T. Koschmann, D. Suthers, & T.W. Chan (Eds.), *Computer Supported Collaborative Learning 2005: The Next 10 Years!*
Mahwah, NJ: Lawrence Erlbaum Associates.

A Structured Chat Framework for Distributed Educational Settings

Jacques Lonchamp
LORIA, BP 254, 54500 Vandoeuvre-lès-Nancy, France
jloncham@loria.fr

Abstract. Chat tools are an integral part of many collaborative e-learning environments. However, standard chat tools suffer from important coordination and coherence deficiencies. The solution proposed in this paper is a generic framework for building malleable structured chat applications. The paper discusses the approach and the main design decisions, emphasizing the importance of dynamic malleability in educational settings.

Keywords: chat, structured chat, computer-mediated communication, process-centric system, generic framework, malleability, on line education, cooperative learning

INTRODUCTION

Chat tools are an integral part of many collaborative e-learning environments. Chat tools can support non collocated instructors and students in several domains. First, they can help to overcome the lack of corridor or water cooler talk, with unplanned informal discussions. Second, instructors or fellow students can schedule a chat session on a certain day, for a more focused but still informal discussion about some project or exam. Third, chat tool can support formal and planned learning sessions with a teacher and a small set of students, for instance under the banner of some constructivist learning practice. Young people in their late teens and early twenties have a great familiarity with chat tools. Those who would have significant difficulty writing a few sentences in a traditional course setting are able to keep a sustained attention and level of energy during chat sessions. However, in all these contexts, standard chat tools suffer from important deficiencies. In the second section we discuss these problems and define our vision for solving them by improving chat technology for educational settings. The third section gives more details about the main design decisions resulting from that vision. The fourth section briefly describes our current prototype. Finally, we discuss and evaluate the approach.

CHAT TOOL DEFICIENCIES

Informal Interaction Support

Many research efforts rooted in the sociological study of conversation have identified important coordination and coherence issues in standard chat tools (Garcia, 1999), (O'Neil, 2003). The most important is the lack of control over turn positioning. Since turns can be sent simultaneously by a number of participants, there is no guarantee that a next-turn, for example a response to a question, will appear directly after the question. Instead other turns may appear between the question and the response, causing confusion over threads. The consequence is a preference for short turns so that the response might be closer to the question, if sent quickly. Standard chats are not places where carefully constructed messages can be sent. Lack of visibility of turns-in-progress, because chat systems only transmit turns when they are completed (ENTER key), and lack of visibility of listening-in progress, because participants do not receive moment-by-moment information about the reaction of those who are listening to them, are other examples of well known coordination issues.

A number of research prototypes aim at addressing these problems with non standard interfaces like threaded interfaces (Smith, 2000), 2D/3D graphical interfaces (Viegas, 1999), (Kurlander, 1996), streaming media interfaces (Vronay, 1999). Innovative interfaces can solve one specific problem but often raise new ones in other domains (Vronay, 1999), and the change is often too radical for many end users. Another approach extends traditional chat tools with additional awareness mechanisms, each responding to a specific need. Researchers have proposed many punctual iconic or textual cues such as the social proxy of Babble for representing graphically user activity (Bradner, 1999), turns-in-progress visualization through the textual 'someone is typing' indicator, or social presence through animated face icons representing facial expression, hand raising, etc. (Fadel, 2004). In our opinion, such awareness mechanisms should be selectively available within consistent interaction styles for avoiding an excessive level of cognitive load. For instance, in a round

robin interaction, user activity, turns-in-progress, and hand raising cues are of little value. The last approach considers that most of the deficiencies are consequences of the unstructured nature of standard chat conversations. By constraining the turn talking, a fundamental aspect of all virtual learning communities (Reyes, 2004), and by dividing discussions into more focused sub discussions, most of coherence and coordination problems will be alleviated. We think that educational settings strongly require such structuring capabilities.

