Jürgen Mangler & Christine Bauer (2004). A CEWebS component for facilitating online cooperative brainstorming processes. Proceedings of the 7th International Workshop on Interactive Computer-Aided Learning. Villach, Austria, 29 September - 1 October.

A CEWebS Component for Facilitating Online Cooperative Brainstorming Processes

Jürgen Mangler, Christine Bauer

Department of Computer Science and Business Informatics
University of Vienna

Key words: Blended Learning, Web based learning (WBL), Learning technology, Learning platforms, Whiteboard, Chat, Brainstorming

Abstract:

Drawing your ideas to a peace of paper is easy, but drawing onto the screen using the mouse appears to be quite complicated. This usually results in some non-formalised, bad-looking scribbling that is hard to extend and which makes it hard to modify or delete fragments.

Currently available web-based whiteboard tools support common drawing features. However, in informatics, teams enjoy the advantage of formalised languages (e.g. UML), which whiteboards do not support and, hence, do not make this advantage effective.

For supporting brainstorming processes, we developed a simple, extendable chat application that can be embedded into existing learning environments through a Web service framework and focuses on a cooperative modelling approach instead of cooperative drawing.

1 Introduction

1.1 Background

In summer term 2004, about 290 students attended the course ”Web Engineering” at our department. In this course students should learn about different types and architectures of Web Based Systems (WBSs) and how to use them. In small-scale teams of 3 to 5, students had to go through the whole process of planning and realising their own small WBS. This process was loosely based on the Rational Unified Process (RUP) [18] and included the presentation of a vision document, an inception phase in which students specified the basic outlines of their systems, an elaboration phase with detailed analysis of their WBSs, and a construction phase resulting in a fully functional prototype. The whole course was supported by a custom Web service based platform (see [11, p 20, p 22]). This platform provided students with helpful resources (e.g. lecture notes, links, etc.), a tool for uploading and sorting students’ contributions, tools for testing the functionality of their home assignments (e.g. XML, DTD, Schema, WSDL), and various communication facilities such as forums and feedback forms.

In the inception phase, one of the suggested steps was the creation of a Rich Picture Diagram

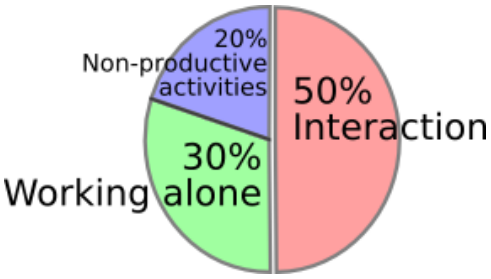

Figure 1: Distribution of a Software Engineer's Time

(RPD), which resulted directly from a team's brainstorming process and formed the basis for all architectural decisions that took place later. Although it is quite difficult to assess a non-formalised diagram, we found student's RPDs quite varying in terms of understandability and efficiency.

1.2 Aim

A typical project is not carried out by an individual but by a closely interacting team. A study conducted by IBM (see [12]) revealed that about 30% of a typical programmer's time is used for interacting with other team members (see Fig. 1). This should not be different for web projects.

We wanted to create a realistic setting for our course. Thus, giving our students not only theoretical and practical knowledge of methodologies and tools used for creating WBSs but also a sense of the interaction that is necessary in order to create such systems.

In order to create realistic working conditions, we took a blended learning approach as discussed in [15] and created a learning setting that supports interaction and teamwork. This setting included discussion and presentations during the presence phases and forums, workspaces, and upload facilities supporting the online / homework phases.

At this point, we want to highlight the group's heterogeneity: participating students came from different fields of study, different universities, and different semesters. The benefit of facilitating online cooperation was obvious for us.

Teams' first tasks included the presentation of their ideas for their self-chosen projects. Following loosely the activities described in "Developing Web Information Systems" (see [20]), students had to use the concept of Rich Picture Diagrams (RPDs) from the Soft Systems Methodology (SSM) to describe the basic outlines of their systems (see Fig. 2).

Using SSM, systems are described by diagrams or "rich pictures" (see [5]) - diagrams "without rules". Such RPDs are used for organisational analysis (see [20, p 32]) and illustrate following aspects:

- involved actors (people),
- their roles and purposes,

Figure 2: Example of a Rich Picture Diagram

- what they want from the systems,
- their concerns,
- additional details to describe the environment as accurately as possible (e.g. different interests in the organisation, similar processes to reach a goal), and
- how interests correspond or conflict.

RPDs look like comics and contain arbitrary symbols. There are no restrictions, although students have shown a certain preference for well-known symbols, e.g. from the Unified Modelling Language (UML) (for UML see [16]).

The Unified Modelling Language is used for visual construction and documentation of information systems. The language is in widespread use, although for WBSs it is sometimes disproportionate to create each and every diagram.

While and after drawing the RPDs, students created Use Case Diagrams (UCDs). Students know this concept, which is part of the technical design (see [20, p 32]), quite well since it is an integral part of their studies. UCDs describe the dynamic behaviour of the system, illustrating what cases an actor can carry out. Therefore, they consist of very few elements:

- Actors: persons or entities that interact with the system,
- Use Cases (UCs): the activities that actors carry out, and
- relationships between actors and UCs, actors and actors, or UCs and UCs.

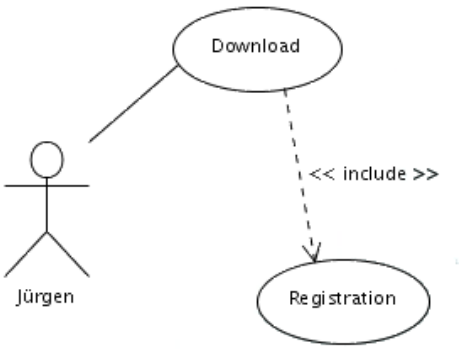

Figure 3: Example of a Use Case Diagram

Fig. 3 shows a small example of a UCD. The actor *Jürgen* can carry out downloads. For each download he has to register (every download *includes* a registration).

