

# **2006-1325: CONCEPTUAL GRAPHS AND STORYBOARDING FOR PROBLEM SOLVING STRATEGIES IN MATHEMATICS**

**Michael Jeschke, Technische Universität Berlin**

**Sabina Jeschke, Technische Universität Berlin, Inst. f. Mathematik**

**Olivier Pfeiffer, Technische Universität Berlin**

**Rudolf Reinhard, Technische Universität Berlin**

**Thomas Richter, Technische Universität Berlin**

# Conceptual Graphs and Storyboarding for Problem Solving Strategies in Mathematics

## Abstract

The major challenge for eLearning courses on undergraduate mathematics is the broadness of the audience they are targeted at. Our proposal how to deal with this challenge is to deploy intelligent assistants using Bayesian learning which, given some initial knowledge on the audience, explore user behavior to build up a model of the learner within the system. This allows us to leave the choice of the most suitable learning material to the learner. Thus, it enables an adaption of the system to individual learning styles while avoiding the risk of overwhelming the user by the plethora of choices of available material.

Starting with models for learner and course, we present a prototypical implementation of such a system within the virtual laboratory VIDEOEASEL developed at the TU Berlin.

## 1. Introduction

Scientists' and engineers' workplaces are about to change: numerical software and computer algebra systems remove the burden of routine calculation, but demand the ability to familiarize yourself with new concepts and methods quickly. Traditional "learning on supply" might be able to provide some basic knowledge, but this learning model becomes more and more unable to deal with the rapid growth of knowledge in today's sciences. Instead, learning and teaching methods have to be established that drive learners towards efficient self-controlled learning. New Media and New Technologies present a turning point in the educational system since they provide the basis to support the necessary changes.<sup>1</sup>

In our understanding, mathematics is the most attractive field for developing and deploying this New Technology:<sup>2</sup> first of all, it is the key technology of the 21st century. Studies in engineering sciences, physics, computer science and many other fields depend on a well-funded mathematical education. Teaching mathematics then, however, means that diverse backgrounds and varying interests of the audience have to be taken into account. Traditionally, the choice of the proper learning material for a course is to the lecturer and the teaching staff, but the broader the audience gets the harder it becomes to select exercises that are not only suitable for all students but also still within their field of interest. Our proposed solution for this apparent conflict is to leave the choice of the learning material to the student: an intelligent agent system aids her/him finding the right decision by providing exercises that proved best for previous generations of students.

To make this technology applicable, the learning material must be well-structured — as for example to ensure that all prerequisites to master an exercise are given. Luckily, mathematics as a scientific field already provides a perfectly worked-out internal structure we can exploit here. It is

in our understanding therefore well-suited for applying methods of computer science. We are confident that other fields might as well benefit from our research, though.

## 2. Virtual Laboratories

Virtual laboratories<sup>3</sup> use the metaphor of a “real”, scientific laboratory, thus provide a framework that emulates a scientific workplace for hands-on training, just in virtual spaces. Virtual laboratories enrich traditional mathematics, typically taught along “proofs”, by providing means to access abstract objects and concepts in interactive experiments; they thus build bridges between the theoretical fields and practical sciences by supplying experiments that run on computer-implemented algorithms that either emulate real devices in idealized situations or represent theoretical concepts. Applications of virtual laboratories range from practical support for traditional lectures (e.g. for demonstration purposes), over homework assignments and practical training for students up to aiding researchers in experimentation and visualization.

The virtual laboratory VIDEOEASEL<sup>4</sup>, developed at the TU Berlin, focuses on the field of statistical mechanics and statistical physics; it implements microscopic dynamics by a cellular automaton<sup>5</sup> that can be dynamically programmed at run-time, and makes its computational results available at run time over a network interface implemented in CORBA. This laboratory acts as the scientific back-end on top of which we develop interactive courses and study intelligent assistants: Software modules that provide learner-adaptive course material, following some earlier ideas.<sup>5,6,7,8,9,10</sup> Even though currently deployed in this framework only, the assistant technology we are developing for this virtual laboratory could be exploited by other systems as well.

