VILAB - A Virtual Electronic Laboratory for Applied Computer Science

Rainer Lütticke, Carsten Gnörlich, Hermann Helbig
University of Hagen, Department of Computer Science,
Chair of Applied Computer Science VII, Intelligent Information and Communication Systems,
Universitätsstr. 1, D-58084 Hagen, Germany; rainer.luetticke@fernuni-hagen.de

Abstract
The virtual electronic laboratory for applied computer science (VILAB) is created for Internet-based distance education at the FernUniversität Hagen, since the transfer of theoretical knowledge of students into practical problem solving without physical presence at an university is an important teaching target as well as an important administrative facilitation. In addition, students having a basic knowledge in a specific field of applied computer science shall get a deeper insight into this field within a motivating learning environment. For this purpose an intelligent tutoring system (ITS) is implemented in VILAB realized as a client-server system. The interactive tutoring system supports the process of problem solving by an analysis of results from exercises, by hints, if it detects deficiencies in the knowledge of a student, and by verification of the student's performance. The reactions of the tutoring system are based on error diagnosis and correction using a lookup table in which the student's errors are anticipated. Therefore no individual student models are needed in VILAB. The knowledge domains of the tutoring system are separated in laboratory stations: different programming languages and rule based systems including databases and information retrieval.

1 Introduction
The conception of the VIrtual electronic LABoratory for applied computer science (VILAB) originates 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 their theoretical and methodical knowledge. This paradigm is used for the realization of the virtual laboratory for the education in applied computer science at universities. Therefore the users of VILAB will also be called "students". Different to traditional practical courses in VILAB the students get access to several software tools, with which the students work on their experiments.

VILAB has three different aims. The instructional aspect is that simple learning by reading of texts or hearing of lectures do typically not lead to the expected success. Three main reasons explain this fact: The knowledge is saved by students only for a short time. The transfer of theoretical knowledge into practice and into strategies for the solving of concrete problems is not possible, and the monitoring of learning success fails (Universität Tübingen 2000). Such effects can often be prevented by the coached work on practical exercises. Therefore VILAB assigns such exercises out of the field of applied computer science and includes an interactive tutoring component (later also called "tutor"). This component helps the students at their work in the laboratory, if they have problems to find the correct solutions.

From the administrative perspective the aim is the creation of a virtual laboratory which relieves the staff of universities for applied computer science. Such a support is necessary since in Germany the growing number of students in computer science is not coupled with a growing number of staff personal.

Thirdly students using VILAB have large personal advantages. They save the time and costs of a long and expensive travel to a real laboratory. This point has to be taken especially into account at distance universities. Furthermore, the students can decide themselves at what time they want to work in the laboratory. This freedom has also instructional advantages we will discuss later.

2 Layers of VILAB
VILAB can be divided in three layers reflecting the typical aspects of an e-learning tool. The technical layer includes all information about software, interfaces, and architecture. The functional layer describes the content of the technical components and the instructional system design reveals the learning method VILAB is based on.

2.1 Technical Layer
The virtual laboratory is realized in a client-server-architecture. The teaching components (the tutor and the systems for the laboratory stations) are installed on an Unix server. This system is open to user clients via the Internet and thereby accessible for the students. To have an uniform communication between server and client only the tutoring component is directly connected with the

clients. After an user has started an instance of the laboratory on his computer a graphical user interface (written in the Lisp dialect Scheme) and a browser (e.g. *Netscape*) are displayed on the screen and the send-receive system of the tutoring component (Java) is started. During a session these and further processes are controlled by the VILAB system. Inside this system every user gets individual ports and directories so that the display of data and the saving of files can be managed. In this way VILAB can be used by many students at the same time.