Formal Interaction Support

In structured chat tools the rules governing the interaction (its process in a broad sense) are mechanically enforced for improving coordination and coherence. Several field studies have demonstrated that structured chat tools can help to support different kinds of formal interactions (Farnham 2000), (Pfister, 2002). This approach embodies into technology "social scripting" (Farnham, 2000), which is commonplace in face to face group interaction such as explicit meeting agendas, more or less implicit scripts for conducting interviews, brainstorming, and most of formal collaborative learning activities. In some prototypes, the rules are hard-coded (Pimentel, 2004). The imposition of inflexible structures is often resisted by participants, when poorly designed or missing the situatedness of human work (Suchman, 1987). People need to feel in control of a system according to their roles. So it is important to provide different forms of malleability to end users. We will discuss in more details our vision of malleability in the next section. The prerequisite is to have soft-coded rules, instead of hard-coded rules, by means of some interaction modeling language. In our opinion, most of the recent chat prototypes which follow this orientation suffer from important deficiencies: lack of expressive power of the modeling language in Lead Line (Farnham 2000), in which a process is simply a linear sequence of regular chat sessions, excessive complexity of the general purpose modeling language (colored Petri nets) in ProChat (Whitehead, 2000), lack of generality of the approach in the Learning Protocols approach (Pfister, 2002).

Our aim is to provide an open source framework for building malleable structured chat applications for distributed educational settings. It is based, among other sources of inspiration, on the most valuable lessons learned from other research areas dealing with flexible process-centric systems, such as process-sensitive software engineering environments (Finkelstein, 1994), workflow management systems (Agostini, 1997), and process-enabled cooperative hypermedia systems (Wang, 2000). Our prototype is called ωChat. Its applet client can be integrated into every Web-based collaborative e-learning environment.

ωCHAT APPROACH

Malleability

Malleability encompasses four different aspects.

- *Model evolution*. A process model is composed of three variations: a template definition (set of types) expressed in a model specification language, one or several enactable instances with the contextual information which makes them possible to execute, and one or several enacting instances created from the enactable ones with their execution states. (Finkelstein, 1994) distinguishes three styles of evolution: delayed change, when the template is modified but only future instances will be impacted, busy change, when the template is modified and the change is immediately propagated to all existing instances, and local change, when a single instance is modified with no impact on the template definition. In ωChat we are interested in delayed and local change. Unlike long term business process, there is no need for simultaneous evolution of all existing instances of a given template, because they are basically independent of one another. Delayed and dynamic local changes should be easy to perform by end users.
- *Model emergence*. Sometimes, the template definition itself can only emerge opportunistically and dynamically during the interaction, which includes a kind of meta discussion about how to proceed (Wang, 2000). No cryptic notation should be necessary for defining and instantiating such a new template.
- *Punctual constraint relaxation.* Any user should be able to relax or sidestep any specific constraint (without model evolution) when exceptional circumstances arise, the system making other users aware of these punctual rule breakings.
- *Customizable information and guidance.* The user interface should reflect in a natural and customizable way the current set of constraints which applies to a specific user playing a given role, and non intrusively provides an adapted guidance.

A Two level architecture for malleability

A central idea of our approach is to distinguish between a macro level (or process level), and a micro level (or protocol level), with different malleability properties. At the macro level, the process model specifies a sequence of phase types. Each phase type is characterized by a name, an informal description, and an interaction protocol

type: open-floor, moderated open-floor, circular floor passing, single contribution, unique contributor (all predefined), and application-specific protocols (defined at the micro level). A library of predefined process model templates is available for reuse at room definition time. These definitions are stored in a declarative form (XML files on the server side), making delayed change easy to perform. When an enactable phase instance is created from the template, the user gives a name (by default the type name with an instance number), who is participating (if the phase has restricted participation), the binding of users to protocol-specific roles (e.g., who is the moderator in a moderated phase), some informal instructions for end users, and a set of optional mechanisms for customizing all client interfaces (use of utterance type labels, use of explicit referencing through the sequence number of the referred utterance in the chat history, …). The four malleability aspects of the previous section are fully supported at this macro level through simple interactive manipulations : the process model can emerge (e.g., a standard chat room is transformed on the fly into a structured, model-driven, chat room), the model can evolve (e.g., a new phase type is created, changed, or suppressed), and all constraints can be relaxed by users playing the predefined Room Operator role (e.g., the sequencing rule is relaxed by jumping to any previous or subsequent phase type, the participation rule is relaxed by kicking off temporarily a participant, the circular floor passing rule is relaxed by skipping a user, and so on). It is worth noting that the evolution power is into the hands of all people playing the predefined Room Operator role (which can be transmitted), not necessarily into the hands of a single heavy-handed dictator.

