Online Teaching and Learning – Developing and Using an eEducation Environment

Manuel Goetz, Stefan Jablonski, Michael Igler, Stephanie Meerkamm
Chair for Applied Computer Science IV
University of Bayreuth
Bayreuth, Germany
(manuel.goetz, stefan.jablonski, michael.igler, stephanie.meerkamm)@uni-bayreuth.de

Matthias Ehmann
Computer Science Education
University of Bayreuth
Bayreuth, Germany
matthias.ehmann@uni-bayreuth.de

***Abstract*— To support schools in teaching computer science, we have started the Informatik@School project. In this project, which is meanwhile funded by "Oberfrankenstiftung", we communicate computer science content to beginners and matured students. As the number of participating students is very large in comparison to the number of advisors and big distances have to be bridged, we separated students into two groups. Students from schools located not far from our university are taught in common face-to-face lessons, while far-off students get taught the same content in online lessons.**

In this paper we present the project, its preconditions and the didactical and content based concepts. We will introduce a web based technical environment which fulfills these issues and facilitates realization of afore mentioned online courses. Finally, we present lessons learned from this project and draw conclusions especially concerning the technical platform which consists of hardware and software used.

Keywords: e-learning; online teaching; online education environment

I. INTRODUCTION

Since the German curriculum at secondary schools was reformed, computer science is an independent subject. Teachers skilled in this discipline are rare as education of computer science teachers has just begun. Furthermore, computer science is a broad area and schools are limited with respect to time they can spend on computer science education. Consequently many interesting fields of computer science cannot be addressed.

Therefore a project named "Informatik@School" was set up (funded by "Oberfrankenstiftung") which supports schools in computer science education and should increase students' interest in computer sciences [1]. As this project is not limited by a curriculum, topics are freely selected; they should be of major interest for the students. Limitations of this project are the amount of time the advisors can spend and the distances between schools and the advisors. According to that some schools are taught remotely while others can be taught in common classes. In this paper we present our online teaching approach consisting of a technical and didactical part and compare success to traditional learning methods.

The rest of the paper is structured as followed: In Section II we give an overview about the didactical teaching and learning concept used in our approach. Section III provides a collection of requirements and their technical implementation needed for realization of the didactical concept. In Section IV we describe an application of our didactical and technical concept in computer science teaching and learning. Finally, Section V summarizes our experiences and provides and outlook to future improvements and research.

II. TEACHING AND LEARNING CONCEPT

Our final goal is the development of an online environment for teaching and learning. To find an adequate solution it is necessary to analyze the situation of online teaching and learning and the development of a fundamental learning concept. After these steps are done, it will be possible to identify necessary parts of a software solution.

A. Teaching and learning situation

From the time of Johann Friedrich Herbart on, the didactic situation of teaching and learning can classically be described by the didactic triangle [23] (Figure 1).

It covers the interdependencies between teacher, learner and content. This figure visualizes the dependencies of the main factors of teaching and learning. Analyzing the didactic triangle can lead to different didactical concepts.

Brain research and psychology made large progress in understanding the human learning process and the underlying cerebral structures during the last century [30]. According to these results learning must be understood as an individual process. A learner builds his own cerebral web and embeds new knowledge into his web. The connections between “chunks” of knowledge are built during the learning process. The

stronger these usage dependent connections are the better the fetch of the chunks works.

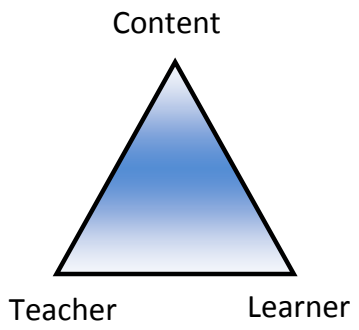

Figure 1. Didactic triangle

Taking all previous points into account we can conclude that learning is more successful if students play an active part in the learning process. The teacher is not the main person; he is in the role of an organizer of the learning environment. These results are similar to the constructivist learning theory.

The student centering is the most important part of our didactic concept.

B. Demands on the Future Generation

In didactics, demands on the future generations are often used to classify the content of teaching and learning. But demands on the future generation also influence the way of teaching and learning – the methodology.

Live long learning is one key word in our society. We must enable our students to learn self-dependently. They need didactical tools to make new fields of knowledge accessible to themselves. This is another motivation (see II.A) for a student centered learning concept.

Another key word is key skills qualifications / soft skills [26]. Employers and educators criticize that graduates are not well equipped with basic general skills which are necessary for their future professional life and full participation in society. Standardized assessments like TIMSS [17] or PISA [20] are tools to get an objective result concerning knowledge and key skills qualifications of students. They partially confirm the criticism.

The resulting demand on the future generation is the ability to solve problems. All other key skills qualifications are tesseras to succeed in problem solving. According to that we identify the main qualifications to develop a didactical model for sustainable learning and teaching:

Communication. Communication is the fundament in our work-sharing society. Students need to practice communication with others. They also need method competence in selecting and using communication tools.

Cooperating with others. Communication is just the key to cooperate with others. Students must learn concepts of cooperation like team play, constructive arguing and taking responsibility. They have to combine these skills with communication.

The concept of pair programming, for instance, from the extreme programming approach contains good ideas of cooperation in computer science tasks [3].

Presenting. Presenting work results becomes more and more important in all areas of work life. Students must acquire presentation techniques and get used to giving presentations in front of an audience.

Improving own learning and performance. As requirements will change much faster in future, we must equip our students with techniques of self dependent learning to give them the chance to adapt to any kind of changes. They must be able to identify targets and work towards them.

Acquiring these previously listed skills works best in social situations. They are all requirements for problem solving. Problem solving is and was an important qualification for any generation. Today the world-wide-web supports finding solutions for many issues. It is the most comprehensive knowledge base in history of mankind. But problem solving is more than investigating the web. Students need to gain an insight in problem solving strategies. A possible process for solving problems can be adapted from computer science. It consists of 4 steps: modeling, processing, interpreting and validating [9] (Figure 2).

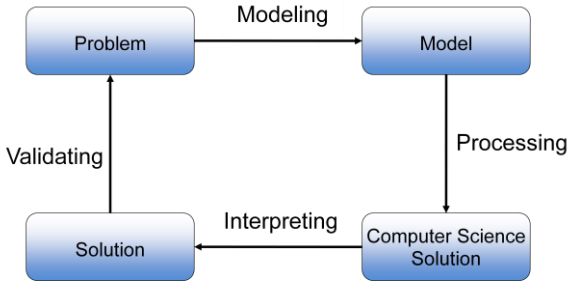

Figure 2. Problem solving

Modeling is necessary to make a problem processible:

- Analyzing the structure
- Identifying main objects, their properties and relationships
- Defining operations within a model

A model is a miniworld view of a given problem. Students need mechanisms like analyzing, structuring and abstracting to succeed in modeling.

