Munics: Multimedia for Problem-Based Learning in Computer Science

Pamela Tröndle[1], Heinz Mandl[1], Frank Fischer[1], Jürgen Hartmut Koch[2], Johann Schlichter[2], and Gunnar Teege[2]
[1]*Ludwig-Maximilians-Universität München, Germany:*[2]*Technische Universität München, Germany*

Key words: Problem Based Learning, Learning Environment, World Wide Web

Abstract: Recently, German university education in computer science in Germany has not been adequately meeting the needs of computer science professionals. Knowledge remains passive, not easily transferred and applied to actual problems. Our project involving problem-based learning in computer science aims to improve this situation at Technische Universität München by supporting realistic case studies as part of students' work. Our approach exploits recent trends in using computers and the Web for improving learning and teaching. We have developed an integrated environment—Munics (Munich Net-based learning in Computer Science), which supports students and teachers working on actual case studies in computer science. We want students to learn how to apply their factual knowledge in future workplace situations.

1. THEORETICAL FOUNDATION FOR THE MUNICS DESIGN

The concept of problem-based learning views learning as a constructive, self-directed, active, situated, and social process. Situated concretely, it balances instruction by a teacher or another expert, with construction realised by the learner. Deriving from these demands, the design of a learning environment should meet the following requirement: Problem-based learning must be set in authentic, multiple, and social contexts surrounded,

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35502-3_19
S. D. Franklin et al. (eds.), *Building University Electronic Educational Environments*

in turn, by an instructional context (Mandl, Reinmann-Rothmeier and Gräsel, 1998).

1.1 Cognitive Apprenticeship

The design of the learning environment follows the design criteria arising from the Cognitive Apprenticeship model by Collins, Brown, and Newman (1989). Cognitive Apprenticeship belongs to the situated-learning-based instructional models, requiring a problem-based learning process. This means the learning environment is supposed to present realistic problems as authentically as possible. This aims to create a suitable context for later transfer of the acquired knowledge.

Authentic activities and social interaction are the main methods by which Cognitive Apprenticeship, just as in craft apprenticeship, aims to introduce the learner to an expert culture. From the very beginning the learner starts out with global tasks to assure that he or she will build up an internal concept of what is to be learned.

In this problem-solving process, the learner receives certain support from an expert in order to correct and enlarge the learner's current knowledge. The first step is a period of cognitive modeling. This means that the cognitive process and strategy leading to a professional solution of a problem are made accessible to the learner (*modeling*). Starting out from this point, the learner deals with a problem alone. Meanwhile, the expert provides active help and hints, especially as support for solving problems that are somewhat beyond the learner's scope (*coaching* and *scaffolding*). The more the learner's knowledge and abilities grow, the more the expert will let the learner work alone and will gradually withdraw this support (*fading*). The learner's communication with the expert or even with other learners, which takes place during the whole learning process, plays an important role: it ensures that the cognitive process and problem-solving strategies are articulated, thereby becoming an object of reflection and, if necessary, improvement (*articulation* and *reflection*). In the long run, the learner is enabled to explore problems actively and to solve them independently (*exploration*).

Within a complex learning environment like Munics, these elements of Cognitive Apprenticeship do not have to be enforced completely and need not follow a certain order; they can be combined and mobilized as needed.

During this whole process, learners themselves decide about their procedures; the learning environment does not prescribe specific steps or specific subtasks. The learner must therefore proceed actively, in a self-directed way, just as suggested by the principle of situated learning.

1.2 Cognitive Apprenticeship Performed by Multimedia

As Reinmann-Rothmeier and Mandl (1997) point out, multimedia are particularly suitable for realising active, situated, self-directed, and cooperative learning. Therefore, any instructional model that advocates such a learning process aligns well with the use of multimedia. Consequently, Cognitive Apprenticeship is a highly appropriate model for creating multimedia-based pedagogy.

1.3 Previous Approaches

Recently, several attempts have been made to provide electronic support for problem-based learning. Typically, this support consists of multimedia-based descriptions of a problem and some simple tools (like a text editor with some templates) for working on this problem (Albion and Gibson 1998). In some cases, there is additional communication support for interactions among learners and teachers (Nuthalapaty et al. 1998). Few projects go further than this yet. Grooters and de Vries (1998) describe a much more comprehensive concept for supporting problem-based learning, and Guzdial et al. (1996) present an environment which includes computer support for synchronous and asynchronous collaboration for learners and teachers.

