LINKING CONVERSATIONS AND TASK OBJECTS IN COMPLEX SYNCHRONOUS CSCL ENVIRONMENTS

Jacques Lonchamp
LORIA, Nancy University, Campus Scientifique, BP 239, 54506 Vandoeuvre-les-Nancy Cedex, France

Keywords: CSCL, collaborative learning, linking, referencing, sticky annotated snapshot, sticky note, intermediary object.

Abstract: In CSCL systems, linking conversations and task objects is essential for establishing joint attention and constructing shared meaning. This paper proposes an application-independent linking mechanism, well adapted for expressing complex inter-tool links as required by the most recent synchronous CSCL environments. The sticky annotated snapshot concept and its more classical derivatives, sticky note and persistent graphical pointer, are described together with their implementation. Preliminary use experiences are discussed which show, in addition to normal linking usages, a somewhat unanticipated way of using sticky annotated snapshots as fully-fledged intermediary objects during creative processes.

1 INTRODUCTION

An important requirement for effective collaborative learning is the combination of communication with shared work artefacts (Suthers and Xu, 2002). Most synchronous Computer-Supported Collaborative Learning (CSCL) environments include two distinct spaces of interaction. The task space is the place where students "do things", by creating and manipulating task objects, i.e., elements of the shared artefacts. The communication space is the place where dialogue-based interaction takes place: students mainly "talk of what they do". In this space, communication is mostly textual. As pointed by Dillenbourg (2005), from a theoretical point of view, this distinction between communication and action is shallow because the task space is clearly also a communication space (each action from a participant conveys a message to the others) and because participants manipulate verbally task concepts in the communication space. Even more confusing, in some specific cases, the task space mediates the construction of a discourse structure (Suthers et al., 1997), (Baker et al., 2003). However, from a practical point of view, these spaces are in general physically dissociated. This separation raises several practical and theoretical issues for CSCL tool designers and researchers (Dillenbourg, 2005). This paper considers the issue of linking conversations and task objects in synchronous CSCL environments. Such links are essential for establishing joint attention and constructing shared meaning (Stahl et al., 2006).

The linking problem becomes more and more difficult as CSCL tools become themselves more and more complex. Early environments provided a single tool in each space: in most cases, some shared graphical editor in the task space and a chat tool in the communication space (Baker and Lund, 1996), (Soller et al., 1999), (Constantino-Gonzáles and Suthers, 2002). Simple solutions, described in section 2, are available for linking a textual contribution to a specific element of a graphical representation and for referencing a graphical element in a textual production. The most recent synchronous CSCL environments provide several tools in each space (Avouris et al., 2004), (Pinkwart, 2003), (Lonchamp, 2006). Therefore, the linking problem encompasses both intra and inter-space linking. Complex references (i.e., with several sources and/or targets) have also a higher probability to occur. Moreover, an application-independent linking mechanism is important for accommodating a wide range of textual and graphical tools. The sticky annotated snapshot (SAS) concept proposed in this paper, together with its more classical derivatives, sticky note and persistent graphical pointer aim at satisfying these requirements.

The rest of the paper is organized as follows. Section 2 introduces the linking problem and shows

Lonchamp J. (2007).
LINKING CONVERSATIONS AND TASK OBJECTS IN COMPLEX SYNCHRONOUS CSCL ENVIRONMENTS.
In *Proceedings of the Third International Conference on Web Information Systems and Technologies - Society, e-Business and e-Government / e-Learning*, pages 281-288
DOI: 10.5220/0001270202810288

why existing solutions are insufficient for complex synchronous CSCL systems. Section 3 defines the principles of the generic solution proposed for solving the linking problem and describes its implementation. Section 4 discusses preliminary use experiences and emphasizes a somewhat unanticipated way of using SAS as fully-fledged intermediary objects during creative processes.

2 THE LINKING PROBLEM

Links can be categorized into three classes.
(1) 1 to 1 links, relating a communication act to a task object or the opposite, two communication acts (in the same or in different tools), or two task objects (in the same or in different tools).
(2) 1 to many links, relating a communication act to a set of task objects or communication acts (in the same or in different tools), a task object to a set of communication acts or task objects (in the same or in different tools).
(3) Many-to-many links, relating a set of elements (communications acts and/or task objects) to another set of elements (in the same or in different tools).