2 Initial Situation

The submission of RPDs was optional; nevertheless, about 90% of students created them. Obviously it seems natural to them to visualise their thoughts in form of RPDs. Students used the rich pictures as basis for the creation of UCDs.

After talking to the students about how they created their RPDs, we found out that there were essentially three ways of creating them:

1. **Students created the RPDs in presence meetings**. They created their RPD together via pen and paper or modelling tools. Afterwards the RPD was digitalised (in the pen and paper case) and put online.
2. **Students created the diagrams in online cooperation**. One person set up a document (e.g. Visio, PowerPoint, Word, etc.) and sent it to a team member via e-mail. The document was modified (elements were added or deleted) and sent around until the desired result was achieved.
3. **Combination of the above**. The students created the RPD in a presence meeting with additional online discussions and revisions.

We also collected feedback from the students in order to gather information about what impacts the creation of these diagrams with a computer (including sending them to their team members) had for them. For the second and third scenario, for which the creation process was computer-supported and online interaction potentially took place, we can summarise as follows:

- Creating the diagram is time-consuming, especially when used to visualise and discuss different ideas.

Figure 4: Stick-man as the Symbol Actor

Figure 5: Comic Figure Lisa Simpson as the Symbol Actor

- In most cases, creating the diagram requires several exchanges until a compromise about a final version is reached.
- Students do not have compatible tools available. They submitted RPDs as PowerPoint, Word, Visio, PDF or image files.
- Students use well-known symbols.

It is remarkable that computerised creation (i.e. using a tool) of RPDs had been slightly more popular than drawing the diagram with pen and paper (although the lecture showed a pen and paper example). Surprisingly there were even three examples, in which the basics were drawn with pen and paper and additional text was added into speech bubbles via drawing tools.

Especially the analysis of symbols used revealed interesting issues. We have to admit that the analysis of symbols used in RPDs is certainly subjective or - better say - do not respect the creator's creativity. For instance, a stick-man (see Fig. 4) has been valued as the symbol *actor* as well as had been the comic figure Lisa Simpson (depicted in Fig. 5) or a stressed employee (see Fig. 6), which both hold artistic creativity. Nevertheless, we could identify that among the total number of 51 submitted RPDs, only 28 distinct symbols were used (see Fig. 7).

In essence, we can summarise following aspects that seem interesting to us and are worth mentioning:

Figure 6: Stressed Employee as the Symbol Actor

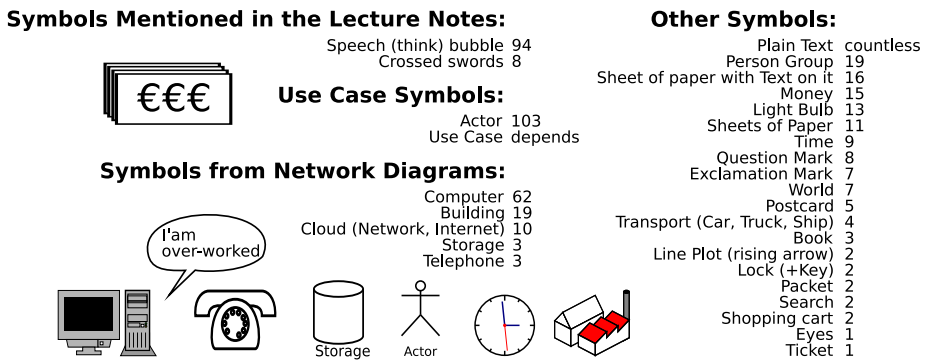

Figure 7: Symbols Used by the Web-Engineering Students

- Most students, especially when using pen and paper, used well-known symbols, e.g. the actor symbol from UML and symbols from network diagrams including computers, network clouds, buildings, storage, and telephone symbols.
- The preference for Use Cases was more a textual than a directly visual one. The students used phrases that they could directly reapply to UCs - and did so. In our opinion, this may be due to two reasons:
  - Students created the well-known UCs before the RPDs (which were in fact new to most of them), and only summarised their UCs as RPDs afterwards.
  - Students used their RPDs as intended to analyse the whole system, to ease the creation of detailed Use Cases but are influenced by their knowledge about UCDs.
- The students did not use different kinds of arrows for describing relationships between symbols although they should know different connection types very well (e.g. from UML). They only used ordinary solid line arrows, sometimes with a caption.
- Visual quality of RPDs submitted varied a lot. The visual representations reveal what students had in mind when creating their RPDs (which ranges from *I have to do it, but I do it fast* to *RPDs are indeed a good idea*).
- RPDs realised with pen and paper used symbols that were easier to understand. Therefore they were overall more comprehensible.
- The diagrams made in Word, PowerPoint and Visio sometimes showed excessive use of ClipArts, which leads to a cluttered appearance (photos of persons or comic figures were not immediately recognisable as actors).
- The diagrams done with ordinary drawing tools resembled the style of the pen and paper version presented in the lecture but suffered from a lack of completeness; they generally consisted of basic elements only.

3 Improving the Situation

Analysis of the is-situation shows that 24 teams (47.06%) used pen and paper for realising their RPDs while 27 teams (52.94%) made use of a modelling tool. Albeit, the analysis reveals that

students realised their RPDs mainly in presence meetings. Main reason for this seems to be that online cooperation would have been cumbersome and complicated. This assumption is affirmed by students' reactions. The tenor can be summarised as follows:

- We tried to make our diagram in online cooperation but we were simply unable because because we lacked in a tool that anyone of our team could install on his computer.
- A lot of discussion was necessary in order to put down the RPD in a version we all could agree on. Exporting files and sending them via e-mail or posting them in a forum had been exhausting.
- We agreed to take the effort to meet personally since, for us, interaction had been important and we could not find a way to have this interaction with immediate feedback when working together online. We like online cooperation, which is mostly possible for writing seminar papers or realising small projects. However, we simply could not figure out how to efficiently cooperate online to do the RPD.