2. THE CASE STUDY WITHIN MUNICS

Problem-based learning in Cognitive Apprenticeship requires working on a problem that is as close to the real world and as authentic as possible. Therefore, Munics is centred around a realistic case study representing a typical problem in computer science: the inefficient distribution of information within the larger organisation of a university department. (We plan in the future to offer students several different case studies within the same Munics learning environment). The students' concrete task is to solve this problem using the methods that a computer professional would use at his or her workplace. In the process of dealing with this problem, students are expected to acquire the competence to handle complex problems and design distributed systems and informational networks.

The aforementioned case concerns a university department. Due to the size of the department, the distribution of information and knowledge can be managed only by the deliberate use of communication and information technology.

One departmental task is organising the educational courses to be offered in the following term. This organisation involves the following duties: coordinating different courses offered by the different lecturers in the department, and ensuring that all courses offered are announced correctly in the different lecture plans and on the WWW. Executing all this necessary planning and administration requires contacting and coordinating a number of different persons and institutions, within and outside the department. The problem at the moment is that these organisational procedures take up much time and cause much extra work. Until now, these organisational processes were carried out without any systematic use of information technology, which explains the exaggerated amount of time spent and the number of incorrect announcements which still appeared.

The head of the department now commissions a computer professional, whose role is taken by the learner, to solve this problem by using information and communication technology efficiently. The end result must not waste too many personnel of fiscal resources, and it must be acceptable to the staff.

3. CONCEPT

Munics consists of the following two components: problem context and instructional support. Both were designed according to the theoretical requirements of the instructional model of Cognitive Apprenticeship.

3.1 Problem Context

According to the instructional background, the learning process starts at the beginning with a global problem and a global task both of which the learner must accommodate within an extensive case study.

The problem context is the part of the multimedia module that contains all the important facts about the case study. However, this information is not provided in a ready-made display. Rather, the context of the problem is designed for interactive use. Learners themselves have to decide which of the offered topics they need more information on in order to solve the problem. We call this multimedia presentation *Interactive Problem Context*.

Conforming to Cognitive Apprenticeship, learners are stimulated to request actively the information they need, instead of just absorbing passively what is presented (*exploration*).

Moreover, situated learning demands that the learning process be set in authentic situations. To meet this requirement, multimedia are used to represent the problem and its context to learners as realistically and as

similar to the professional situation as possible. Learner themselves have to navigate within the "multimedial university department" as a computer professional would probably navigate within the real university department. While talking to different members of the department, asking them questions, and taking a look at different problem-related objects, learners try to gather all the information they consider necessary to solve the problem.

3.2 Instructional Support

Cognitive Apprenticeship offers a number of instructional possibilities to support the learner in a complex learning environment. The following aspects are particularly suitable for multimedia realisation.

3.2.1 Cognitive Tools

Computer programs designated as Cognitive Tools are specifically created to support the process of handling complex information, for example, programs for the visualisation of information.

In the Munics learning environment, such tools are made available to learners as suitable coaching and scaffolding for their cognitive activities during the whole working process.

The most important tool is the Modeller Tool, which enables the modelling, analysis, simulation, and visualisation of the flow of information within an organisation. It thereby offers assistance in drafting, planning, simulating, and analysing a solution to any problem resulting from an inefficient information flow.

Munics also lets student use external applications. Being able to work with familiar electronic tools should promote creativity and unrestrained problem solving.

3.2.2 Expert Modeling

Expert modeling aims to enlarge learners' knowledge, to model their cognitive concepts and strategies, and to enhance their learning processes.

Expert modeling can be realised in different ways. It can be the presentation of a complete professional solution to the problem, or it can be an expert demonstrating problem-solving steps, clarifying cognitive concepts, and explicating working procedures for learners. A given professional solution can be created either for the learner's future use, or for comparison with the learner's proposed solution. Depending on the learner's concrete task, Munics mobilizes different types of expert modeling.

3.2.3 Cooperation (Computer-Based Tutoring and Chat Groups)

One of the basic principles in Cognitive Apprenticeship, as well as in constructivistic models of situated cognition in general, is cooperative working and learning. Social interaction and cooperative learning is translated into action by computer-based tutoring just as much as by computer-based chat groups.

