# Problem solving in an interactive Internet-based learning environment

*Rainer Lütticke, Hermann Helbig*

FernUniversität Hagen

**Key words:** *Web-based learning, Interactive learning, Virtual laboratories*

## Abstract:

*The virtual electronic laboratory (VILAB) was developed for Internet-based teaching in computer science at the FernUniversität Hagen, the biggest open and distance university in Germany. There VILAB is used for regular teaching since summer 2002. The learning environment is realised as client-server system in which the clients are connected via Internet (remote-login and browser) with the server of the university. In this way the students get fast access to complex software-tools for their work on exercises in VILAB divided by their topics. At the same time the server indirectly controls the content of a browser opened on a client. Additionally, an interactive tutoring component, based on the newest theories for intelligent and adaptive feedback, supports and motivates the students during their problem solving processes.*

## 1 Introduction

The opportunities which distance learning provides of tailoring studies to suit the individual time budget gives people, who cannot study on a conventional university due to their personal situation (work, family, disability, etc.), a chance of a real university education. Distance learning is the ideal form of lifelong learning, continuing education, and advanced training. However, disadvantages are sometimes difficulties at self-studies and time-consuming and expensive travels to courses which need the presence of the student on the campus of the university (e.g. practical courses). By its fundamental decision in favour of the blanket use of the New Media and the development of the Education and Knowledge Space: Virtual University[1] [11], the FernUniversität has phrased its response to society‘s demands for more flexible education and new models of lifelong learning. In this context the FernUniversität was also searching for new concepts of practical courses using the virtual mobility, new ways of communication, and modern technologies for teaching.
To reach this aim for the field of applied computer science we developed a conception for a learning environment originating from the laboratories used for the education in natural sciences. In these labs the students are guided by a curriculum to several different experimental stations. There they can work on exercises and additionally, can test and deepen their theoretical and methodical knowledge getting support by an assistant. This paradigm is used for the realisation of our Internet-based learning environment for the education in applied computer science at our university. Therefore we have called it "virtual electronic laboratory" (VILAB). Its functionality is described in chapter 2. Analogue to traditional practical courses

[1] http://vu.fernuni-hagen.de

the learning content in VILAB is divided in several stations inside of the lab. There the students have access to several software tools, on which they work on their experiments and problems (chapter 3). To integrate functions of a personal human assistant of a real lab into our learning environment and not to leave an user with a problem during a session in the virtual lab alone an interactive tutoring component described in chapter 4 is implemented in VILAB. We are closing with evaluation results and a conclusion in chapter 6.

# 2 Functionality of VILAB

## *2.1 Architecture*

VILAB is realised in a client-server architecture in which the Linux-based server is operated at the FernUniversität. The server saves and manages the content of the lab via a navigation tool for the students. The server also harbours the tutoring component (Java) and the software-tools and provides the communication infrastructure to databases and WebAssign, which is an established online system for standard exercises (e.g. multiple choice) with modules for automatical correction of such standard exercises and allows the integration of further modules for the analysis of more complicated problems [2], [7]. This system is in VILAB integrated in such a way that users of the lab do not notice when they work with functions of Web-Assign.
The access to the server is managed by decentral user clients (Windows- or Unix-based computers) which must be connected with the Internet. Via remote-login to the server the students are directly routed to a navigation tool which is written in the Lisp dialect Scheme and is one of the two user interfaces (Fig. 1). After a login the server starts a new instance of the lab and the send/receive system of the tutoring component. Thereby the students cannot get access (intended or not) to the Linux-system of the server. For the display of the content of the lab the student has to open a browser on his client with an individual URL as second user interface (Fig. 1). Via navigation tool local hypertext or PDF documents as well as documents out of the WWW can be selected by buttons. An action which shall lead to a display of a hypertext document result in an automatic reload in the browser of the user. This is realised by loading local documents in the individual directory of the user inside of VILAB and copying them to the URL of the user. PHP scripts inside of the hypertext document notice the change of the content in the user's URL and start the reload. In this way it is possible that actions on the server indirectly control the browser of the client. Documents of the WWW are embedded in a frame so that higher-level PHP-scripts can manage also the display of this kind of documents in the browser of the user.
Via navigation tool the software-tools installed on the server can be activated. These tools can communicate via sockets with the tutoring component leading to the fact that every user have individual ports for this socket communication.
Since every user possesses individual URLs, directories and ports inside of VILAB it is possible for the system to manage handling, saving, and display of data of many users at the same time.