The micro level specifies interaction protocol types. Such a definition may require complex rules specification that cannot be performed interactively by an average end user. At this level, dynamic malleability is not fully supported. New protocols are only specified off-line with a declarative XML-based protocol specification language. An interaction protocol is defined by a protocol name, a non empty set of protocol-specific role names, a non empty set of protocol-specific utterance type names, and a non empty set of transition rules: <utterance_type> <role_type_1> → [<utterance_type_list> | all] [next | next_ circular | any] <role_type_2>, where [a | b] denotes a choice, and the keyword 'all' replaces all the values of a given set. It means that, depending on the type of the previous utterance and the role of the contributor, the next possible speaker (or role) is known with the set of possible utterance types. The table below specifies with this language one example of application-specific type (Pfister, 2002).

Protocol	*Role types*	*Utterance types*	*Transition rules*
Explanation protocol (Pfister, 2002)	Tutor Learner	Question Explanation Comment	Question Learner → Explanation any Tutor
			Explanation Tutor → all next-circular Learner
			Explanation Learner → all next_circular Learner
			Comment Learner → all next_circular Learner

THE CURRENT PROTOTYPE

The default client for interacting within an unstructured room or an open-floor phase of a structured room looks like any standard chat. This is important for people who are happy with such a basic tool. This default client can be customized both globally, by specifying interaction features at phase instantiation, and individually, through the Options Menu. For instance, displaying an 'Info Panel' for unsolicited awareness messages (such as 'mary joined the room' - see Figure 2) and query results. The 'Can you talk?' (see Figure 1) and 'Are you op (Room Operator)?' visual indicators are examples of individual customizations.

During structured phases, clients reflect predefined and protocol-specific roles. For instance, users playing the Room Operator role have 'Next' and 'Jump' buttons for instantiating and starting the next phase or any other phase of the structured room. In a moderated open floor phase, the Moderator's client is the only one where new messages are immediately displayed (with the [MODERATE] label - see Figure 2). A publish window allows the Moderator to choose either to accept (broadcast) new messages or to refuse them (triggering a refusal message in a private chat session with their authors). During a phase using an application specific protocol such as the Explanation protocol specified above, users can only choose between a list of protocol-permitted utterance types when they have the floor (see Figure 1). All messages have a type label (e.g. {Explanation} - see Figure 1). This kind of controlled discourse should probably be restricted to specific and short time phases for avoiding the straight jacket effect pointed out in coordinator tools in particular (Winograd, 1986).

The following scenario illustrates different aspects of dynamic malleability. Jack (a teacher) is a Room Operator during a disorderly open-floor phase. At some point, Jack decides to kick off Peter (a student) for one minute (message 12 in Figure 2). Besides the use of this punctual mechanism, Jack decides also to change the current open floor room into a moderated one (himself being the Moderator) for a better control of the contributions. All clients instantly reflect that change: message 13 was un-moderated while message 16 is now moderated by Jack. Finally, Jack decides to add a 'Summarization Phase' after the current phase with a circular

floor passing protocol: this local model evolution is performed interactively (see Figure 2). Instantiation will be done later, when the 'Summarization Phase' instance will start ('Next' or 'Jump' buttons).