Computer-based chat groups and learning groups engage students' social interaction in order to work together, to discuss different solutions to the problem, and different problem solving strategies, to get important tips, and to argue about difficulties.

A tutor also provides for an exchange of ideas about the working process. It may also be his or her role to preside over the discussion of a chat group, to give out invitations to the chat group, and to manage different ideas about possible solutions.

Discussing one's own working process with a tutor or a chat group requires one to articulate problem-solving steps, as well as to reflect upon them. Moreover, this has a modeling effect on learners' own cognitive processes.

3.2.4 Background Information as Hypertext

The problem which has to be solved within the Munics learning environment was chosen to correspond with lectures on "Distributed Systems" and "CSCW Systems". The Department of Computer Science currently offers supplementary, online, lecture notes. Since these notes provide useful guidance and background knowledge for solving the problem, they are organised as hypertext, and students can access them during the whole problem-solving process for support in actively solving the problem.

4. MUNICS - A MULTIMEDIAL, NET-BASED LEARNING ENVIRONMENT

Basically Munics consists of two large parts: A multimedia presentation (the Interactive Problem Context mentioned in 3.1) and a number of applications to support the students in cooperating and solving the problem. (These are the Cognitive Tools mentioned in 3.2.1.) In this section we describe in detail the Interactive Problem Context and the most important application, the *Modeller Tool*.

4.1 Interactive Problem Context

The problem context contains all important facts about the case study. All information that the students need in order to solve the exercises can be found here. Most currently available learning environments present this information in a very structured way. For Munics, we chose a different approach.

4.1.1 Information Acquisition as Adventure

As already mentioned, it is essential that the process of gathering information required for the problem solving process be kept as close as possible to real life. We therefore designed the problem context as an "adventure". The students collect not treasures, but information they need to solve their exercises.

Very similar to the situation of a computer professional in the real world, the students conduct interviews. Every interviewee explains his or her very personal view of the problem; some details may be useless, some others may be important. It is the students' job to ask the "right" questions and to realise which information is really important for solving the problem.

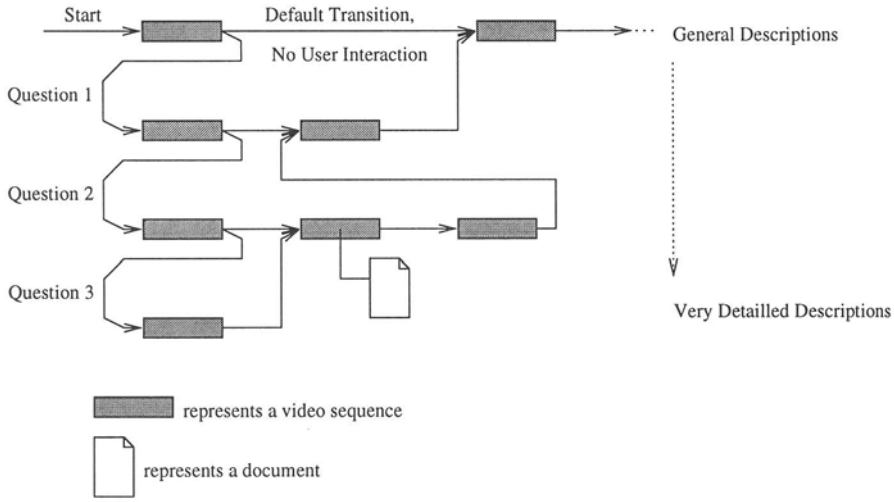
We use animations, simulating movement around a chair, to enable students to choose interview partners and to start interviews. This simulated movement is very similar to navigating in a VRML world with today's popular VRML viewers.

4.1.2 Structure of an Interview

Each interview consists of a set of video sequences. An interview begins with the first video sequence. Whenever a video sequence is finished, possible questions are displayed in a text area. The students can "ask" a question by clicking on it. Then, according to the question selected, the next video sequence is played.

Sometimes the interview partners not only explain details but also present some extra material to the interviewer, for example, documents like booklets, charts, etc. The students can collect these documents and store them in the document repository for later use.

Imagine an interview as a set of trees of video clips. Each topic in an interview represents a tree of video sequences; the video sequence with the most general information lies at the root of the tree.



Example: Tree of Video Sequences

The level of detail increases with the level in the tree; the more details the students ask, the deeper they descend into the tree, getting more and more detailed information.

