Individualization and Flexibility through Computer Algebra Systems in Virtual Laboratories

Sabina Jeschke
Berlin University of Technology
MuLF-Center, Department of
Mathematics & Natural Sciences
10623 Berlin, Germany
sabina.jeschke@math.tu-berlin.de

Thomas Richter
Berlin University of Technology
MuLF-Center, Department of
Mathematics & Natural Sciences
10623 Berlin, Germany
thor@math.tu-berlin.de

Abstract

Virtual labs enable field specific experiments and open them for collaborative and distributed usage. In order to realize comprehensive laboratory set-ups providing a scientific broadness and user adaptivity, several challenges regarding the integration of different software technologies have to be solved. We propose an eLearning framework consisting of a learner and a course model; exercises within this framework are supplemented by virtual laboratories and computer algebra systems. We discuss the potentials of this setup on the example of the laboratory VideoEasel and its interface to Maple.

1. Introduction

Studies in physics, computer and engineering sciences and other fields depend on a well funded mathematical education. Given that more and more of the computational tasks are solved by the computer today, the demand for *understanding the concepts* and *interpreting the results* of the electronically performed operations becomes a major task of the mathematical education. Teaching mathematics then, however, means that diverse preknowledge and varying interests of the different student groups and the varying learning targets of the different fields of study have to be taken into account. To support this broadness of audience however, one has to go go beyond the first generation of eLearning [6], which was often not much more than computer assisted document management.

Intelligent technology, providing enough flexibility to adapt to the requirements of the field and the learning process, is needed [7]. Therefore the concepts of 'Intelligent Assistants", which have been recently developed in the field of Artificial Intelligence have to be adapted to eLearning Systems and their components. Intelligent Assistants thus can provide the necessary adaptivity to usage patterns to support the learning or research process actively, provided the course material is structured properly. We will describe our proposition for a suitable structure in section 2.

The potential of the proposed course structure is that it allows the integration of components like virtual laboratories — see section 5 — and computer algebra systems (CAS). This provides students not only access to experiments and tools to evaluate the outcome of experiments, but also requires them to learn how to handle these tools on realistic data. This allows us to overcome one problem of traditional teaching methods: teacher-centered lessons provide the essential basic knowledge, but they hardly allow for a more active approach, nor do they allow for individualized training scenarios. Instead, we develop the handling of standard tools as they are used by working professional engineers and scientists on data gained by the students themselves as part of the course. Thus, we believe that New Media and New Technologies present a turning point in the educational system providing the base to support the necessary chances [6].

We will introduce the concept of virtual laboratories in section 5 on the example system VideoEasel, a laboratory for statistical mechanics demonstrated in section 6, and describe its interface to the Maple algebra system in section 7.

2. A Course Model for eLearing

If we desire less-static, but user-adaptive eLearning courses, then this requires that the available material

Proceedings of the Eighth IEEE International Symposium on Multimedia (ISM'06)
0-7695-2746-9/06 $20.00 © 2006 IEEE

within a course is well-structured and equipped with all the necessary meta-data to drive electronic agent programs, also called "Intelligent Assistants" in the following. Following the ideas of [12, 14, 9, 1, 2], it is the purpose of these agents to detect usage patterns and classify users in order to provide them with suggestions as which learning material fits them best.

We therefore propose to structure this learning material along the following three level hierarchy:

Courses on the Content Level:
A course is the coarsest unit we consider: the abstraction of a series of lectures held on a topic. Courses are represented by directed graphs whose edges are *Knowledge Atoms* of which each encodes one individual learning unit of a course. A learning unit in the field of mathematics could be a theorem, or a definition, or a motivation for a definition, a proof of a theorem and so forth.

The vertices in this graph are dependencies between learning units: a vertex is drawn from A to B if B is a precondition for A, i.e. B must be taught in order to make A understandable. Dependencies themselves are classified into three groups: hard *Requirements*, resulting from the ontology of mathematics. *Recommendations* that are useful for didactic purposes, though not imposed by the mathematical structure. And, finally, *Suggestions* an interested student might want to follow, but which are neither required for didactical nor for inner-mathematical reasons. A visualization of the combined *recommendation* and *requirement* subgraph is called "HASSE Diagram" in educational sciences [9, 1].