Students' feedback shows that they seem to favour online cooperation. This may be due to the group's heterogeneity; students came from different universities and different semesters, which means that they rarely met at university. The benefit of supporting the course by an online platform had been obvious for us. Students' feedback strengthens our assumption.

To put it short, though students favour online cooperation, most teams decided to realise their RPDs in presence meetings. First, they did not have appropriate tools available that supported their needs. Second, students do not have compatible tools at all. Thus, instead of modifying a team member's version of the RPD, one has to redo the diagram with the desired change. Obviously, this inefficiency is a matter for improvement. Furthermore, this fact leads to inefficient communication via e-mail, forums, etc.

3.1 Can Web-based Whiteboards Challenge the Problem?

Challenging these drawbacks by providing our students with a web-based whiteboard system in order to give them a compatible tool was obvious.

3.1.1 What are Web-based Whiteboards?

Traditional whiteboards (or chalkboards) help facilitators to make important coherences clear to students or, for instance, to outline important facts concerning a lesson's structure in order to make them more understandable. In brainstorming processes, whiteboards are often used to collect ideas or to explain them to the group.

Web-based whiteboards are used similarly. Existing images can be loaded and changed with basic drawing tools. These tools are comparable to small applications like MS-Paint allowing

the user to use pencil, rectangle, fill, text, and erase tools for modifying an image. Users are connected synchronously (not possible with HTTP, which is asynchronous).

In existing learning environments, web-based whiteboards are realised and integrated as java-applets (see e.g. [2]) or through external components like MSN-Messenger or MS-NetMeeting (see [13]).

One approach of web-based whiteboards is to load a presentation into the whiteboard, allowing the user to make his (private) notes and save them locally (comparable to the scenario that a student writes his personal notes on his printed script).

Traditionally, whiteboards were used to provide students with information. Nowadays, this can be easier achieved by making the material available online. So, whiteboards are more and more used to visually enhance discussions between students, tutors, and facilitators. A web-based whiteboard can do essentially the same in online settings. Technically speaking, it is a tool that distributes a user's local input to all participants.

Briefly, web-based whiteboards:

- provide synchronisation between all users,
- enhance traditional chat systems,
- are sometimes further enhanced by voice or video conferencing possibilities,
- offer the possibility to modify existing images and / or create new ones, and
- are highly integrated into learning environments or realised via external components.

3.1.2 Feasibility of Web-based Whiteboards

Students did not have compatible tools available. Instead, students had different tools and thus could not modify other team members' versions of RPDs. Providing students with a particular tool that is available for all students could challenge this problem.

Furthermore, web-based whiteboards support synchronous collaboration of users. Consequently, the inefficiency of sending a version of RPD to the other team members via e-mail for modification (or posting it in a forum and waiting until revision) disappears.

We tried out three web-based whiteboards, which should be representative for the current market. Below, we give a brief overview about th most common features:

- **Groupboard** (see [6]): Consists of java applets, including a whiteboard, a chat application, and a message board. One of its strengths is that everyone can embed a version with limited features for free in his website. Basically, it is a web-based drawing application that support lines, circles, and rectangles in different colours. It enables the user to save, load, and print images.

- **NetMeeting** (see [13]): Its functionality is very similar to the one of Groupboard. It uses the well-known MS-Paint. The same component can also be found in MS-Messenger.
- **WebCT** (see [25]): WebCT is the provider of the same called e-learning platform. At the University of Vienna, WebCT Vista is available. Like Groupboard, the whiteboard is implemented by a java applet. It offers more advanced capabilities, like moving or reshaping elements. But it also supports primitives like straight lines, ovals, and rectangles.

As can be seen from above overview, all of these tools support common drawing features. However, that is the point - whiteboards have a focus on painting. In informatics, teams enjoy the advantage of a formalised language, but whiteboards do not support any formalisation and, hence, do not make this advantage effective.

The fact that in 51 RPDs students used only 28 distinct symbols seems to be due to two reasons: First, projects to be described as RPDs are of technical nature (informatics); second, our students have technical background and, thus, are familiar with standard symbols from UML and network diagrams.

Though our findings show that web-based whiteboards can possibly improve the current situation, the results are not satisfactory.

3.2 Our Tool

Our findings reveal that available tools cannot keep up with the demands of our students. Therefore we decided to develop a tool that supports the cooperative creation of diagrams, which involves:

- shared drawing of diagrams that consist of common used symbols to exploit the advantage of a semi-formalised picture language and
- a chatting facility in order to comment, describe, and discuss decisions.

In the subsequent sections, we further discuss the components our system consists of.

3.2.1 Synchronous Diagram Creation Feature

In order to exploit the advantage of a semi-formalised picture language, the tool shall easily support common used symbols.

Since the creation of a RPD is, generally, a cooperative task, the tool has to support cooperative creation. Due to the fact that the creation process of a RPD is time consuming per se, it is important that the tool used does not further expand the time necessary to create the diagram. Instead it should make the finalisation of the diagram possible within reasonable time.

The process of creating a RPD is complex. It is easy to understand a finished diagram, but it is rather impossible to understand a diagram that is only halfway done. The context is very important in this latter case; without the context of immediate feedback the ability to correct interpretation may be upset and disrupted (see [23, p 25]).