4.2 The Modeller Tool

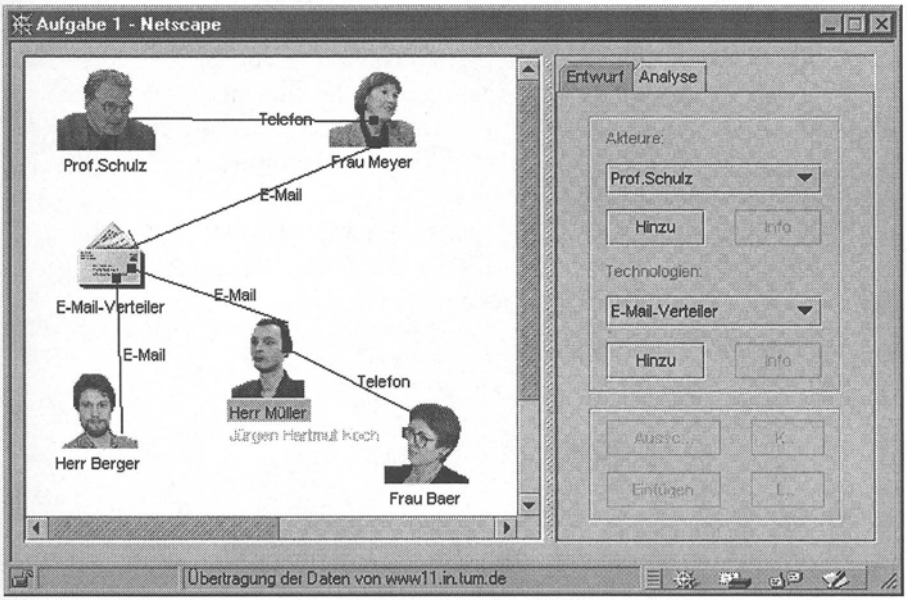
4.2.1 Purpose of the Modeller Tool

In our kind of case studies (IT infrastructure in organisations) we often have to deal with distributed systems, especially groupware systems. Our aim is to present students with the "nuts and bolts" of these distributed systems, in particular how information is exchanged among the system's different components. Since distributed systems may exhibit non-deterministic behavior, a tool like the Modeler Tool is immensely useful to help our students observe, model and analyse the dynamic behavior of various demonstration systems.

4.2.2 Modeling Informational Networks as Graphs

In the Modeler Tool distributed systems and informational networks can be modeled as graphs: The components which receive, submit, and process information are the vertices, and the edges model the connections between these components. The properties of real world components and connections define attributes of vertices and edges. Since we make no assumptions about

the complexity of the components and the connections between them, this concept is flexible enough to model very different systems. We can model core- level technologies like client/server systems where the vertices are the client and server programs, and the edges are existing network connections like an Ethernet. Here "flow of information" is the flow of TCP or UDP packets over a real network connection. We can also model very high-level situations like workflows or other informational networks where the vertices are humans or information systems like database systems, and the edges represent any flow of information between the vertices, for example phone calls, database queries, etc. The following picture shows an example.



An Informational Network in the Modeler Tool

Once an informational network is modeled as a graph, all the techniques used to analyse graphs can be applied to our informational network.

4.2.2.1 Actors, Technologies, and Connections

For most real world informational networks that we wish to observe, the classic view of a graph–vertices and edges–is far too abstract. We therefore divide the vertices into two classes: *actors* and *technologies*.

Actors model all people who work with information, and we call all hardware and software which the actors use technologies. A systems engineer would be modeled as an actor, as would a factory worker or a secretary. The information systems they use (mailing lists, databases, etc.)

would be modeled as technologies. Note that there is a fundamental difference between actors and technologies: While technologies handle information automatically—once implemented they do exactly what they are told to do—actors can react much more flexibly.

Edges in the graph model the flow of information between actors and/or technologies in the real world. To stress this, we call edges *connections*.

4.2.2.2 Modeling Behavior

The entire behavior of both actors and technologies is defined by *rules* and *attributes*. Rules define how actors and technologies react to certain events; attributes are used for "fine-tuning".

For technologies, the rules cannot be changed by the users; the rules define how a technology has been implemented. For actors, some rules may be fixed while other rules may be defined by the students. This means that the students can "teach" actors how to react to certain events and situations; the students just add a new rule that tells the actor what to do in some specific situation.