## *2.2 User access*

Realising the access to the learning environment to the main fraction of the students it is important to support the client operating systems Windows and Linux since we know by surveys that ~75 % of our students use Windows, ~20 % Linux, and only ~5 % other systems. Therefore we support access for these two operating systems and other Unix-like systems, of

course. Additionally, we demand for the client only a x86 processor and any standard browser (e.g. Netscape or Explorer).
The remote-login to the server of VILAB (via ssh) is for students using Linux trivial. Windows user have to install the open source packages *cygwin* and *xfree86.4.x* to get the analogous Unix functions needed for the access. The students can either download these packages from our web pages or ask for a CD produced at our institute and containing all needed software packages.
This kind of access has the advantages that all existing tools for Unix can easily be integrated in our learning environment, we do not need any commercial software, the students (only Windows user) need only free software which can be easily and in a short time installed, and we have large technical freedom for the realisation of our didactical concept. On the other side the exclusive use of a browser environment seems to be simpler at the first glance, especially for the Windows user, but there are some large disadvantages. Since one of our main purposes is the integration of software tools, in this alternative variant we had to rewrite the tools as Java-Applet (in fact this is not possible due to the complexity of tools) or had to run the tools within other systems (e.g. “Tarantella” of the company SCO) on a special server redirecting in- and output interfaces of the tools to a Java-Applet. The disadvantage of this solution is the dependency for a key component of VILAB on the availability of future product support and on license fees for the commercial systems. Additionally, and this is even more important, tests have shown that the reaction time of the software tools using the remote-login is much smaller than in the redirection-Java-Applet variant.
Offline working with a CD-ROM have even larger disadvantages and the both advantages, saving money and faster reactions of a system, are today weaker arguments than in former times. Like in the remote-login variant Windows user had also to install the open source packages for the Unix-based tools. However, offline working is not suitable for the aims of our practical learning environment because some tools need the Internet access (communication with other systems), collaborative work would be excluded and access to databases for generation of adaptive feedback would not be possible. If students like to work offline they have also in the realised remote-login variant the possibility to download all text documents of VILAB.