Creating a model is just a first – but important – step. The model itself is not the solution of a problem –

for example – just as little as an entity relationship model is the final implementation of an enterprise resource planning system. Creating and working with the model helps to structure a problem and find a solution.

Processing the model leads to a first solution, which is produced based on the model created in the previous phase. In computer science, the processing phase can be an implementation in a programming language.

Interpreting the solution is a kind of inverting the modeling. It re-translates the solution from the miniworld's "language" back to the "language" of the entire problem. Considering an implementation, we interpret the result delivered and draw conclusions.

Validating the interpreted solution is the final step. It helps to identify errors in the whole process and is a kind of quality measurement. It is necessary to go through all steps to get a validated solution.

This process of problem solving has a recursive structure and is valid for problems from many disciplines. It can be applied to widespread problems; included smaller problems can be identified and solved using the same strategy.

Taking all results of our previous analysis into account, we have to implement a didactical concept for individual as well as cooperative problem-oriented teaching and learning in an electronic environment.

C. Structuring the learning process

We will introduce our teaching and learning concept by describing the structure of the learning process. Later on we will go into further details of important phases.

1) Teaching usage of communication tools

As education via online courses is uncommon to German students, first an introduction to the technical environment has to be given. Although some students were used to parts of our environment because of free time activities, no student was familiar with all tools of our online teaching environment (see III.D, III.F).

Independently of the tools that should be introduced, we could observe a huge cooperativeness of students to help each other in this phase.

2) Introduction to theoretical and practical concepts of teaching content

Basic concepts of the application domain are introduced. In case of computer science and web technologies, important concepts are the OOP (Object-Oriented Paradigm) [16] and especially the structuring and modularization of problems. Furthermore, an introduction to the applications that should be used for applying the learned theoretical concepts was given (Squeak [31] and Scratch [18] for beginners and Java with Eclipse WTP [7] or jMonkey Engine (JME, [25]) for advanced students). In this phase, consequent feedback from students is extremely important to achieve a good learning curve. Also direct and fast support is essential for motivating students.

3) Applying concepts learned in simple exercises

In order to train transfer and problem solving techniques, the introduced concepts are applied to simple problems. With supervised implementation of simple applications, students are forced to think about the concepts learned.

Although this is an extra phase, it cannot be completely separated from the previous phase as there is an overlapping; distinction is done concept and content based and not because of the timeline of teaching.

We found two factors which are especially important for success of this phase. Firstly, there is a need to avoid a higher degree of frustration at students. Therefore, consequent encouragement for asking questions concerning their (probably not working) solution has to be done. This has to be combined with fast and clear answers, when support is needed (see also [22]). Secondly, the possibility to "revisit" a lesson with watching a video-on-demand containing the lesson is used broadly.

4) Solving a daily problem single-handed

After applying the concepts learned on simple problems in a supervised environment, students work in teams to solve a daily and more complex problem. They should get a feeling for the complexity of common domain problems and a rough understanding for the different modules a structured solution is composed of. As a real problem of a domain can mostly not be solved from one lesson to another, about two months are given for finding a solution. Also the character of lessons changes. Advisors no longer have an active part, but answer questions on special problems. Furthermore, support should not be that detailed as in the previous phase, but only give a rough solution that then should be detailed by the students. This way a self dependent working style is trained.

D. A basic concept for individual and cooperative problem-oriented teaching and learning

Especially the realization of step 3 and 4 cannot be covered by traditional teaching. Here our approach for individual and cooperative problem-oriented teaching and learning is applied. A concept that fulfills the requirements is the "I-You-We" approach (Figure 3). It is a problem-oriented way of teaching and learning with three stages of individuality [33]. We adopted this concept to online learning and teaching (see IV.C, IV.D).

In the "**I phase**" the students are confronted with a problem and have to explore the situation by themselves. They should try to find an individual solution. In contrast to common teacher centered education, the students take an active part in the learning process: They go their own ways and they make their own decisions and experiences. The

solutions may not be perfect, but also mistakes help them to improve understanding. For example, they are asked to create an object diagram to visualize the objects and relationships regarding the online chat. The advisor becomes some kind of coach or tutor.

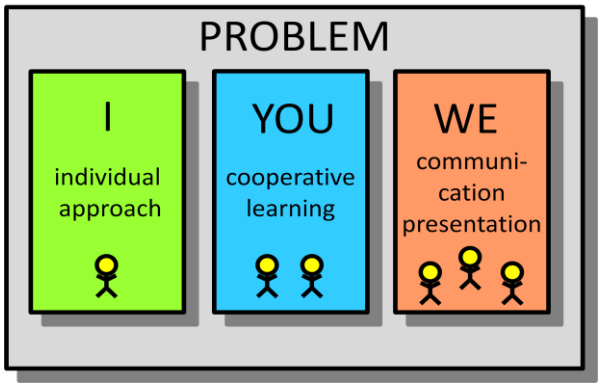

Figure 3. I-YOU-WE Concept

Cooperation characterizes the "**You phase**". Together with a partner, students rethink their solutions. Discussing about the problem and explaining their ideas support this process. They put thoughts into words to communicate with the partner. This is another active learning process. Errors can be identified and different solutions can be combined.

The collaboration with the partner is a kind of dress rehearsal for the presentation in the whole group and helps them to overcome their inhibitions.

The two phases of individual exploration culminate in the common presentation of the students' work, the "**We phase**". Some learners show their ideas to the whole group with reporting their results and difficulties. The listeners complete the remarks. As every student was engaged in the given problem in the two phases before, he should be familiar with it. The teacher is the moderator. He leads the discussion and he combines the gathered work with new content. Discussing and communicating about a situation deepens the understanding of the problem and opens each students mind for manifold approaches.

In our concept there are two possibilities for the last phase. The presentations can take place in on site lessons in every school. Then the local teacher moderates the session. The online alternative uses web technologies.

We apply this concept also to the on site lessons of the project. It can also be used in traditional teaching in different subjects as "islands" of self dependent and cooperative learning [6]. This can be the beginning of a new way of teaching and learning.

The duration of units covering the concept reaches from parts of a single lesson up to projects lasting several weeks.

