to active all service planned on the session and it must receive all the past communications exchanged within the system.
- Lost messages management: when a client is out of sync in the communication, it must receive all the lost messages.

To the best of our knowledge, our technique is the first that is suitable for F2F collaborative applications, that keeps the generality of the solution, and is suitable for multiple groups. Moreover, our mechanism is automatically activated for each tool that is embedded on the CoFFEE platform; in fact, communication about the state is transparently managed without any knowledge of the tool semantics or message format. Of course, if one wants a more efficient state management for a specific tool, the state communication methods can be overwritten.

When a latecomer logs into the system two events are fired:

- The current CoFFEE state (which is identical for all collaborators) has to be transferred to the latecomer. This will synchronize the latecomer to every other client.
- During the preparation of the state snapshot and its transmission to the latecomer, events from (other) clients must be collected and queued to preserve the interactivity of the whole system.

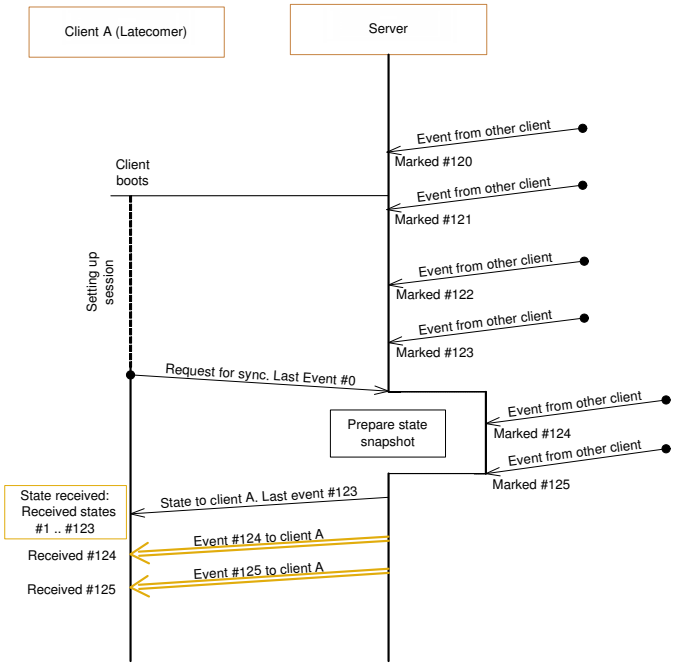

Fig. 6. The latecomer management sequence diagram.

The state management takes into account both situations: latecomer's arrival and lost messages. In the client side, each activated service sends a state request message to the service server side and discards each received message while it waits for a state response message. On ordinary (i.e. not latecomer) clients, the start-up state is an empty set of messages; in latecomer clients, the state is a set of all the messages

that is processed by the latecomer client that is, therefore, synchronized.

During its activity, when a service receives a message it updates the current sequence number with the received sequence number message; if there is a discontinuity in this update, the service has lost some messages and is out of sync and must ask to be updated by using a new state request. So, the service sends a new state request message and it discards all received messages while it waits the state response. The state response message contains the updated current sequence number and the new state (the set of all last messages). When the service receives the state response, it is in sync and stops the messages discarding mechanism.

This mechanism is managed by CoFFEE architecture (via ECF event processor) at the service level and it is transparent to all CoFFEE plug-in implementations. It simply exchanges a set of messages and an updated sequence number of each active service (plug-in). The components involved on this mechanism do not know the messages classes implementation and their content.

On the client side, the state mechanism is managed by Client State Manager (*CSM*) (an ECF event processor): at the beginning, it sends a state request message with the current sequence number (zero for both normal and latecomer clients).

This mechanism is also activated on out-of-sync situations: the processor tests the out-of-sync situation for each received messages and it sends a new state request message (with a 10 seconds timeout before sending again the request).

On the server side, mechanism is simply managed by a server state manager that handles state request messages.

