A Flexible and Tailorable Architecture for Scripts in F2F Collaboration

Furio Belgiorno, Rosario De Chiara, Ilaria Manno, and Vittorio Scarano

ISISLab
Dipartimento di Informatica ed Applicazioni "R.M. Capocelli"
Università di Salerno
Fisciano (Salerno), 84084, Italy
{furbel,dechiara,manno,vitsca}@dia.unisa.it,
http://www.isislab.it

Abstract. In this paper we introduce the architecture of the script engine of a collaborative co-located discussion support system, named CoFFEE, and, in particular, we describe its extendibility and flexibility as a macro-script engine for CSCL activities [7].

1 Introduction

Among educational researchers, it is well known how critical is the design of the educational setting since it strongly impacts on learners activities. Their research has proved that fruitful collaboration require careful design and not occur *per-se* [6], and that, otherwise, the partners may not engage in productive collaboration.

One of the possible solutions in order to stimulate productive interactions within the collaborative process are the *scripts for collaborative learning* [16] that are considered useful instruments to facilitate and stimulate equal participation in collaborative learning activities with respect to the unscripted collaborative learning activities. This research is often referred to also as "learning scenarios" or "task sequences". Several examples of scripts date back even before the term was coined such as the well-known Jigsaw [1] but more recent examples are Concept Grid script [5], ArgueGraph script [11] and RSC script [2].

While scripting is generally considered useful and fruitful, the researchers are clearly aware of the dangers of too much structure, called over-scripting [5] where, in general, the scripts become obstacles to learners' own strategies. In [7], the author investigates the trade-off between the structure provided by the script and its *flexibility*, i.e. the possibility that the teacher and the learners can modify some features of the script. In the same paper, the distinction between intrinsic (to the pedagogical design) constraints and extrinsic (i.e. induced by the technology) constraints is made, so that the goal of a script engine can be clarified as to help maintaining the intrinsic constraints and to provide flexibility on the extrinsic constraints.

Following [7], (macro-)scripts can be defined as pedagogical methods that aim at producing desired interactions, and consists of several phases, namely, individual phases, collaborative phases (intensive interactions within small groups) and collective phases (looser interactions with larger groups e.g. classroom discussions).

Besides flexibility, another characteristic among CSCL systems has been emerging recently, the well-recognized underlying principle that the software should be easily modifiable ([12]) is based on the fact that users (and their needs) evolve over time and that the groupware should be able to follow the evolution of users' expectations, as well as adapt itself to different settings and new scenarios [8]. With different terms and slightly different meanings, the characteristics has been also referred to in literature as tailorability [18], malleability [14]) and composability [20].

We adopt the following definition of *user-centered tailorability* given in [3] and based on the work on [17, 18] with the explicit instantiation of the stakeholders of tailorability's advantages. In this definition, four different types of tailorability are distinguished:

• **Tailorability by Customisation:** it is the simplest form: it allows to configure the basic properties in such a way to slightly modify the behavior of a functionality. In general, this is the level where "ordinary" users (e.g., learners, novice teachers, etc) are most interested in.

• **Tailorability by Integration:** it allows the user (e.g. experienced teacher or facilitator) to select the desired functionalities (tools) from a predefined set that is given within the system. It requires predefined communication interfaces between the components.

• **Tailorability by Expansion:** the user is empowered to widen the set of available functionalities by adding new, compatible tools to the system. In this case, the needs of a more experienced user (such as pedagogical researchers) are addressed.

• **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 and is of particular interest to developers.

1.1 A flexible and tailorable script engine

In this paper we present the architecture of a flexible and tailorable script engine that is currently being developed in the CoFFEE environment [3, 4, 15] developed within the EU VI Framework Lead project [13]. The project already delivers a face2face cooperative environment with several tools already available and an environment and support to write new tools to be included in the platform. CoFFEE delivers a script mechanism that allows teachers and researchers to write/edit/modify their "sessions" by using a Lesson Planner component that offers the flexibility of choosing tools, their configuration, their layout and their interaction within the session.

The organization of the paper is as follows: in the next section, the architecture of CoFFEE is outlined, emphasizing the parts relevant to the scripting of the cooperative activities. Then, in Section 3, we describe the part of CoFFEE that deals with the scripting and compare its flexibility with other results. Finally, in Section 4, we conclude the paper with some comments and the description of status of the project and the work currently undergoing.