Exercises in the Training Level:
Knowledge atoms in the course network may also link to training units in the exercise network; in general, this is an n-to-m mapping as one knowledge atom might refer to more than one exercise, and one exercise might be useful for more than one knowledge atom. It is this, and the following level we are currently exploring within our virtual laboratory, and where our agent programs work on.

The exercise layer is again a directed graph; however, its edges are now representing *exercises*: one exercise defines learning material a student might want to use to repeat and train contents of the lecture. The vertices in the exercise network encode dependencies of the exercise units: a node A is linked to a node B if B is an exercise for a sub-problem that is required to solve A. Following these links, a student might be delegated to simpler sub-problems of a harder assignment.

Similar to the above, the vertices are annotated by the type of dependency, and we also find *requirements* for mathematically dependent sub-problems, *recommendations* and *suggestions* here. Requirements might be satisfied by more than one node, i.e. there are also cases were one out of several requirements is sufficient. It is exactly this ambiguity that allows the deployment of user agents: even though all exercises might provide the same learning material to the professional reader, they might be not equally suitable for all users. It is up to an assistant program to make a *suggestion* about the learning path through the exercise network that tries to achieve a learning goal; however, it should be left to the student to have the final say about the decision as it is important to have a system that does not try to patronize the learner by reacting in an incomprehensible way, and thus would rather cause more confusion than it would be able to help. The exercises a student picks over time define a path in the exercise network. We will call this path the *learning path* of the individual user. The learning path has to be recorded until its success is evaluated, let it be either by an independent mandatory test within the system or by an (external) exam, see section 3. The role of this evaluation is to update weights for the decisions the tutoring system performs in future sessions, individualized to the user group, see sections 3 and 4 for details.

Asset Level:
One exercise consists of one or several *assets* on the asset level hierarchy of the network, and form here again the nodes of a directed graph. An asset is one elementary operation that must be performed to solve an exercise. Vertices in the asset graph define now the reaction of the system on user input, e.g. performing the wrong operation would redirect the user to an asset that demonstrates why the proposed solution would not work, and hints could be given by the system. At this level, the graph is a representation of a *Storyboard* [5].

In our current implementation, the training and the asset level make use of virtual laboratories, to be introduced in section 5, to provide work items for the student to follow. The Agent program operating on the eLearning material for that interact with the laboratory to recognize how the learner performs on the stated problems. To this end, minimal programs called "evaluators" can be linked into the laboratory user interface at runtime, and collect data from the running experiment. This data is then used to drive decisions to be made within the storyboard at the asset level of the course.

3. Learner Model

The role of the learner in the proposed system is twofold: first, the learner is in the traditional role of

the recipient of the learning material. But then, by picking learning material and by participating in an intermediate or final evaluation, the learner *also* evaluates the learning path and thus drives the learning system towards providing more suitable learning material for future users.

In order to construct such a learning system, we need to make a couple of simplifications and assumptions on the learner: within our model, a learner is part of a community, defined by a common language and notation, and a common goal to be achieved by the study. Using the metaphor that this community is often identical to the visitors of one lecture, we call this the *audience* the user is part of.

Note that even though the same course material has to be taught to all above audiences, the notation, formulation and exercises to be given will differ. However, this does not necessarily impose that the exercise network will look completely different, and that exercises for one group will be unsuitable for the other. Specifically, the vertices on the content level dictated by the ontology of the field are likely to be independent of the audience.

Furthermore, we assume an objective method that qualifies the learning success after following a learning path through the exercise graph. This evaluation method should be, within all limitations we are of course aware of, objective enough to update the database of the tutoring system. This evaluation therefore defines, a posteriori, the learning goal to be achieved, and thus has to be defined by the teaching university staff, e.g. the professor responsible for the course.