VI. CoFFEE Tailorability

As described in Section II, tailorability of groupware is fundamental to assure users satisfaction by evolving and customising the system as new needs and requirements come out. Following this guideline, we have designed CoFFEE so that it provides several kinds of tailorability, as we previously defined them. In the following, we want to highlight how some design features fully address and fulfill the tailorability levels we defined. Of course, we are aware that the Tailorability is based on Java and Eclipse, but the implicit limitation of the framework is ameliorated by the wide diffusion of Java, and existence of a number of wrappers toward different frameworks.

Tailorability by Customisation. As described in Section IV-B, CoFFEE tools provide many options to configure the major functionalities and properties, so that each tool is generic enough to be adaptable to many different contexts and scenarios. The teacher can fully configure each tool as desired, in the session design phase, through the Session Designer component.

To get the best trade-off between configurability and usability, CoFFEE will soon also provide templates of tools configuration that will also be used to launch a tool at runtime, when there is no time to make carefully configurations.

The high configurability of the tools makes our system fulfill the Tailorability by Customisation requirement; this kind of tailorability is granted to the teacher (power user) in the session design phase.

Tailorability by Integration. As described in Section IV-A, the system allows to structure the collaborative process in a Session, defining the sequence of collaborative steps and selecting for each step the desired tools. Offering the capability of composing the desired set of functionalities in each step, CoFFEE provides what we called *Tailorability by Integration*. Similarly to the previous case, this kind of tailorability is granted to the power user in the session design phase. To balance the tailorability and usability, CoFFEE will provide also a set of predefined sessions and of session templates, to facilitate the session definition.

Tailorability by Expansion. Similarly, the Tailorability by Expansion is provided since the teacher (or the researcher) can download and upgrade new tools (RCP provides a mechanism to install new plug-ins and update old ones), developed by independent developers within the CoFFEE framework (see Tailorability by Extension, next).

Tailorability by Extension. During the design of CoFFEE we aimed to achieve Tailorability by Extension, i.e. to add new components or integrate components from third parties in the groupware without changing the existing ones, as well as to make the extension process as simpler as possible. In particular, using the Eclipse plug-in based architecture and designing carefully the tools integration mechanism, we have achieved a system where new tools can be integrated with minimum effort: as described in the section IV-B4, the implementation of new tools is extremely easy, but also the integration of tools from third party is facilitated.

However, the extension process requires some technical capabilities that are beyond the medium skill of learner, teacher or pedagogical researcher, so this kind of tailorability is granted to the Developer.

VII. Conclusions

As discussed in the previous sections, groupware can provide several degrees of tailorability, on the basis of the kind of tailorability and the target user. By describing the architecture of CoFFEE we presented how it addresses the four different kinds of tailorability that are well-suited to be mapped to the four categories of users.

Currently, CoFFEE is actually in Alpha release, whose realization was paralleled (in the last 2 years) with some early experiments and test within classes and in labs by the pedagogical partners of the Lead project.(see [2], [5], [26], [41]). The final release, after an evaluation next fall, and a successive round of development with the empirical feedback provided by the experiments will be available (open source) at the end of the project in November 2008.

Acknowledgments: The authors thanks for many fruitful discussions and comments Delfina Malandrino and Ugo Erra. The authors also gratefully acknowledge the technical contributions to CoFFEE by Giuseppina Palmieri

and Furio Belgiorno. All the partners of the Lead project are likewise acknowledged for the useful discussions that inspired and motivated some of the CoFFEE characteristics. Part of this research was funded by VI Framework EU IST STREP project "Lead: Technology-enhanced learning and Problem-solving Discussions: Networked learning Environments in the Classroom", Contract number: 028027 (http://www.lead2learning.org/).

References

[1] Groupsystem web site. http://www.groupsystems.com/.

[2] M. Annarumma, G. Marsico, A. Iannaccone, B. Maroni, and F. Martini. Collaborative learning for teachers in italian educational contexts: knowledge and practices. In *(In Press) In EARLI 2007 Book of Abstracts, in the "Computer support for face-to-face collaborative problem solving" Symposia*, 2007.