2 CoFFEE system: an overview

In order to present the script engine provided by CoFFEE, we need to describe, first, its software architecture. CoFFEE [3, 4, 15] is being developed within the EU VI Framework Lead project [13]; some preliminary versions are already available for download and the final versions will be available at the end of the project (November 2008). The result (already available for downloading in a preliminary 2.0 version) is a steady, extensible and effective platform that has been developed and used in a variety of schools by the pedagogical partners of the project.

From the purely technological point of view, CoFFEE architecture principles are inspired by the tailorability principles described in the introduction but also, from the pedagogical point of view, by the guidelines [8] that support a *sustainable and reusable* CSCL environment, that allows to "*amplify, transform, and extend their work to new or additional outcomes*".

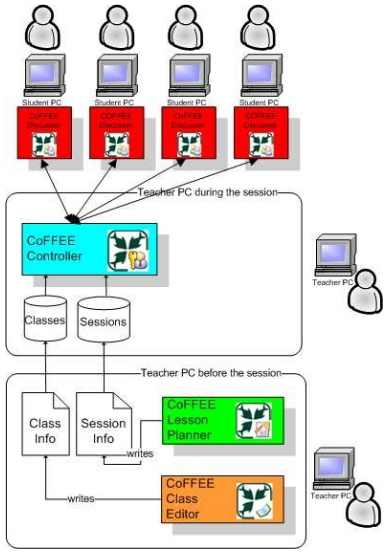
Both the principles in our architectures aim at ensuring a steady and extensible architecture that can be used by a long-term research activity, allowing comparisons over long range of time and extensive experiments, by avoiding the research and evaluation conducted in "commando" style, where researchers and technology are "parachuted" in the classroom for departing as soon as the short experiment is finished, with no significant first-person involvement of schools, teachers and learners.

2.1 CoFFEE at a glance

CoFFEE is a suite of applications: the Class Editor, the Lesson Planner, the CoFFEE Controller and the CoFFEE Discusser. The CoFFEE Controller (launched by the teacher) and the CoFFEE Discussers are the applications developed to support the face to face collaboration in the classroom. They provides access to a predefined set of collaborative tools. The main tools provided at this moment by CoFFEE are the Threaded Discussion tool and the 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. [19] 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 discussion aims to address the lack of control over discussion structure in the standard chat. It must be said, that the structure of a 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.



Both the Threaded Discussion and the Graphical Discussion tool can be configured so that each contribution is tagged by the user according to a notation system, e.g., contributions are tagged as [Q] for Question, [A] for Answer and [C] for Comment (see the example given in Fig. 2). The notation system for each tool is fully configurable: for each tag we can be define name, label, color, shape (for the graphical). Moreover, for the Graphical Discussion tool also the connections can be configured (color, linestyle, arrowheads, text label, number of allowed bendpoints, etc.)

CoFFEE provides also other tools, like a Group Presence tool to provide presence and group membership awareness within the groups, a CoWriter tool to allow cooperative writing with turn taking (just one user writes into the text at a time), a Private Notes tool to provide a personal workspace to write some textual notes and a Positionometer to support voting and giving one's opinion.

2.2 CoFFEE Sessions

The collaboration in classroom via CoFFEE is structured into *sessions* representing the scripts designed to support the collaborative activity.

A session is a sequence of *steps*; each step can have one or more *groups* of students; each group has a set of tools chosen from a set of predefined tools. The groups of students can have the same set of tools or even a different set of tools. Moreover, within each group, a tool can be present more than one time with different configurations.