In class (1), several techniques are available for linking a communication act to a task object:
– deictic references, using spatially indexical references (e.g. "the blue object on the right") or temporally indexical references (e.g., "your last object"),
– deictic references using non-persistent mechanisms supporting gestural deixis, such as telepointers (Hayne et al., 1994), ephemeral graphical gesturing (e.g. pointing by drawing an arrow which fades away after a short period of time) (Dongqiu and Gross, 1999), highlighting of objects when the cursor pass over them or when they are clicked (Suther et al., 2003),
– deictic references using persistent mechanisms, such as persistent graphical pointers with unique identifiers that can be pasted near task objects, and the explicit graphical linking mechanism provided by Concert-chat (Mühlpfordt and Wessner, 2005). This last solution allows direct access from a communication act to the appropriate part of the artefact and direct access from some artefact area to the concerned discourse contributions, while both parts remain separated on the screen.

Linking a task object to a communication act is mostly performed with communication acts embedded (anchored) into the task space, such as "sticky notes" (Fidas et al., 2001). The message is put next to the object, making the deictic reference of this particular text evident.

Links between communication acts are frequent within chat tools. Textual referencing by repetition (quoting elements of an earlier posting), textual referencing by position ("the third line of the second paragraph"), and explicit referencing of the author name are the most frequently used techniques (Nash, 2005). Communication act numbering is another possible solution for links within a single chat tool. The explicit graphical linking mechanism of Concert-chat can also be used between communication acts.

Linking between task objects is possible through graphical annotating within a single artefact: objects can be linked with arrows, underscoring, boxing, and circling (Giordano and Mineo, 2005). Embedded communication acts can also reference one (or several) other task object(s) by using deictic or textual referencing.

There are fewer mechanisms for one to many links of class (2). The graphical linking mechanism of Concert-chat has been extended for multiple designations of task objects or communicating acts. Anchored annotations can take the form of anchored discussion threads (Trahasch and Lauer, 2005) which allow linking a task object to a set of communicative acts.

Finally, many to many links of class (3), can only be specified by complicated textual descriptions or by graphical annotations when they are associated to a single artefact.

As most CSCL environments provide a textual communication space, textual referencing is always possible. But some techniques that are common in spoken discourse have a greater potential of leading to ambiguity and misinterpretations in synchronous textual communication (Nash, 2005). It is the reason why most CSCL systems provide additional specialized linking mechanisms. In the context of complex CSCL environments, what is needed *is a simple application-independent mechanism which provides most of the previous possibilities and can deal with the most complex situations, i.e., many-to-many inter-tool links*.

3 THE SAS CONCEPT

3.1 Requirements

A participant should be able to link any source element (or complex set of elements) to any target element (or complex set of elements) appearing in

any textual or graphical tool of the environment (both in the task space and in the communication space). The mechanism should be persistent for supporting latecomers (and inattentive participant) in their comprehension of the missed part of the session and for allowing ex-post analysis after the session has ended. The mechanism should allow some kind of threaded communication rooted into a link, for supporting argumentative discourses.

3.2 Design Choices

In the more complex case of many to many links between elements displayed in different tools, graphically annotating a snapshot of the environment is the only solution that is fully generic: participants can use arrows, boxes, circles, braces, underlinings, or any other graphical notation, complemented with textual notes. This is the basic idea behind the sticky annotated snapshot (SAS) concept.

A SAS is anchored somewhere by right-clicking with the mouse at any visible position of any tool of the environment. A SAS editor is then launched which allows drawing on a re-centred snapshot of the whole environment. A small camera icon playing the anchor role is inserted at the mouse location and labelled with the name of the author followed by a sequentially increasing integer (e.g., suzan3). By double-clicking the icon any learner can see the annotated image in a read-only SAS viewer. Snapshot creation is also possible in the viewer, for commenting the original annotated snapshot, leading to a form of threaded discussion. Embedded snapshots have a composed name reflecting their inclusion path (e.g., suzan3-peter2). When stored, a SAS cannot be changed. The author (and actors playing the moderator role) can hide its icon, the snapshot remaining stored for further analysis and session replaying.

In a display, the most recent SAS has a specific colour (red) which distinguishes it from previous (yellow) instances. As this last snapshot can be included into other ones, SAS including the most recent snapshot have an intermediary colour (orange).