We do not believe that a "credit system" that assigns credit points to users passing an exercise should be used to drive and update the decisions of the learning system. First, to achieve a uniform learning goal within an audience, all possible learning paths would have to provide the same, or similar learning units, which is hard to accomplish. Second, if an intelligent self-learning tutoring system is trained by the learners, then the optimal learning path is that providing the maximal number of credits for minimal effort. Given that a "lazy" learner would pick the easiest possible exercise providing a given number of credits, the system would be trained to optimize the wrong goal, namely best possible credits/lazyness ratio. This is different from maximizing the learning success unless we can really objectively assign credits to each exercise that measure their contribution to the learning path — but this might turn out to be much harder to realize than the initial assumption, namely that of an objective exam. In order to avoid learners to pick learning paths that are unsuitable to achieve the desired learning goal, i.e. to pass the exam, the number of choices offered to a learner has to be restricted. Within our learning system, training nodes are therefore qualified by metadata defining the audiences a node is suitable for.

4. Bayesian Learning

To improve the user adaption of the system, our learning system includes a Bayesian decision system that aims at finding the optimal exercise for a given user and thus drives the system within the exercise level introduced in section 2. For that, denote the random event that a learner is part of a specific audience by U, and the event that a learner successfully managed the evaluation resp. the exam is named S.
Exercises are denoted by $e \in E$ in the exercise graph $G = (E, F)$ where F is the edge set of the exercise graph. Furthermore, denote the random event that the learner has visited nodes $e_1, \ldots, e_k$ in this order by $\Phi(1, \ldots, k)$. Assume now that at this stage the learner reached a decision point: amongst all suitable outgoing nodes of the node e_k, namely the set $F(e_k) = \{e_l \in E | (e_k, e_l) \in F\}$ the learner resp. the learning system has to pick one, and by that extend the learning path by one step. For brevity, we write $\Phi := \Phi(1, \ldots, k)$ for the unextended and $\Phi_l := \Phi(1, \ldots, k, l)$ for the extended path in the following.
The optimization problem is now finding a node $e_l \in F(e_k)$ such that the probability of passing the exam successfully is maximal for the given audience U, namely to maximize $P(S|U \cap \Phi_l)$. Using Bayes' formula [10], one finds

$$\text{maxarg}_l P(S|\Phi_l \cap U) = \text{maxarg}_l \frac{P(\Phi_l|U \cap S)P(U \cap S)}{P(\Phi_l|U)P(U)} . \quad (1)$$

The numerator contains now the probability of finding the extension Φ_l of the path in the subgroup of successful learners of the audience U times the probability of being successful in U, the denominator the similar probabilities for the full audience. All probabilities can be estimated by first running the system through an initial training phase where relative frequencies of all events are measured and the probabilities are estimated, and the system can keep updating the probabilities as students keep using it. By Laplace's rule [10], we can estimate all terms by relative frequencies and by that find:

$$\text{maxarg}_l P(S|\Phi_l \cap U) = \text{maxarg}_l \frac{|\Phi_l \cap U \cap S| + 1}{|\Phi_l \cap U| + 1} \quad (2)$$

i.e. the best extension e_l of the path is that which provided the best ratio of students of the audience U

passing the test so far, quite what one would have expected naively in first place. It thus remains an easy task for the learning system to identify the exercise paths picked by the user by querying a database, and update the counts appropriately as soon as a student fails or passes the final test for an exercise group.

We conclude this section with several remarks: First, to make the estimation eqn. (2) useful, numerator and denominator should be large and thus the sample size must be large. This imposes a restriction on the granularity of the audience since a finer granularity results in smaller members for each audience, reducing the sample size. Similarly, the number of paths to consider should be small enough to have useful samplesizes. This can be realized by two mechanisms: First, one could define the learning goals small enough and by that limit the number of valid paths, i.e. provide a lot of small evaluations within one course. Second, note that the number of possible paths grows very fast with the path length: by conditioning the expressions above only by the last N steps taken by a student, the number of possible paths to take into account is also greatly reduced. This restriction of the path length has also the nice interpretation as modelling a system of finite memory, where "memory" quite nicely coincides with the memory of the average student.

