

# **Interactive Presentation Support for an Electronic Lecture Hall - a practice report -**

**H. U. Hoppe, W. Luther, M. Mühlenbrock, W. Otten, F. Tewissen**

*Gerhard-Mercator-Universität – GH Duisburg,  
Fachbereich 11, Fachgebiet Informatik Lotharstr. 65, 47048 Duisburg,  
{hoppe, luther, muehlenbrock, otten, tewissen}@informatik.uni-duisburg.de*

“Electronic lecture halls” or (electronic) “teaching theaters” have been installed in many places world-wide. Usually, it is suggested that these be used for “multimedia-supported” teaching or for tele-learning scenarios including the broadcasting of lectures and communication with remote lecturers. However, in our current practice of non-distance or campus universities this is yet a rare scenario. This paper reports practical experience of using interactive presentation equipment in standard, single-site undergraduate computer science lectures. The problem with typical electronic presentations, as with transparencies, is that they tend to result in using ready-made presentations instead of facilitating the free elaboration of ideas and development of models as it used to be done on the traditional chalkboard. So, our aim is to gracefully integrate interactive whiteboards and pen-based input with the presentation of prepared electronic documents. Particular attention is given to minimizing the effort for providing both presentation material for the lecturer and electronic WWW scripts for the students, and to smoothly integrating the results of free pen-based elaboration with the scripts. First results of practical usage are available.

**Keywords: lecturing support, interactive presentation, pen-based input**

## **1 Introduction**

“Electronic lecture halls” or (electronic) “teaching theaters” have been installed in many universities and colleges all over the world (e.g. [1], [10]). These installations typically include high quality projection of computer displays, video conferencing facilities, and often also computer-based interaction on the part of the students. Usually, it is suggested that these lecture halls be used for “multimedia-supported” teaching or for tele-learning scenarios including the broadcasting of lectures and communication with remote lecturers. However, in our current practice of non-distance or campus universities this is yet a rare scenario. The work reported here is focused on the everyday practice of lecturing in a presence scenario of a campus university.

To better motivate our approach, let us first contrast our rationale to other lines of work:

- This report is not about any kind of real-time distance lecturing, so not about video conferencing etc.
- We do not believe that synchronous computer-mediated communication and cooperation between lecturers and students is a relevant issue for lecture halls with 30 or more students. So, we do not deal with real-time collaborative technologies here (in spite of some of us working on collaborative scenarios for smaller groups - cf., e.g., [5]).
- Multimedia and CBT learning materials are typically designed for individual use by the student. They are not the kind of material to be presented in an electronic lecture hall, and - they are still very costly to produce and do not cover complete curricula!

Particularly the last point about individual learning materials gives raise to the question if traditional lecturing is potentially obsolete in our days, and “didactic efforts” should be put

into the development of individualized courseware rather than modernizing lecturing using new technologies. So, what are the characteristics of lecturing which might make it valuable also in the future?

Typically, the lecturer takes the active part, not only exposing material but also *exposing herself or himself in relation to the subject matter*. This aspect of accompanying the exposition of a subject by giving a personal example is of utmost importance in our positive judgement of traditional lecturing. It introduces a reflective (and reflexive) element into the teaching/learning process, which is very difficult to achieve in individualized automated instruction. In addition to reflection, the presence of a lecture and the given rhythm of lectures may produce a higher form of personal commitment on the part of the student. Good lectures create a form of *dialogue* with the audience in two ways: First, the lecturer may lead a virtual or internal dialogue within her or his own arguments; and secondly, there is an explicit dialogue in terms of taking up questions and comments from the audience or also by questioning the audience.