Rules consist of two parts: The first part, the *If-Part*, holds some conditions; the second part, called the *Do-Part*, specifies what actions to do if all conditions in the If-Part evaluate to TRUE.

Rules can look like this:

```
if
  changed(paper)
  paper_already_finished==(const)FALSE
  current_time<990228
do
  send paper to Pamela via e-mail
  set waiting_for_comment=(const)TRUE
```

The If-Part may be empty. In this case the Do-Part will always be executed.

The behavior of actors and technologies can be fine-tuned by changing some of their attributes. Some attributes may be predefined by the case study description; others may be changed by the students. The Modeler Tool provides forms in which all component attributes can be observed and eventually changed.

Connections do not have rules but only attributes. (They do not need rules since their one and only job is to transport data from a source to a sink.) Using attributes for edges, we can model properties of the connections in the real world. Examples may be reliability and maximum bandwidth of network connections.

Similar to actors and technologies, students can use forms to observe and eventually change attributes of connections.

4.2.2.3 Modeling Constraints of the Real World

The most important attribute of a connection is its type. The *type* attribute defines which kind of connection an edge models. For example, an edge between two actors *A* and *B* may be of type "e-mail" to model that *A* sometimes sends e-mails to *B*.

Actors and technologies may accept only certain specific types of connections. This allows us to model that, for example, a factory worker has no e-mail account and, therefore, cannot receive or send e-mail. In this case, the Modeler Tool does not allow an e-mail connection between the factory worker and his or her boss.

4.2.3 Analysing Informational Networks

Once a graph modeling a real world informational network is defined, we can use all the techniques provided by classic graph theory. But these techniques cover only "static" properties of the graph; they do not capture the behavior of the informational network over time. The Modeler Tool provides methods to analyse not only the static aspects, but the dynamic aspects of the informational network as well. For better flexibility, the methods for graph analysis (static as well as dynamic) are not hard-coded into the Modeler Tool but reside in *analysis modules* which the students can plug into the Modular Tool at runtime.

4.2.3.1 Analysis of Static Properties

Attributes of actors, technologies, and connections are the basis for analysing the network's static properties. Values of attributes are used to calculate certain numbers, for example, total costs of the informational network, maximum delay of a message between two actors, etc. These numbers are used to check if the informational network, modeled by the students, fulfills certain requirements that have been postulated by a case description.

4.2.3.2 Analysis of Behavior via Visualisation and Simulation

Analysing the behavior of an informational network over time is very interesting. Here the analysis modules must handle dynamic aspects like the change of attributes over time, reliability of connections, delays in message transport caused by overload of certain components, etc. We wrote different monitor modules to visualise the dynamic behavior of dedicated components of the informational network. For example, a load meter (similar to the Unix

tool xload) shows the "work load" of actors, that is, the number of jobs still to be done.

Monitors like the ones mentioned above only make sense when used in simulations. Currently, two simulation modules are available. One allows the running of the whole informational network in slow motion (the user can set the delay between two time slots), and another lets the student single-step through the informational network. The students can switch between simulation modules whenever they want.

5. ARCHITECTURE

To present students with an integrated learning environment, rather than a set of different applications, a special framework program for integrating all the other applications is necessary.

In the Munics learning environment, the Session Manager is this program. It provides the so-called "working environment", a framework for all other Munics applications and multimedia presentations. Additionally, the Session Manager provides commonly used services to the other Munics applications.

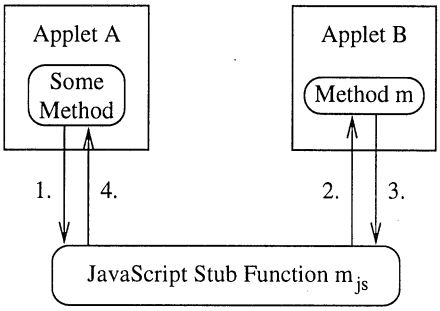
In order to be accessible from all over the world, we decided to use Web browsers as our sole runtime environment. Therefore, we implemented all Munics programs as Java applets.

5.1 Inter-Applet-Communication in the Munics Learning Environment

Most Munics applets reside in different frames. This has certain advantages for maintaining a nice screen layout, but on the other hand causes some serious technical problems. The applets reside in distinct namespaces that means that they do not have references to each other. This makes it quite difficult to share data.