## 3. Course Model

To build an effective and realistic model of an eLearning course, we propose the following three-level hierarchy of learning units (cf. Fig. 1):

### Level of content elements:

A *course* is the largest/coarsest unit we currently consider. It is the abstraction of a series of lectures held on a topic in a university. For example, a course could have the topic of “Linear Algebra”. 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. Typical examples for learning units of “Linear Algebra” are: the definition of the determinant, the invertible matrix theorem, stating that a matrix is invertible if and only if its determinant is nonzero, and so on. 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 has to be taught

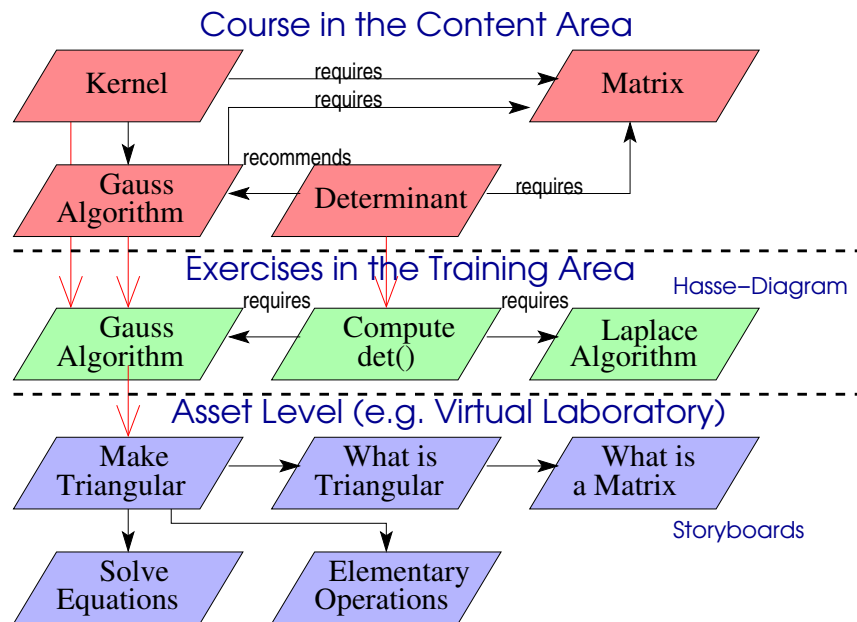


Figure 1: The proposed Course Model as a three-layer design of content, exercises and assets, here for an eLearning course on Linear Algebra

foremost in order to make A understandable. In the above example, the definition of the determinant is the precondition for the invertible matrix theorem. Dependencies themselves are classified into three groups:

1. *Hard Requirements*, that follow from the ontology of mathematics, as in the example presented above. That is, one cannot teach the invertible matrix theorem without having introduced the concept of the determinant.
2. *Recommendations* that are necessary for didactic purposes, though not imposed by the mathematical structure: for example, the node describing the determinant could recommend the Gauss algorithm as one suitable algorithm how to calculate it. And, finally,
3. *Suggestions* an interested student might want to follow, but which are neither required for didactical nor for inner-mathematical reasons. Historic remarks how the definition of the determinant evolved over time may aid as an example here.

Clearly, the network created this way depends on the ontology chosen for the field. Specifically, the *requirement* subgraph encodes one possible ontology of linear algebra. A visualization of the combined *recommendation* and *requirement* subgraph is also called “HASSE Diagram” in educational sciences.<sup>6,8</sup>

**Level of exercise elements:**

Recommendations 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 several knowledge atoms. It is this, and the following level we are currently exploring (cf. Fig. 1). The exercise layer in the proposed hierarchy is again a directed graph; however, its edges are now representing *exercises*: one exercise defines learning material a student might consider to repeat and train contents of the lecture. For example, one exercise in the course “Linear Algebra” would be to compute the determinant of a  $4 \times 4$  matrix, or to solve a linear differential equation with constant coefficients. 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 where one out of several requirements is sufficient, cf. the middle layer in Fig. 1 for example: In order to compute the determinant, a student might choose the Gauss algorithm or the Laplace Algorithm.