III. TECHNICAL ENVIRONMENT

After introducing the teaching and learning concept, our concrete realization is described in detail.

A. Preconditions

Goal of our project is to support schools in computer science education [10]. Participation of students is optional and they can leave the project every time. The curriculum should be interesting and directed to students' interests in order to increase their motivation to deal with and apply computer science techniques. This means we can communicate content concerning computer science in a form students appreciate.

In this project more than 200 students per year from 15 schools participate and are attended by only three advisors. Three of these schools with about 30 - 40 students can be visited directly; students of the other schools need to be taught remotely because of the local distance which is up to 70 kilometers. For online teaching we use an internet based e-learning approach [4].

In our online schools local mentoring is needed, especially for setting up and introducing students to the online environment. Therefore a teacher of every participating school is prepared for usage of technical environment and is taught basics of the lessons' content.

B. Requirements to an online teaching environment

There are manifold requirements for an online teaching environment (OTE). During the first two years of our project Informatik@School we gained much experience in the field of e-teaching and gathered information for a requirements review. We can categorize the demands in three scopes: functional requirements from students, functional requirements from advisors and technical requirements. The functional requirements take into account the demands of our learning concept (Section II).

Functional requirements from students. From students' point of view, an OTE has to be a tool easy to use. In our project students have the chance to participate in online lessons in all places. Consequently an OTE should be available at school and at home. It has to combine several single applications in just one user interface and should cover several use cases.

We offer live online lessons. During these lessons students should see advisor's desktop and hear his voice. Students should also have the possibility to ask questions during a lesson (see II.B). This requires unidirectional video and bidirectional audio transmission. To arrange an almost face-to-face learning situation for our students it would be nice to have a web cam transmission of the advisor.

During online lessons, students work with their own project files. These files are normally stored on their local PC, but in our use case more flexibility is needed as students may start a project at school and finish it at

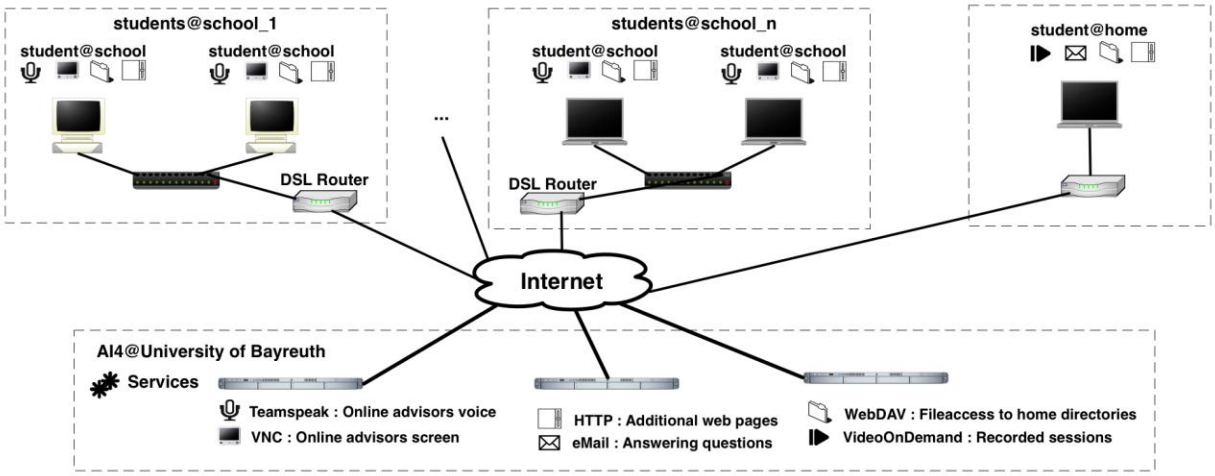

Figure 4. Overview of first topology

home. To enable seamless portability of files, an OTE should offer an online storage solution for students' files called data pool. This data pool has to be accessible from anywhere and should be independent from the computer students currently use. The online storage solution is also important for collaboration of students (see II.B). We want to encourage our students to exchange their ideas and work together in groups (see II.B, II.D). So they need the possibility to provide access to their files for other students. An OTE should offer a simple right management solution for the data pool.

The cooperation between students can be real or virtual. For virtual cooperation we need forums and online chat functionality. Some students do not have the chance to participate in the live online lessons; other students want to revisit a lesson. For these participants we should offer video streams of all online lessons.

Functional requirements from advisors. The requirements of communication during and after online lessons are also valid from advisor's view. The live audio communication should be under advisor´s control. He manages the voice rights and grants voice privileges to students after a request. This is a way to offer the possibility of asking questions and giving a lesson in a controlled way.

Many questions of students can be answered easily without watching what students have done at their PC. But especially in the phases in which student's work on larger projects (see II.C.3, II.C.4), questions and problems become more and more complex. In these situations qualitatively good and fast support can only be given if the advisor can see and also access students' desktop. The advisor also needs a storage solution for his files. He has to publish project files and scripts for all students.

An OTE has to provide a tool where students' exercise solutions can be collected. It should be a central repository where the advisor can access the files, correct them and provide some comments for the students.

Technical requirements. Beside the functional requirements of the users we also have to keep the technical requirements in mind. Especially the available hard- and software equipment at schools can be a limiting factor.

The client environment is very heterogeneous. Students use different PCs with different operating systems at school and at home. We can meet with this obstacle by using portable client software on students' side.

Also security restrictions at school have to be faced. The client software has to run with standard permissions. Network applications have to use standard ports.

Taking all this into account, a web application is the best solution to fulfill the technical requirements.

Furthermore the client application needs a user-friendly installer to support PC administrators at schools in the best possible way and facilitate installation for students at their PCs at home.