An example of a CoFFEE session with three steps will be used for illustrating the run-time architecture of CoFFEE and is shown in Fig. 1: in the first step there is just one group where there are occurrences of the Threaded Discussion Tool with different configurations, and an occurrence of the Graphical Discussion Tool; tipically, the two different configurations for the same tool allow to have a private (personal) version of the tool as well as a a shared collaborative one.In the second step there are two groups: both groups have an occurrence of the

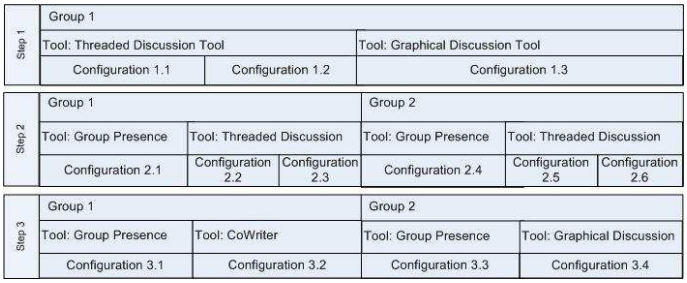


Fig. 1. A CoFFEE session

Group Presence and two occurrences of the Threaded Discussion Tool. Finally, in the third step there are two groups: both the groups have an occurrence of the Group Presence; moreover the first group has an occurrence of the CoWriter Tool, while the second group has an occurrence of the Graphical Discussion Tool.

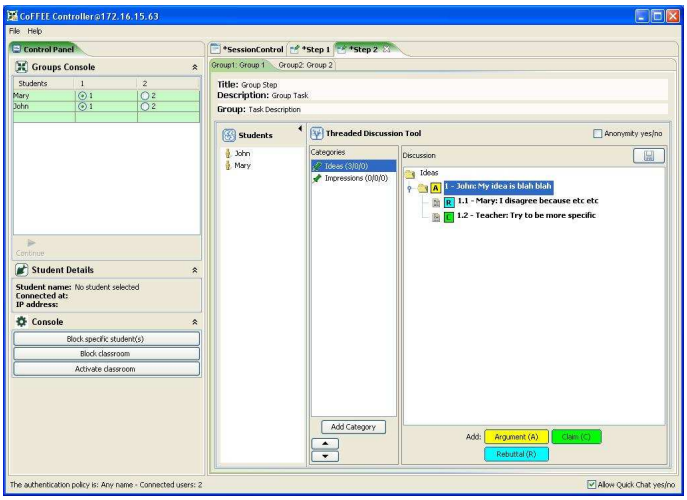


Fig. 2. A screenshot of the CoFFEE Controller. The notation system can be see in the Threaded Discussion tool: each contribution is tagged by the owner and, if configured properly, it can be later changed by the owner/anybody.

In Fig. 2 we show a screenshot of the CoFFEE Controller. On the left the teacher has the Control Panel to manage grouping, blocking and turn taking. The main area is covered by a set of views: the first view (Session Control view) shows the session and allow to control the session execution. The other views are the steps in the session. If a step has more than one group, its view contains a tab for each group. Tools are laid-out within the group tab (or within the

step view, if there is just one group) according to the configuration (that allows up to 5 tools, in positions Center, North, South, East, Wes). The CoFFEE Discusser has a simplified similar interface without the Control Panel and the Session Control view.

The teacher organizes the session by using the Lesson Planner, 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. However, the Lesson Planner allows the teacher to design a new session from scratch. Then the session is played at run-time by the server (CoFFEE Controller) that allows to choose also the authentication mode (no nickname, or nickname/password as specified in the roster of the class). More details on this mechanism will be provided in the next section.

2.3 Software Technology

As previously described, tailorability is a crucial issue for CSCW software architectures. Following this principle, we have designed a component based architecture leveraging on Eclipse Rich Client Platform (RCP) [10]. Eclipse is an open component-based Integrated Development Environment; its architecture allows Eclipse to offer its own core, RCP, to build general purpose applications leveraging on its own component based model.

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. This kind of architecture allows to develop a system composable and extensible. Indeed, designing the tools as components to integrate over a core component allows to achieve a model where is possible both choosing the desired tool from a predefined set and expanding that set by adding a new tool.

The network communication between the distributed components (the server side and the client side) is based on the Eclipse Communication Framework [9], a subproject of the Eclipse community that provides a communication framework for supporting the development of distributed Eclipse-based tools and applications requiring 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.)).

2.4 CoFFEE Architecture

CoFFEE Applications have a component based architecture: we distinguish between the core component (respectively, the CoFFEE Controller Core and the CoFFEE Discusser Core) and the tools components (see Fig. 3). Each tool

has a server side extending the CoFFEE Controller and a client side extending the CoFFEE Discusser.