The role of external representations and media may be the one of an illustration or a surrogate of the spoken message, or may just another redundant representation, or -most importantly- it may serve as an *external memory* which captures important information in persistent form. Certain cognitively demanding and complex contents, as e.g. mathematical proofs or the construction of complex formal models, rely on using such an external memory. Here, it is important that the external representation is not ready-made but grows incrementally following the spoken argument. "Computer presentations" are typically superior to chalkboard lectures in terms of their potential complexity as well as their graphical quality and readability. However, in their everyday usage they are often inferior with respect to subjective/reflective aspects and with respect to the spontaneity and incrementality of the presentation.

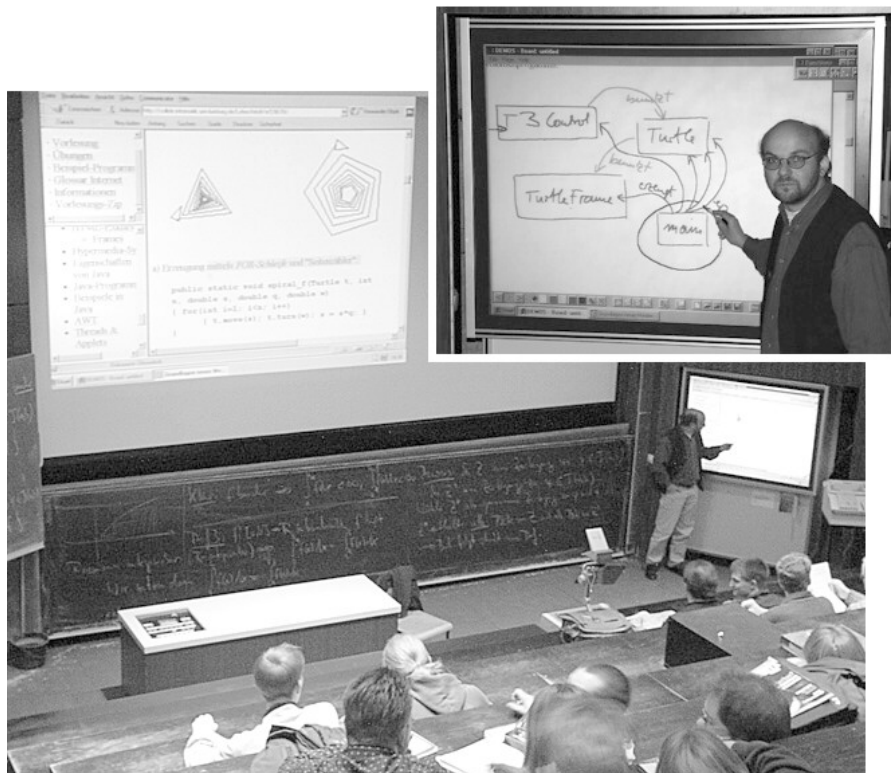


Fig. 1. Interactive electronic board + projection in a lecture hall

This work has been guided by our active and passive experience with lecturing in formal areas such as mathematics, physics, and computer science. Here, the free development of

ideas and the incrementality of presentations with external memory function are indispensable. For our lecturing in computer science, we would distinguish “know how” and “know what” subjects (within one course). Wherever we want to teach “know how”, i.e. cognitive skills or reasoning procedures, we would not like to rely on ready-made material. Based on these convictions, we have set up a lecture hall with specific hardware arrangements and software tools, which will be explained in the next sections.

## **2 Scenario and basic tools**

We were convinced by previous experience ([6]) that big interactive computer displays, also called “interactive electronic whiteboards” (e.g. Xerox LiveBoards as described in [3]), are well-suited for supporting human presenters exposing computerized presentation material or free-hand sketches and exposing themselves in relation to the material in the above mentioned sense. The size of these displays (about 1.3m x 1m) corresponds to the area covered by the hand of a standing presenter. For a classroom or a seminar room scenario, such a board is sufficient for supplying visual information to the audience. A practical “rule of thumb” says a screen may serve an audience up to a distance of 5 times its diagonal (here, approx. 8 m). In a lecture hall, such a board is by far not sufficient. Accordingly, we have designed an arrangement in which the lecturer uses the board as a facility for presentation and pen input, but the content of the display is also projected in sufficient size and luminosity on the wall (cf. Fig.1).