C. Technical Environment at schools

Main limitation in designing the technical environment was a heterogeneous technical infrastructure at the participating schools. We had to consider very different hardware configurations ranging from up to date computers to PCs that are aged more than 10 years. The same applies to the used screens, i.e. we needed to be compatible to quite low resolutions on students' side. Additional to the hardware, already installed software on the computers had to be regarded. On the one hand, schools work with different operating systems (and some schools even use very special configurations of an operation system). On the other hand, compatibility of our software to installed security software needs to be provided. Especially schools have very hard restrictions concerning security which leads to school environments where students cannot save files

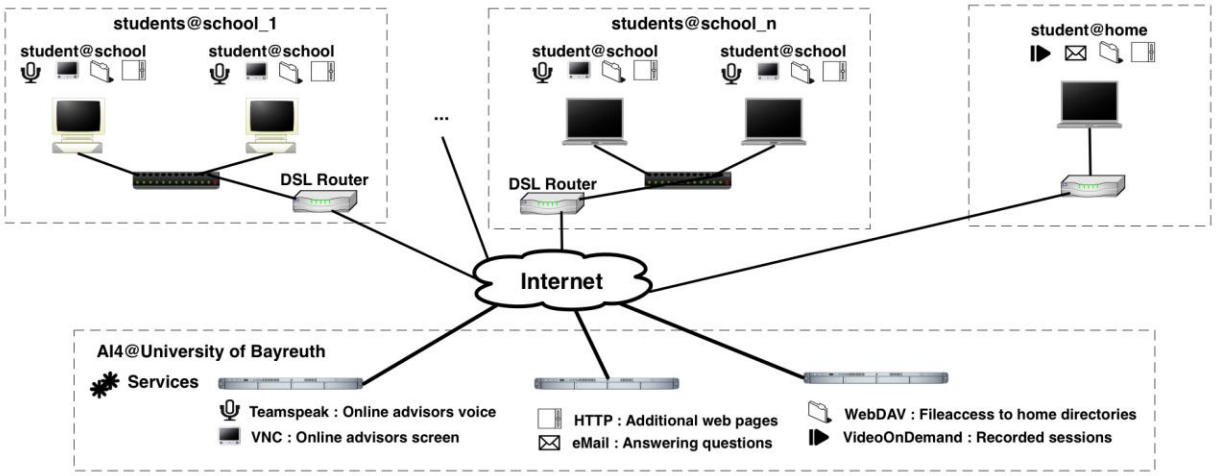

Figure 5. Overview of the eEE topology

or just have very restricted access to the internet (which we need to use to provide communication). Consequently, we had to realize a platform spanning solution on client side.

As we are in the second run of our Informatik@school project meanwhile, we have a first implementation of this software (Figure 4) which was used in the first term and an improved second version which we are using currently (Figure 5). Both versions will be presented in the following and the lessons learned during usage of our first version will be discussed as these lessons lead to the implementation of our second version.

D. Online Teaching Environment Version 1.0

To provide an audio communication channel during online sessions, we decided to install a VOIP (Voice Over IP) environment. After evaluating different systems, our decision fell to the free application (for non commercial entities) "Teamspeak" [32]. This solution is available on common operating systems like Microsoft Windows, Apple OS X and Linux. It provides a spanned solution for the most configurations on the client side. Due to the web based administration control panel and the highly scalable user permissions system, it is very flexible and comfortable to administrate the accounts in our server environment. During the online course all students hear the advisors voice and we can grant a student the right to talk if he is requesting voice for a question. All other participants in the online course can hear his question.

Screen content of the advisor's computer is transmitted via a VNC-server [27] to the VNC-clients on the classroom side. Students can watch the transmission live on their own screens or on a video projection in the classroom. Accessing the transmission is possible via the VNC server's integrated web service. So a connection using a java-capable browser is possible, which avoids installing a VNC client on classroom computers.

Online lessons are recorded containing advisor's and students' voice as well as screen content of the advisor. It is saved in the windows media format WMF [36] which are provided as on-demand video streams. Video streams are accessible on our website and students can use them for postprocessing the online lessons at home.

Resources of the online lessons like PDF [1] files of presentations or programming libraries are downloadable through our website. So they are readable and presentable for any later references.

As data exchange tool between students and the advisors at university, we use a WebDAV [35] system in this version. WebDAV is an extension of the Hypertext Transfer Protocol (http). It allows bidirectional file transfer. The WebDAV service is accessible with login and password through the Internet Explorer [13] by entering the URL [34] of our WebDAV server. WebDAV folders are mapped to the file explorer of windows automatically. So this procedure presents the folder structure in a well-known way.

All necessary files of each lesson (WMF, PDF, project resources) are also stored on the WebDAV system; so the online lessons can be reused later for reference. Consequently the WebDAV system is a central storage unit for all course resources.

Questions occurring after a lesson can be asked by email or phone. Especially questions addressing students' problems with their current implementation are mostly asked by email as students can send their current source code in the attachments or as a link to their WebDAV folder.

E. Lessons learned

Almost every school network has a high bandwidth asynchronous DSL [29] connection to the internet, which is mostly secured by a firewall [21]. Although

there is no standardized bandwidth of schools' internet connection, the minimal configuration of approximately 1 MBit/s downstream was sufficient to receive voice and screen content transmission during our online sessions without latencies.

The upstream bandwidth of 128kBit/s with asynchronous DSL connections was adequate to receive students' questions during online lessons in good quality. Additionally, it is desirable to switch to the computer of a certain student in order to help him solving a problem with his development environment or source code during the online lesson. Our didactical concept could be realized better if this kind of feedback would be available, too.

In the beginning of the project most common problems in initiating connections between schools and our server came up from different policy restrictions of the firewalls installed at schools. As long as there is a stateful firewall [21] it is not necessary to open any incoming ports. For more restrictive firewalls some configuration effort has to be done to enable a problem-free transmission.

Students had no basic problems concerning the usability of our environment. Switching between the windows of several stand alone applications (Web Browser Window for WebDAV, Web Browser Window for VNC video transmission, Teamspeak) was a bit unpleasant sometimes.

Based on our experience during the first run of the project "Informatik@School", we designed and realized an all-embracing solution. This new environment helps to minimize client side software installation effort by using web services. Furthermore it offers a single user interface for all services to avoid switching between several applications.

F. eEE – The next step in online teaching

Finally, the developed system for online teaching called eEE (eEducation Environment) is presented and discussed. In comparison to the version presented in [1] and III.D, progress regarding of some important features could be done. Firstly the functionality offered to the user could be integrated into one single web application which facilitates and accelerates the access to the available functionalities and makes learning and teaching more comfortable. Furthermore we succeeded in implementing the access to student's desktop by the advisor, which improves the quality of teaching a lot.

1) Conceptual layout of eEE

We decided to deal with the requirements mentioned in Section III.B by creating a web application. This application is deployed on a dedicated server which is connected to the internet and provides the server components of a client-server architecture. Students and advisors log in to this environment by login name and password through a web browser. eEE can be accessed from all schools and from students' homes.