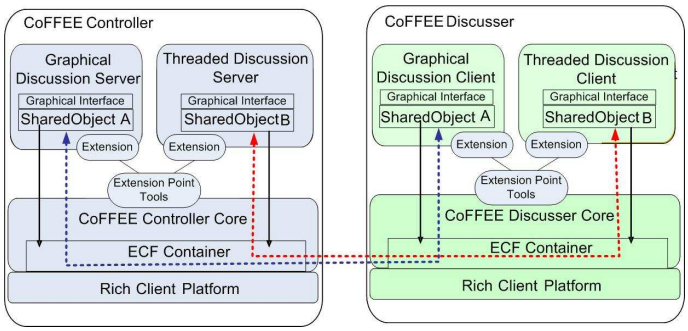


Fig. 3. CoFFEE technical architecture.

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 CoFFEE Controller and the CoFFEE Discusser is based on ECF. We use two kinds of communication objects provided by ECF: the containers and the shared objects. The containers provide access to a communication protocol while the shared objects (hosted by a container with a unique ID) manage the messaging. 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 CoFFEE Controller and of the CoFFEE Discusser, while each tool uses the shared objects (see Figure 3). In detail, each tool defines a *Service* as a pair (GUI, SharedObject) where the GUI (Graphical User Interface) provides the tool functionalities to the user while the shared object is in charge of communication. Each Service represents (an instance of) the tool functionalities and, potentially, a tool can have several independent Service instances running at the same moment. This is one of the key architectural points where the flexibility is grounded.

The state of the GUI of a Service on a CoFFEE Discusser determines the state of that Service for that student; the graphical interface can be *visible* or *invisible*, *enabled* or *disabled*:

- the service is *active* for a student if the GUI is visible AND enabled on its CoFFEE Discusser ;
- the service is *frozen* (i.e. it does not allow interactions but can be seen) for a student if the GUI is visible AND disabled on its CoFFEE Discusser; the

situation when a service is frozen include previous steps or turn-taking is going on in the classroom (managed by the teacher on the Controller);
- the service is *unavailable* for a learner if the GUI is not visible on its CoFFEE Discusser.

In the following, we describe the system architecture during the execution of a session. In Fig. 4 on page 15 we show the state of the system during the execution of the example session represented in Fig. 1.

The first step of the session defines just one group which uses two instance of the Threaded Discussion Tool and one instance of the Graphical Discussion Tool. While executing of the first step, the CoFFEE Controller activates the Threaded Discussion Tool and the Graphical Discussion Tool. Within the Threaded Discussion Tool it creates two Services (remember that a service consists of the Graphical User Interface and the Shared Object) corresponding to the two instances of the Threaded Discussion Tool required by the session. Within the Graphical Discussion Tool, the CoFFEE Controller creates one Service corresponding to the instance of the tool required by the session. Since during the execution of the first step there is just one group, all the CoFFEE Discussers have the same state of the CoFFEE Controller and all the students have the Services active (that is, with GUI visible and enabled).

The second step of the session defines two groups; each group uses two instances of the Threaded Discussion Tool and one instance of the Group Presence Tool. When the teacher choose to move on to the second step, the CoFFEE Controller freezes the previous step but does not eliminate the Services previously created: simply, it disables the Graphical User Interface of each Service of the previous step, keeping the Graphical Interfaces visible (even if disabled) and the Shared Object active. By leveraging on this characteristic, we can easily reactivate[1] any previous step by simply enabling its Graphical User Interface.

At the beginning of the second step, the Controller creates four Services of the Threaded Discussion Tool, two instances for the first group and two instances for the second group. It creates also two instances (one for each group) of the Group Presence tool. Indeed, the students of one group receive also the Services of the other group, but with the GUI not visible: in other words, all the CoFFEE Discussers have the Services of all the groups, but only the Services of the group which the learner belongs to are visible. This allows the flexibility of moving a learner from a group to another simply by making not visible, on its CoFFEE Discusser, the graphical interfaces of the "old" group and making visible the graphical interfaces of the "new" group. Furthermore, at the end of the step, it is possible to show the artifacts of each group to the others by simply making visible on all the CoFFEE Discussers the Graphical User Interface of all the groups.

