Open Hypermedia Educational Environments:
A Feasible Approach To Overcome Some Difficulties

Institut für Informatik — Technical Report No. 91

Chr. Bacher, R. Müller, Th. Ottmann, M. Will
Institut für Informatik
Universität Freiburg
e-mail: {bacher, rmueller, ottmann, will}@informatik.uni-freiburg.de

Abstract

This report describes our efforts to develop an interactive hypermedia environment to enhance traditional education at university level. While we have chosen the field of *Algorithms and Data Structures*, which is a key area in the curriculum for computer science students, the concepts we use are equally valid for other disciplines as well. As the common result of a number of well-tuned projects we establish a computer based teaching and learning environment which provides both lecturers and students with a large variety of information resources, tools and services: They have access to electronic libraries including journals and books, animations and simulations of algorithms and data structures, courseware, collections of programs, tests and notes. The environment in particular includes so–called AOF (Authoring On The Fly) products. These are recorded lectures delivered by using a computer whiteboard and transferred into multimedia documents. We report about the tools necessary to generate and post-process AOF products and discuss in particular the difficulties arising when controlling animations and simulations both in on-line and in off-line mode. We describe what an advanced computer based teaching and learning environment in the future might look like. Our view is not pure speculation but solidly based on considerable experience gained in the past.

1 Introduction

Courses on *Algorithms and Data Structures* are a standard topic both in undergraduate and graduate computer science curricula at universities. About two years ago, we started to develop a computer–based teaching and learning environment for this area at the University of Freiburg. Fortunately, the area of algorithms and data structures is not only covered by a number of good textbooks but also a preferred experimental area for simulation and animation programs enhancing the content of paper-bound documents. There are even a few sample topics covered by animated, interactive texts already available on the Web.

Thus, instead of having to develop everything from scratch, we were able to reuse other people's achievements, regarding both the scientific content and the tools needed to create such an integrated, computer–based environment. From the very beginning, however, we wanted to go beyond merely collecting and linking together documents from various sources by taking into account the paradigms of traditional university education, such as those encountered in class lecturing, while at the same time trying to automate the process of courseware production.

This paper, which reflects our current achievements and future activities, is organized as follows: First, we describe the fundamental ideas of our efforts, which are based on the experiences gained within the last two years. Next, we focus on the automatic production of hypermedia documents from online lectures. This is followed by a section which addresses the problem of controlling animations and simulations both online and offline. Finally, we sketch our ideas to extend the current prototype into a high–end educational environment which fully integrates heterogeneous technologies such as multimedia presentation, CAI, distance teaching and learning to support both lectures and students to the greatest possible extent.

A brief sketch of the prototype system

Currently the teaching and learning environment contains:

- a textbook (700 pages, postscript),
- animations and simulations (about 40 running under X Windows),
- recorded lectures (about 10 hours),
- assignment collections,
- templates and class libraries for programming,
- organizational and administrative information,
- search engines,
- access to electronic libraries containing computer science topics,
- bibliography: database of cross–references (BibTEX).

2 Principles for producing reusable hypermedia courseware

2.1 Rely on other people's work

We were convinced from the very beginning that it is hopeless to establish a rich environment even for a limited subject like *Algorithms and Data Structures*, if everything, i. e. in particular all animations and simulations as well as courseware of the traditional type, has to be developed from scratch to cover the whole area. Therefore, we started to systematically collect all kinds of relevant material available either offline (on CD-ROM or disks) or online (via Internet services) to either download or use it as interactive text. Although we were able to find a substantial amount of useful material for both lecturers and students, we ran into the well–known problems of platform dependence, difficulty to customize material to specific needs, to control simulations over the net, to replay animations and many more (not to mention nontechnical issues such as copyright).