Two derivates of the sticky snapshot concept are also proposed for simpler cases.
(1) A sticky note (embedded textual annotation) is managed as a SAS having a text-only content. The graphical editor and the graphical viewer are replaced by a textual editor and a textual viewer. An envelope icon is associated to sticky notes. Long textual annotations are difficult to read when they are drawn on a snapshot. Sticky notes constitute a better alternative for textually commenting or reacting to an annotated snapshot. They are also useful in the classical situation of a textual communication act linked to a single task object.
(2) A persistent graphical pointer is managed as a SAS with no content. A plain arrow icon is associated to persistent graphical pointers.

Figure 1 shows the SAS editor during the creation of a simple annotated snapshot (launched through a contextual menu available in all panels where an annotation can be created). In this example, Suzan, Jack and Mary use Omega+ environment (Lonchamp, 2006) customised for an object-oriented design course. The task space provides a read-only textboard and a shared graphical UML class diagram editor. The communication space provides a regular chat. The three participants are collaboratively transforming the wording of a situation into a UML class schema. Figure 1 shows Suzan's reaction to the last initiative from Jack. She creates a SAS linking a relationship in the proposed schema with a word in the problem definition justifying why she disagrees (1 to 1 link between the diagrammer and the textboard). When saved, her SAS named Suzan1, is anchored in the class schema. The coloured (red) button panel on top of the SAS editor helps for distinguishing the editor from the actual CSCL environment and provides a set of whiteboard-like functionalities. As graphical annotations are drawn on a transparent overlay, it is possible to erase drawings and texts in the SAS editor without altering the underlying snapshot.

Figure 2 shows the SAS Viewer launched by Jack by double-clicking on suzan1 icon (on top of Jack's client). For avoiding confusion, editors and viewers have a differently coloured button panel.

Figure 3 shows, in a SAS viewer launched by Suzan, a threaded conversation example. First jack has answered Suzan's criticism with a SAS named suzan1-jack1. In this SAS, Jack has sketched a solution with a freehand drawing and an additional comment. Mary has reacted with a sticky note, named suzan1-jack1-mary1, in which she agrees with Jack's solution.

This sticky note is shown in the note viewer. In all viewers, the 'Close all' button helps for closing with a single action a set of stacked viewers. In SAS viewers, snapshots have a slightly degraded rendering. So it is easy to distinguish visually between icons that are part of the snapshot (e.g., suzan1 and suzan1-jack1 in Fig. 3) and bright ones that can be clicked (e.g., suzan1-jack1-mary1).