5. The concept of Virtual Labs

Several options are available to implement the asset and training level of the eLearning framework introduced in section 2: Java applets may, for example, provide the necessary interactivity and flexibility to run eLearning courses. However, the most attractive choice for experimental sciences, e.g. physics, is a virtual laboratory tailored to the interest of the course. They here completent the traditional hands-on courses performed in a "real" laboratory by not only providing experiments that would be hard to realize for resource constraints, but can also allow experimental access on abstract concepts discussed in theoretical courses. Thus, virtual laboratories can build bridges between theory and application. We will describe such an application in the field of statistical mechanics in section 7 below. Last but not least, virtual laboratories can be intergrated smoothly into an eLearning system as the one described above and thus are also for that reason an interesting research subject for us.

As an attempt to define this term, virtual labs use the metaphor of a scientific lab as guiding line for the design of the software. That is, similar to real labs, they provide the framework and equipment to setup and run experiments, i.e. qualitative and quantitative explorations of physical or mathematical phenomena of the model under examination. A virtual lab either simulates the dynamics of a system, or represents a mathematical algorithm to be studied. Furthermore, virtual labs provide tools to measure on these experiments and to explore their behavior. By that, a virtual laboratory is more than just a simulation: It is a flexible *framework* to study several systems of an entire domain, possibly including simulations.

Specifically for research purposes, flexibility and, more importantly, the integration and interconnection of the labs with other software elements and existing infra-structures are desirable. From our viewpoint, Maple is one of these components, an eLearning system running an interactive course using laboratories another. It is also attractive to think about the possibility to combine labs from different fields for complex experiments beyond the limit of one specific software program. That is, modular software design is imperative, and virtual labs cannot be monolithic applications one finds just too often.

6. The Virtual Lab VideoEasel

A virtual lab prototype in the above sense demonstrating the impacts of the demands on pedagogics and software design is the virtual lab VideoEasel [13], developed at the DFG research center Matheon of the Berlin universities. VideoEasel focuses on statistical mechanics, lectured in the "Mathematical Physics" course. Our audience is a mixture of mathematics and physics students in their 6^{th} semester.

Within VideoEasel, probability, analysis, dynamical systems and cellular automata are the mathematical disciplines that are brought to action in an environment of applications like image compression and denoising, phase transitions and irreversibility of large systems [8]. VideoEasel implements the microscopic rules of interest, e.g. the classical Ising model [4] or lattice gas models, by using so-called *Cellular Automata* [15]. We use here a modular software design separated into a computation kernel implementing the microscopic dynamics of a physical system, lacking any kind of graphical user interface, and several graphical front-ends allowing users to observe and manipulate the experiment.

These interfaces are, even more important, flexible enough to be used for completely different purposes, namely to integrate the laboratory into an eLearning system as described in section 2, or to extend the possiblities of the laboratory by the powers of a computer algebra system like Maple, see section 7.

The rules defining the microscopic physical laws are written in a C-like programming language and are

compiled and linked to the kernel at runtime. A set of predefined programs implementing various experiments are available, though the user is always invited to modify and change these rules if desired. Similar to the simulation itself, measurement tools, also defined by algorithms, can be linked into the experiment.

It is now in the hands of the controlling front-end to define the measurements to be made, to modify the parameters of the model for the purpose of an experiment and to evaluate the results of these measurements.

7. The Maple Interface

Combining virtual labs with CAS is attractive for many reasons: first of all, labs provide the framework for practical exercises and thus train students for their future profession; the proper handling of numerical and mathematical toolkits is, however, an important skill in all applied sciences. It is therefore desirable to train the usage of these tools right away on realistic, real-world data.