2.2 Rely on teaching experience and maintain as much as possible from good lectures

Traditionally, course preparation is done by gathering scientific material (books, journal articles etc.) from various sources. This material is consulted to produce own course material (slides, overhead transparencies, course notes) and made available to students. If the course was given before, teachers usually want to re–use a large quantity of the material by editing it to keep it up–to–date. Since lecturers are familiar with this procedure, it is advisable to adapt it to the production of hypermedia courseware.

University teachers usually have gained a lot of experience in explaining difficult concepts to students in traditional classes. Unfortunately, most of this valuable information is lost after the end of the class though many students would appreciate it very much if they could retrace a lecture when using printed material to review a complicated issue within a specific area. This observation was the starting point for our *Authoring-on-the-Fly* project. Our approach is based on the conventional lecturing paradigm. However, instead of traditional tools such as blackboard, chalk and overhead transparencies, the computer, more specifically an electronic whiteboard, is directly used for teaching. We record everything (audio, video, whiteboard actions) synchronize the corresponding data streams, transfer them into multimedia documents and store them in a hypermedia database for offline use.

2.3 Share your efforts with your colleagues

Since the field of algorithms and data structures is a standard topic at each computer science department, we started to cooperate with our colleagues at other universities in two ways: Using the teleteaching facility of the MBone whiteboard *wb*, we both exported and imported lectures delivered via the MBone tools. Furthermore, we encouraged guests and colleagues to cover their favorite topics by delivering lectures via the computer, using the whiteboard *wb*, to transfer them into AOF products, using our own recording software. Thus, we have not only covered a series of standard topics but also a new kind of textbook on *Principles of Algorithm Design* including papers, AOF–products, simulations and animations that will appear soon as the outcome of this effort.

2.4 Do not reinvent the wheel, but develop your own tools, if necessary

When looking for an electronic substitute for blackboard and chalk we found the whiteboard *wb* within the MBone toolset. We have used it not only to deliver lectures for our own students; together with the other MBone tools *vat* and *vic* we were able to transfer the lectures to other universities (Mannheim, Munich, Frankfurt, Karlsruhe) as well. This is a welcome side-effect of the multicast facility of the MBone tools. On the other hand, *wb* has a number of deficiencies which make it difficult to use. Moreover, no appropriate recorders and editors were available for lectures delivered using *wb*. This was the reason why we started to develop our own tools, in particular an enhanced whiteboard *AOF-wb* (see [BaO96] for more details). Unfortunately, as a consequence of using our own whiteboard instead of *wb* the teleteaching facility is lost. Therefore, we just started a new project to fully integrate AOF–wb into a set of tools for teleteaching and recording of lectures.

2.5 Use top–technology for prototypes

Compared to the life cycle of modern computers, it takes a long time to create multimedia documents, simulations, animations and courseware, as well as to implement appropriate tools. Therefore, we started to develop our recording software, viewers and editors for AOF products as well as the enhanced whiteboard (AOF–wb) on a high-end graphics workstation. Even though we cannot expect students to privately own computers of this category, multimedia PC's are rapidly approaching the computing power of workstations. Therefore, once the feasibility of the AOF approach has been demonstrated, it seems reasonable to develop (at least) PC– based receivers and viewers.

2.6 Use open systems

Most of the courseware currently available comes as a closed package which is difficult to customize for specific needs. Teachers, however, want to choose from many sources.

The material used to prepare a lecture, which consists of heterogeneous documents such as text in various formats, graphics, videos, audio, animations, simulations, hypertext, and hypermedia documents, can be gathered from various information sources consisting of multimedia databases, second–generation hypermedia servers, which make it possible to extract structured information, electronic libraries, locally stored documents and, to a certain extent, offline publications such as books and journals[1]. This material is used to create new documents such as lecture notes, edited (in case of bringing own documents up–to–date), converted into other formats, e. g. turning an animation into a MPEG movie for platform independence, or structured to meet specific needs. Some of the documents will be used in online lectures (see section 3) to extend the abilities of conventional tools such as overhead transparencies, slides or chalk and blackboard.