Synchronous tools allow immediate response (see [21, p 81]), which makes it possible to integrate new information into work and thinking processes immediately. On the other hand, most asynchronous conferencing systems represent context with sequencing markers, various structural features, etc. (see [23, p 25]), which we could implement in our tool for establishing context. However, processes behind RPDs are personal and creative. Structural features and markers do not provide help in understanding these processes. As a consequence, due to the missing context, a user would have to document his thinking process (in written) for his collaborators to understand the meaning of his RPD's inception. This scenario would result in a written documentation supplemented by a diagram. This, in fact, misses a RPDs purpose. Processes behind RPDs are that complex that a written documentation fails to give an account of these processes in a way that it is easily understandable. Therefore RPDs have been established aiming to make these complex processes explicit and understandable.

Interaction within context (whether in presence meetings, via telephone, chat or other synchronous systems) needs fewer cues in order to make one's train of thoughts traceable and comprehensible than communication out of context.

As a consequence, we suggest a synchronous environment, which allows a high level of interaction (see [21, p 81]) and synchronicity (see [19]) in context, for our tool in order to best support the processes.

Furthermore, as already described in detail above, students make use of a limited pool of symbols only (e.g. taken from UML standard). Providing *autoshapes* (i.e. predefined elements / symbols) could ease the design process. First, it is easier to find an adequate symbol out of a pool than freely think of any (though this definitely narrows creativity). Second, it takes less time to insert a predefined symbol and just adapt size and position than having to draw it freehand. Third, from a technical point of view, data is saved more efficiently with autoshapes, for which data can be saves as text (position and scale), than with freehand drawings, for which every single pixel has to be saved. Fourth, the realisation of the *undo*- and *redo*-functionality gains meaning (undo whole symbols instead of individual design steps).

This last argument leads us to a further important feature: the deletion and movement of elements. When realising a RPD with pen and paper, it is (almost) impossible to move or remove elements that have already been drawn. Tools generally support this feature, while it is evident that autoshapes can easily be moved and removed whereas deleting or moving freehand drawings demand a great deal of the user as well as of the tool.

Our analysis has shown that diagrams realised with Word, PowerPoint and Visio showed excessive use of ClipArts, comic figures or photos of persons, whereas RPDs realised with pen and paper generally held symbols that were easier to understand. Generally speaking, the latter were as a whole more comprehensible. Consequently, we suggest that the possibility to implement ClipArts, comic figures, etc. into the diagrams does not improve the results but impair them. Hence, a sufficient, though minimal, pool of symbols should serve as a good solution.

3.2.2 Chat Facility

The drawing facility builds the core component of our tool. However, though the proverb says, *A picture paints a thousand words.*, the drawing facility needs to be supported by an additional communication component for interaction. Words give meaning to what is going on in the drawing part; they give the context.

As outlined above, synchronous collaboration is to be preferred. Consequently, the communication tool has to be a synchronous system as well. Communication via telephone or audio conferences are neither reasonable nor useful since deployment of these systems is expensive (running costs) and the average student does not have the required equipment at his disposal. Instead, the deployment of a real-time chat system seems appropriate.

A real-time chat is a synchronous, written, computer-based communication system. It enables synchronous communication with other users of the system. Usually several people communicate with each other at the same time.

Generally, users insert short messages line by line via the keyboard and submit these (see [1], [4, p 19, p 29]). Furthermore, there are chat systems that allow transmission character by character in real-time (teletype style; e.g. Unix Talk).

A beneficiary of chat communication may be in the lack of physical (nonverbal) cues. Absence of these cues may increase one's cognitive resources devoted to message construction and perception because there is no need physically to backchannel (nod, smile, remember to look interested, etc.). Due to this high level of private self-awareness (see [23, p 23], [3, p 84]), the entire attention can be drawn to the task. Since paralinguistic phenomenons (like e.g., intonation, cadence, breaks, sighs, etc.) (see [24, p 51] are transmitted via oral communication systems (e.g. audio conferences), chat systems, which lack in these cues, better support private self-awareness than the former.

Furthermore, in a chat, students are present via their messages only. Consequently, individuals are less distracted than in presence meetings and can, hence, concentrate alternately on absorbing information and phrasing messages (see [3, p 84]). As regards this issue, deployment of a chat system adds value compared to conversations in presence meetings.

For the same reason, people tend to advance their opinions more openly than they would in presence meetings. Though this may result in more badmouthing and flaming (see [9, p 150], [7, p 74]), it also increases the variety of opinions expressed, which is a valuable asset for the process of RPD creation.

Albeit, it is not to be forgotten that typing a text is cognitively more demanding than reproducing the same text orally and takes about four times longer to exchange the same number of messages as in presence meetings (see [22, p 189]). However, [7, p 74] point out that this depends on users' experience with the technology. Since our students are computer scientists, we assume them to have a high level of experience with chat systems.

Concluding, we believe that using a chat system instead of audio conference systems is not only a compromise but gives additional value to the process of creating RPDs.

4 Requirements and Description of the System

Considering above discussion, we established the functional and non-functional requirements for our tool.

4.1 Functional Requirements

To meet our students' needs, we could identify the following functional requirements. The tool has to enable:

- synchronous communication via chat,
- synchronous cooperative drawing on a shared whiteboard,
- the sharing of diagrams with trusted people (everyone should be able to invite people to his own whiteboard),
- efficient brainstorming through the combination of chat and whiteboard within a single window,
- its use with different web browsers on different Operating Systems,
- the export of diagrams to a common image format,
- management of multiple diagrams (save whiteboard, load to whiteboard, rename, delete), and
- going back and forth in creation history to see how the diagram was created step by step.

The demand for a web browser based environment is apparent. To avoid an installation process and to be able to control / use the application from a trusted environment (the server) is a good idea for both, the maintainer and the user.

Exporting to a common image format should be as simple as possible. We decided to use PNG, as it is both, license free and provides good image quality (unlike jpeg, which is patent-covered and more suitable for photographs than for images).