Second, Maple is a highly sophisticated software package that allows us to extend virtual labs by providing an interface to a powerful programming language and to powerful mathematical tools not available within the lab itself. E.g, Maple programs can be used as the process control for experiments. Furthermore, Maple can also act as a software bridge — a connector — allowing the user to link several lab toolkits together to simulate even more complex setups beyond the limit of each of the components.

Let us we demonstrate an experiment on the Ising model [4] for ferromagnetism. The Ising model is one of the most prominent systems considered in statistical mechanics as it demonstrates the effect of phase-transitions in many-body systems, here between a magnetic and non-magnetic domain: similar to real ferromagnets, the Ising model shows ferromagnetic behavior below, and non-ferromagnetic behavior above a critical temperature T_c, also referred to as the "Curie Temperature". The existence of a phase transition can be rigorously proven [11], thus making the model interesting for mathematical treatment. In this setup, the simulation of the Ising model is performed by VideoEasel, implementing the microscopic dynamics of the spins from which the model is built.

The aim of a student experiment is to show the relation between the free energy and the magnetization of the model, below and above the critical temperature, and to analyze their dependence on an external field for both temperature domains. A Maple-script, to be written by the students, controls the temperature and external magnetic field and collects data from the measurements of the free energy and magnetization. The data is returned, ready for plotting and numerical evaluation within Maple.

We deployed this setup in the mathematical physics course at the TU Berlin, and students observed that the magnetization as a function of the external field becomes noncontinuous, and the free energy non-differentiable for temperatures below T_c. This is exactly the mathematical definition of a phase transition. Furthermore, from the form of these curves they conjectured correctly that the first is proportional to the derivative of the second. Most students then were able to prove this relation exactly in the thermodynamic limit from the theory taught in the lecture.

For the Maple side, the VideoEasel interface appears as an external package that forwards appropriate calls to the kernel. This package provides all the basic functions to control and handle the numerical core of the lab, including the attachment of measurement devices, the adjustment of parameters or the modification of the microscopic rules.

8. The Course Interface

As described above, the lab integrates into a prototypical implementation of an exercise and tutoring system following the ideas of section 2. The *Course Model* consists of a database keeping elementary asset nodes in a textual representation. The asset nodes formulate the assignment given to the user, the way how to evaluate the solution presented by the student and the reaction on the solution, i.e. they encode a storyboard. To classify the solution provided by the learner, an asset node specifies one or several external java classes to be linked into the system on the fly; it is the purpose of these classes to return textual evaluations for the learner's behaivour. Despite providing the assignment, the asset node also defines hints to be presented to the learner on request, a name and a set of audiences the node is suitable for, and all requirements and suggestions this asset depends on. By that it groups assets into exercise units, implementing the middle level of our course model, cf. section 2.

Since more than one node might be available to satisfy the preconditions of an exercise, the learning system has to decide which exercise to present. It is here where the Bayesian estimation described in section 4 jumps in. However, the learner has the last say here: The system offers several routes in the exercise graph, but makes only a suggestion for the node to follow.

The *learner model* of the tutoring system is implemented as a data base which, indexed by the user, keeps information on the *audience* (see section 3) of the user,

the credits obtained by the user in the asset nodes and the learning path taken so far; this information therefore encodes the learner's *User Profile*. Given this and using the information which asset nodes have already been successfully visited by the user, the software agent can decide which asset nodes should be visited in future assignments, can update the Bayesian estimator once the exam has been taken, or present suggestions which exercise to take next in the exercise network. As explained in section 3, the credit points are not used to drive the Bayesian estimator; they just provide a convenient feedback mechanism for the student to give a rough approximation on the learning success.

9. Conclusion and Outlook

The presented intelligent assistant system [7] follows a storyboard, to be prepared by the lecturer of an audience, thus following hard links between asset nodes given by the vertices in the asset graph. Even though exercises can be adapted by the system to a certain degree by evaluating user profiles and picking suitable assets from the dependency graph, the means to escape this network are limited.