These are the reasons why we chose to use a second-generation hypermedia system (HyperWave) to store and link all documents (including AOF–products) in our teaching and learning environment.

In the following, we will now briefly sketch the current status of our developments and describe the problems yet to be solved in order to fulfill future needs. The scenario which we will describe is soundly based on past experience and already operational in most essential aspects in a prototype version

It is important to distinguish between content supply and the presentation of information. While the former is usually done by the lecturer and cannot be automated, since it involves the compilation of information from various sources and the structuring of ideas according to varying needs, the latter is basically the integration of various document types into a scheme, which both takes into account and extends conventional studying techniques to support the student to a maximum degree. These are roughly the same independent of the research area; they are used to design the basic framework for an educational environment. Its structural components, which will be discussed in more detail in section 5, are compiled into hypermedia documents along with the content supplied by the lecturer, and which includes multimedia documents generated from online lectures. They can be made available in local or wide area networks or used for the production of CD–ROMS for offline use. Additionally, they can be inserted into electronic libraries either as a whole or partially (e. g. only the document generated from the online lecture). Either way, it is possible — assuming that copyright laws are not violated — to re–use parts of the hypermedia document, since its structure is visible to the user. Thus, due to the possibility to integrate various media types to create new content, the production of electronic courseware is facilitated considerably and does not require more effort than a well–structured lecture delivered conventionally. The generation of courseware is thus a by–product of the lecturer's course preparation, since the integration into a courseware scheme reflecting the needs of students is largely automated.

[1] Various research projects in the area of searchable, distributed scientific libraries lead to the assumption that the material available electronically will steadily increase, apart from journal articles often made available to the public by their respective authors

3 Production of hypermedia documents on the fly: current possibilities and future options

In [OBa95], we successfully demonstrated that AOF is indeed feasible if the whiteboard *wb* of the MBone toolkit is used. Although a lecturer can perform actions using *wb* which are difficult or even impossible when using blackboard and chalk (like moving objects around copying, overlaying, form-filling etc.), he readily notices that *wb* indeed lacks many desirable features. Therefore, we started to develop our own enhanced whiteboard (AOF–wb) [BaO96]. However, the interaction with large distributed hypermedia information systems as well as the integration of animations and simulations into lectures are not solved satisfactorily yet even using the enhanced AOF–wb. Thus, we will briefly describe what *Authoring on the Fly* might look like in the near future. In addition, we will talk about the technical problems to be solved in order to turn these ideas into reality. Note that our results are based on a considerable amount of experimental experience. It is natural to structure the AOF-process into three phases which are all affected by the change of technology.

3.1 Preprocessing

The main work in this phase is collecting, adapting and converting course–related material to prepare the lecture as well as creating or editing own documents, which are presented and edited during the lecture. These procedures are partially supported by the AOF–wb's ability to manipulate documents consisting of text and graphics. The question is, however, how to support the collecting and manipulating of multimedia data for lecture preparation. In order to access all the documents used for presentation in online lectures, we will implement a gateway to the underlying second–generation hypermedia server (HyperWave) as well as an appropriate client to directly access the needed documents. These will automatically be converted into the internal format of the enhanced whiteboard. For adaptation of HTML documents, a HTML formatter is used. This has the advantage that the ability to record and replay lectures is preserved. Animations, simulations and movies can be stored appropriately to start them during the lecture by simple interactions with the AOF–wb using external viewer software.

3.2 Online Scenario

The AOF-wb currently allows to deliver lectures and to record them for later replay using our own custom viewer software. It is also possible to use the MBone wb along with our multicast recording software for subsequent offline use. In a related project, however, we are working on adding a whiteboard stream transmission mechanism via MBone to our own enhanced whiteboard. This includes the implementation of MBone receivers for different platforms and defining protocols for the communication between the AOF–wb and the receivers. Section 4 will focus on the related question how to control animations and simulations over networks. The transmission of whiteboard streams via MBone makes it possible to separate the author's and the students' view of the whiteboard. Thus, we will be able to implement a complex authoring station without overloading the student's view with unnecessary information.