Because of the limitations most web browsers impose upon applets, we could not use component technologies like CORBA for communication among applets. Instead, we use JavaScript.

For each public method *(m)* an applet exports, we wrote a JavaScript stub function m_{js}. When an applet *A* wants to call a method *m* of applet *B*, *A* calls a JavaScript stub function m_{js} (step 1). Now, m_{js} first tries to get a reference to the "target" applet *B*, then it invokes the corresponding applet method *m* (step 2) and finally it returns the result - if any - to applet *A* (steps 3 and 4).



Communication between Two Applets A and B

Thus, the JavaScript stub functions serve as a "bus" to exchange data and events between all Munics applets.

5.2 Communication between the Munics Learning Environment and the Munics Servers

As already mentioned above, the Session Manager applet provides interfaces to commonly used services, such as persistent properties or the event service that connects locally distributed learning groups. Applets do not connect to the server programs directly, but instead call service methods the Session Manager applet provides. The Session Manager applet then connects to the corresponding server program. So all network connections are handled by the Session Manager applet which makes it very easy to evolve protocols, install client-side caches to reduce network traffic, bundle network connections to operate through firewalls, and so on.

6. CURRENT STATUS AND FUTURE DIRECTIONS

We have currently finished implementing the first prototype of the Munics learning environment. Early releases of the working environment framework, the Interactive Problem Context and the Modeler Tool have been finished.

By the end of 1999, we plan to integrate a collaborative notebook where learning group members can collect ideas, note important information, etc. Additionally, we want to enhance the rule language that is used to model the behavior of actors and technologies. This will help us to simulate complex informational networks much more realistically ways than we have been able to do so far.

In July 1999, we tested the prototype as one step in formative evaluation. Our aim was to analyse the learning process in order to identify possible problems in the use of Munics. We focused particularly on the usability of the whole learning environment and the efficiency of the learning process. Moreover, we were interested in how the students accept Munics, in terms of how well they like to work with it and to what degree it fulfills their expectations. First results will be available in autumn 1999. Our next step will be to improve Munics accordingly.

REFERENCES

Albion, P.R. and Gibson, I.W.: 1998, Interactive Multimedia and Problem-Based Learning: Challenges for Instructional Design, in T.Ottmann and I.Tomek (eds), Proceedings of ED-MEDIA/ED-TELEKOM 98, Freiburg, Germany, pp. 117-123.

Collins, A., Brown, J. and Newman, S.: 1989, Cognitive apprenticeship: Teaching the craft of reading, writing and mathematics, in L.Resnick (ed.), Knowing, learning and instruction: Essays in honour of Robert Glaser, Erlbaum, Hillsdale, NJ, pp.453-494.

Grooters, F. and de Vries, S.: 1998, Design of a Project-based Study Environment on the World Wide Web, in T.Ottmann and I.Tomek (eds), Proceedings of ED-MEDIA/ED-TELEKOM 98, Freiburg, Germany, USA, pp.493-498.

Guzdial, M., Kolodner, J., Hmelo, C., Narayanan, H., Carlson, D., Rappin, N., Hübscher, R., Turns, J. and Newstetter, W.: 1996, Computer Support for Learning through Complex Problem Solving, Communications of the ACM 39(4), 43-45.

Mandl, H., Reinmann-Rothmeier, G. and Gräsel, C.: 1998, Gutachten zur Vorbereitung des Programms "Systematische Einbeziehung von Medien, Informations- und Kommunikationstechnologien in Lehr- und Lernprozesse", Bonn, Bund-Länder-Kommission für Bildungsplanung und Forschungsförderung.

Nuthalapaty, F.S., Oh, J., Elsner, C. and Altman, M.: 1998, Interactive Electronic Problem Based Learning (iePBL): An Internet based Application for Clinical Medical Education in the PBL Case Format, in T.Ottmann and I.Tomek (eds), Proceedings of ED-MEDIA/ED-TELEKOM 98, Freiburg, Germany, USA, p. 2066.

Reinmann-Rothmeier, G. and Mandl, H.: 1997, Lernen mit Multimedia (Forschungsbericht Nr. 77), München, Ludwig-Maximilians Universität, Lehrstuhl für Empirische Pädagogik und Pädagogische Psychologie.