To improve the personalization of the system, despite the "hard conditions" encoded into the if-clauses in an asset node, "soft conditions" are to be introduced: user behavior – individual and group behavior – could be observed, and could be used to build up a psychological profile of the user. Psychological variables one could exploit are for example whether the user prefers graphical or textual representation of material, or prefers a serialistic or holistic approach when learning new material.

An important aspect is to keep the system transparent. That is, the decisions made by the system, e.g. why which asset node has been chosen, have to be made apparent to the learner. Otherwise, it is likely that students will not accept the feedback provided by the tutoring system since they are unable to comprehend the reasons the decision was based upon. Also, the user has to be allowed to "escape from the storyboard" because a tightly and statically linked asset network might possibly kill the user's creativity.

Thus, we have demonstrated that the combination of virtual labs and Maple has attractive applications in educations. Future work will be directed towards two major goals: First, several components of individualized feedback will be extended by developing a model for user profiling. Second, VideoEasel will be integrated in cooperative knowledge spaces [3]:

References

[1] D. Albert and J. Lukas. *Knowledge Spaces - Theories, Empirical Research, and Applications*. Lawrence Erlbaum Associates, New Jersey, 1999.

[2] D. Albert and M. Schrepp. Structure and design of an intelligent tutorial system based on skill assignments. In D. Albert and J. Lukas, editors, *Knowledge Spaces-Theories, Empirical Research, and Applications*, pages 179–96. Lawrence Erlbaum Assoc., New Jersey, 1999.

[3] S. Cikic, S. Jeschke, and U. Sinha. ViCToR-Spaces: Cooperative Knowledge Spaces for Mathematics and Natural Sciences. Conference Proceedings E-Learn 2006, Honolulu, HI, accepted, to appear October 2006.

[4] E. Ising. Beitrag zur Theorie des Ferromagnetismus. *Zeitschrift für Physik*, 31:253–258, 1925.

[5] K. P. Jantke and R. Knauf. Didactic Design through Storyboarding: Standard Concepts for Standard Tools. First Intl. Workshop on Dissemination of E-Learning Systems and Applications (DELTA 2005). Proc. of ACM Press, 2005.

[6] S. Jeschke and R. Keil-Slawik. Next Generation in eLearning Technology – Die Elektrifizierung des Nürnberger Trichters und die Alternativen. Informationsgesellschaft. Alcatel SEL Stiftung, 2004.

[7] S. Jeschke and T. Richter. *Intelligent Assistant Systems/Concepts, Technologies and Applications: Mathematics in Virtual Knowledge Spaces – User Adaptation by Intelligent Assistants*. Idea, Hershey, PA 17033, USA, 2006.

[8] S. Jeschke, T. Richter, and R. Seiler. VideoEasel: Architecture of Virtual Laboratories on Mathematics and Natural Sciences. Proceedings of the 3rd International Conference on Multimedia and ICTs in Education, June 7-10, 2005, Bandjoz/Spain, 2005.

[9] R. Kraußе and H. Körndle. TEE: The Electronic Exercise. In Jantke, K. P. and Fähnrich, K-P., and Wittig, W. S., editor, *Marktplatz Internet: Von e-Learning bis e-Payment*, Lecture Notes in Informatics, pages 281–286. Gesellschaft für Informatik, 2005.

[10] D. J. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, Cambridge, 2003.

[11] L. Onsager. A two-dimensional model with an order-disorder transformation. *Phys. Rev.*, 65:117–149, 1944.

[12] P. Pangaro. Thoughtsticker1986: A Personal History of Conversation Theory in Software, and its Progenitor, Gordon Pask. *Kybernetes*, 30(5/6):790–806, 2001.

[13] T. Richter. VideoEasel. `http://www.math.tu-berlin.de/~thor/videoeasel`.

[14] B. Scott. Conversational Theory: A constructivist, Dialogical Approach to Educational Technology. *Cybernetics & Human Knowning*, 5(4), 2001.

[15] T. Toffoli and N. Margolus. *Cellular Automata Machines*. MIT Press Cambridge, 1987.