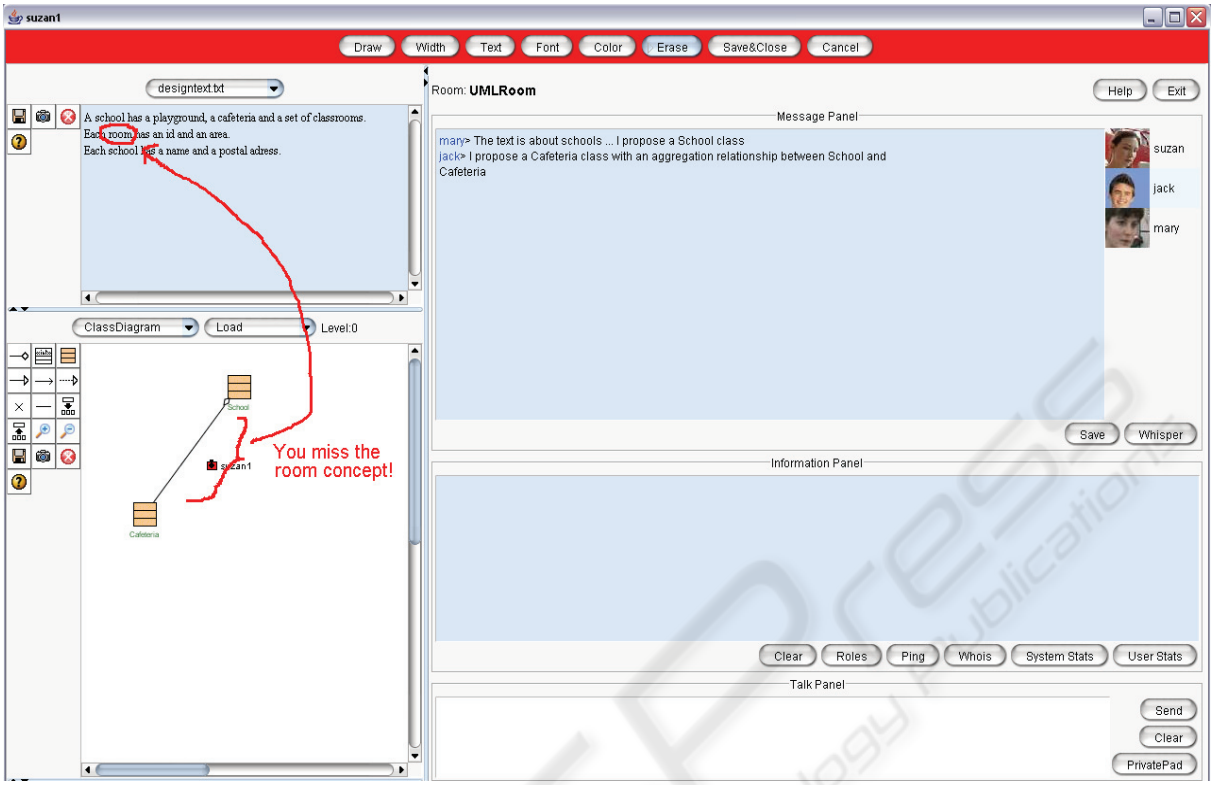

Figure 1: The SAS editor.

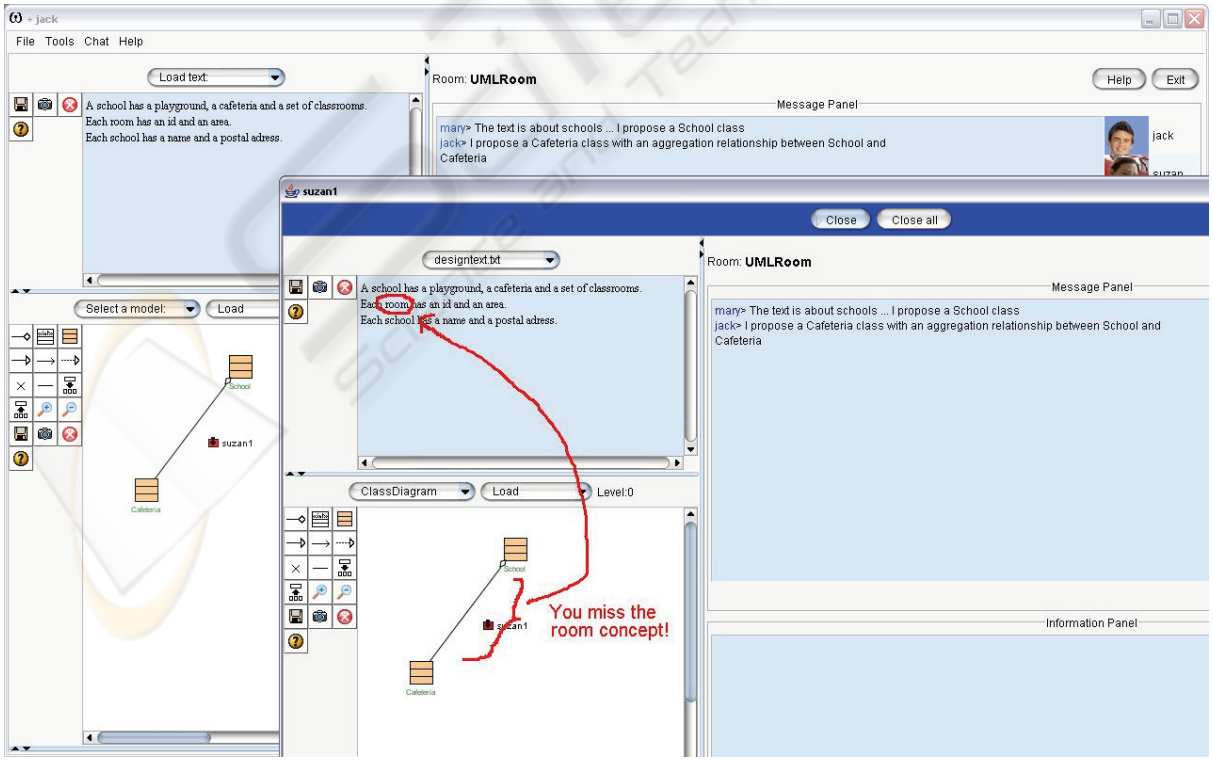

Figure 2: The SAS viewer.