[3] M. Baker. Argumentative interactions, discursive operations and learning to model in science. *The Role of Communication in Learning to Model*, pages 303–324, 2002.

[4] G. Chung, K. Jeffay, and H. Abdel-Wahab. Accommodating latecomers in shared window systems. *Computer*, 26(1):72–74, Jan. 1993.

[5] R. De Chiara, I. Manno, and V. Scarano. Design issues for a co-located collaborative learning system. In *(In Press) In EARLI 2007 Book of Abstracts, in the "Computer support for face-to-face collaborative problem solving" Symposia*, 2007.

[6] P. Dillenbourg. *Over-scripting CSCL: The risks of blending collaborative learning with instructional design.*, pages 61–91. Paul A.Kirschner (Ed.), Heerlen: Open Universiteit Nederland, 2002.

[7] P. Dillenbourg and P. Tchounikine. Flexibility in macro scripts for computer-supported collaborative learning. *Journal of Computer Assisted Learning*, 23 (1):1–13, February 2007.

[8] A. Dimitracopoulou. Designing collaborative learning systems: current trends & future research agenda. In *CSCL '05: Proceedings of th 2005 conference on Computer support for collaborative learning*, pages 115–124. International Society of the Learning Sciences, 2005.

[9] A. Dix, J. E. Finlay, G. D. Abowd, and R. Beale. *Human-Computer Interaction (2nd Edition)*. Prentice Hall, 1998.

[10] Dialogical Reasoning Educational Web tool (Drew). http://scale.emse.fr/download/drew.html.

[11] Eclipse Communication Framework (ECF). http://www.eclipse.org/ecf/.

[12] Eclipse. http://www.eclipse.org.

[13] A. El Saddik, D. Yang, and N. D. Georganas. Tools for transparent synchronous collaborative environments. *Journal Multimedia Tools and Applications*, 2006.

[14] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6):784–803, Dec. 1997.

[15] W. Geyer, A. J. Witt, E. Wilcox, M. Muller, B. Kerr, B. Brownholtz, and D. R. Millen. Chat spaces. In *DIS '04: Proceedings of the 2004 conference on Designing interactive systems*, pages 333–336, New York, NY, USA, 2004. ACM Press.

[16] J. D. Herbsleb, D. L. Atkins, D. G. Boyer, M. Handel, and T. A. Finholt. Introducing instant messaging and chat in the workplace. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 171–178, New York, NY, USA, 2002. ACM Press.

[17] S. Herring. Interactional coherence in cmc. *Journal of Computer-Mediated Communication*, 4(4):xx, June 1999. see also.

[18] T. Illmann, R. Thol, and M. Weber. Transparent latecomer support for web-based collaborative learning environments. In *Proc. of CSCL 2002, Colorado, USA,*, November 2002.

[19] E. Isaacs, A. Walendowski, S. Whittaker, D. J. Schiano, and C. Kamm. The character, functions, and styles of instant messaging in the workplace. In *CSCW '02: Proceedings of the 2002 ACM conference on Computer supported cooperative work*, pages 11–20, New York, NY, USA, 2002. ACM Press.

[20] R. Johansen. *GroupWare: Computer Support for Business Teams*. The Free Press, New York, NY, USA, 1988.

[21] D. W. Johnson and R. T. Johnson. *Learning Together and Alone: Cooperative, Competitive, and Individualistic Learning*. Allyn & Bacon, 1998.

[22] M. Koch and G. Teege. Support for tailoring cscw systems: adaptation by composition. In *Parallel and Distributed Processing, 1999. PDP '99. Proceedings of the Seventh Euromicro Workshop on*, pages 146–152, 3-5 Feb. 1999.

[23] I. Kollar, F. Fischer, and F. W. Hesse. Collaboration scripts - a conceptual analysis. *Educational Psycology Review*, 18(2):159–185, June 2006.

[24] B. R. Krogstie. The potential of instant messaging for informal collaboration in large-scale software development. In *Workshop: The Social Side of Large-Scale Software Development.*, 2006.