The third step of the session defines two groups. The first group uses an instance of the CoWriter Tool and an instance of the Group Presence Tool; the

[1] This possibility is not implemented in CoFFEE since the pedagogical requirements of the LEAD project explicitly required to freeze previous steps.

second group uses an instance of the Graphical Discussion Tool and an instance of the Group Presence Tool.

As done before, at the beginning of the third step, the Controller freezes the second step but does not eliminate the Services previously created: simply, it disables the Graphical User Interface of each Service of the second step, by keeping the interface visible (even if disabled) and the Shared Objects active. Then, the CoFFEE Controller creates a Service of the CoWriter Tool and a Service of the Group Presence Tool for the first group, a Service of the Graphical Discussion Tool and a Service of the Group Presence Tool for the second group. The students in each group have also the Services of the tools of the other groups, but with the related GUI not visible.

The architecture we have described through the execution of the example session ensures a good level of flexibility in the managing of scripting, as we will now describe and detail in the next section.

3 Scripting in CoFFEE

Now, we describe the flexibility in scripting that is exhibited by CoFFEE architecture, just described in the previous section. The principle of the architecture is to support flexibility in the script design process at several levels according to user role, user expertise and to the timeline (before or during the script execution).

In fact, scripts in CoFFEE are managed in this order (with different objectives) by the following components:

- Template Chooser
- Lesson Planner
- Class editor (for the default group composition)
- Controller (for the flexibility at runtime)

The Template Chooser is in charge of providing access to the user to several templates of scripts, whose parameters can be instantiated and saved with a ready-to-run script. The Template Chooser offers a "high level" interface, with a wizard that asks the teacher some questions about the activities to be supported in order to identify the reference pedagogical scenario. At the end, it returns the predefined script template that is better suited to the context as described by the teacher. Then, the teacher can fill in the context-specific contents.

In particular among the parameters that can be instantiated from the template we include:

- title and subject of the session;
- for each step, it is possible to configure the title of the step and the task;
- for each group step, the number of groups;
- the notation system.

Then, the Lesson Planner offers to the experienced users (teachers/researchers) the possibility to fine-tune the script as originated by the Template Chooser or,

alternatively, to create a new script from scratch. The Lesson planner offers a more complex interface than the Template Chooser's, that allows to manage every detail of the script.

The user can define the step sequence, the tools for each step, the groups (that can have same or different configuration) and where to place each tool in the layout. For each step, it is possible to decide to show or hide a group's work to the other groups when the step is finished. Importing a tool's artifacts from previous step for the main collaborative tools (currently, Threaded Discussion and Graphical Discussion tool) is also possible. Furthermore, each tool has a number of configuration parameters to adapt its functionalities to the specific needs of the script: the Lesson planner delivers the complete control over the tools included into the script.

The group composition at runtime is facilitated by the Class Editor that allows to define a default group for each student so that groups can be pre-assembled by the teacher. The teacher can still change the groups composition at any time during the execution.

The Controller executes the script created by the Template Chooser and/or the Lesson Planner, eventually using a class file created by the Class Editor that contains the class composition and, possibly, the groups initial composition. The Controller is managed by the teacher, while the students run the Discusser. The Controller has a synchronized visualization of all the groups activities, with the same layout as in the Discusser, and can interact with them. The Controller is in charge of:

- decide when to start/end script execution and when to move to next step (steps are synchronized);
- manage turn/taking, block/unblock/disconnect specific students;
- turn on/off anonymity for tools that offer this capability;
- manage groups composition.

In Fig. 5 we show the interactions among the CoFFEE components just described for scripts design and execution.

3.1 Main components of CoFFEE script flexibility

According to [7], four main components are needed by a platform that wants to provide flexibility in scripting: a model of the script, a model of the actual interaction patterns, the script intrinsic constraints and the script extrinsic constrains (to take contingent decisions). CoFFEE architecture provides them as follows.

- *a model of the script*: in CoFFEE this is accomplished by the Template Chooser and the Lesson Planner applications, that allow an organization of the script in two levels of detail. With the Template Chooser the teacher has the choice among generalized scripts that implement different pedagogical scenarios with predefined models of interaction, and can fill in the specific values for the context, such as strings, contribution tags, number of groups.