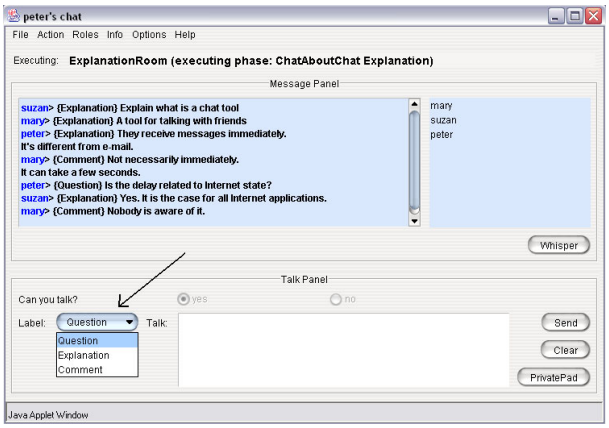

Figure 1. The Explanation protocol at work

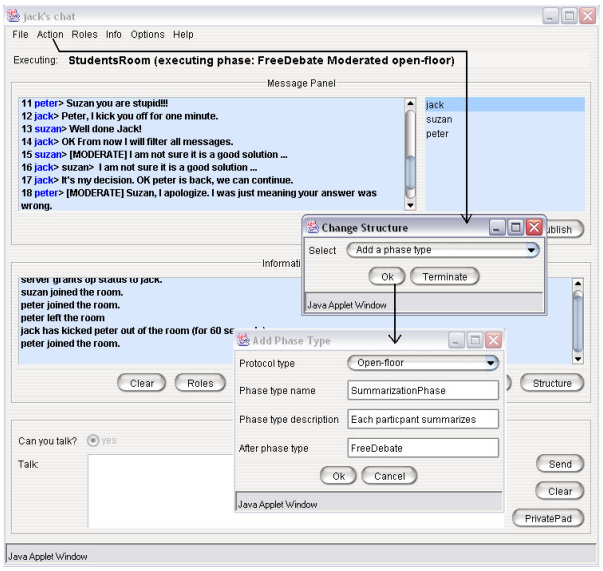

Figure 2. Examples of dynamic evolutions

DISCUSSION AND EVALUATION

From the collaborative work perspective, ωChat is a fully-fledged generic framework for building flexible chat applications. It combines the process view of Lead Line (Farnham, 2000) and the protocol view of ProChat (Whitehead, 2000) and Learning Protocols (Pfister, 2002). One can argue that ωChat does not "invent a new way of chatting" but only a practical way of building flexible, domain specific, chat-based collaborative tools. However, through its malleability properties, ωChat provides new ways of controlling chat sessions, which can deeply change the way of chatting: end users can strengthen (or relax) constraints when it becomes necessary during a chat session. This is very important in the educational context. Teachers can control many parameters, such as the content of the turns (in the moderated style), the flow of turns (with the predefined and application-specific protocols), who are the participants, their roles, the overall knowledge construction process (in structured sessions). Teachers can react easily to concrete problems such as lack of participation, flying fingers domination, control of disturbing persons, etc. For formal learning sessions, it is possible to support different collaborative knowledge construction theories, such as reciprocal teaching, guided peer questioning (Weinberger, 2003), or Socratic group discussion (Hoeksma, 2004). All these theories aim to facilitate collaborative learning by specifying activities in collaborative settings, sequencing these activities and assigning the activities to individual learners through roles definition. ωChat allows to implement them flexibly, alone, or in conjunction with other collaborative tools.

Evaluation of ωChat is a complex task. As a generic framework, we must prove that the tool can support a large scope of formal learning sessions, i.e. evaluate its process and protocol modeling languages. Work is on progress for finding in the literature various scripted learning approaches and for supporting them with ωChat. The prototype must also be evaluated from the ergonomic point of view. Internal tests have already suggested several improvements to the initial version (now included in the first public release): providing a multi line talk zone instead of a single line for avoiding the "one line/one thought" hypothesis and encouraging more carefully constructed contributions, providing a private zone distinct from the talk zone for preparing these contributions, making private sessions (whispering) a controlled optional mechanism, etc. ωChat adoption is promoted in two ways. First, we provide the tool freely, as on open source software (http://omegachat.sourceforge.net). Second, we are integrating ωChat into a workflow-enabled community platform for CSCL practice and dissemination. The example of open source software demonstrates that a large exposure to a community of practice is a very effective way for testing and improving innovative tools.

CONCLUSION

This paper describes our approach for defining a multi purpose chat technology for educational settings. For a long time, the value of chat tools for sharing insights and thoughts, for making decisions and reaching consensus, for quickly clarifying ambiguities and obtaining immediate replies has been acknowledged (Talamo, 2001).

ωChat is a generic framework for dynamic and fluid management of structured learning processes. It provides dedicated "interfaces" adapted to four categories of users having different requirements for evolution: passive chatters, who just communicate in accordance with the current interaction rules, active chatters, who are interested in maintaining the best organization for the ongoing interaction process, interaction designers, who prepare in advance, without programming, new application-specific ways of interacting, and tool developers, who customize the framework at the code level and integrate it within larger collaborative environments. We hope that ωChat could be more successful than previous attempts for imposing structure to synchronous text-based discussions because it allows to introduce innovative interaction modes progressively and reversibly during the collaborative sessions.

REFERENCES

Agostini, A., De Michelis, G., Grasso, M.A. (1997) Rethinking CSCW Systems: The Architecture of Milano. Proc. of ECSCW'97 (Lancaster, UK, September 1997), Kluwer Academic Publishers, 33-48.