[25] Lead - technology-enhanced learning and problem-solving discussions: Networked learning environments in the classroom, 6th Framework Programme Priority IST. http://lead2learning.org/.

[26] M. B. Ligorio, L. Tateo, I. Manno, R. De Chiara, and A. Iannaccone. Coffee: a software to blend face-to-face and written communication in collaborative problem solving-based scenarios. Summer School "Building Knowledge for deep Understanding", at the Institute for Knowledge Innovation and Technology, August 2007.

[27] J. Lonchamp. A structured chat framework for distributed educational settings. In *CSCL '05: Proceedings of th 2005 conference on Computer support for collaborative learning*, pages 403–407. International Society of the Learning Sciences, 2005.

[28] J. Lonchamp. Multi-dimensional model-based genericity in omega+. In *Advanced Learning Technologies, 2006. Sixth International Conference on*, pages 730–734, 05-07 July 2006.

[29] J. Lonchamp. Supporting synchronous collaborative learning: a generic, multi-dimensional model. *International Journal of Computer-Supported Collaborative Learning*, 1(2):247–276, June 2006.

[30] S. Lukosch. Transparent latecomer support for synchronous groupware. In *CRIWG*, pages 26–41, 2003.

[31] MeetingWorks. http://www.entsol.com.

[32] G. Olson and J. Olson. Distance Matters. *Human Computer Interaction*, 15(2-3):139–178, 2000.

[33] J. O'Neill and D. Martin. Text chat in action. In *GROUP '03: Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work*, pages 40–49, New York, NY, USA, 2003. ACM Press.

[34] M. G. Pimentel, H. Fuks, and C. J. P. de Lucena. Co-text loss in textual chat tools. In *CONTEXT*, volume 2680/2003, pages 483–490, 2003.

[35] D. M. Russell, C. Drews, and A. Sue. Social aspects of using large public interactive displays for collaboration. pages 229–236, 2002.

[36] C. Shen. Ubitable: Impromptu face-to-face collaboration on horizontal interactive surfaces. 2003.

[37] R. 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.

[38] R. Slagter, M. Biemans, and H. ter Hofte. Evolution in use of groupware: Facilitating tailoring to the extreme. In *CRIWG '01: Proceedings of the Seventh International Workshop on Groupware*, pages 68–73, Washington, DC, USA, 2001. IEEE Computer Society.

[39] R. Slagter, H. ter Hofte, and Stiemerling. Component-based groupware: An introduction. In *In Proc. of Component Based Groupware Workshpo of CSCW2000*, 2000.

[40] M. Smith, J. J. Cadiz, and B. Burkhalter. Conversation trees and threaded chats. In *CSCW '00: Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 97–105, New York, NY, USA, 2000. ACM Press.

[41] L. Tateo, G. Ammaturo, M. Annarumma, A. Iannaccone, I. Manno, A. Di Matteo, R. Prinzi, and F. Belgiorno. Un coffee è più buono in compagnia! design partecipitavo di coffee software a supporto del collaborative problem solving. Poster in the Symposium "Tecnologie emergenti e costruzione di conoscenza", Università degli Studi di Cassino, March 2007.

[42] G. H. ter Hofte. *Working Apart Together: Foundations for Component Groupware*. PhD thesis, Telematica Institute, Enschede, The Netherlands, 1998.

[43] D. Vronay, M. Smith, and S. Drucker. Alternative interfaces for chat. In *UIST '99: Proceedings of the 12th annual ACM symposium on User interface software and technology*, pages 19–26, New York, NY, USA, 1999. ACM Press.

[44] A. Weinberger, B. Ertl, F. Fischer, and H. Mandl. Epistemic and social scripts in computer-supported collaborative learning. *Instructional Science*, 33(1):1–30, 2005.

[45] L. Xiao and J. S. Litzinger. Unraveling the ordering in persistent chat: a new message ordering feature. In *GROUP '05: Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work*, pages 450–451, New York, NY, USA, 2005. ACM Press.