To manage multiple diagrams (and images), it is necessary to provide a form of Diagram Management (DM). For our first version, we decided to take an approach common to Internet Relay Chat (IRC) (see [8]). We agreed to implement IRC style commands (with User Interface (UI) elements to launch them) to save, load, and list the available images.

Following commands are required (note that IRC commands typically start with a slash):

- a */list* command to display all saved diagrams,

- a */load* command to load an image into the whiteboard,
- a */save* command to save a whiteboard diagram state,
- a */delete* command to delete a diagram,
- a */clear* command to clear the whiteboard,
- a */invite* command to send invitations to other users that enable them to join a whiteboard, and
- a */ban* command to revoke a users' invitation to a whiteboard.

The whiteboard itself should have the characteristics as presented below:

- Every user has his own whiteboard.
- Every user can invite people to his whiteboard.
- The owner of the whiteboard can load diagrams and clear his whiteboard.
- Every user should be able to save diagrams from the whiteboard, he is participating. He can load the saved diagrams into his own whiteboard at a later point in time.
- The whiteboard owner should be able to revoke an invitation.
- When a whiteboard owner leaves, the other participants can continue with cooperative drawing but are not able to load diagrams or clear the whiteboard.
- When a user leaves, his invitation stays active. After returning to the chat he can still join the whiteboard and participate in the drawing.
- While in whiteboard mode, the chat messages of the non-participating users are invisible, only the messages from the whiteboard users are visible. Though, it should be possible to send a message to a whiteboard session from outside by explicitly naming the owner of the whiteboard (IRC style).

4.2 Non-functional Requirements and Design Decisions

There are also additional requirements that do not explicitly affect the functional range of the system. They are mainly technological design decision resulting in easier maintainability but can as well influence the user.

First, our system builds strictly on open source tools and is open source itself. We think that in an academic environment it is not fruitful to share the results only. We want to share also the work itself and benefit from possible contributions by third parties.

One important decision is that we want to deliver the chat and drawing part in the form of Shockwave Flash (SWF) component (see [10]). One may argue that SWF is kind of an artistic

tool. However, we believe that it is quite suitable for Rich Internet Application development (as it is advertised for on its website). The Ming project (see [14]) provides us with a set of tools and a library that enables us to write our application without the use of the commercial authoring software.

Furthermore, our decision to implement everything in action script instead of adding static elements to the SWF component should help to stay independent. For the communication between the Client (SWF component in the users' browser) and the server that collects data from the clients and distributes it back to them, we use XMLSockets. XMLSockets basically realise a stateful connection (HTTP itself is stateless and the browser only has a connection to the server during the loading of a page) over which small pieces of XML are exchanged.

The symbols should be loaded upon use from a separate SWF image repository.

The server consists of two parts, a CEWebS (see [11]) and a Chat Server (CS) component. A CEWebS is basically a Web service (WS) with a standardised interface, being easy embedable in existing infrastructures (e.g. an e-learning platform), and provides facilities like distributed user administration and configuration. Obviously, we expect the chat / whiteboard tool to be integrated perfectly and easily into our existing infrastructure. CEWebS also provides the developers with facilities to create UIs for various contexts (e.g browsers, wireless devices, etc.), which is generally a good idea. The CEWebS also delivers the SWF component to the client.

However, since WSs are stateless (at least in reality), we need the additional Chat Server component that can handle the stateful connections to the clients in order to realise synchronous communication between users. This CS, for sure, is tightly integrated into the CEWebS to take full use of its authentication facilities. The chat logs and created images should be accessible through a web interface provided by the CEWebS.

The CEWebS and the CS are written in Ruby (see [17]), which is characterised as combination of the object-oriented strength of Smalltalk and the expressiveness of Perl with an easy-to-use syntax. It proved to be perfect for object-oriented Rapid Application Development (RAD) and also brings in the advantage of the result running on different flavours of Unix (Mac OS X, Linux, Solaris, etc.) and Windows.

Finally, the tool has to be ready for international usage. Consequently, the internal data flow should be entirely unwinded in UTF-8 in order to guarantee compatibility with as many languages (and their special characters) as possible.

5 ChatStorm - Architecture and First Impressions

In this section, we want to provide first impressions and walk through an example brainstorming.

Resulting from the requirements, we established a fully functional prototype, which we call *ChatStorm*. This "branding" underlines that our system builds a symbiosis between chat and brainstorming.