The Lesson Planner allows "fine tuning" of the script model by defining a number of tool-specific parameters, adding/removing steps, or even creating a different script from scratch.
The component based architecture of the system allows to integrate new tools with the Lesson Planner, so that the set of tools available to define the scripts can grow. Furthermore, the component based architecture allows to achieve the flexibility on the tool layout management: each "placeholder" in the layout can be covered by any tool (even if some tool fits better than others in some position).

- *a model of the actual interaction patterns*: in CoFFEE the actual interaction is visible by the teacher at any time, as the Controller application shows every group's activity in collaborative tools. This allows the teacher to intervene when he notices some incorrect deviation from the expected behavior, or wants to stimulate the discussion in some way: this can be done directly by voice (face to face), interacting in the collaborative tools, or even with a direct communication tool that the teacher can activate with any specific student (also the students can be allowed to communicate with the teacher this way). Furthermore, successive analysis of the interaction can be done by examining the trace of the session (that reproduces the whole evolving of the interaction) and/or the printed session (that reproduce the final result, or a snapshot, of the interaction).
- *the script intrinsic constraints* i.e. the design rationale of the script, defined by the script designer, that must be respected in order to maintain the specific mechanisms that underlie the script. In CoFFEE the model of interaction (choice of the tools, organization of the steps, etc.) is defined by the script design facilities (Template Chooser / Lesson Planner), and the actual interaction (step timing, group changing at run time) is regulated by the teacher (via Controller), so the intrinsic constraints should be respected due to the constraints imposed by the software application. The management of step sequence-and step-synchronization is a consequence of the general architecture of the system: the collaborative process is driven by CoFFEE Controller on the basis of the Session previously defined. The architecture can ensure Turn Taking handling the Services graphical interfaces (see 2.4).
- *the script extrinsic constraints*: in some cases the actual evolving of the interaction may suggest some changes in strategy. In CoFFEE the teacher can decide to change the groups composition at run-time, to block/unblock the classroom or specific students, to perform turn-taking. Both client management and changing group composition are achieved by handling the visibility and the level of activity of the Services Graphical User Interfaces (see Section 2.4); among the future planned enhancements of CoFFEE, already fully supported by the current architecture, the teacher will be also able to add a step at run-time, or to move to a previous step.

In Table 1 on page 12, a comparison of the flexibility exposed by CoFFEE and other similar proposal is provided, to support the claim of flexibility of the platform we proposed.

Flexibility		CoFFEE	ArgueGraph	ConceptGrid	RSC
Script modeling	Tool for script Modeling	Yes	Fixed structure	Fixed Structure	Yes
	Tool layout management	Yes	Fixed layout	Fixed Layout	External tools can be used so layout cannot be controlled completely
	Collaborative script modeling	It can be done with a collaborative session followed by a non-collaborative actual script modeling	No	No	Yes
Actual interaction patterns	Run-time visualization of the activity	Yes	? (contains individual phases)	Yes (only for collaborative phases)	No
	Complete activity log	Yes	?	No	No
	Automatic interaction analysis	No	No	No	No
Script intrinsic constraints (managed by the application)	Step-sequence	Yes	Yes	Yes	No
	Step synchronization	Yes	Yes	Yes	No
	Turn taking	Yes	No	No	No
	Control group size	No (group composition can be managed at design- and run-time but no constraints can be put on their sizes)	Yes	Yes	No
Script extrinsic constraints (contingent decisions)	Changing group composition at run-time (during the step)	Yes	No	N.A. (may violate intrinsic constraints)	Yes
	Switching anonymity on/off	Yes	No	No	No
	Modifying script structure at run-time	Possible to add a new step at run-time (planned by Nov. 2008)	No	No	Yes
	Move to a previous step	Planned by Nov. 2008	No	No	Yes
	Latecomer accepted and included into the activities	Yes	No	No	Yes
	Client management (freeze of a client, forced disconnection)	Yes	No	No	No

Table 1. A summary of the flexibility provided by CoFFEE and compared to other results. Information about other CSCL system has been taken by their relevant bibliography and, to the best of our knowledge, is correct but may not be totally accurate.