Depending on the actions of the user VILAB starts the WebAssign system[1] (Brunsmann et al. 1999), software tools, or HTML/PDF-documents. These documents belong either to the VILAB system itself, to the "Virtual University"[2], or are parts of the WWW. Out of the HTML-documents inside the WebAssign system requests can be sent via CORBA to servers (so called correction-servers written in Java). In such a case these servers initiate reactions in the VILAB system. After possible subsequent processes (e.g. compilation of programs) are managed, the send-receive system of the tutoring component, which communicates via sockets with the rest of the VILAB system, creates the final answer. For this purpose a HTML-document including Javascript files is loaded into the individual directory of a VILAB user and is displayed.

The software tools communicate directly via sockets or indirectly via the WebAssign system with the send-receive system of the tutoring component. Depending on the content of the software tools and the functions of the correction-servers both, tools and servers, can be joined with compilers for the respective programming language, a relational SQL database, or other software components.

The software implemented in VILAB can be divided in five categories. Newly developed software includes the components of the platform of VILAB (control of the processes inside VILAB, functionality of the user interface, interfaces to the different systems in- and outside of VILAB, and send-receive system of the tutoring component), the correction-servers, and the software for the creation of the answers of the tutor on requests. Corresponding to the needs of VILAB the tool for knowledge representation (MWR[3], Gnörlich 2000) was modified and is fully developed. Additionally, VILAB uses administrative functions of WebAssign (e.g. access of data of the students and saving of results of exercises) and standard correction-servers for automatical correction of exercises without the use of the send-receive system of the tutoring component of VILAB. WebAssign is an open source project inside of "CampusSource"[4] and is specially adapted for VILAB. Furthermore, commercial software components, compilers and Oracle databases, are implemented in VILAB. The last category is formed by gateways. They are used by the tools of VILAB, but are not part of it (e.g. the natural language interface, Helbig and Hartrumpf 1997).

2.2 Functional Layer

Via the user interface the students navigate inside of VILAB. Using this navigation menu the different laboratory stations of VILAB can be selected.

The implementation of different programming languages in VILAB is indispensable since programming is a substantial content of practical courses in applied computer sciences. In VILAB the students can experiment with the four most important programming paradigms: procedural, object oriented, functional, and logical programming. This aim is reached by the implementation of the languages C, Java, the Lisp dialect Scheme, and Prolog as laboratory stations. The students can either work on exercises with tutoring help or write own programs with display of their results (only text, no graphics) as help for the exercises or to get experience with a computer language.

The knowledge engineering tool MWR is used in the station "rule based systems" for the fields of knowledge acquisition and rule based inferences and transformation into requests on databases. The students need this tool for the creation of solutions (e.g. graphics, rules, answers to questions) of exercises. Additionally, they can experiment with MWR. In this way this station also touches the forthcoming stations "technology of databases" (teaching the concepts and functionality of database systems, the basic principles of the architecture, and the implementation of data) and "information retrieval" (including the search for information in databases, MULTINET knowledge bases, and in networked digital libraries or at similar data provider via the Internet, Helbig et al. 2000).

Depending on the laboratory station and the software tool VILAB offers different kinds of activities for the students: navigation in VILAB via user interface, reading of literature in a browser or in an Acrobat Reader window, graphical real time actions in the editor window of MWR by mouse control, actions in the menu of MWR, file upload via WebAssign, writing into textareas or marking of buttons in HTML-documents of exercises,

[1] WebAssign is a system for the realization of exercises of lectures in universities via the Internet developed by the FernUniversität Hagen.

[2] "Virtual University" means here the transfer of teaching, research, and administration of the FernUniversität Hagen into virtual space via the information technology and infrastructure of the Internet (Schlageter and Mittrach 1998).

[3] MWR(MULTINET-Wissensrepräsentation) is a software tool for the graphical representation of MULTINET terms. In this graphical 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.

[4] http://www.campussource.de

or starting a request for the correction of an exercise to the send-receive system of the tutoring component.