Figure 5 gives an overview of the technical aspect of the architecture of our eEE. Students' computers are placed in the top part of this diagram. Students can be connected to the internet directly or through a local area network using a router. When a connection to our server is established and users are identified, students can use all services provided by our server software like audio conference, desktop transmission or functions independent of online lessons like chat or forums. In comparison to the first version (see 3.5.) now all services are integrated in one single environment. Regarding the additional functionality the advisor needs, it is valid as well. He just needs to connect to the server to give a lesson or use other features the software is providing.

eEE itself is now based on the tool Moodle [19]. Moodle is a learning management system focused on course management, but lacks most features needed for live online teaching. It provides a widespread plugin structure, i.e. some components can be added or exchanged easily without having any effect on the rest of the system. We developed a new plugin that supports online lessons to adapt Moodle to our needs. As we used this plugin structure we established a loose coupling. Updates or other extensions do not interfere with our plugin and vice versa, i.e. we are not restricted to one special version of Moodle.

With Moodle and the additional plugin for the online lesson we have an integrated concept for online teaching and a direct and facilitated access to all the functionality is possible.

2) eEE for students

As already mentioned, in this version of our online teaching system the entire functionality is offered in one single system. Other systems are not necessary anymore.

At first, students need to select a course to login. After authentication, students can select many different options and have access to all working materials like scripts etc.

Now they can download these resources or open them in their browser. Additionally communication tools like a forum or a chat are available. This all time available communication platform raises collaboration between students and establishes a community in our project, although students may be separated by potentially big distances.

Students often need to switch their PC and working place (they may work at home, at school or meeting at a team member). Having the right data at the right moment on the right place was not easy to organize and handicapped the work a lot. Thus in this version we also offer an online storage which is accessible with our application. Students can upload and download files in their personal folder. They can also share files with other students; this sharing is based on the accounts, i.e. for every file or folder that should be shared the privileges to access this file or folder can be given to

other students individually. That way teamwork in small groups is facilitated a lot.

The features presented so far deal with the administrative part of our environment, i.e. with the distribution of data and students' communication between the lessons. Lessons are available in two modes: students can participate in currently given online lessons or they can watch videos of former lessons. All lessons are recorded with sound and desktop of the advisor. These videos are available as streams and can be watched directly in the students' browsers. In older browser versions additional plugins need to be installed to play these video streams. Current versions of Internet Explorer or Mozilla Firefox, for instance, can start the videos directly as necessary video plugins are already integrated. To support a broad variety of systems, also links are given to access each stream from any potential player.

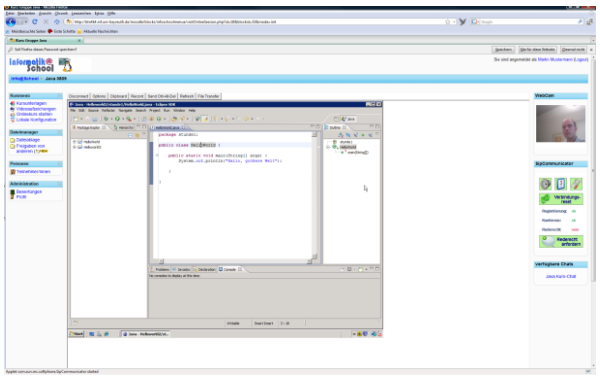

Figure 6. Screen of students during online lessons

While an online course is given by an advisor, students can participate without prerequisite software installations (see Figure 6). In order to minimize requirements and to guarantee portability between systems, all components needed for online lessons on client side are based on Java Applets [12]. First, students can watch the advisors desktop. This is done via VNC streaming [27] and a Java client on students' side which is started automatically as an applet when students participate in an online lesson. Using the same approach, students can see the advisor as a webcam stream is delivered to the students. Lastly, students can listen to the advisor. This is managed using an online conference tool called Java Voice Bridge [14] which was developed by Sun. We integrated this tool and implemented a new user interface. Joining an online course, a student also joins the corresponding audio conference. This happens automatically so that students do not realize it. By default, students have no voice privileges, i.e. they can listen, but their voice is not transmitted to the other listeners. Therefore, students also have a button to ask for voice privileges. When this permission is granted by the advisor, students can ask questions; all other participants can listen to that question and the advisor's answer.

As we teach computer science, software on students' side is needed (currently JDK [8], Eclipse [11] and Scratch [18]). To provide this software, we offer an installer created with InnoSetup [28]. This installer also contains a VNC server which is needed for the optional access of an advisor to a student's desktop.

3) eEE for advisors

After discussion of the students' options in our system, the features for the advisors are presented. As for the students, in this version all features can be offered on one single platform which makes teaching much more comfortable. Furthermore we succeeded in implementing the access to students' desktops for the advisor during an online lesson. This was one of the requirements regarded as very helpful.

First, advisors can create courses. A course complies with a subject in school in our understanding. Courses may have one or more advisors who have the privileges to add scripts or new videos of lessons. Furthermore they can initiate online lessons.

After an advisor started an online lesson, his desktop is transmitted to the students automatically. This happens with a VNC client that streams the desktop to all registered clients. If a student participates in an online lesson, this registration is done automatically. The advisor can observe which students are currently visiting the lesson (see Figure 7).

Figure 7. Screen of advisor during lesson

Via the same interface, the advisor is informed when students ask for voice privileges. Then he is able to grant this privilege to one or more students. He can also revoke this privilege individually after answering the question.

A feature that is extremely helpful, but technically hard to implement is the access to the students' desktop for the advisor during an online lesson. It is helpful as problems students currently have can be identified more

effective if the advisor can really have a look on students' manuals. Consequently frustration for students is diminished. Usually access to students' desktops is forbidden by security restrictions. Furthermore, most students' PCs are placed behind some firewalls and/or routers and they do not have an own public IP address that could be addressed. Therefore, there is only one solution to provide this feature: the connection has to be established from the student's side, but without any action that needs to be done by the student himself. The advisor is the initiator.

If an advisor wants to access a student's desktop, a flag in a database is set. The database containing that flag is polled periodically from all participating clients. If this flag is set, then the student's browser starts a VNC streaming server and streams the students desktop to the advisor. As the connection is established from inside the student's network, there are no problems with firewalls. When the connection is not needed anymore, the flag is reset in the database and the connection is closed from the student's side. So there are no security gaps remaining.