3.3 Postprocessing

In this step, editing and indexing of the recorded lecture as well as its insertion into the HyperWave database is accomplished, which includes linking it to other related documents. Thus, we have developed a tool to edit the audio, video and whiteboard streams in order to cut out pauses or mistakes while preserving the data stream syncronization (see [Maa96]). We currently use the whiteboard stream to index a recorded lecture such that every change to a new page or slide is interpreted as the beginning of a new section. This makes it possible to generate a table of contents in HTML, which is automatically inserted into the HyperWave server along with the digitized lecture.

In order to enhance editing and indexing, our own AOF tools have to be enhanced with speech recognition. Thus, the editor would present a transcription of the lecture in order to facilitate the process of finding possible errors. Even without this feature, the editor can automatically detect long pauses. Also, a glossary can automatically be created from the audio stream of a set of lectures using a speech recognition tool.

4 The Problem of Controlling Animations and Simulations

Animations and simulations have become a substantial part of computer–supported education because they support comprehension of methods and algorithms by supplying step-by-step execution. Animation packages are available for a wide range of educational topics, in particular for the field of *Algorithms and Data Structures* in computer science (e. g. [Gl96]). A variety of tools, such as XTango [St92], Polka [St95] or Zeus [Br+92] support the generation of animations while adding control commands to implemented algorithms. As a consequence, the variety and amount of available animations and simulations is rapidly increasing. In order to integrate any kind of animations into an online lecture as well as the resulting hypermedia document to be inserted into the database server, we want to make these available to the lecturer within our educational environment.

In the following, we briefly describe the three main scenarios along with their respective drawbacks appearing in the discussion of controlling animations. We will then talk about the present solutions and possible future ones in respect to these scenarios.

4.1 Animations in Differing Scenarios

We can distinguish between various scenarios when considering animation control, namely the *online scenario* (see section 3.2), the *replay scenario* and the *offline scenario*, which will be discussed in this section.

In the online scenario, the lecture will be transmitted to the connected host via the multicast protocol. All documents used during the lecture, including animations and simulations, have to be displayed on the remote host. This raises a fundamental issue: should the application only be displayed or should it be executed as well on the remote host? The former leads to the problem of restricted network capacities, since the transmission of an application movie or the exportation of an application's display requires a substantial bandwith. The latter causes the problem of multiple platform execution, since every single application presented must be explicitly ported to all platforms (server and clients) used for the teleteaching session. Additionally, the animations on the remote hosts must be synchronized both with the animation running on the lecturer's host and with the remainder of the lecture, i. e. the audio, video and whiteboard data streams. Depending on the current load on the network or on the remote hosts, the synchronization can slow down the lecture delivery as a whole if handled automatically. Otherwise, a feedback mechanism is needed to inform the lecturer about the run time state of the currently active animation in order to allow him to keep in sync with his comments.

In the replay scenario, the data streams which were recorded and automatically converted to form a single multimedia document are synchronously replayed in a special viewer. During this process, all animations used during the lecture must be presented exactly in the same form and at the same speed as in the live lecture. This also means that interaction with any displayed application is neither necessary nor required in this context. In order to enable the offline use for multiple platform types, platform independence is mandatory.

The use of animations within educational environments, where digitized lectures as a combination of animations, simulations and AOF products constitute an integral part, is called the offline scenario. In addition to the presentation as movies without interaction, all animations used by the lecturer should additionally be inserted as simulations, i. e. as real interactive applications, since learning is much improved if it is possible to start animations with own parameters and to interact with the algorithm. As in the two previously mentioned scenarios, platform independence is required, whereas synchronization is not an issue, since the animations run as stand–alone applications.