2.3 Instructional System Design

Real labs are only useful for students having some background in a specific field of science. In the same way our virtual laboratory is not a learning tool for basic knowledge. Before an user of VILAB starts to work in the laboratory this knowledge should previously be acquired elsewhere (e.g. lectures or literature). However, the student's knowledge has not to be wide. The difficult transfer process of theoretical into practical knowledge is supported in VILAB by instructional transaction. This particular interaction with a student in an e-learning system is characterized by a mutual, dynamic, real-time give-and-take between an instructional system and a student with an exchange of information (Merril et al. 1992).

The user of VILAB gets practice by working on exercises assigned to a laboratory station out of a pool (which will grow with the time). The important concept of VILAB is that the problem solving happens with the assistance of an interactive tutoring component. The differentiated reactions of the tutor directly point to the error of the student. The tutor supports him, if it finds missing knowledge, and evaluates his solutions. Furthermore, the possibility of own experiments with the software tools will additionally promote the student's practical knowledge. In this way VILAB applies the last four out of nine learning steps (utilization of knowledge, feedback of controlled results, evaluation of learning success, and ascertaining of knowledge saving and knowledge transfer) of the instructional design theory by Gagné et al. (1988). VILAB can also be used as a constructivistical learning environment, in which learning is problem based and takes place in the practical use of knowledge, if the exercises of a laboratory station are the starting point (with some basic knowledge) for the learning of a domain.

The differentiated feedback on the actions of the user by the tutoring system will motivate the students because together with the assistance of the tutor they have a larger chance or it is easier to solve the problems of exercises. The immediate feedback of the tutor about their learning success will give additional motivation. Thereby the necessity of the students for their extension of competence and for autonomy (Creß 1999) can be satisfied. Additionally, the variegated reaction of the tutor will keep the attention of the students alive. Illustrating at first the learning target and the knowledge background of an exercise shows the students the relevance of their doing. Thus VILAB includes all main categories of the ARCS model (Attention, Relevance, Confidence, and Satisfaction) for the promotion of learning motivation by Keller et al. (in Reigeluth 1983).

3 Tutoring Component

The tutoring component is the central part of VILAB transferring the instructional system design. Via user interface introductory texts as instruction manual, guided tour, and lists of FAQs and of the responsible contact persons in case of problems are presented. Furthermore, the laboratory stations can be selected in the user interface. Every laboratory station welcomes the user with background information (HTML) about its content, learning target, and units of lectures. The dedicated software tools and exercises are introduced and requirements on the student's knowledge for a successful problem solving of the exercises are illustrated.

The exercises of a specific laboratory station are listed in the user interface. To guide the student for his selection they are divided in different levels by our experience with real practical courses in applied computer science

- very easy: transfer of knowledge to problems which are comparable to model problems in the accompanying instructional texts in VILAB or problems are trivial; ideal for beginners
- easy: new (resp. the accompanying texts in VILAB) and easy problems; ideal for advanced beginners
- moderately difficult: finding of solution need some experience; ideal for advanced students
- difficult: the student need a lot of experience in this field of science to solve the problem; ideal for students in their last phase of study

and in different degrees of complexity.

- separate: the exercise is not connected to others
- moderately complex: the exercise is connected with others mainly situated in the same laboratory station
- complex: the problem has a large extent including exercises out of different laboratory stations

A selection of an exercise leads to a display of a HTML-document. This document presents the learning target, the task, the relevance of the exercise for the field of knowledge, and helpful literature and explains what software tools have to be used, how the exercise is embedded in VILAB, how the solution has to look like, and how it is controlled by the tutoring component. Links to the literature and the software-tools can only be activated in the user interface to retain control about the processes in VILAB. All texts are graphically optimized integrating theories of screen design (Universität Tübingen 2000).

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. 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 via user interface.