4) eEE User interface

Finally we present the eEE user interface integrating the entire functionality. Layout for students, advisors and administrators is quite similar to facilitate usage with a clearly defined structure; the workplace is divided into three columns (see Figure 8):

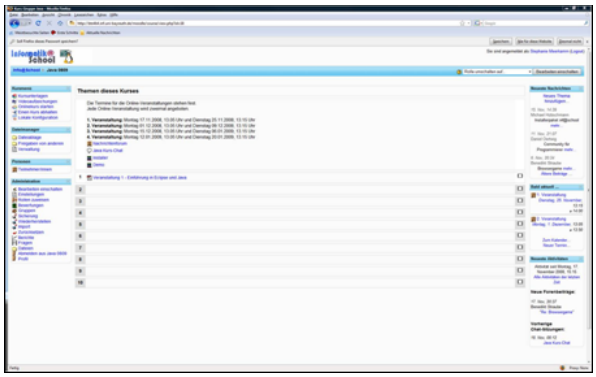

Figure 8. Screen of the workplace

- The left column offers a list of different activities and navigation possibilities. It is classified into "Course Menu", "Data Manager", "Participants" and "Administration". With the "Course Menu" the functions for the online-teaching are offered, as for example the courseware, the online session or the video recording. The "Data Manager" handles uploads and downloads of data files together with sharing of files between users. An overview about who is attending the course is given by the entry "Participants", whereas "Administration" offers general configuration possibilities.
- The middle column displays the content of each navigation topic. This can be all currently available scripts or, during an online lesson, the advisor's desktop.
- The right column contains additional information like news or next appointments. During an online lesson, the web cam shows the advisor.

This layout structure is always the same, whereas the offered activities in the left column change due to the role a person is logged in with. For instance, an advisor has the possibility to start an online lesson whereas students have the possibility to participate in a running lesson.

IV. REALIZING THE CONCEPT WITH COMPUTER SCIENCE CONTENT

A. General organization of our project

We offered two courses for students aged between 14 and 19. While younger students participated in a Squeak eToys [31] or Scratch [18] project, the older ones took part in a Java (e.g. [8]) project using the Eclipse developing framework [11]. In every group we had approximately 100 students at the beginning.

Due to introduction of a new curriculum, students participating in the Squeak / Scratch group already had some basic knowledge concerning OOP (Object-Oriented Paradigm) [16] and control flow modeling. The members of the Java group had no preliminary knowledge in computer science. In the face of the different precognition levels, the teaching concept described above was applied to both courses.

We offered one session for every course in two weeks at local schools which could be supervised directly. For online schools, an hour every week with one repetition was offered, i.e. every second week was a repetition of the preceding week. As we had different time slots at repetitions, more online schools could participate.

B. Preparation of teachers

Although most courses are held online there is at least one teacher in every participating school for supporting the students on site. On the one hand this assistance is necessary because of the differences in software installations, network environments and rights management. On the other hand technical problems concerning the infrastructure in schools during an online lesson can be solved much easier on site which is an important factor for success of an online teaching project ([37]). In order to enable teachers to deal with these problems, they were prepared in a one day face-to-face course at university.

First the course concept and the technical preconditions were introduced to them. In the second part teachers got to know the technical environment and

the installation of the necessary software products. The final and most extensive section of the in-service teacher training addressed the computer science content of the project.

Brief presentations gave a first overview about the aims. The content of our lessons was introduced and teachers dealt with the exercises of the first third of the online courses.

C. Courses with Squeak / Scratch

The curriculum for our younger students emphasizes basic, long lasting computer science concepts. As these students already have basic knowledge in Object Oriented Modeling (OOM) we use this as a connection point. We want to improve their knowledge in OOM and introduce object oriented and general programming concepts. To keep motivation high we have to combine the concepts with an attractive topic. The last two years we have chosen the field of computer games. All students have experience with computer games and we offer the possibility to look behind the scenes.

We used Squeak eToys during the first year and Scratch in the second year. Both development environments are based on Smalltalk [15] and offer an aged-based user interface for (object oriented) programming.

We apply our concept presented in 2.3. and 2.4. to our courses. During the first lessons we introduce the technical environment to the students together with basic computer science content. This makes the introduction of the tools more target-oriented. For instance, students have to download a draft of an animation film project from the central file repository; they “program” their own animation film and upload it as an exercise. The advisor provides a comment in the eEE and students can retrieve this comment. During the online lessons students use audio communication and chat to interact with the advisor. We observed no problem of the students concerning usage of eEE.

Also during introduction of new computer science concepts we try to make the online lessons student centered. They are always called upon to try out new concepts. They have time to work on their own and asked to give feedback. Always students have to present or describe their solutions. We also use exercises where students have to cooperate. For example, they have to model the concept of a car racing simulation and write a to-do list.

We inure students to be in a more active role during lessons. So the step to the third stage of our concept (II.C.3, II.D) is not that big. In this phase tasks become more complex and students have to apply and acquire knowledge. One of the tasks is to develop their own version of the arcade game “Pong” [24]. We offer a version they can use for playing but they cannot look behind the scenes. Each student has to develop a model of the game (I phase). He discusses and improves the model together with a partner (You phase). The “pair” implements the model together. Some of this work is done during online lessons. Students can ask the advisor for help if they have problems. They continue at home and ask for feedback using eEE tools. Finally the groups present their results (We phase). The presentation can be offline in their class or online with all students. The problems students solve in this phase help them to acquire new computer science concepts related to their present knowledge, i.e. they discover different kinds of condition-controlled loops and learn to use them. The kind of working according to II.C.3, helps students to improve their key skills qualifications.

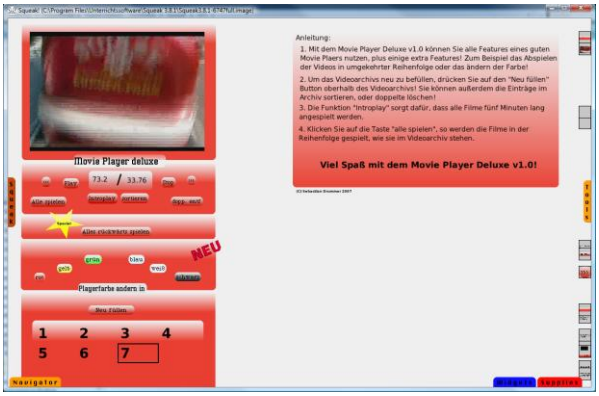

Figure 9. Student's solution – Video player

In the fourth phase (II.C.4) students work in a project for about two months. One of the problems they have to solve is the development of a video player in Squeak (Figure 9). The task is constructed in a way that teamwork is necessary. Students organized themselves in teams with two or three members. They were responsible for the whole project management. Most of the work is done at home. We offer online lessons in this phase as well. Students have the chance to ask questions. The access to students’ PCs is very helpful for the advisor so he can identify problems much faster and provide more precise help. In the last runs of our project, students' solutions partially exceeded requirements. We recognized that students really cooperated and applied previous learned computer science skills as well as soft skills.