In a first application of this scenario, we have used HTML documents and a standard browser for the presentation of prepared material. An existing whiteboard tool (see below) was used for free pen or finger input. It was important for us to enable the usage of this installation by lecturers who would not do much more than preparing a normal script as a textual basis. We also wanted to avoid having to prepare two version of the script, one as reader’s WWW version, and another one, more in the form of electronic transparencies, to be used for presentation. To achieve this, the script must be highly structured and should contain a high portion of images, diagrams, tables, bullet points etc., and of course problem-oriented components for further free elaboration during the lecture. However, the script should still be readable as a “flowing hypertext” (as opposed to a series of presentation frames). This turned out to be feasible particularly in “know how” areas such as formal languages (e.g., programming in Java or databases and database languages). It was more difficult in areas such as computer architecture and networks, which were taught to our students more as “know what” subjects.

## **3 Integration of interactive pen-based presentation and direct archiving**

To use pen input in parallel with the presentation of prepared HTML or other documents, basically any kind of whiteboard tool which allows for freehand drawing can be used. However, for practical reasons, the following additional functions are important:

- a page structure of the whiteboard with paging (next/previous page) and free page selection in the range of pages created so far;
- a convenient “snap” function for transferring parts of other documents into the whiteboard for further annotation;
- flexible ways of deleting and undoing (e.g. stroke-wise or bitmap-oriented);
- for cooperation with external participants (e.g. a remote guest lecturer), telepointer and synchronization functions are needed.

In a first version of our lecturing scenario, we have used a cooperative whiteboard tool developed in a previous European Telematics Applications project (DEMOS) which complied with all the above mentioned requirements. The most important function this kind of usage which is non-standard for whiteboard or paint tools for is paging. The creation of new pages should happen implicitly, when the presenter say “page forward” when being on the last existing page. Also the insertion of new pages must be as easy as possible. The rearrangement of pages is less important from a specific usability point of view.

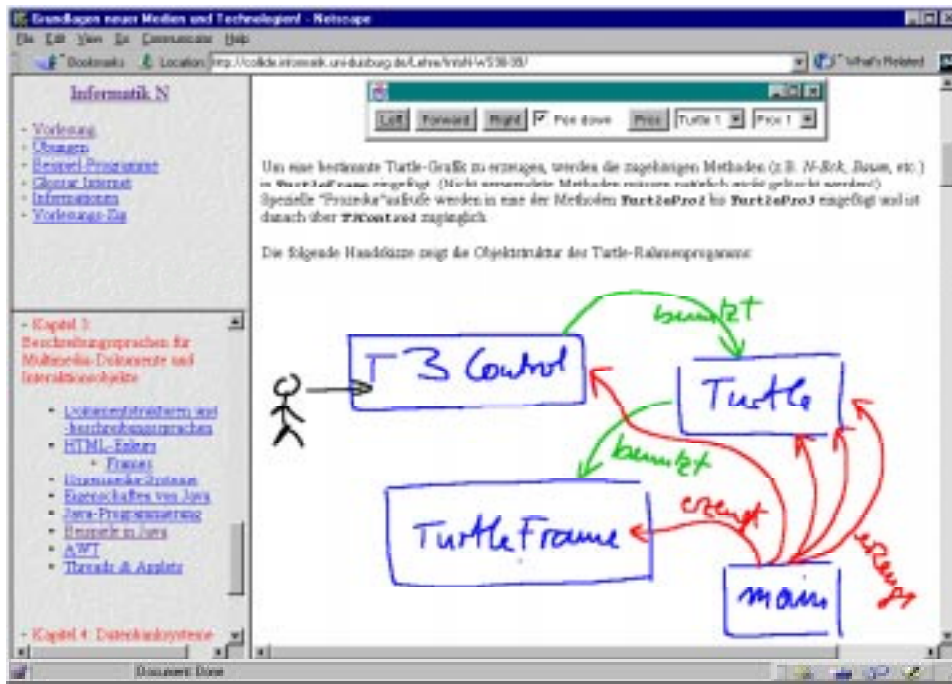


Fig. 2. Integration of free-hand sketches into a WWW script (reader’s interface).