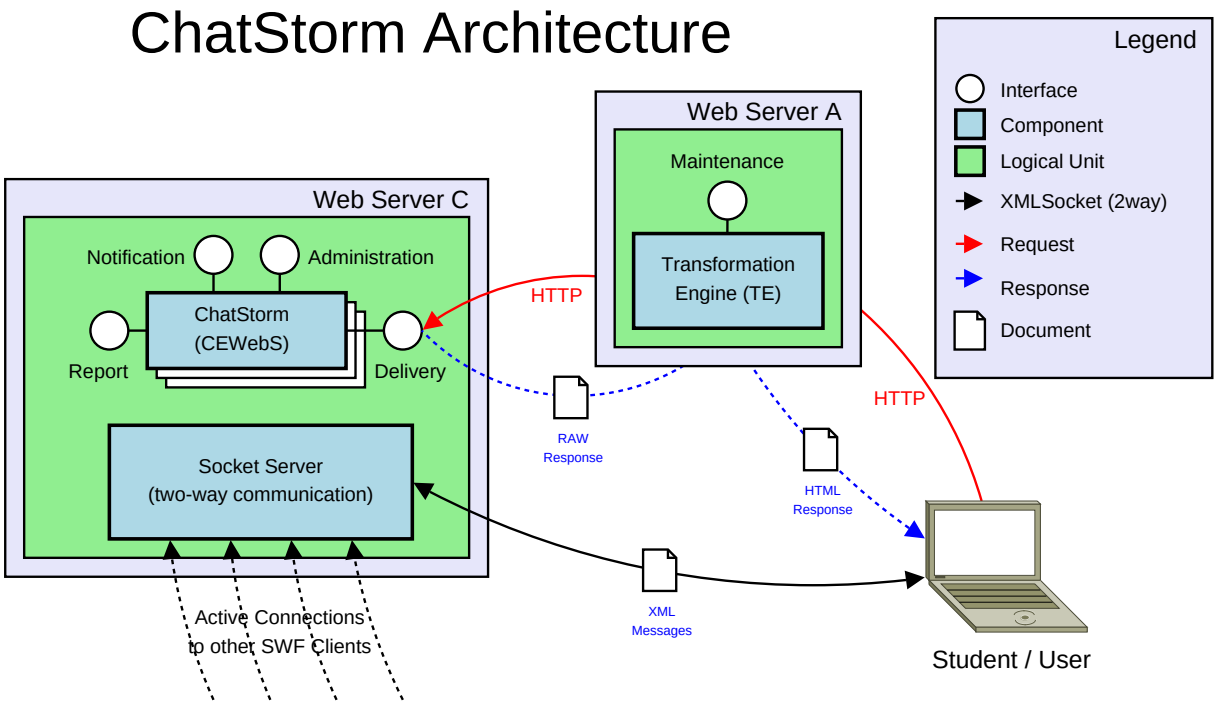

Figure 8: ChatStorm Architecture

Fig. 8 sports ChatStorm's architecture. It shows clearly that ChatStorm significantly differentiates from a standard CEWebS insofar as the synchronous connection between its users is not handled by standard Web service calls but through separate XMLSockets. It has to be mentioned that this can pose a problem when Web server A and B do not reside within the same domain, since the SWF client can only connect to servers from the same domain it was delivered from (though buying a certificate from Macromedia could solve this problem).

The system works basically as described in the in the preceding chapters. Fig. 9 points out the simplicity of the user interface (UI) in the chat mode:

The UI consists of only four parts:

- The main area where all the messages are printed,
- an input field for writing messages (which are delivered to the server when pressing the enter key),
- an area *Users* where all users are listed, and
- an area *Whiteboards* where all whiteboards, a user is authorised to access, are listed (of course, the *Own* whiteboard is always listed).

In our showcase, two persons meet via chat and agree to switch to the whiteboard. *Christine* invites *Jürgen* to draw with her on her whiteboard. From now on the *Whiteboard* area of user *Jürgen* shows a link to the whiteboard of user *Christine* (although it is possible that his invitation is revoked).