4.2 Possible Solutions For Animation Control

Up to now we circumvent the problem to control external applications, since all animations used within the lecture are installed on all connected hosts beforehand. Thus, the platforms used in the online and the replay scenarios must be identical. In order to control the remote applications, the lecturer sends start and stop commands via multicast. This obviously is not an appropriate solution, since an adequate fine–grained resolution is required to stop at a specific (discrete) moment.

A more promising solution is the generation of a Quicktime or MPEG movie from the animation itself. This guarantees the platform independence in an elegant way. Even though it is a possible solution for the replay scenario, it is not suitable for either the online and the offline scenario, the former because of net overload, the latter due to the lack of interactivity.

Both the use of Java to program the animation and the Mocha model (see [Ba+96]) solve the platform independence problem. The Mocha model is a distributed system with client–server architecture, which separates the algorithms running on a remote server from the graphical user interface and algorithm visualization modules. The latter are implemented in Java and run on the calling client. As elegant as this solution may appear, an important drawback is that the pool of existing animations cannot be reused, since everything must be reimplemented from scratch.

Shared Displays are a variation of Shared Applications used in computer supported cooperative working environments. The main difference is that the former do not support interaction on the remote clients. An implementation of this model are X clients communicating with different X servers to export their display. This is not a solution for the offline scenario since interaction is needed. The use of shared displays in the online scenario leads to a high network load and can slow down the lecture as mentioned above. Additionally, in order to replay a recorded lecture, it is necessary to protocol the corresponding X session, which requires a mechanism to record the protocol messages between client and server. This mechanism must be able to replay the recorded stream, i. e. it has to open a connection to the X Server and to send all recorded messages. As far as CSCW environments are concerned, recorded X sessions have been sucessfully used (compare [Ga+96] and [Ga96]). The resulting drawback in the context of shared displays is that an X server is required on all hosts to be considered. In particular, it is a nontrivial task to transform the recorded data stream of protocol messages to generate a similar graphical user interface on other window systems such as Windows(95/NT).

As with the use of Java or Mocha, the porting of the used application is not an appropriate solution, since it restricts the number of usable animations to those available in source code and suitable for compilation to the required platform.

In conclusion, no straightforward solution is available which encompasses all three scenarios for animation control.

5 Optimal Support of Students – The Integration Process

Let us now follow more closely how course material is gathered and compiled for courseware production. This process is closely related to course preparation and lecturing, which was described in section 3. Usually, the list of courses, possibly structured according to course type, is compiled first, with a description whose author is the respective lecturer. It is made available to the public as hardcopies before the semester starts. Likewise, an electronic repository is generated for each course, which will be filled with content later on, starting with the course description. A HTML form is used to get the necessary data, from which both hardcopy and electronic version can be generated. Since the procedures for preparing a course are similar even though its content may differ, we will focus on single course preparation.

5.1 Basic Course Material

The lecturer gathers all kinds of documents from various sources which he intends to use as a background for his own course preparation or as a recommendation for further studies. It is common to include relevant bibliographic data in a database which can be used for various purposes[2] like searching for specific publications or referencing documents in scientific papers. Thus, access to bibliographic databases as well as the integration of locally used bibliographic sources eases the compilation of recommended literature for students as well as individual research. Thus, if the bibliographic data is supplied with keywords, a simple search automatically generates a core list of relevant material, which can then be edited by the lecturer and made available as a complement to the course description. Additionally, a local database of all the primary documents for a course is automatically generated, which requires that basic bibliographic data (author, title, publisher) is supplied or inferred from other bibliographic information upon insertion into the local database server. Thus, the documents which are available on–line form a searchable electronic bookshelf. A course syllabus, which is very valuable for the students as a guideline, completes the preliminary course information available. It includes the dates reserved for lab sessions and tutorials, which are provided to consolidate the subject area. Note that the basic course material is not constrained to textual documents, as it is the case in conventional educational settings, but instead includes all kinds of educational software, especially high–quality lectures from experts in specific areas (AOF products), computer simulations or VR models. In case of a lecture delivered within the current semester, it is inserted into the database, along with external helper applications, graphics or animations, and linked to the syllabus.