A typical interactive presentation scenario is the following:

1. The lecturer presents a certain concept or problem which illustrated in the presentation script by an incomplete table or diagram.
2. After explaining and problematizing the incomplete diagram, the lecturer “snaps” it into the whiteboard and completes it as a result of a dialogue with the students.
3. When the lesson is over, the whiteboard content is saved and - if considered appropriate - integrated page-wise into the script as a GIF or JPEG image.

In our original scenario step 3 was problematic, since the lecturer could forget to save the whiteboard content, and the integration with the script required some “manual” post-processing. Fig. 2 shows the result of such an action, viewed from the reader’s version of the script. We have been asked (not by our students, but by colleagues) why we did not polish these sketches using e.g. office graphics. However, we believe that our scripts should not be confounded with textbooks. They should be seen as persistent versions of the lectures, and as such they should reflect the individuality of the lecturer and the spontaneity of the lecturing process. This is, in our view, supported by inserting such free-hand sketches (for a further elaboration, see section on “perspectives”).

In a recently completed new version of our software environment, we use a Java-based Whiteboard with essentially identical basic functionality. This whiteboard can be run as a separate application, but it can also be embedded into the script as an applet. At any place, where the lecturer foresees free exposition, a copy of the applet can be inserted on beforehand. The applet content is XML-structured and allows for combining free-hand graphics (strokes) with standard image formats. Often, the lecturer will want to provide an “embed-

ded whiteboard applet” with predefined content, e.g. an empty or incomplete table or diagram (see above). For this purpose, the applet can be invoked with initial content as a parameter. The modified content can directly be stored during the lecture from within the browser and will be immediately available to any reader.

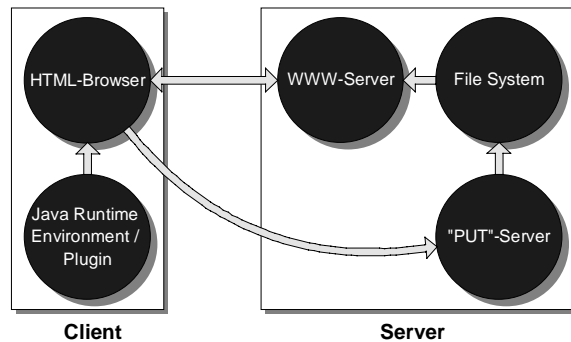


Fig. 3. Function of the Whiteboard Applet.

Figure 3 shows how this functionality is implemented: The HTML-Browser downloads the HTML-page, the Java applet code and necessary libraries. Once initialized and running in a local Java virtual machine, the applet loads whiteboard pages including images from an XML file and displays the content in the browser’s window where it can be modified interactively. It is also possible to share whiteboard pages with other instances of a whiteboard applet (or application) running on different hosts. The storage function is realized by a “PUT” Server (a kind of servlet, i.e. a service module of a WWW server, written in Java) which is responsible for performing the HTML “put” command by storing the edited XML file locally in the file system.

## 4 Evaluation

Our scenario has so far been used in six standard lectures in computer science and psychology over two semesters. The author’s experience results from the following computer science lectures: [L1] Computer science foundations of multimedia and communication systems (an introduction for non CS students, 45 students), [L2] Algorithms and data structures (35 students), and [L3] Computer architecture and foundations of programming languages (30 students).