Then, our solution is tailorable in three of the four levels of tailorability described in the Introduction: it is *customisable* for the teacher/facilitator, since it allows to tune the script selecting parameters of the tools configuration; it offers *tailorability by integration* since it allows to use predefined templates as starting points and offers freedom in assembling a script from the available tools; finally, it provides *expandability* since the scripting mechanism allows to include new tools into the scripting design, because the Lesson Planner is designed to be general, dealing with the all the tools that export some known configuration capabilities.

4 Conclusions and future work

We have presented the architecture and the features of the script engine of CoFFEE. Its flexibility is significantly enhanced by the tailorability of the architecture at different levels, since it offers a sustainable, reusable, personalizable environment for F2F CSCL applications.

The CoFFEE version currently available is version 2.0 (codenamed "*Express*") that can be downloaded at the Lead project web site `http://www.lead2learning.org` under Technologies (available in four languages and for Windows and Mac Os X). Its scripting facilities include a (non-expandable) Lesson Planner, the Class Editor and the Controller that offer flexibility in the design of the sessions (choice of tools, layout, groups, etc.), the group and class management both at design-time and at run-time, management of latecomers, and saving and reloading of sessions (to allow script to spawn several meetings).

Current work on CoFFEE will be available on the next releases and will include (about the scripting) an expandable Lesson Planner (version 3.0 by the end of May 2008), the Template Chooser (version 3.5 by the end of July 2008) and the exploitation of the run-time flexibility of the architecture by adding steps/groups at run-time, and allowing to move back to a previous step (that simply consists of activating the corresponding UIs and the Shared objects of the step) (version 4.0 by the end of the project in November 2008).

References

1. E. Aronson, N. Blaney, J. Sikes, G. Stephan, and M. Snapp. *The Jigsaw Classroom.* Beverly Hills, CA: Sage Publication., 1978.
2. M. BetBeder and P. Tchounikine. Symba: a framework to support collective activities in an educational context. In K. Lee and K. Mitchell, editors, *Proc. of the International Conference on Computers in Education*, pages 188–196. AACE, Hong-Kong, 2003.
3. R. De Chiara, A. Di Matteo, Ilaria Manno, and V. Scarano. Coffee: Cooperative face2face educational environment. In *Proceedings of the 3rd International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2007), New York, USA, November 12-15, 2007*, 2007.

4. 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.
5. 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.
6. P. Dillenbourg, M. Baker, A. Blaye, and C. O'Malley. *The evolution of research on collaborative learning*, pages 189–211. Oxford: Elsevier/ Pergamon, 1995.
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. Eclipse Communication Framework (ECF). http://www.eclipse.org/ecf/.
10. Eclipse. http://www.eclipse.org.
11. P. Jermann and P. Dillenbourg. Elaborating new arguments through a CSCL scenario . In J. Andriessen, M. Baker, and D. Suthers, editors, *Arguing to learn: Confronting Cognitions in Computer - Supported Collaborative Learning Environments)*, CSCL, pages 205–226. Kluwer, Amsterdam, The Netherlands, 2003.
12. 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.
13. Lead - technology-enhanced learning and problem-solving discussions: Networked learning environments in the classroom, 6th Framework Programme Priority IST. http://lead2learning.org/.
14. J. Lonchamp. Supporting synchronous collaborative learning: a generic, multi-dimensional model. *International Journal of Computer-Supported Collaborative Learning*, 1(2):247–276, June 2006.
15. I. Manno, F. Belgiorno, R. De Chiara, A. Di Matteo, U. Erra, D. Malandrino, G. Palmieri, D. Pirozzi, and V. Scarano. Collaborative Face2Face Educational Environment (CoFFEE). In *Proc. of First International Conference on Eclipse Tecnhologies (Eclipse-IT), Oct. 4-5 2007, Naples (Italy)*, 2007.
16. A. O'Donnell and D. Dansereau. Scripted cooperation in student dyads: A method for analyzing and enhancing academic learning and performance. In R. Hertz-Lazarowitz and N. Miller, editors, *Interaction in cooperative groups: The theoretical anatomy of group learning*, pages 120–141. London: Cambridge University Press., 1992.
17. 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.
18. R. Slagter, H. ter Hofte, and Stiemerling. Component-based groupware: An introduction. In *In Proc. of Component Based Groupware Workshpo of CSCW2000*, 2000.
19. 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.
20. G. H. ter Hofte. *Working Apart Together: Foundations for Component Groupware.* PhD thesis, Telematica Institute, Enschede, The Netherlands, 1998.