Bradner, E., Kellog, W.A., Erickson, T. (1999) The Adoption and Use of Babble: A Field Study of Chat in the Workplace. Proceedings of ECSCW'99 (Copenhagen, Denmark, September 1999), Kluwer Academic Publishers, 139-158.

Fadel, L.,M., Nazareth, A. (2004) Animated-Chat, Facial expression to support social sense of presence. Proc. of 2nd Int. Workshop on Designing Computational Models of Collaborative Learning Interaction (Maceio, Brazil, August 2004), http://sra.itc.it/people/soller/ITS2004Workshop/papers/ Simao.pdf.

Farnham, S., Chesley, H.R., McGhee, D.E., Kawal, R., Landau, J. (2000) Structured Online Interactions: Improving the Decision-Making of Small Discussion Groups. Proc. of CSCW'00 (Philadelphia, Pennsylvania, December 2000), ACM Press, 299-308.

Finkelstein, A., Kramer, J., Nuseibeh, B. (1994) *Software Process Modeling and Technology*. John Wiley, Advanced Software Development Series.

Garcia, A., Jacobs, J. (1999) The eyes of the beholder: Understanding the Turn Talking System in Quasi-Synchronous Computer Mediated Communication. *Research on Language and Social Interaction*, 32, 4, 337-367.

Hoeksma, K, Virtuelle Sokratische Gespräche-Umsetzung einer Idee aus dem Philosophieunterricht, Proc. Mod 2004 Workshop, http://www.collide.info/ILLS/Workshop_Mod2004/TagungsbandMod2004.pdf

Kurlander, D., Skelly, T., Salesin, D. (1996) Comic Chat. Proc. of SIGGRAPH'96 (New Orleans, Louisiana, August 1996), Addison-Wesley, 225-236.

O'Neil, J., Martin, D. Text Chat in Action (2003). Proc. of Group'03 (Sanibel Island, Florida, November 03), ACM Press, 40-49.

Pfister, H.-R., Mühlpfordt, M. (2002) Supporting discourse in a synchronous learning environment: The learning protocol approach. Proc. of CSCL'02 (Boulder, Colorado, January 2002), Lawrence Erlbaum, 581-589.

Pimentel, M.G., Fuks, H., Lucena, C. (2004) Mediated Chat 2.0, Embedding Coordination into Chat Tools. Proc. of COOP'04 (Hyères, France, May 2004), IOS Press, 99-103.

Reyes, P., Tchounikine, P. (2004) Redefining the Turn-Taking Notion in Mediated Communication of Virtual Learning Communities. Proc. of ITS 2004, LNCS 3220, Springer-Verlag Berlin Heidelberg, 295–304.

Smith, M., Cadiz, J.J., Burkhalter, B. (2000) Conversation Trees and Threaded Chats. Proc. of CSCW'00, (Philadelphia, Pennsylvania, December 2000), ACM Press, 97-105.

Suchman, L. Plans and Situated Actions (1987). *The Problem of Human-Machine Communication*. Cambridge University Press, Cambridge, NY.

Talamo, A., Ligorio, B. (2001) Strategic Identities in Cyberspace,CyberPsychology and Behavior, special issue G. Riva, C. Galimberti (Eds) *Mind in the Web: Psychology in the Internet age*, 4, 1, 109-122

Viegas, F.B., Donath, J.S. Chat Circles (1999). Proc. of CHI'99 (Pittsburgh, Pennsylvania, May 1999), ACM Press, 9-16.

Vronay, D., Smith, M., Drucker, S (1999). Streaming Media Interfaces for Chat. Proc. of UIST'99 *(*Asheville, NC, November 1999), ACM Press, 19-26.

Wang, W., Haake, J. (2000) Tailoring Groupware: The Cooperative Hypermedia Approach. *Computer Supported Cooperative Work*, 9, 1, 123-146.

Whitehead, R.K., Stotts, D. (2000) ProChat: Dynamic Formal Collaboration Protocols in a Chat Tool for Handled Collaboration, Technical Report UNC TR00-016, University of North Carolina.

Weinberger, A. (2003) Scripts for computer-supported collaborative learning. Effects of social and epistemic cooperation scripts on collaborative knowledge construction, Dissertation, Ludwig Maximilians-Universitädt, http://edoc.ub.uni-muenchen.de/archive/00001120/01/Weinberger-Armin.pdf

Winograd, T., Flores, F. (1986) *Understanding computers and cognition: a new foundation for design,* Addison Wesley.