In lectures L1 and L3 (both introductory for non-specialized and specialized students, respectively), students filled in questionnaires to assess their acceptance of the new form of lectures. In both samples, at least 70% rate the new form of lectures positively. Both groups are dominated by students whose main experience is with classical chalkboard lectures in “mathematical style“ in which they are mainly engaged in copying from the chalkboard during a lecture. As compared to these other lectures, students attribute a better structuring of the content to our web-based electronic lectures. A slight majority tends to favour classical lectures with respect to memorization. A minority of students (up to a maximum of 30% in L1) see the new form of lectures as inferior to chalkboard lectures in an overall judgement. However, this negative judgement coincides almost exactly with a non-acceptance of the computer science lectures from a content-oriented point of view. These figures indicate that we should not believe that we can win students just by using a certain technical scenario in our lectures. Also, there is a small number of students who really believes in the virtues of the traditional copying/listening paradigm.

The experience of the lecturers and free-form responses of students indicate the following main difficulties:

- Lecturers tend to close or discard free-hand elaboration too quickly by switching back to prepared material (as compared to the traditional chalkboard in which the lecturer's writing remains visible for a longer time).
- In general, switching between presentation modes (document viewing, interactive simulation tools, freehand tools) is seen as distracting. It might be better to have more persistent parallel presentation in different modes (e.g., different displays for document viewing and freehand input).
- Presentation of prepared documents tends to be too fast as compared to the "natural rhythm" of writing on the chalkboard or freehand use of a whiteboard. This does also happen with transparency-based presentation, in general. Freehand writing on a whiteboard is still limited in the complexity of what can be displayed at a time.

Apart from what the students stated in the survey, we were interested in the actual usage of the material on the web server. Two different versions of the material were available to the students during the course of lecture. The first one was a pre-existing version from a former course on the same subject that served as a source. This material was revised and extended according during the course of the lecture, e.g. by including material that was freely developed in the electronic lecture hall. On the average, each student accessed each subsection about six times. This number is surprisingly high and shows an intensive usage and practical acceptance of the medium by the students.

## 5 Comparison with similar approaches

Our approach is not based on a previously set up large project. It was initiated when the authors had the chance to influence the design of the technical installations in the lecture hall (with a given budget of approximately 55.000 US \$). In the first trials, available software, partly public domain and partly from a previous project, was used to run the scenario. The development of the new Java-based Whiteboard toolset is a result of about nine person months of student assistants' work. All these developments were driven by practical needs and were always targeted at providing a lean and lightweight solution, not only in terms of the software tools and environment, but also in terms of setting the threshold for a lecturer using this technology as low as possible. In this work we were inspired by the following ideas and examples of existing work:

We were convinced that new learning environments with traditional or non-traditional group settings should be conceived from a "roomware" perspective as formulated by Streitz et al. ([9]). This means that the aspect of physical arrangements and space should be part of the explicit design. Also, specific hardware devices should be considered (here, particularly a large interactive display with pen or finger input), and software should be adapted to this scenario. The roomware approach is closely related to principle of "ubiquitous computing" as developed and exemplified at Xerox PARC in the late eighties ([8], [11]). Also pen-based input technologies for interactive and collaborative scenarios have been conceptualized in this framework. Moran et al. ([7]) describe the principles of freeform interaction with graphical objects in pen-based systems. Although our technical solution, i.e. particularly the Java Whiteboard, does not support all the features of Moran et al., it is based on the same principles and its advantage lies in the better integration with standard presentations (as an applet) and in net-based archiving (servlet functionality).

The "Classroom 2000" project at Georgia Tech ([1]) aims at supporting a range of teaching and learning activities, namely presentation, public and private annotation, and discussion.