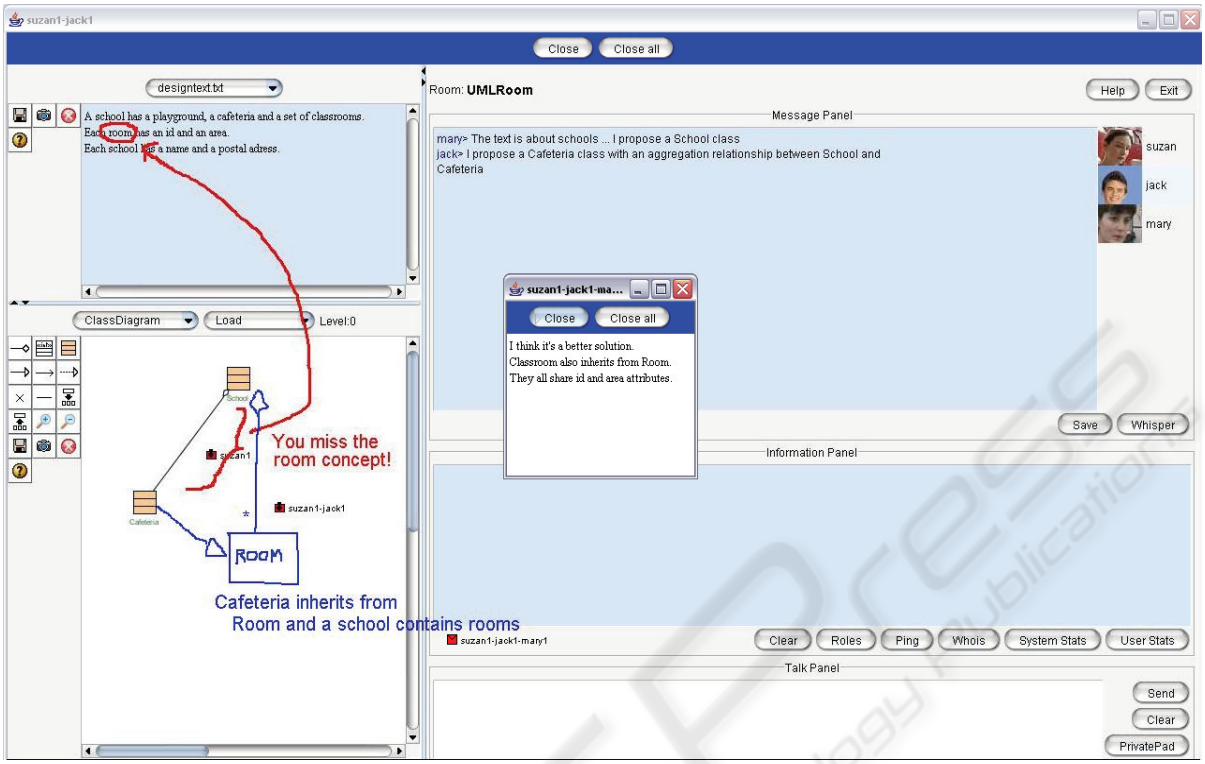

Figure 3: A threaded conversation.

3.3 Implementation

The SAS concept and its derivatives are implemented into Omega+ synchronous CSCL environment (Lonchamp, 2006). Omega+ is a generic framework which applies "model-based genericity" to the four dimensions of collaborative learning: the situation, the interaction, the process, and the way of monitoring individual and group performance. These four aspects are explicitly specified in a set of models (process, protocol, artefact, and effect model) that serve as parameters for the generic framework which is designed to model systems that are flexible and can be tailored to a wide range of users, communities, goals and contexts. Omega+ client looks like a classical dual space CSCL system (Dillenbourg, 2005), as shown in Fig.2, with a communication space on the right part and a task space on the left part. The chat (in the upper part of the communication space) is either a regular chat or a protocol model-driven chat. An information panel (in the middle part of the communication space) allows displaying textual or graphical information, in particular model-generated individual and grouping performance measures. The task space may contain up to three tools as requested by the executing process model definition. Tools are either predefined editors (shared text editor, shared whiteboard) or shared graphical editors customised by artefact models. Omega+ is implemented in Java/Swing.