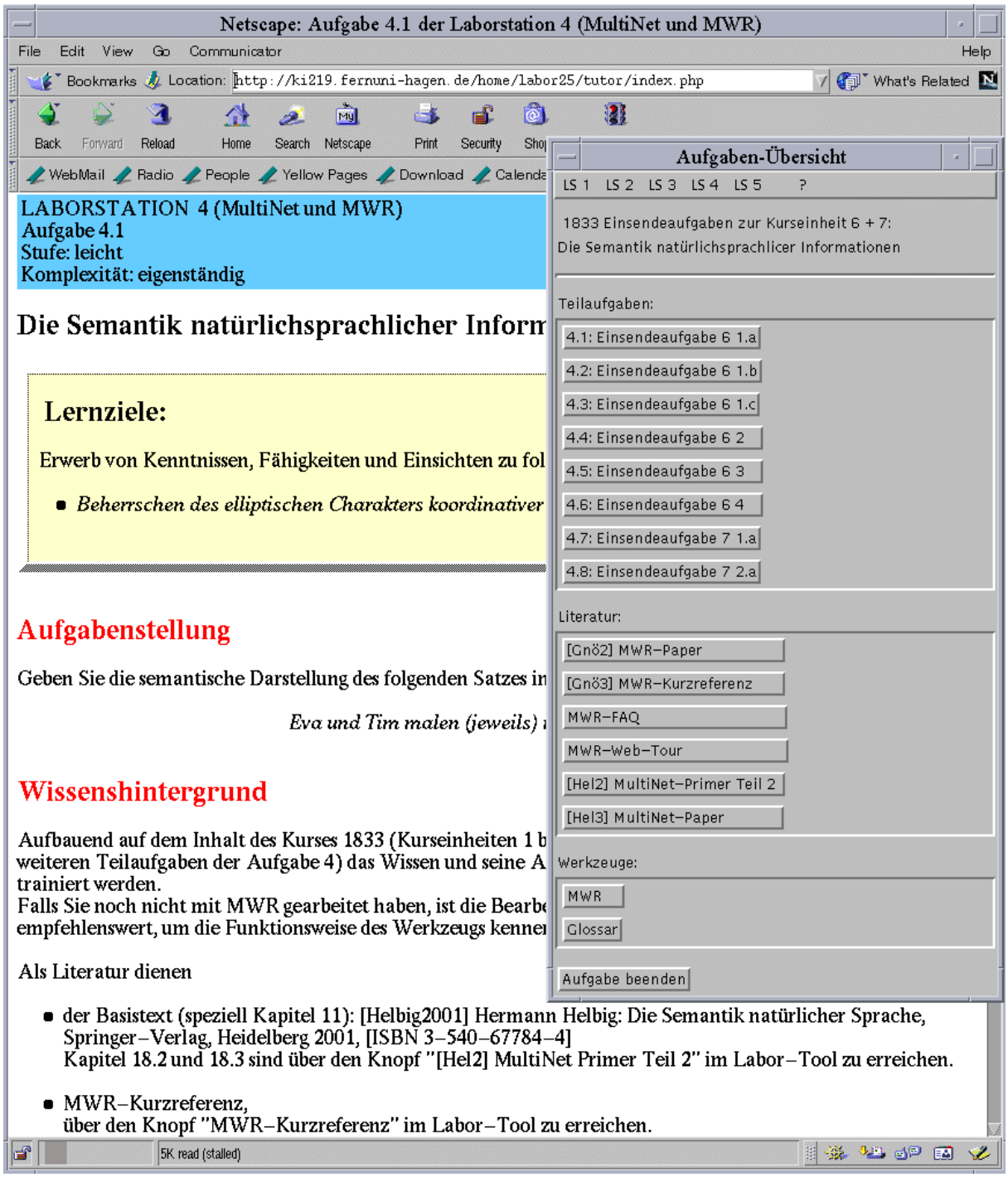


Figure 1: Screenshot of two user interfaces of VILAB. The browser window is supposed by the navigation tool. (Of course in practise the user places the windows side by side.)

# 3 Presentation of the exercises and problems

Conception and platform are nearly independently developed from the teaching content. In this way new laboratory stations, which contain the content, can be added and further Unix-based software tools can easily be integrated. Currently there are five stations in the laboratory: Programming with the languages Java, C, Prolog, and Scheme (procedural, object orientated, functional, and logical), neuronal networks, relative databases, as rule based systems (semantic networks and natural language transformation in SQL), and computer linguistics.

Every station has a main page giving information about purpose of the station, background, reference to other stations, and an overview about the exercises and software tools in the station. Via navigation tool the students can select the different exercises (Fig. 1, right) presented on hypertext documents which are divided in learning targets, description of the task, relevance of the exercise for the field of knowledge, knowledge background for solving the exercise, helpful literature, description of the required software tools, characteristics of the solution, and kind of control/help by the tutoring component (Fig. 1, left).
The students can either work on the given exercises and problems with tutoring help or get practical experience by free work with the software tools (e.g. request on databases, constructing neuronal networks with the SNNS[2] [12], or transform natural language information in semantic networks using the graphic knowledge engineering tool MWR[3] [4]).
For the creation of a new laboratory station and new problems we have developed a hypertext-based tool for the authors. However, the tutoring module for each problem has to be developed by the authors themselves. As help they get a commented list of interfaces and some examples.

# 4 Tutoring component

While a student is working on an exercise or in a software tool, the send-receive system of the tutoring component has two functions: "error diagnosis" and "error correction". Additionally, the tutor saves the student's result of an exercise, which was requested for correction, via the WebAssign system or the software tools. In this way the student starts every new editing of an exercise with his last result and a teacher has the possibility to examine results of the students. Depending on the learning target a possibly existing model result can be presented to the student. However, in most cases an analysis of the student's solution is didactically better.

## *4.1 Error Diagnosis*

Diagnosis means in this context the decision whether a solution is correct or not and to find out what and why (explaining module) something is wrong or incomplete. The tutoring component has an passive and an active error modus for two different kinds of problem solving support which are discussed in analyses of ITSs (Intelligent Tutoring Systems) [3]. In the passive modus ("intelligent analysis of student solutions") the send-receive system of the tutoring component has to identify errors from the student's final result after he has requested for a correction. This modus is needed for the non interactive (therefore the term "passive") programs (e.g. SQL, Java) in which errors are only detectable by a complete analysis of the student's input. This analysis is realised by correction modules and possibly following processes. The output of such an analysis integrating also an student model is an error code with accompanying parameters. The student model is based on data collected during the student's work in the laboratory and saved in a MySQL database. In the active modus ("interactive problem solving support") the send-receive system of the tutoring component does not wait for the student's final solution. If an interactive software tool detects an error, it sends at once the specific error code with accompanying parameters to the send-receive system of the tutoring component. Thereby the student is provided with intelligent help on each step of problem solving.

[2] SNNS: Stuttgart Neuronal Network Simulator
[3] MWR (MultiNet-Wissensrepräsentation) is a software tool for the graphic representation of MultiNet terms. In this graphic environment the terms can be produced, displayed, and manipulated. MultiNet (multilayered extended semantic networks) is a paradigm of knowledge representation created for knowledge processing and specially for the semantic representation of natural language information [6].

### 4.2 Error Correction

The error correction has the goal to show the student how to remove the errors asserted in the diagnosis by identification of missing or incorrect knowledge which is responsible for errors. The overriding goal of the tutor is the finding of the logical errors and not the complete listing of syntax errors as in standard program diagnosis systems. The error diagnosis and the proposals for the problem solving are derived from the error codes and the accompanied parameters. These codes are processed by the send-receive system of the tutoring component with an error table. Every error code leads to the display of hypertext documents which are individually modified by Javascript variables extracted out of the parameters. The content of the documents includes a statement about the correctness of the solution, the description of possible errors, hints to the avoidance of these errors, possibly a listing of useful literature to derive lacking knowledge, an assessment of the student's performance, and a motivation [10]. In this way the students get adaptive feedback to their individual solution.