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: 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; it is the equivalent of the “history” in a web browser: A learning path is the sequence of learning units visited over time.

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 4. 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 section 4 for a detailed definition, and section 5 for a description of the update process.

**Level of assets:**

One exercise consists of one or more *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, e.g. an elementary row operation in the Gauss algorithm. Vertices in the asset graph now define 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 could not work, and hints could be given by the system. At this level, the graph is a representation of a *Storyboard*.<sup>11</sup>

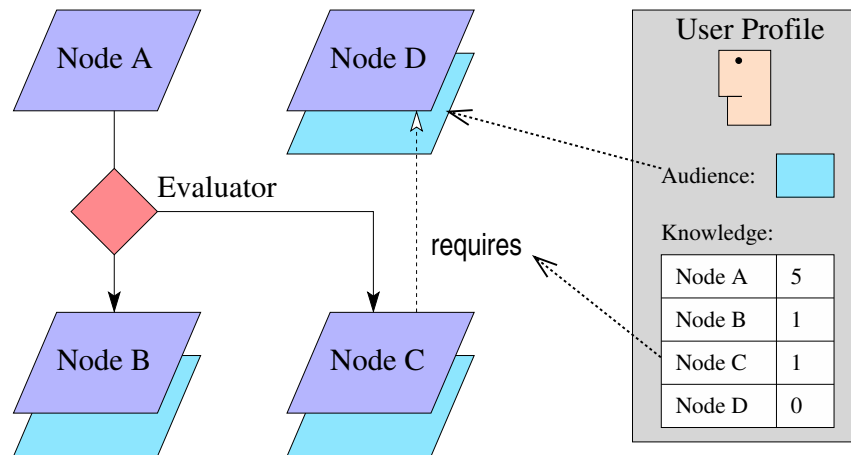


Figure 2: The learner model navigation in the exercise/asset network. The Evaluator (left) picks a suitable target node for the solution presented by the user. Dependencies between nodes and the users knowledge can redirect the user attention to a more basic asset. The audience selects the suitable implementation of the node from a set of nodes all providing the same content.

#### 4. 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. Degrees are the adequate characterization for the study goals, e.g. “Bachelor’s Degree”, “Master’s Degree” etc. Then “Electrical Engineers”, “Physicists”, “Mathematicians” are examples for audiences defining the notation, language and interests. Thus, a student has to classify himself in a two-dimensional matrix, defining his interests and community.

Note that even though the same course material – in our example “Linear Algebra” – has to be taught to all above audiences, the notation, formulation, and exercises to be given will differ significantly. 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 on 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 the learning goal to be achieved a posteriori, and thus has to be defined by the teaching university staff, i.e. the professor lecturing the course and her/his team.

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 (cf. Fig. 4) 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 the same 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 VIDEOEASEL, training nodes are therefore qualified by meta-data defining the audiences a node is suitable for.

## 5. Bayesian Learning

In this section, we present a Bayesian decision system that aims at finding the optimal exercise for a given user. 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. By that  $G$  is a directed graph where an edge runs from vertex  $e_2$  to  $e_1$  if and only if exercise  $e_1$  is a precondition for exercise  $e_2$ . For example, in the Linear Algebra exercise graph an edge would run from the determinant exercise to the Gauss Algorithm exercise, and another edge would run from the determinant exercise to the Laplace Algorithm exercise.

Furthermore, denote the random event that the learner has visited nodes  $e_1, \dots, e_k$  in this order by  $\Phi(1, \dots, k)$ . 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. Going back to the above example, the decision to make would be to choose either the Gauss or Laplace Algorithm exercise to compute the determinant. For brevity, we write  $\Phi := \Phi(1, \dots, k)$  for the unextended and  $\Phi_l := \Phi(1, \dots, k, l)$  for the extended path in the following. That is,  $\Phi$  is the “history” of the eLearning system for this specific student, and  $\Phi_1$  would denote the history plus Gauss,  $\Phi_2$  the history plus the Laplace Algorithm exercise.