On the client side, SAS descriptions (name, type, location, visibility …) are stored into a class named RootPanelWithSAS which inherits from JPanel and serves as content pane for the Omega+ client. The display area of each tool where annotated snapshots can be created (class X) is transformed into a class XWithSAS which inherits from class X and just redefines the painting method. This painting method calls a method of RootPanelWithSAS which repaints all the currently visible SAS. The mouse listener associated to X has to be changed (or created). In case of a right button click, a method which displays the contextual menu is called. For a SAS creation, a graphical editor (instance of an internal class SASEditor within RootPanelWithSAS) is created and the resulting annotated snapshot is transmitted to the server where it is stored. The creation notification (i.e., the snapshot description) is broadcasted by the server to all the connected clients. When a SAS is hidden a notification is also sent to the server and broadcasted to the clients. A

visibility boolean attribute in the SAS description indicates if the SAS is visible or not. In case of a double click with the left button, the designated SAS is found, its image is retrieved from the server and displayed into a viewer (instance of an internal class SASViewer within RootPanelWithSAS). The method which receives from the server notifications of events occurring on other clients is also located in RootPanelWithSAS: it can add locally a new SAS description or register that an existing SAS is hidden. The annotated snapshot is transformed into a string (Base64 coding) in the RootPanelWithSAS class, so transmission to the server, storage in the server, transmission to connected clients, and latecomers' management can reuse the existing mechanisms provided by the chat server. By this way, any existing java CSCL system including a chat component can be easily "SAS-enabled": all complex methods are located into the RootPanelWithSAS class that can serve as content pane for its client window.

4 FIRST USE EXPERIENCES

Our first use experiences show that the sticky annotated snapshot concept is used as expected in situations where complex messages with multiple references are needed. Two typical examples are described in section 4.1. A somewhat surprising and interesting usage has appeared during creative processes with SAS playing the role of fully-fledged intermediary objects. Two examples are discussed in section 4.2.

4.1 Complex Referencing

A first example of complex referencing is a SAS created by a participant during a textual debate and anchored into her own chat posting. The chat production summarizes a set of ideas previously proposed by several other participants which are circled into the attached snapshot. This example of 1 to many link among communication acts well illustrates the frequently observed distribution of a message between a SAS and a companion chat message, taking advantage of both textual and graphical expressiveness.

The second example is taken during a pyramid collaborative learning process (Asensio et al., 2004) for designing UML class schemas. A group of four students compares two schemas they have previously constructed independently as two dyads. During this phase of the process, Omega+ is configured with three class diagram editors: the first two for displaying dyads' solutions (in read-only mode) and the third one for constructing the reunified group solution. As anticipated, similarities/differences descriptions are mainly expressed with annotated snapshots. The more complex ones are embedded into chat contributions and use the explicit spatial referencing capabilities of SAS (with lines explicitly relating elements). In simpler cases, the implicit spatial referencing capability is used: the SAS is anchored into one of the graphs, its location implicitly defining the source element; only the target element is explicitly designated (pointed, underlined, circled) in the snapshot. Similarly, simple comments and positive or negative evaluations are generally expressed thanks to sticky notes anchored within the graphs or within snapshots.

4.2 Intermediary Objects

Intermediary objects are defined as ephemeral and shared representations appearing during collaborative design processes which serve as mediators to discussion and reflect some transformation or translation of the designed artefact (Vinck and Jeantet, 1995) (Boujut and Blanco, 2003).

During the collaborative production of an UML class schema by a dyad using Omega+, one participant has spontaneously produced in a snapshot an analysis of the possible candidate classes by underlying the more relevant terms in the problem wording appearing in the text editor (resized for occupying most of the screen). This analysis has raised a discussion with the other student through chat postings and a child SAS where two candidate terms were crossed out with a short explanation in a companion sticky note. The first snapshot has all the attributes of an intermediary object: ephemeral and spontaneous, shared, serving as mediator for a discussion and involving a transformation from a state of the product into a subsequent state (Boujut and Blanco, 2003).