D. Courses with Java

For students in higher grades the Java course is offered. Goal of this course is to give an introduction into computer science contents through usage of professional tools and concepts.

In the first phase (see II.C.1), students need to get to know the communication tools which in our case is eEE (III.F). There are 3 sections that are especially interesting for students at the beginning:

- Resources that contain teaching material can be downloaded

- Online lessons can be participated through the system
- Communication functionality to the advisors and other students is provided by chat, forum, private messages etc.

Students get introduced to the environment by their teachers after they visited the preparation course. This introduction is mostly quite short in time as many students are used to work with some kind of electronic communication tools. Therefore, they succeed by mainly following their intuition.

In the second phase, students are taught theoretical and practical concepts of the course content. In this course, they learn basic OOP concepts like information hiding, modularization or generalization / specialization. These theoretical concepts are directly used in the 3rd generation programming language "Java" in order to visualize them and show the effects. Java code is created in the professional IDE Eclipse [11]. Of course, a basic introduction to such a powerful tool also must be given to the students. All this content is also taught remotely in the online courses, i.e. we are just using our environment eEE to communicate to the students as advisors.

In the third phase, students practice concepts they have already learned during the second phase in order to raise the learning effect. Therefore, small examples are programmed which include a "HelloWorld" program or some kind of Banking Account example. In this example, for instance, students need to program an account at a bank as a class which fulfills several requirements. It needs to provide data fields for the account number or the account balance, which should be hidden in the implementation (information hiding). Furthermore, different kinds of sub-accounts should be implemented for families or companies (generalization / specialization). While implementing these small examples, students mainly reflect actively what they learned before and this deepens their understanding.

In the last phase, students should use all the contents of the course so far to manage one big problem. Typically, they have 8 to 10 weeks to provide a solution; teamwork is favored a lot in this phase. As an example, we wanted the students to implement a multi-user MP3 player which could be controlled through many workstations. Before, we taught them the usage of the web framework JSF [5] with the programming tool Eclipse WTP [7]. Figure 10 presents a screenshot of one of our students' solution. Different paths to the folders containing music can be configured. Furthermore, different strategies can be chosen to merge different playlists from different users. It is possible to merge them having equal rights or setting priorities as well as to configure the number of songs that needs to be waited until a song can be repeated.

In the end, students had to present their solutions in order to get feedback from other students or (through watching other solutions) getting more insights what could have been solved more elegantly in their own solution.

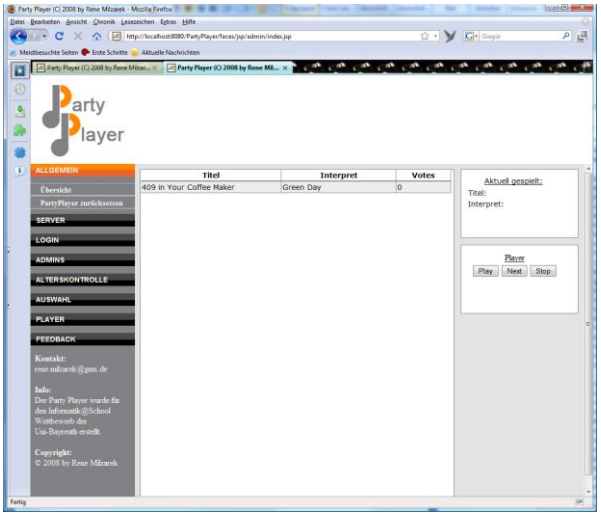

Figure 10. Multi-user MP3 Player

Altogether, we observed a very high level of the provided solutions. Furthermore, the students' ability to structure and solve problems was obviously very considerable and they only needed very little help to build up own solutions.

V. EXPERIENCE OF REMOTE TEACHING

A. Technical Environment

Main goal of a technical environment used in a school project must be usability. We increased usability a lot by designing and developing eEE. Using this software, most obstacles for students from a technical point of view were eliminated. Additionally, it enables us to realize the didactical concept in a more appropriate way.

B. Preparation and Organization of Online Teaching

The in-service teacher training turned out to be an excellent preparation of the teachers who care for the students in the "online schools". They are integrated in the project from the beginning. Most of them take part in the online lessons and some also offer additional mentoring for the students.

Together with the teachers we organize information meetings in every school. During these meetings we introduce our courses to all interested students. This face-to-face event is very important for the students to meet their online advisors on site.

The students enrolled for the courses using an online form on our website. We collected all registration data in a database; this data was used to create accounts automatically.

It is not possible to bring all online students together on one fixed day. We offer each online lesson twice on two different weekdays. The video streams of recorded lessons can be used by students who cannot take part in online sessions; these streams are also used frequently to revisit lessons. Additionally students can ask questions by email, phone, chat or forum.

After the first third of the course we visit all online schools to intensify the personal contact with the students. This helps to hold up motivation and it is a useful feedback for us as online advisors.

C. Assessment of Online Teaching

As mentioned above up to three schools take part in the project in on site lessons. These groups play an important role in the online concept. Before an online lesson is held we use the corresponding on site lesson as a kind of dress rehearsal. The advisor receives direct response from the students and recognizes problems in the learning process. These are important experiences for the online lesson.

To nearly all students our student centered approach described in the "I-You-We" was completely new. They mostly had no experience in solving more complex problems on their own or in cooperating with a partner during a lesson. But after some lessons we observed in the on site classes that students started working together. Although each student had his own computer they sat in front of one computer in groups of two or three students. They started to discuss about the exercises and began to cooperate. After planning a strategy for a solution all students of a group returned to their own computer and started their work on a sub-problem.

In online classes this process took much longer. In the beginning it was harder for them to cooperate in small groups during online lessons. But the less information they received from the online teacher the more self-dependent they became. In the end of the third step "Applying concepts learned in simple exercises", described in II.C.3), there was nearly no difference between the two groups. Some online students described their way of cooperation during a meeting at their school. They explained that they also met in their spare time to solve the exercises in groups.

The on site classes can also be compared with the online classes directly. At the moment this is more an informal comparison than a statistical assessment. The number of participants is too small to receive valid results. A statistical evaluation is already planned for the next year project.

But also the subjective experiences show interesting details.

The solutions of exercises show no significant differences in quality. Although we expected a discrepancy in speed of learning between the two groups we could not confirm that. The withdrawal rates for the volunteer courses do not differ.

"Online students" had no problems using our learning environment. Some students seem to have stoppages asking questions during the lessons using the voice communication tool. They mostly use the chat function.