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  $U$ . If we model the event of passing the exam as a random event  $S$ , we therefore need to maximize  $P(S|U \cap \Phi_l)$ : we need to maximize the probability for  $S$  under the condition that the user is part of audience  $U$  and has visited the course nodes in  $\Phi$  followed by node  $e_l$ . Using Bayes' formula<sup>12</sup>, one finds

$$\max \arg_l P(S|\Phi_l \cap U) = \max \arg_l \frac{P(\Phi_l|U \cap S)P(U \cap S)}{P(\Phi_l|U)P(U)}. \quad (1)$$

The numerator now contains the probability of finding the extension  $\Phi_l$  of the path in the subgroup of successful learners of the audience  $U$  times the probability of being successful in  $U$ . The denominator consists of the similar probabilities for the overall audience. In the example at hand, the numerator for  $l = 1$  is the probability of having picked the Gauss algorithm given that the learner in course  $U$  — say a master student of physics — passed the exam  $S$  times the probability of being a successful master student of physics. The denominator drops the condition of being successful.

Now all these 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<sup>12</sup>, the estimation for the first term of the denominator would be, for example:

$$P(\Phi_l|U \cap S) \approx \frac{|\Phi_l \cap U \cap S| + 1}{|U \cap S| + 1}$$

where  $|\Phi_l \cap U \cap S|$  is the number of events the sub-path  $\Phi_l$  has been found in the observation for successful  $U$  students, and  $|U \cap S|$  is the total number of successful students in audience  $U$ . It is easy to find similar expressions for all other probabilities in eqn. (1) to see that finally

$$\max \arg_l P(S|\Phi_l \cap U) = \max \arg_l \frac{|\Phi_l \cap U \cap S| + 1}{|\Phi_l \cap U| + 1} \quad (2)$$

i.e. the best extension  $e_l$  of the path is the one which provided the best ratio of students of the audience  $U$  passing the test so far – quite what one could have naively expected in first place. That is, the decision of whether the Gauss or Laplace exercise should be suggested as training exercise for the determinant computation depends on where most students following the same learning strategy so far have been most successful. It can be suggested that, following this decision rule, it might be more appropriate to pick the Gauss Algorithm if and only if the student has looked into this algorithm before on his or hers way through the learning material, for example.



It thus remains an easy task for the learning system to identify the learning 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 get sound results from estimation eqn. (2), 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 less members of each audience, reducing the sample size. Similarly, the number of paths to consider should be small enough to have useful sample-sizes. This could be realized by two mechanisms: first, one could define the learning goals small enough thus limiting the number of valid paths, i.e. providing 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. Restricting the path length has also yields a vivid interpretation as modelling a system of finite memory, where “memory” quite nicely coincides with the memory of the average student.