## 5 Conclusion

The first use and the didactical testing took place in summer 2002 with concrete content of the field applied computer science. Since this summer term VILAB is used for teaching at the FernUniversität Hagen. Results of evaluations show that the system works stabile, the access to VILAB (also for Windows user) is easy, the reaction times of the systems are very small, the students like the kind of learning with VILAB, its design, and the support and motivation of the interactive tutoring component. From the view of the teacher the most imortant point is that the students have learned more effective compared to former courses without VILAB [8]. Therefore we conclude that we reached our aims with our learning environment:

VILAB is training practical skills in computer science by working with complex software tools rendering unnecessary and unreasonable software installation for the work with them. The students get practical experience by the problem solving of different exercises and the fast automatic feedback motivates them to search for the right solution of a given problem. The advantages for the students are obvious:

- Instructional aspect: Coached learning by doing on practical exercises leads to a longer saving of the knowledge than pure reading of texts, the transfer of theoretical knowledge into practice and into strategies for solving of concrete problems is trained, and the monitoring of learning success can be managed [9], [5], [1].

- Mobility: The students can work independently of space and time. In this way especially students of a distance university save costs and time of a long and expensive travel to the university (traditionally necessary for practical courses).

- Software-tools: The students get access to tools which otherwise they could not use from home (installation to complicated, no rights of use, tools need too much disk space, wrong operating system, etc.).

- Motivation: The interactive tutoring component motivates and helps the students so that learning processes are supported.

An integration of collaborative software tools, corresponding exercises, and tutoring feedback in the future will be the next step to train also the social skills during problem solving in groups.

## References:


[1] Ballstaedt, S.-P. et al.: Planung, Entwicklung, Durchführung von Fernstudienangeboten. Deutsches Institut für Fernstudienforschung an der Universität Tübingen, 2000.

[2] Brunsmann, J.; Homrighausen, A.; Six, H.-W.; Voss, J.: Assignments in a Virtual University – The WebAssign-System. In Proc. 19th World Conference on Open Learning and Distance Education, Vienna, Austria, 1999.

[3] Brusilovsky, P.: Adaptive and Intelligent Technologies for Web-based Education. In: Künstliche Intelligenz, 13(4), pp. 19-25, 1999.

[4] Gnörlich, C.: MultiNet/WR: A Knowledge Engineering Toolkit for Natural Language Information. In Technical-Report 278, FernUniversität Hagen, Germany, 2002.

[5] Haynie, W.J.: Effects of Anticipation of Tests on Delayed Retention Learning. Journal of Technology Education, 9, 1997.

[6] Helbig, H.: Die semantische Struktur natürlicher Sprache: Wissensrepräsentation mit MultiNet. Springer, Berlin, 2001.

[7] Lütticke, R.; Gnörlich, C.; Helbig, H.: Internet-basierte Lehre im Rahmen von VILAB. In Schubert, S., Reusch, B.; Jesse, N. (ed.), GI-Lectures Notes in Informatics (LNI), P-19, “Informatik bewegt”, Bonner Köllen Verlag, 2002, pp.293-296.

[8] Lütticke, R., Helbig, H.: Das Virtuelle Informatik Labor als Teil des Lernraums Virtuelle Universität der FU Hagen. In von Knop, J.; Haverkamp, W. (ed.), GI-Lectures Notes in Informatics (LNI), “Moving Expertise”, 17. DFN-Arbeitstagung über Kommunikationsnetze, Bonner Köllen Verlag, 2003, in press

[9] Morrison, G.R.; Ross, S.M.; Gopalakrishnan, M.; Casey, J.: The Effects of Feedback and Incentives on Achievement in Computer-Based Instruction. Contemporary Educational Psychology, 20:32-50, 1995.

[10] Musch, J.: Die Gestaltung von Feedback in computerunterstützten Lernumgebungen: Modelle und Befunde. Zeitschrift für Pädagogische Psychologie, 13:148-160, 2000.

[11] Schlageter, G., Mittrach, S.: Virtuelle Universität. In Informatik Forschung und Entwicklung, Volume 13, 1998, pp. 159-162.

[12] Zell, A.: Simulation neuronaler Netze. Oldenbourg, München, 1997.


## Authors:


Rainer Lütticke, Dr.
Hermann Helbig, Prof. Dr.
FernUniversität Hagen, Department of Computer Science,
Intelligent Information and Communication Systems
Universitätsstr. 1, 58084 Hagen, Germany
rainer.luetticke@fernuni-hagen.de