VI. Conclusion and future Work

The technical concept of our e-learning environment follows the content and didactical concept ("I-You-We"). Each component of our infrastructure plays a dedicated role in teaching and learning. We use server based web services to minimize client side problems. Based on our experiences, we integrated bidirectional real time audio, web cam and screen content transmission and file access in one web based e-learning environment as a Moodle [19] extension.

Our experience shows that student centered learning is successful in online teaching. To achieve this we had to lead the students from a more teacher centered to a self dependent way of working step by step. Our technical environment builds the base for communication and cooperation of students among each other and with the online advisor.

Nevertheless, face-to-face contact between advisors and students should not be underestimated. It is necessary for students to hold up motivation and to get personal feedback. To lessen this disadvantage we implemented web cam transmission of the advisors themselves during online lessons to personalize the contact between students and advisors.

Also an on site contact person is very helpful to solve problems concerning local conditions. These local teachers are involved in the whole project, accompany the students and give feedback to the online advisors.

In future, we plan to adapt our teaching and learning concept in combination with our technical environment eEE to other domains; for instance, university language courses or further education at operational level in companies can be realized with our approach.

References

[1] M. Goetz, M. Ehmann, S. Jablonski, and M. Igler, "Experiences in Online Teaching and Learning", First International Workshop on Virtual Environments and Web Applications for e-Learning, with ICIW 2008, IARIA, Athens, 2008.

[2] Adobe PDF, http://createpdf.adobe.com, last revisited 2009-05-11.

[3] K. Beck, Extreme Programming Explained: Embrace Change, Addison-Wesley Longman, Amsterdam, 1999.

[4] Z. Berge, "Computer - mediated communication and the online classroom in distance learning", Computer-Mediated Communication Magazine, Vol. 2, Number 4, Hampton Press, Cresskill (NJ), 1995.

[5] H. Bergsten, JavaServer Faces, O'Reilly Media, 2004.

[6] D. Bocka, C. Miller, and M. Ehmann, "Teaching and Learning Mathematics with Dynamic Worksheets", International Journal of Continuing Engineering

Education and Life-Long Learning (IJCEELL), Volume 18, Issue 5/6, Inderscience Publishers, Geneva, 2008.

[7] N. Dai, L. Mandel, and A. Ryman, Eclipse Web Tools Platform. Developing Java Web Applications, Addison-Wesley Longman, Amsterdam, 2007.

[8] B. Eckel, Thinking in Java, Prentice Hall, Upper Saddle River (NJ), 2006.

[9] M. Ehmann, "Roboter zum Anfassen und virtuell – Problemlösendes Arbeiten im Informatikunterricht", in Spektrum – das Wissenschaftsmagazin der Universität Bayreuth, Universität Bayreuth, Bayreuth, 2006.

[10] D. Frayer and L. West, Creating a new world of learning possibilities through instructional technology, http://horizon.unc.edu/projects/monograph/CD/Instructional_Technology/Frayer.asp, last revisited 2009-05-11.

[11] S. Holzner, Eclipse, O'Reilly Media Inc., Sebastopol (CA), 2004.

[12] K.C. Hopson and S.E. Ingram, Developing Professional Java Applets, Sams Publishing, Indianapolis (IN), 1996.

[13] Internet Explorer, http://www.microsoft.com/windows/products/winfamily/ie/default.mspx, last revisited 2009-05-11.

[14] J. Kaplan, jvoicebridge, https://jvoicebridge.dev.java.net, last revisited 2009-05-11.

[15] A. Kay, "The Early History of Smalltalk", History of Programming Languages Conference (HOPL-II), Preprints, Cambridge (MA), 1993.

[16] J. Keogh and M. Giannini, OOP Demystified, McGraw-Hill/Osborne, Emeryville (CA), 2004.

[17] E. Klieme and J. Baumert, TIMSS – Impulse für Schule und Unterricht, Bundesministerium für Bildung und Forschung, Bonn, 2001.

[18] Massachusetts Institute of Technology – Media Lab – Lifelong Kindergarten Group, Scratch, http://scratch.mit.edu, last revisited 2009-05-11.

[19] Moodle, http://docs.moodle.org, last revisited 2009-05-11.

[20] OECD, PISA Results, OECD, 2000, 2003, 2006, http://www.oecd.org, last revisited 2009-05-11.

[21] B. Oberhaitzinger, H. Gerloni, H. Reiser, and J. Plate, Praxisbuch Sicherheit für Linux-Server und Netze, Hanser Fachbuchverlag, München, 2004.

[22] R. Palloff and K. Pratt, Lessons from the Cyberspace Classroom: The Realities of Online Teaching, Jossey-Bass, San Francisco (CA), 2001.

[23] W.H. Peterssen, Lehrbuch Allgemeine Didaktik, Ehrenwirth, München, 1983.

[24] Pong Arcade Game, http://en.wikipedia.org/wiki/Pong, last revisited 2009-05-11.

[25] M. Powell, jMonkey Engine User's Guide, http://www.jmonkeyengine.com/wiki/doku.php?id=user_s_guide, last revisited 2009-05-11.

[26] Qualifications and Curriculum Authority, The key skills qualifications standards and guidance, Qualifications and Curriculum Authority, London, 2004.

[27] Real VNC, http://www.realvnc.com, last revisited 2009-05-11.

[28] J. Russell, InnoSetup, http://www.jrsoftware.org/isinfo.php, last revisited 2009-05-11.

[29] A. Sikora, Technische Grundlagen der Rechnerkommunikation, Fachbuchverlag Leipzig, 2003.

[30] M. Spitzer, Lernen. Gehirnforschung und die Schule des Lebens, Spektrum Akademischer Verlag, Heidelberg, 2006.

[31] Squeakland, http://www.squeakland.org, last revisited 2009-05-11.

[32] Teamspeak, http://www.goteamspeak.com, last revisited 2009-05-11.

[33] V. Ulm, Objekte in Grafiken, Z-MNU Universität Bayreuth, Bayreuth, 2003.

[34] W3C, Architecure domain, Naming and Adressing: URIs, URLs, …, http://www.w3.org/Addressing/#rfc3986, last revisited 2009-05-11.

[35] WebDAV, http://www.webdav.org, last revisited 2009-05-11.

[36] Windows Meta File, http://www.microsoft.com/windows/windowsmedia/de/format/default.aspx, last revisited 2009-05-11.

[37] A. Zucker and R. Kozma, The Virtual High School: Teaching Generation V, Teachers College Press, New York (NY), 2003.