## 6. A Course System in VideoEasel

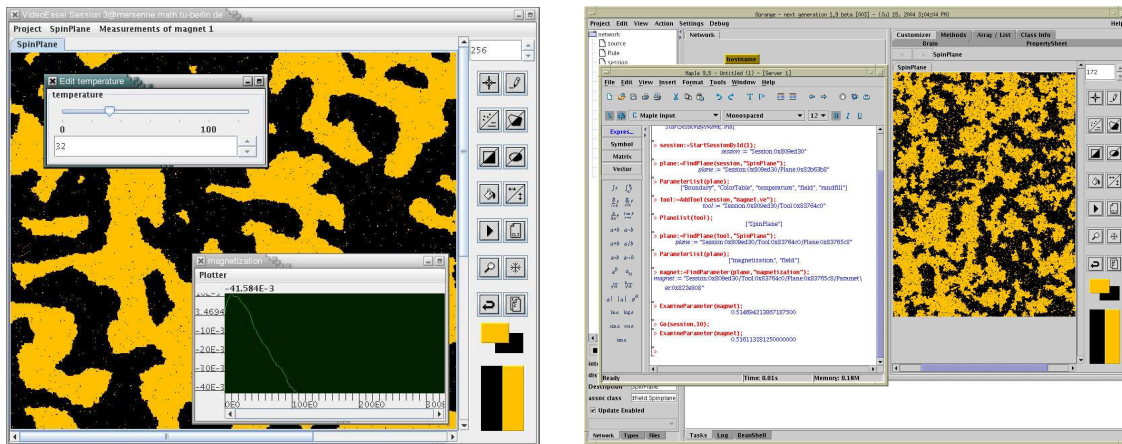


Figure 3: Exemplary Laboratory Front-Ends: Java Front-End on the left, Orange Interface communicating with Maple alongside.

The virtual laboratory VIDEOEASEL<sup>3,4</sup> introduced before is user adaptive in several ways: first of all, more than one user interface is available (cf. Fig. 3); depending on the desired deployment, Java applets, standalone Java front-ends, Orange<sup>13</sup> plug-ins or Maple-plugins are available. In addition to this passive adaptiveness the laboratory comes with a prototypical implementation of an exercise and tutoring system (cf. Fig. 4) following the course and learner models presented in the previous sections. The models are implemented as follows:

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 presented solution and the reaction on the solution, i.e. they encode a storyboard. To classify the solution provided by the learner, an asset node specifies the name of an external java class that is linked to the system on the fly. This class then returns a textual evaluation for the learner's solution, see Fig. 2. Beyond 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 credit points to be assigned to the user upon successful completion of an asset.

Leaving the "hard" links within the asset level storyboard aside, an asset node also formulates requirements and suggestions, and thus dependencies. By that it groups assets into exercise units, implementing the middle level of our course model, cf. section 3. 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. The system here makes a suggestion using Bayesian estimation as described in section 5; unlike hard branches, here the final say is to the learner: the system offers several routes in the exercise graph, and only makes a suggestion which node to follow.

The *learner model* of the tutoring system is implemented as a database which keeps information on the *audience* (see section 4) of the user, the credits obtained by the user in the asset nodes and the learning path taken so far. This information therefore encodes a *User Profile* of the learner (cf. Fig. 2). Given this and using the information which asset nodes have already been successfully visited by the user, the tutor program 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 noted in section 4, the credit points are not and should not be 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.

## 7. Outlook

The presented tutoring system follows a storyboard that has to be prepared by the lecturer for an audience, and thus follows hard links between asset nodes given by the vertices in the asset graph. Furthermore, the system reacts in an adaptive way to user behavior because, over time, it learns which exercises proved most successful for the learning goal. It does not, however, allow the user to "escape from the storyboard" completely: this might be a highly desirable feature because a tightly linked list of pre-defined exercises may kill the user's creativity. The only option a learner currently has is to abort the tutoring system and explore the laboratory on his own, though tutorial help is not available then.

Last but not least, it might help students a lot to interrogate the system, let it be by means of a graphical point-and-click front-end, or even by usage of a linguistic parser, allowing users to formulate queries in natural language. By that, it would be possible to deploy the system even in

problem-oriented, rather than exercise-oriented work, e.g. research and self-study scenarios.