3.1 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 active and a passive error modus for two different kinds of problem solving support which are discussed at large in analyses of ITSs (Intelligent Tutoring Systems) (Brusilovsky 1999). In the active 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 programs (e.g. SQL, Java) in which errors are only detectable by a complete analysis of the student's input. This analysis is realized by the correction-servers and possibly following processes (s. section 2.1). The output of such an analysis is an error code with accompanying parameters.

In the passive 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. In this way this tutoring component has not to be activated for the error finding. Thereby the student is provided with intelligent help on each step of problem solving.

3.2 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 sketched in Tab. 1. Every error code leads to the display of PDF- or HTML-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, and possibly a listing of useful literature to derive lacking knowledge. Additionally, the student can get concrete instructions, if technical actions are necessary to improve his solution of an exercise (s. Tab. 1).

In the literature (Lelouche 1999) such a tutoring system is called "frame-oriented" because to every anticipated wrong student solution or action inside this interactive system there exists a frame encompassing predefined tutoring actions.

Table 1: Table of errors

Error	Correction proposal	Action of the correction
code of the error: e.g. wrong result, incomplete result, or syntax error	*textual references:* display of HTML- or PDF-documents	*technical actions:* e.g. press button xy, give a specific answer, or kill process

3.3 Comparison with other tutoring systems

Following the study about distance education of the Universität Tübingen (2000) a CBT/CAI system is considered to be "intelligent" if it has functions which are based on the analysis of user requests and which influence the feedback of the system over several teaching/learning cycles. Using this definition the solution analysis of the tutoring system of VILAB can also be called "intelligent" with regard to our method of error diagnosis/correction (Brusilovsky 1999). Therefore we count the tutoring system of VILAB among the ITSs.

The traditional "frame oriented" ITS concept is selected for VILAB because students in a practical course should have a basic knowledge in the respective field in which they work. Therefore the typical main drawback of such systems, the dependence of course progression on the author's expectations and not on the learner's deep understanding and learning preferences (Lelouche 1999), is not relevant. Since the source of student's errors in a practical course is mostly not missing knowledge, but missing understanding of the connection between theoretical knowledge and concrete problems, the experience from real courses has shown that students make typical errors during the knowledge transfer. Thus, the solutions for these errors can be generalized. Therefore the ITS of VILAB concentrates on the complete recognition of errors and not on individual student models as in many modern ITS, since such models are not effective for all ITSs (Gugerty 1997). In this way the tutoring of VILAB is adaptive to the user by the specific reaction on the user's individual errors.

Different to the most ITSs (e.g. VC-Prolog-Tutor with a the similar concept as VILAB, Peylo et al. 1999) the tutoring inside of VILAB is not limited to one single

domain, but includes several domains represented by the laboratory stations. Thereby the students get the advantage that they have to be familiar and to work with only one system covering many fields of applied computer science. Such a system is adequate for the education on universities, since the goal of practical courses for students is not the training of experts in one single domain, but the overall education.

4 User Access

The users of VILAB have to work with tools for Unix, but they start VILAB from different operating systems. A typical problem is that a student has a Windows based computer as client but he has to use a tool for Unix in the virtual laboratory. To solve this problem we have analysed two different approaches and have picked out the better one after a test phase of VILAB. Both solutions imply that the user of VILAB works online in the virtual laboratory, since we have gained the experience that offline working via CD-ROM is not accepted by the students. The fear for damage resulting from the installation of software packages on their computer often used for their professional job is larger than the cost of online working.

4.1 Exclusive use of a browser environment

In this variant the commercial system "Tarantella" of the company SCO is used. The tools of VILAB run within this system on a special server. In- and output interfaces of the tools are redirected to a Java-Applet running inside of the Internet browser of the client. The advantage of this approach is that the student only needs a Java-compatible browser. However, we have rejected this variant because a key component of VILAB would depend on the availability of future product support and license fees for the Tarantella-platform have to be paid in respect of the number of clients. Since the number of students in computer science in Germany is growing and the first practical use of VILAB for teaching has shown that the community of the students seems to accept the concept of a virtual laboratory, we would expect too large costs.