5.2 Interlinking multimedia data

The course material, which is structured and organized according to the lecturer's ideas, constitutes a network, which may be represented in an acyclic graph. The individual documents are related to each other both syntactically (e. g. parent–child relationship) and semantically (e. g. comments or additional data on a document). The structure of the graph reflects the concepts of the lecturer, which he develops and amplifies while developing the course. Therefore, an editing tool to create graph structures has to be provided, which permits document and data organization, including atomic and composite (generic) nodes as well as typed links at various levels.

It may well be, however, that a student's view on the course structure may locally differ, either because own material is added or additional links are established, or because a reduction of the graph's

[2] Since scientific publications are most often written in LaTeX, a bibliographic database in BibTeXis commonly used.

complexity is favoured to enhance the student's understanding. Thus, local views on the course material have to be enabled, which can only be edited by their respective owners, without, however, modifying the original structure as designed by the lecturer.

5.3 How Can Learning Be Promoted?

As we have seen, students can reinforce their knowledge by solving assignments, which have to be turned in regularly. In computer science, theoretical assignments alternate with programming tasks. Descriptions of individual assignments are handed out, inserted as read–only documents and associated with the syllabus. Each student is then assigned a space where it is possible to create documents containing the solutions to weekly assignments, which are then sent to the tutor assigned upon registration[3]. The tutor then edits the solved assignment by adding comments and a grade, which is then added to the student's record. Similarly, programming tasks are made available to the tutor both in source code and as executable files[4].

Individual tutoring sessions serve to discuss specific problems[5]. Upon registering, every course participant is put onto a course–related mailing list, which is useful to ask relevant questions, to contact lecturers, tutors or fellow students, or for class announcements. A mail archive, which is automatically archived, can be consulted by course participants at a later stage.

So far, we have not talked about group collaboration, which is very common for larger assignments. Even though there are many tools that ease communication, it is imperative to meet regularly at all stages of a project. Nevertheless, group planning can be supported by shared tools; thus, an adaptation of the whiteboard discussed in section 3 could be used to jointly create and merge diagrams. To support this, an integration of CSCW tools into the education environment is mandatory for student support.

5.4 Annotations at Various Levels: A Fundamental Problem

If students use the recommended material, possibly enhanced by other documents found while searching in hypermedia databases, it is necessary to be able to annotate documents at various granularities (e. g. whole documents, individual sections, paragraphs) , to create associative link structures, to write own summaries or to draw diagrams, just to name a few. All this significantly enhances the studying process. For new documents, it is easy to provide specific editors, but making annotations at various levels has to guarantee the integrity of the corresponding document to avoid keeping multiple physical copies. The process of annotating electronic documents, however, should not only reflect and extend the conventional mechanisms of inserting bookmarks between pages (or sections) of a document, to write cards and insert them into file–card boxes, to make notes on the margin of a written publication, to underline or highlight arbitrary small parts (subsections, paragraphs, single words), but also has to guarantee document integrity to avoid keeping multiple physical copies[6]. A first try would be to allow the students to highlight an area in a document and let the user write an annotation, which is

[3] This is done either by e–mail, by linking the document with the solved assignment(s) to the tutor's repository with automatic notification for sender and receiver.

[4] This requires the students to know which platform the tutors use, which can be agreed on upon registration

[5] Even though this could be done in form of a newsgroup or online discussion, this would only be advantageous for remote studying.