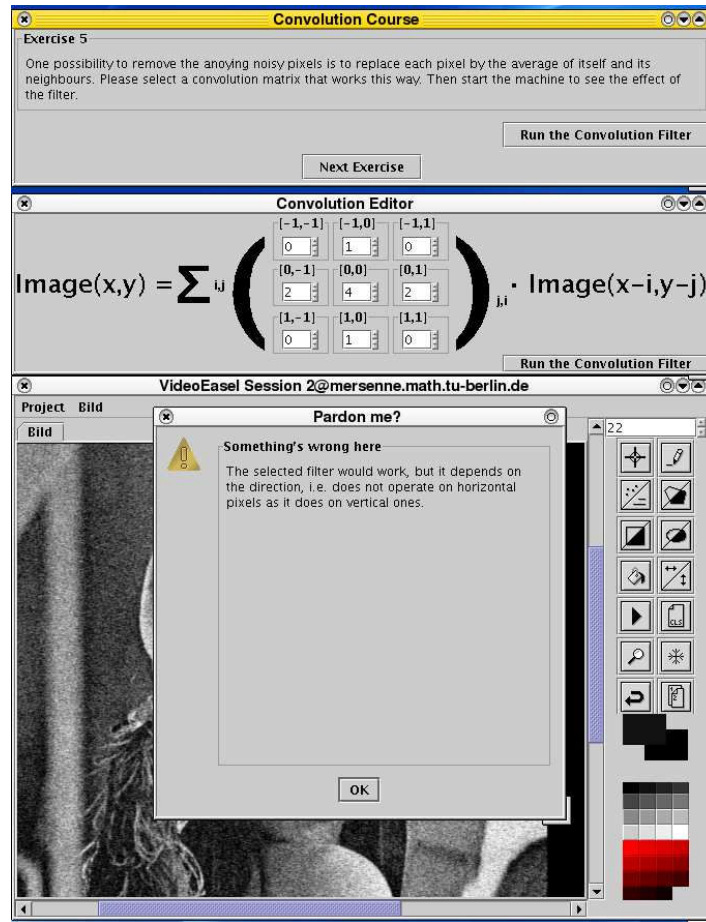


Figure 4: A tutoring assistant in a laboratory on matrix convolution. In the topmost window the current assignment to be performed is displayed; the middle window shows the configuration wizard for the convolution automaton, the target image is shown below (in the background). On top of the target image the assistant gives some feedback to the learner.

## Bibliography

1. S. Jeschke and R. Keil-Slawig. *Next Generation in eLearning Technology - Die Elektrifizierung des Nürnberger Trichters und die Alternativen*. Informationsgesellschaft. Alcatel SEL Stiftung, 2004.
2. S. Jeschke, M. Kohlhase, and R. Seiler. *eLearning-, eTeaching- & eResearch-Technologien – Chancen und Potentiale für die Mathematik*. DMV-Nachrichten, July 2004.
3. 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, Caceres/Spain, June 2005.

4. T. Richter. *VideoEasel*. <http://www.math.tu-berlin.de/~thor/videoeasel>.
5. T. Toffoli and N. Margolus. *Cellular Automata Machines*. MIT Press Cambridge, 1987.
6. D. Albert and J. Lukas. *Knowledge Spaces - Theories, Empirical Research, and Applications*. Lawrence Erlbaum Associates, New Jersey, 1999.
7. D. Albert and M. Schrepp. *Structure and design of an intelligent tutorial system based on skill assignments*. In D. Albert and J. Lukas (eds.) *Knowledge Spaces - Theories, Empirical Research, and Applications*, p. 179-196. Lawrence Erlbaum Associates, New Jersey, 1999.
8. R. Krauß and H. Körndle. *TEE: The Electronic Exercise*. In K.P. Jantke, K.-P. Fähnrich and W.S. Wittig (eds.) *Marktplatz Internet: Von e-Learning bis e-Payment*, Lecture Notes in Informatics, pages 281-286. Gesellschaft für Informatik, 2005.
9. P. Pangaro. *THOUGHTSTICKER 1986: A Personal History of Conversation Theory in Software, and its Progenitor, Gordon Pask*. *Kybernetes*, 30 (5/6): 790-806, 2001.
10. B. Scott. *Conversational Theory: A constructivist, Dialogical Approach to Educational Technology*. *Cybernetics & Human Knowing*, 5(4), 2001.
11. K.P. Jantke and R. Knauf. *Didactic Design through Storyboarding: Standard Concepts for Standard Tools*. 1st Intl. Workshop on Dissemination of E-Learning Systems and Applications (DELTA 2005). Proc. of ACM Press, 2005.
12. J.C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, Cambridge, 2003.