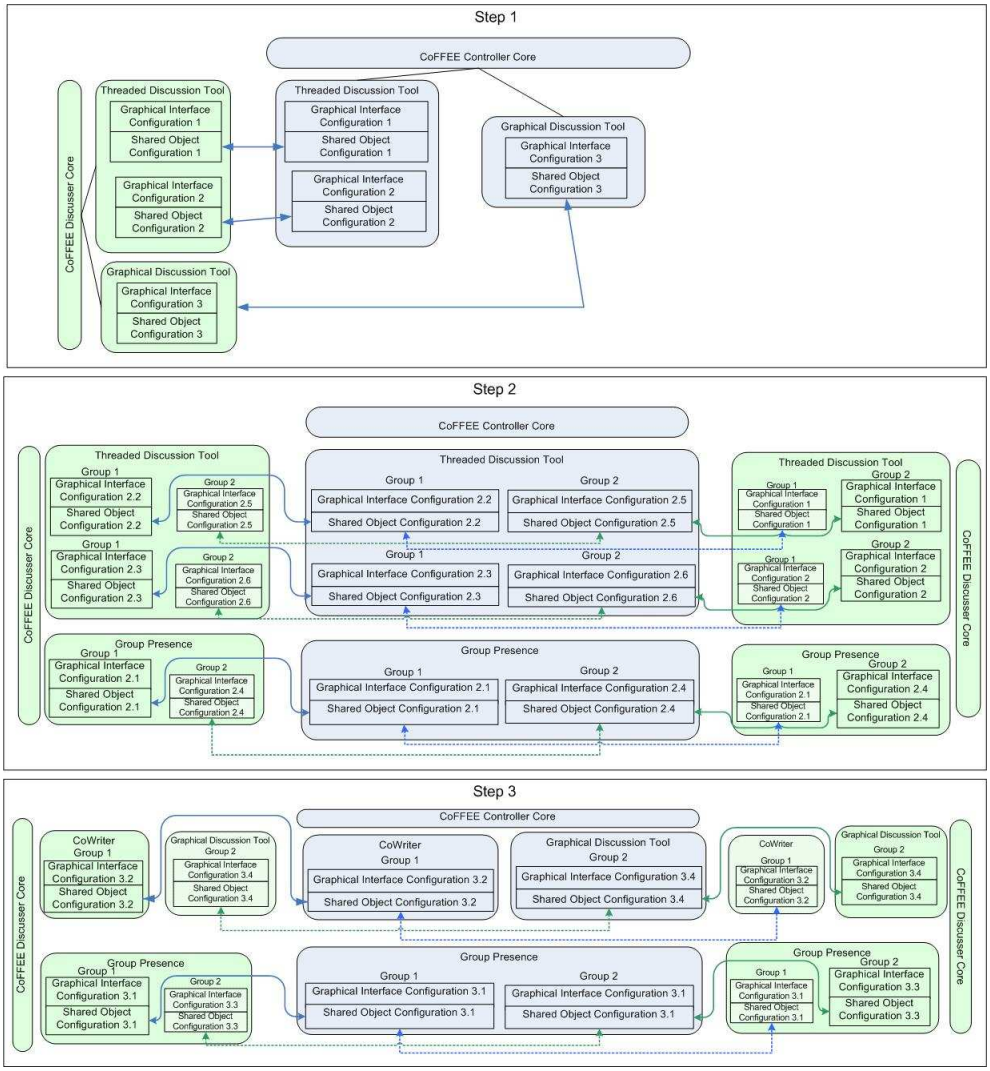


Fig. 4. Session Execution

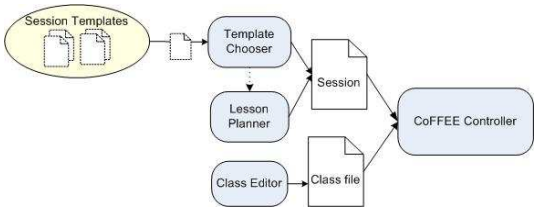


Fig. 5. The roles of each of CoFFEE components with regard to the script design and execution.