[6] Note that editing electronic documents means altering them, so all annotations have to be be kept separate from the original documents in order not to violate copyright laws.

then linked to the original data[7]. However, since the concept of annotating is closely related to the semantic context and logical structure of documents, it is not enough to record the area which was highlighted. Suppose that a document is edited or replaced by a new version. Then, the annotations of a student are no longer valid if only the highlighted areas were recorded. Although version control mechanisms may be feasible for locally stored documents, it is a problem if the original version is no longer available, unless the annotations are hard–coded into the document, which does not preserve its integrity, apart from not being possible for remote documents. This constitutes a fundamental problem yet to be solved.

6 Conclusion

As we have seen, the production of multimedia documents from online lectures and the integration of heterogeneous document types into a scheme to create hypermedia courseware can be seen as parallel processes. A by–product of our approach is the transmission of lectures to remote locations, such that the tasks of lecturing, teleteaching and the production of high–quality course material can be merged. Furthermore, due to the separation of content and presentation, the lecturer's tasks of preparing and delivering a course unit is largely similar to the traditional style of lecturing, while the presentation and integration of hypermedia documents is automated to the greatest possible extent. At the same time, the integration of interaction mechanism into the courseware not only encourages discussion processes and feedback between students and lecturers, but also the possibility to learn from previously answered questions. In this way, the educational environment is dynamically evolving and reflecting the needs of lecturers and students alike. Also, due to its modular structure, existing courseware can easily be enhanced by adding content or adapted to specific needs, and existing parts can be re–used in other contexts. Finally, the production of courseware is detached from its delivery, thus enabling both online and offline use. Thus, while our approach is based on traditional lecturing mechanisms and past experiences in computer–based education, the disadvantages of previous approaches in this area are sucessfully circumvented.

References

[AA96] Abowd, G. D. et al.: *Classroom 2000: Enhancing Classroom Interaction and Review*. Proceedings of the ACM, Multimedia '96.

[Ba+96] J. E. Baker et al.: *The Mocha Algorithm Animation System*, Proceedings Advanced Visual Workshop, 1996.

[Br+92] M. H. Brown: *Zeus: A System for Algorithm AnimationMulti–View Editing*. IEEE Workshop on Visual Languages, October 1991.

[Ga+96] L. Gannoun et al.: *Recording and replaying of an X–Session: An Approach for an asynchronous interaction method*. Proceedings of ED–TELECOM 96, Boston, MA, June 1996.

[Ga96] L. Gannoun et al.: *Asynchronous interaction method for a Remote Teleteaching Session*. Technical Report, Insitut Eurecom, Sophia Antipolis, France, 1996.

[7]Of course, it should only be visible by its author, since it constitutes a personal view, but it may be shared by others on demand.

[Gl96] P. Gloor et al.: *Animated Algorithms: A Hypertext Learning Environment for Introduction to Algorithms* (CD–ROM). MIT Press, Cambridge (MA), USA, 1996.

[Maa96] G. Maass: *Editor für Multimedia–Dokumente*. Master's Thesis, Universität Freiburg, Institut für Informatik, 1996.

[Ma96] H. Maurer: *Hyper-G now Hyperwave. The next generation web solution*. Addisson-Wesley Publishing Company, Harlow, England, 1996.

[OBa95] Th. Ottmann, Chr. Bacher: *Authoring on the Fly*. Journal of Universal Computer Science, Vol. 1, No. 10, October 1995.

[BaO96] Chr. Bacher, Th. Ottmann: *Tools and Services for Authoring on the Fly*, EDMEDIA '96, Boston, June 1996.

[Ott95] Th. Ottmann: *Hochschullehrer als Autoren und Nutzer von Lernsoftware*, Universität Freiburg, December 1995.

[LMa94] J. Lennon, H. Maurer: *Lecturing technology, a future with hypermedia*. Educational Technology, pp. 5–14, April 1994.

[St92] J. T. Stasko: *Animating Algorithms with XTANGO*. SIGACT News, Vol. 23, No. 2, 1992.

[St95] J. T. Stasko: *Polka Animation System*. Available online at http://www.cc.gatech.edu/gvu/softviz/parviz/polka.html.