As our approach it relies on pen-based techniques and uses large interactive displays to some extent. Particular attention is paid to the requirements of specific phases preparation, recording/transcription, and post-production, both from a student and a teachers point of view. Considering these three dimensions, learning activities (styles) - phases - roles (student/teacher), Classroom 2000 is certainly much broader than our project. Our work is focused on presentation by a teacher and archiving of material for students. Our main concern is a smooth and lean integration of both aspects. Whereas Classroom 2000 uses data-intensive forms of recording (incl. audio and video streams), we rely on compact data and easy-to-handle feedback loops from presentation to archived material. Probably the full scenario of Classroom 2000 cannot be handled by a teacher alone. Also, the aspect of how recorded information from within a classroom is fed back into WWW material is not explicitly described for Classroom 2000. Again, an advantage of our approach is its reliance on Java technology which is crucial for issues of portability and interoperability.

The approach of “Authoring on the Fly“ (AoF) developed by Bacher et al. ([2]) elaborates on the issue of producing multimedia learning material directly based on an actual lecture. As in the full scenario of Classroom 2000, audio and video streams are used in addition to event streams originating from a whiteboard and from a slide presentation. The AoF platform is Unix-based and relies on the Mbone tools ([4]). Mbone’s multicast functionality naturally supports scenarios with communicating lecture halls, on the other hand usability and functionality of its basic tools (incl. a graphical whiteboard and slide presentation) is far from optimal. From a technical point of view, AoF deals with synchronization requirements for the different data streams. It is certainly not a lightweight approach: Mbone is still a burden for ease-of-use, and huge amounts of data must be handled. AoF explicitly addresses the problem of production and archiving. However, its reliance on video data is an obstacle for using content-oriented retrieval strategies and for distribution over low or medium bandwidth networks. In comparison to AoF, our approach is characterized by its better accessibility in everyday lecturing, Java-based standardization, and by its reliance on interactive pen-based presentation. As explained in the last section on perspectives, we intend to extend our approach with post-production facilities that are practically comparable to the ones of AoF but easier to handle and less data-intensive.

## 6 Conclusions

Our experiences with the provision of both presentation material for the lecturer and electronic WWW scripts for the students by smoothly integrating the results of free pen-based elaboration with the scripts are encouraging. The integration of material that is developed during the lecture into the script is supposed to reflect the spontaneity of the lecturing process and to help memorizing the concepts developed in the presentation. However, due to the static nature of the whiteboard content (snapshot), only the results, but not the process of the development are covered by the integrated material in the script. To overcome this limitation, we envision to enhance the whiteboard functionality to reflect also the process of the development of the material, i.e. to represent the single steps the lecturer took in his explanation in addition to the static whiteboard content.

From a technical perspective, to achieve this functionality, three components are necessary. First, the single steps have to be *recorded* together with the whiteboard content, i.e. a protocol of the development has to be produced. This is achieved by collecting time stamps in relation to a whiteboard page and the drawing strokes during the phase of free exploration. This is already the case in the current whiteboard implementation. i.e. every action the lecturer performs in the whiteboard application is recorded. However, this protocol is not yet saved to a file together with the whiteboard content. Second, to reproduce the actions at a

later time, a modified version of the whiteboard that *replays* the protocol has to be distributed as a Java applet together with the script. Third, the whiteboard presumably has to be extended by a component that features the *post-processing* of the protocol in order to obtain an adequate format for presentation, for instance by reducing lengthy periods of inactivity from the protocol or to alter the sequence of actions.

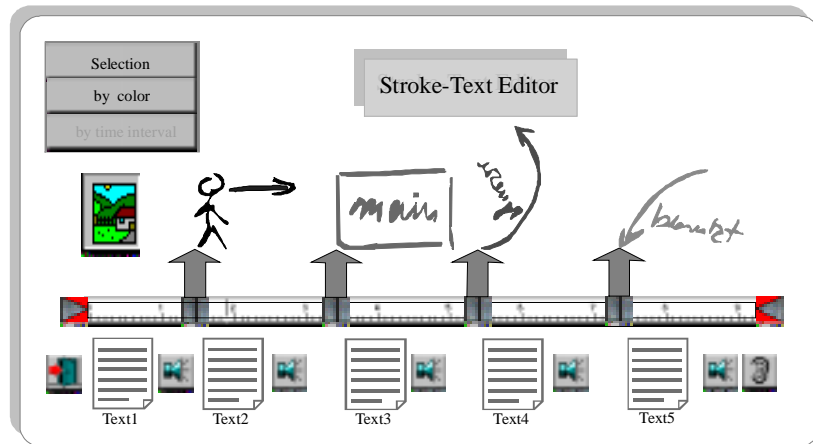


Fig. 4. Postprocessing of whiteboard content, actions and text.