During the next phase of the same process, when dyads contrast their solutions, we observed an attempt to build a similar kind of intermediary object with an annotated snapshot relating with lines similar elements in the two proposals, displayed side by side in two editors, and circling some elements in both representations because they did not have counterparts in the other schema. Comments were given in the chat posting where the SAS was anchored.

This practice can be better understood when analysed in relation with the coordination approach used during synchronous sessions for disallowing anarchic interaction. In synchronous CSCL environments floor control (FC) is important mainly for user-oriented reasons: facilitating turn-taking, mutual focus of attention, and maximum synergy among users (Boyd, 1996). Omega+ provides a set of global FC policies at the environment level, specifying who can talk through the communication space and who can act through the task space (Lonchamp 2006a).

At one extreme, every participant can chat and act freely (global free-floor policy). Confusion can be reduced if students accept to "think aloud" and "draw aloud" by commenting their intents, ideas and actions. In this setting, sessions include generally a large number of intertwined contributions with a high production rate, in the usual chat-style. Creating an intermediary object with a SAS is a way to work in isolation for a moment, as the annotated snapshot is only shared when it is saved. This spontaneous individual production step allows conducting some personal and more structured reasoning. This corresponds to the idea of "near synchronous working" (Mann and Garner, 2005) and of a "personal reflective conversation space", where users can externalise, reflect, edit and develop their own thinking prior to communicating conjecture to the group (Schön, 1992).

At the other extreme, exclusive control policies can be applied to the whole environment or to the task space. By this way, only operations from the current floor owner are allowed to cause changes to the shared artefacts under construction. In this setting, sessions are much slower. Participants have often to wait and contributions are less spontaneous. Creating an intermediary object with a SAS is a way to bypass the lock and to contribute immediately to the shared artefact, as annotated snapshots are not controlled by the floor control mechanism. This spontaneous parallel production step allows conducting some immediate and "forking" contribution with regard to the main controlled stream of work.

In both cases, annotated snapshots used as intermediary objects provide some interesting flexibility with regard to the coordination policy.

5 CONCLUSION

The concept of sticky annotated snapshot is presented in the first part of this paper as an application-independent linking mechanism, well adapted for expressing complex inter-tool links as required by the most recent synchronous CSCL environments. At a first glance this proposal may appear as a quite straightforward extension of well-known mechanisms such as sticky notes and anchored communication threads.

However, preliminary use experiences have shown an unanticipated impact of sticky annotated snapshots on the core of the collaboration process itself when they play the role of fully-fledged intermediary objects during creative activities. These intermediary objects are produced during unplanned individual production phases which provide interesting forms of flexibility with regard to the coordination constraints that apply during the synchronous session.

REFERENCES

Asensio, J., Dimitriadis, Y., Heredia, M., Martinez, A., Alvarez, F., Blasco,M., Osuna, C., 2004. Collaborative Learning Patterns: Assisting the Development of Component-Based CSCL Applications. In *PDP'04, 12th Conference on Parallel, Distributed and Network-Based Processing*, IEEE Computer Society Press, 218-224.

Avouris, N., Komis, V., Margaritis, M., Fidas, C., 2004. Modelling Space: A tool for synchronous collaborative problem solving. In *ED-MEDIA' 04*, *16th World Conference on Educational Multimedia & Telecommunication*s, AACE Press, 381-386.

Baker, M., Lund, K., 1996. Flexibly structuring the interaction in a CSCL environment. In *EuroAIED'96*, *European Conf. on Artificial Intelligence and Education*, Colibri Ed., 401-407.

Baker, M. , Quignard, M., Lund, K., Séjourné, A., 2003. Computer-supported collaborative learning in the space of debate. In *CSCL'03*, *5th International Conference on Computer-Supported Collaborative Learning,* Kluwer Academic Publishers*,* 11-20.

Boujut, J.F., Blanco, E., 2003. Intermediary Objects as a Means to Foster Co-operation in Engineering Design., *Computer Supported Cooperative Work*, 12, 205–219.

Boyd Jr., J.A., 1996. Floor Control in Synchronous Groupware, *PHD Thesis, The Ohio State University*.

Constantino-González, M., Suthers, D., 2002. Coaching Collaboration in a Computer-Mediated Learning Environment. In *CSCL'02*, *4th International Conference on Computer-Supported Collaborative Learning*, Lawrence Erlbaum Associates., 583-586.