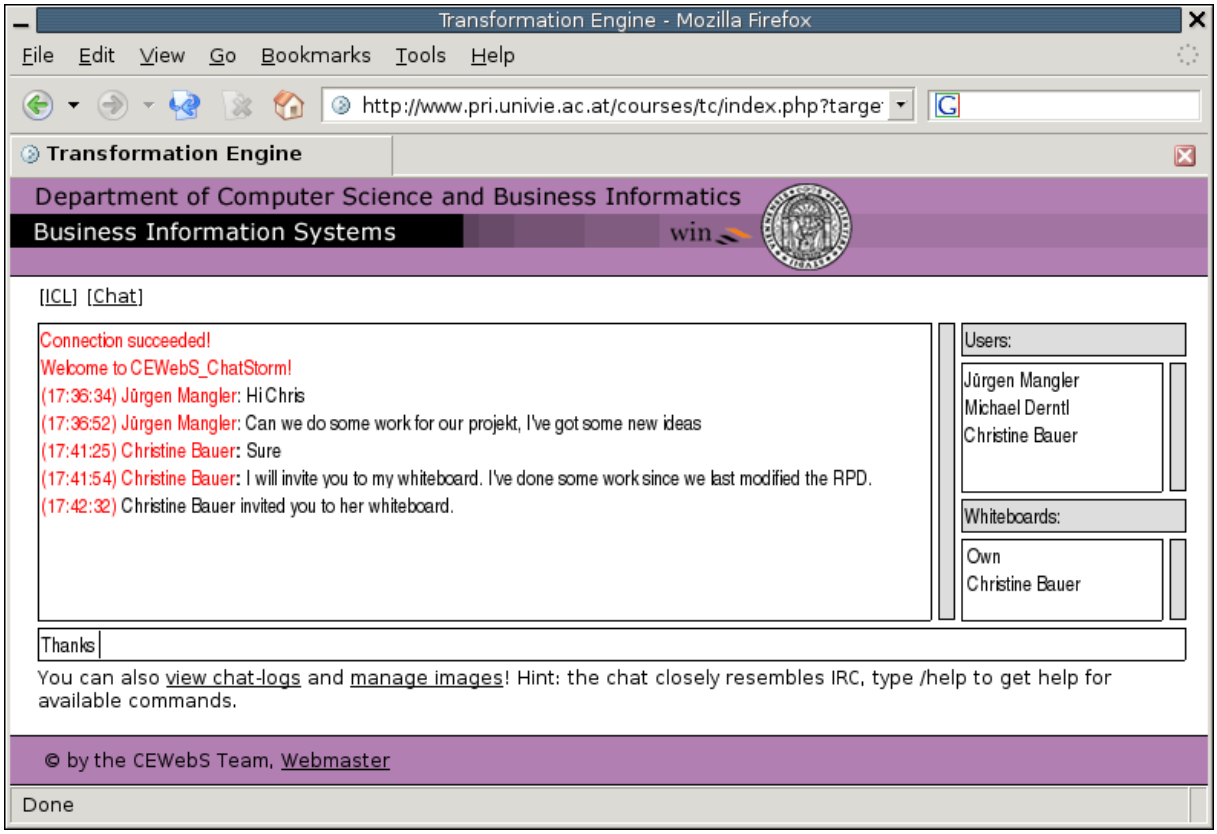

Figure 9: Two Persons Arrange to Draw on a Whiteboard

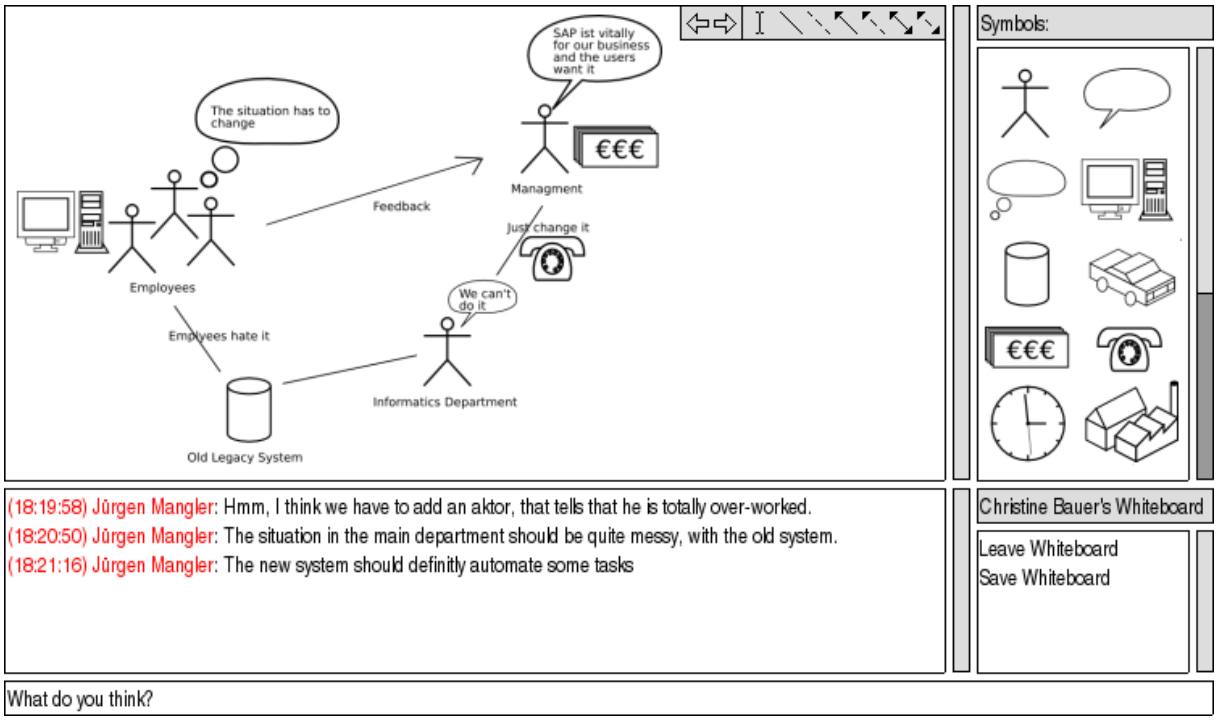

Figure 10: A Rich Picture Diagram Depicting a Complex Situation is Modified

After switching to the whiteboard the UI changes slightly (see Fig. 10):

- A drawing area including a toolbar (the whiteboard) is shown on top of the chat area.
- The chat area is much smaller now and shows only messages of users participating on this whiteboard.
- The area *Whiteboard* is replaced by a command area which holds, in this case, the commands *leave* and *save*.
- The area *Users* is replaced by the area *Symbols*.

The drawing area, the toolbar, and the *Symbols* area work together as follows:

- Per default the user is in the edit mode. He can select and move around symbols on the drawing area and delete selected symbols by pressing the *delete* key.
- A user can add a symbol by selecting it from the *Symbols* area. By clicking in the drawing area the symbol is placed there and the system switches back to edit mode.
- The toolbar provides undo / redo and holds also some basic functionalities like:
  - writing text,
  - connecting symbols with solid lines and dashed lines,
  - connecting symbols with one-way arrows, and
  - connecting symbols with two-way arrows.

6 Conclusions and Outlook

When analysing our students' cooperation for realising Rich Picture Diagrams, we noticed a lack in an adequate tool.

Currently available web-based whiteboard tools support common drawing features - similar to MS-Paint - with additional video and voice conferencing possibilities, as well as the functionality of sharing documents or online presentations. However, in informatics, teams enjoy the advantage of formalised languages - e.g. the Unified Modelling Language (UML). Since whiteboards do not support any formalisation, they do not make this advantage effective.

Drawing onto the screen using the mouse appears to result in some non-formalised, bad-looking scribbling that is hard to extend and which makes it hard to modify or delete fragments.

Our analysis revealed that informatics and computer science students make use of UML and Network Diagram symbols and methods even for creating Rich Picture Diagrams (RPDs), which do not demand any formalisation.

We developed a simple, extendable chat application that can be embedded into existing learning environments through a Web service framework and focuses on using a cooperative modelling approach instead of cooperative drawing. With this tool, we want to enable the use of an (at least) semi-formalised language, giving effect to its advantage in an online cooperation setting.

The prototype provides the implementation of autoshapes, which can be adapted by size and position. The main benefits of our component include easy extensibility or replacement of existing solutions in an OS / platform-independent manner, and a focus on a formalised way of exchanging ideas. This should result in a simplified user interaction and transparent recording of brainstorming processes. Further research will include the identification of useful sets of primitives, not only for informatics and computer science students but also for students from other domains.

In the pilot phase (winter term 2004 / 2005), we want to evaluate if minimal additional functionalities like choosing different colours for symbols as well as enable rotation and flipping will enhance this tool's adequacy.