4.2 Remote-login to the VILAB server

The alternative and realized approach is a remote-login of the students via *ssh* to a server, on which all tools of VILAB are installed. After the login the VILAB system starts directly and the students can navigate inside the laboratory using the graphic user interface (s. section 2.1). Students with an Unix client need no additional software. Only users of Windows have to install the free software *Cygwin* and *XFree*. Tests have shown that these small software packages are quickly and easily to install for students. This variant has the advantages that we do not need commercial software, have to pay only for CDs containing the software for Windows users, and have more technical freedom for design in VILAB.

5 Conclusion

We have presented a virtual laboratory with an Internet-based ITS as central part. Such systems still are rather rare inside the ITS area (Brusilovsky 1999), although the benefits of Internet-based education are clear: platform, time, and place independence.

The function of the tutoring system of VILAB is the support of students working on exercises or practical courses in applied computer science. For this purpose the system uses frame oriented intelligent analyses of student's solutions and interactive methods.

The conception and the platform of VILAB are almost independent on the content of the individual stations of the laboratory. In this way a continual addition of further stations in the future is possible to cover the whole field of applied computer science.

Currently a prototype of VILAB is being realized. It will be used in a seminar about intelligent information and communication systems in SS2002 by a larger number ($\sim$20) of students. After an evaluation of the system its future use will be in the context of a practical course at the FernUniversität Hagen.

Acknowledgments

This work is part of the project "medin" (Multimediales Fernstudium in der Medizinischen Informatik), which is supported by the BMBF inside the program "Neue Medien in der Bildung".

References

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

Brusilovsky P., Adaptive and Intelligent Technologies for Web-based Education, in Künstliche Intelligenz, 13(4), 1999, pp. 19-25

Creß U., Personale und situative Einflussfaktoren auf das selbstgesteuerte Lernen Erwachsener, Regensburg, S. Roderer Verlag, 1999

Gagné R.M., Briggs L.J., Wagner W.W., Principles of instructional design, New York, Holt, Rinehart, and Winston, 1988

Gnörlich C., MULTINET/WR: A Knowledge Engineering Toolkit for Natural Language Information, in:

Technical-Report 278, FernUniversität Hagen, Hagen, Germany, 2000

Gugerty L., Non-diagnostic intelligent tutoring systems: Teaching without student models, Instructional Science 25, 1997, pp. 409-432

Helbig H., Gnörlich C., Leveling J., Natürlichsprachlicher Zugang zu Informationsanbietern im Internet und zu lokalen Datenbanken, in: Sprachtechnologie für eine dynamische Wirtschaft im Medienzeitalter, Schmitz K.-D. (ed.), Wien, TermNet, 2000, pp. 79-94

Helbig H., Hartrumpf S., Word Class Functions for Syntactic-Semantic Analysis, in: Proceedings of the 2nd International Conference on Recent Advances in Natural Language Processing (RANLP-97), Tzigov Chark, Bulgaria, 1997, pp. 312-317

Lelouche R., Intelligent tutoring systems from birth to now, in Künstliche Intelligenz, 13(4), 1999, pp. 5-11

Merrill M.D., Jones M.K., Li Z., Instructional Transaction Theory: classes of transaction, in: Educational Technology, 32 (6), pp. 12-26

Peylo C., Teiken W., Rollinger C., Gust H., Der VC-Prolog-Tutor: Eine Internet-basierte Lernumgebung, in Künstliche Intelligenz, 13(4), 1999, pp. 32-35

Reigeluth C.M. (ed.), Instructional design theories and models: An overview of their current status, Hillsdale, NJ, Erlbaum, 1983

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

Universität Tübingen; Planung, Entwicklung, Durchführung von Fernstudienangeboten, Deutsches Institut für Fernstudienforschung an der Universität Tübingen, 2000