Dillenbourg, P. & CSCL SIG of Kaleidoscope, 2005. Dual Interaction Spaces. In *Proc. CSCL'05 workshop presentation*. http://www.cscl2005.org/Workshops/workshop5.htm.

Dongqiu, Q., Gross, M.D., 1999. Collaborative Design with NetDraw. In *CAAD Futures'99*, *8th International Conference on Computer Aided Architectural Design Futures*, Kluwer Academic Publishers, 213-226.

Fidas, C., Komis, V., Avouris, N., 2001. Design of collaboration-support tools for group problem solving. In *PC HCI'01,Panhellenic Conference on Human Computer Interaction*, Typorama Publications, 263-268.

Giordano, D., Mineo, S., 2005. A graphical annotation platform for Web-based e-learning. In *m-ICTE'05, 3rd International Conference on Multimedia and Information and Communication Technologies in Education*, Formatex Research Center, 1255-1260.

Hayne, S., Pendergast, M., Greenberg, S., 1994. Implementing Gesturing with Cursors in Group Support Systems. *Journal of Management Information Systems,* 10, 42–61.

Lonchamp, J., 2006. Supporting synchronous collaborative learning: a generic, multi-dimensional model. *International Journal of CSCL*, 1(2), 247-276.

Lonchamp, J., 2006a. Floor control in complex synchronous CSCL environments, *WEBIST 2007*.

Mann, P., Garner, S., 2005. The Role of Sketches in Supporting Near-Synchronous Remote Communication in Computer Supported Collaborative Design. In *CSCWD'05, 9th International Conference on Computer Supported Cooperative Work in Design*, Springer Verlag 72-81.

Mühlpfordt, M., Wessner, M., 2005. Explicit Referencing in Chat Supports Collaborative Learning. In *CSCL'05*, *6th International Conference on Computer-Supported Collaborative Learning,* Lawrence Erlbaum Associates, 662-671.

Nash, C.M., 2005. Cohesion and Reference in English Chatroom Discourse. In *HICSS'05*, *38th Annual Hawaii International Conference on System Sciences - Track 4, Volume 04*, IEEE Computer Society Press, 108.3.

Pinkwart, N., 2003. A Plug-In Architecture for Graph Based Collaborative Modelling Systems. In *AIED'03*, *11th International Conference on Artificial Intelligence in Education*, IOS Press, 535-536.

Schön, D.A., 1992. Designing as reflective conversation with the materials of a design situation. *Knowledge-Based Systems*, 5, 3-14.

Soller, A., Linton, F., Goodman, B., Lesgold, A., 1999. Toward Intelligent Analysis and Support of Collaborative Learning Interaction. In *AIED'99*, *9th International Conference on Artificial Intelligence in Education*, IOS Press,75-82.

Stahl, G., Zemel, A., Sarmiento, J., Cakir, M., Weimar, S., Wessner, M., Mühlpfordt, M., 2006. Shared Referencing of Mathematical Objects in Online Chat. In *ICLS'06, 7th International Conference of the Learning Sciences*, Lawrence Erlbaum Associates, 716-723.

Suthers, D., Girardeau, L., Hundhausen, C., 2003. Deictic Roles of External Representations in Face-to-Face and Online Collaboration. In *CSCL'03, 5th International Conference on Computer-Supported Collaborative Learning,* Kluwer Academic Publishers, 173-182.

Suthers, D., Toth, E., Weiner, A., 1997. An integrated approach to implementing collaborative inquiry in the classroom. In.*CSCL'97*, *Second International Conference on Computer-Supported Collaborative Learning,* Lawrence Erlbaum Associates, 272-279.

Suthers, D. Xu, J., 2002. Kukakuka: An Online Environment for Artifact-Centered Discourse. In WWW'02, *11th WWW Conference*, ACM Press, 472-480.

Trahasch, S., Lauer, T., 2005. Scripted anchored discussions of e-lectures. In *E-learn'05*, *10th World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education,* AACE Press, 2436-2443.

Vinck, D., Jeantet, A., 1995. Mediating and Commissioning Objects in the Socio technical Process of Product Design: a conceptual approach. *In D. Maclean, P. Saviotti and D. Vinck (eds.) Designs, Networks and Strategies*, COST Social Science Series, 2.