In addition to saving the whiteboard content and production steps, recording also spoken parts of the free exploration is supposed to be useful to retain the implicit or inherent structure of the freehand interaction. The idea is to integrate *text* into the protocol in the post-processing phase. By replaying the protocol, the lecturer can freshen up his memory and then insert text blocks at the appropriate positions in the protocol. For the integration of whiteboard content, actions and text, a stroke-text editor has been projected (see figure 4). While replaying the enhanced protocol, text blocks are presented by a *text to speech-module*. With the projected extension, a professional animated sketch, even spoken in a foreign language, would be available with small amount of post-processing effort.

## References

- [1] Abowd, G.D. Atkeson, C.G., Feinstein, A., Hmelo, C., Kooper, R., Long, S., Sawhney, N., Tani, M. (1996). Teaching and learning as multimedia authoring: the classroom 2000 project. <http://www.cc.gatech.edu/fce/c2000/pubs/mm96> (also published in Proceedings of Multimedia '96).
- [2] Bacher, C., Müller, R., Ottmann, T., Will, M. (1997). Authoring on the Fly: a new way of integrating telepresentation and courseware production. In Proc. of ICCE '97, Kuching (Malaysia), Dec. 1997.
- [3] Elrod, S., Bruce, R., Gold, R., Goldberg, D., Halasz, F., Janssen, W., Lee, D., McCall, K., Pedersen, E., Pier, K., Tang, J., Welch, B. (1992). LiveBoard: a large interactive display supporting group meetings, presentations, and remote collaboration. In Proc. of ACM CHI '92 (pp. 599-607). Monterey (CA), May 1992.
- [4] Eriksson, H. (1994). Mbone: the multicast backbone. Comm. ACM, vol. 37 (8), 54-60.
- [5] Gassner, K., Tewissen, F., Mühlenbrock, M., Loesch, A., Hoppe, H. U. (1998). Intelligently supported collaborative learning environments based on visual languages: a generic approach. In Proc. of the 3rd International Conference on the Design of Cooperative Systems (pp.47-56), Cannes (France), May 1998.
- [6] Hoppe, H.U., Baloian, N., Zhao, J. (1993). Computer support for teacher-centered classroom interaction. In Proc. of ICCE '93 (pp. 211-17), Taipei (Taiwan), December 1993.
- [7] Moran, T.P., Chiu, P., van Melle, W., Kurtenbach, G. (1995). Implicit structures for pen-based systems within a freeform interaction paradigm. In Proc. of ACM CHI '95.
- [8] Stefik, M., Foster, G., Bobrow, D., Kahn, K., Lanning, S., Suchman, L. (1987). Beyond the chalkboard: computer support for collaboration and problem solving in meetings. Comm. ACM, vol. 30(1), 32-47.
- [9] Streitz, N.A., Geißler, J., Helmer, T. (1998). Roomware for cooperative buildings: integrated design of architectural spaces and information spaces. In N. Streitz, S. Konomi, H. Burkhardt (Eds.), Cooperative Buildings - Integrating Information, Organization, and Architecture (pp. 4-21). Berlin et al.: Springer.

[10] University of Maryland (1995-99). Teaching theaters. <http://www.umd.edu/TeachTech/>.

[11] Weiser, M. (1991). The computer for the 21st century. *Scientific American*, vol. 265 (3), 94-104.