References:

[1] Adams, Lia, Toomey, Lori and Churchill, Elizabeth: "Distributed Research Teams: Meeting Asynchronously in Virtual Space"; Journal of Computer-Mediated Communication, 4 (4),1999, available at http://www.ascusc.org/jcmc/vol4/issue4/adams.html [last access December 19, 2001].

[2] Adusumilli, Krishna K., Al-Halabi, Bassem and Hsu, Sam: " SOFTBOARD - A Web-based Application Sharing System for Distance Education", IEEE International Conference on Information Technology: Coding and Computing (ITCC 2000), March 27-29, 2000, Las Vegas, Nevada, pp. 338-341.

[3] Boos, Margarete: "Computergestützte Problemstrukturierung in Arbeitsgruppen", in: Boos, Margarete, Jonas, Kai J. and Sassenberg, Kai (eds.): "Computer-vermittelte Kommunikation in Organisationen". Göttingen: Hogrefe, Verlag für Psychologie, 2000, pp. 73-87.

[4] Brill, Andreas and de Vries, Michael: "Cybertalk: Die Qualitäten der Kommunikation im Internet", 1997, available at http://www.uni-wh.de/de/wiwi/virtwirt/konf/cybertal/cybtalk.pdf [last access December 18, 2001].

[5] Checkland, P., Scholes, J.: "Soft Systems Methodology in Action", Wiley, Chichester, 1990.

[6] User Data Connections Limited, "Groupboard - Interactive website tools for education, business and fun", information available via http://www.groupboard.com/ [last access September 17, 2004].

[7] Hollingshead, Andrea B. and McGrath, Joseph E.: "Computer-assisted groups: a critical review of the empirical research", in: Guzzo, Richard A. and Salas, Eduardo (eds.): *Team effectiveness and decision making in organizations*. San Francisco: Jossey-Bass, 1995, pp. 46-78.

[8] Various Authors: "Internet Relay Chat", available at http://en.wikipedia.org/wiki/Internet_Relay_Chat [last access August 29, 2004].

[9] Ishaya, Tanko and Macaulay, Linda: ”The Role of Trust in Virtual Teams”, Electronic Journal of Organizational Virtualness, 1 (1), 1999, pp. 140-157, available at http://www.virtual-organization.net/files/articles/Ishaya_US.pdf [last access October 17, 2001].

[10] Macromedia Cooperation: ”Macromedia Flash”, information and plugins available at http://www.macromedia.com/software/flash/ [last access June 15, 2004].

[11] Mangler, Jürgen and Derntl, Michael: ”CEWebS - Cooperative Environment Web Services”, Proceedings of I-Know 2004, Graz.

[12] McCue, G.M.: ”IBM’s Santa Teresa Laboratory - architectural design for program development”, IBM Systems J., 17 (1), pp. 4-25, 1978.

[13] Microsoft Cooperation: ”MS NetMeeting”, information available at http://www.microsoft.com/windows/netmeeting/features/whiteboard/default.asp [last access June 15, 2004].

[14] Ming Contributors: ”Ming - a SWF output library and PHP module”, available online via http://www.sourceforge.net [last access July 5, 2004].

[15] Motschnig, Renate and Derntl, Michael: ”Can the Web Improve the Effectiveness of Person-Centered Learning? Case Study on Teaching and Living Web-Engineering”, Proceeding s of the IADIS International Conference WWW/Internet, Algarve, Portugal, 2003.

[16] OMG: ”OMG Unified Modeling Language Specification, Version 1.5, March 2003”, available at http://www.omg.org/cgi-bin/doc?formal/03-03-01 [last access May 12, 2003].

[17] Yukihiro Matsumoto and Others: ”Ruby”, available at http://www.ruby-lang.org [last access July 6, 2004].

[18] IBM Cooperation, ”IBM Rational Unified Process”, information available via http://www-306.ibm.com/software/awdtools/rup/ [last access September 17, 2004].

[19] Schwabe, Gerhard: ”’Mediensynchronizität’ - Theorie und Anwendung bei Gruppenarbeit und Lernen”, in: Friedrich, Herlmut F. and Hesse, Friedrich W. (eds.): *Partizipation und Interaktion im virtuellen Seminar*; Waxmann Verlag, Münster, 2001.

[20] Vitgen, R.: ”Developing Web Information Systems”, Butterworth-Heinemann, 2002.

[21] Walther, Joseph B.: ”Interpersonal Effects in Computer-Mediated Interaction: A Relational Perspective”, Communication Research, 19 (1), 1992, pp. 52-90.

[22] Walther, Joseph B.: ”Relational Aspects of Computer-mediated Communication: Experimental Observations over Time”, Organization Science, 6 (2), 1995, pp. 186-203.

[23] Walther, Joseph B.: ”Computer-Mediated Communication: Impersonal, Inter-personal and Hyperpersonal Interaction”, Human Communication Research, 23 (1), 1996, pp 3-43.

[24] Watzlawick, Paul, Beavin, Janet H. and Jackson, Don D., ”Menschliche Kommunikation: Formen, Störungen, Paradoxien”, 10th unchanged edition. Bern: Huber, 2000.

[25] WebCT, Inc., ”WebCT”, http://www.webct.com/ [last access September 17, 2004], WebCT whiteboard basics available from http://www.its2.uidaho.edu/help_docs/webct_whiteboard/2_whiteboard_basics.htm [last access September 17, 2004].

Authors:

Jürgen Mangler
University of Vienna
Department of Computer Science and Business Informatics
Rathausstraße 19/9
1010 Vienna, Austria
juergen.mangler@univie.ac.at

Christine Bauer
PhD. Student
University of Vienna
Department of Computer Science and Business Informatics
Rathausstraße 19/9
1010 Vienna, Austria
christinebauer@gmx.at