# A PRESENTATION AND TUTORING ENVIRONMENT FOR COURSES IN THEORETICAL COMPUTER SCIENCE

**Eva Giani, Philipp Rohde, Wolfgang Thomas**
Lehrstuhl für Informatik VII
RWTH Aachen
D-52056 Aachen
E-Mail: {giani, rohde, thomas}@informatik.rwth-aachen.de

## ABSTRACT

*We report on the development of e-lectures in theoretical computer science (more specifically, automata theory). The emphasis is on the development of a simple and flexible infrastructure, which reduces the effort in preparing e-lectures and tutorial units in this field of study. Two components are essential: an integrated and context-independent hardware and software system for the preparation and presentation of the course material, and a tutorial component with dialogue features, which allows to set up problems accompanying the lectures.*

**KEYWORDS: e-lecture, automata theory, online course, tutorial system**

## INTRODUCTION

A central problem in building up course material for e-learning is the lack of resources in a standard university environment. Under usual conditions, the web-based courses and tutorial components have to be developed under severe constraints, regarding both technical equipment and human resources. Only in exceptional cases, a docent will be granted a free semester for the conception and realisation of a course and will get the necessary assistance for the programming work which is involved. So it is a central question how to design an infrastructure with which a docent can prepare and present e-lectures with minimal possible effort. Ideally, he should be able to prepare recorded courses and tutorial units "online" during a running semester.

In this paper we report on a solution to this problem, which was developed within the BMBF project ULI ("Universitärer Lehrverbund Informatik"[1], see also Kandzia 2001; Lauer et al. 2002). Within ULI, the Aachen team has to develop course material in theoretical computer science. The course on which we report here was a basic course on automata theory in the second year of the computer science curriculum, given with three hours per week and one hour problems class. There were about 400 local participants and about a dozen external participants. There was no reason to run the problems classes electronically; intensive personal tutoring by student tutors was done in the weekly problems classes and the external participants were supported via e-mail. Additionally a chat address was offered (moderated by a student tutor). The lectures were recorded (audio track and annotated slides), and a tutorial system for non-compulsory self-tests was offered. This tutorial component was designed to reduce the burden of personal tutoring; with this infrastructure, many basic problems in mastering the material could be settled immediately, without contacting a tutor. Details on this tutorial system are presented in section 3.

[1] See http://www.uli-campus.de/english/project.html

The technical infrastructure should support the docent meeting the following constraints:

- The e-lectures have to cover the complete contents of the course. So in automata theory we could not restrict to special chapters as in the interesting work of Tscherter et al. (2002).
- The presentation of the course should be possible in any lecture hall.
- Given the infrastructure, the preparation and presentation of a course and the generation of tutorial units should be feasible by the docent with the support of a single person (say, a student assistant) for technical assistance during the semester.

Additional constraints are listed in the next section and a description of our solution is given.

## TECHNICAL INFRASTRUCTURE

In the lectures, electronic slides were presented by a DV projector and grabbed by a software recording tool. The voice of the docent, the slides, and additional annotations like drawings on the slides had to be recorded. On the average, about two thirds of the material were prepared before, while about one third (mainly figures) was added during the lecture. It did not seem to be necessary to have an additional video of the acting docent. The system had to produce a video file in a standard format, to be played with any common media player for the standard operating systems.

To realise the constraint of full mobility and to present and record the lectures in any lecture hall, we designed an integrated hardware solution. The essential idea was to use standard components – for the software as well as for the hardware – whenever available and to reduce the "in-house development". This is in contrast to other solutions, which were developed within the ULI project,[2] and was adopted in order to obtain professional support and updates of the supplier, as well as to concentrate the efforts on the course material itself.

### Hardware

For the hardware we combined standard components of personal computers, professional audio recording tools for broadcasting tasks, and a high-end input / output device which is normally used for professional graphic design / CAD. The graphic tablet *Wacom Cintiq 15X* consists of a TFT display, which is sensitive to inputs of a pen.[3] Therefore the docent was able to write electronically on the slides.

We installed these components in a portable case (built by a company specialised in making cases for audio devices). We obtained a mobile solution, which was ready to use in any lecture hall within a few minutes without any further set-up procedures. One had only to connect the device to a DV projector, plug in the microphone and switch on the machine.[4]

---

[2] For example *Authoring on the Fly* (AOF), see Pomm and Widmayer (2002); Müller and Ottmann (2000).
[3] See http://www.wacom.com/lcdtablets/index_15x.cfm
[4] For pictures of the portable case see
http://www-i7.informatik.rwth-aachen.de/d/projects/uli/photos/e-lecture.html

**Software**

Several demands on the software environment had to be met:

1. The environment should support extensive use of mathematical formulas.
2. The docent should be able to reuse material from his (electronic) lecture notes of previous courses.
3. It should be possible to write on the slides giving illustrations and annotations.
4. The presented slides can be recorded together with the annotations and the voice of the docent.
5. The post-processing of the recorded raw material should be simple. The purpose was to produce video files in a standard format, readable by the standard media players for *Windows*, *Mac OS* or *Linux / Unix* without additional set-up.

Furthermore the file size of the videos had to be kept in small bounds since they were offered – usually within one day – for download. We had to take into account that students might have private internet connections with low transmission rates.

To realise 1 and 2 we decided to use the *LaTeX* publishing environment. We combined common *LaTeX* packages for making electronic slides with some in-house developed macros and features to obtain the desired layout. We converted the slides to the *Adobe PDF* format and used the *Adobe Acrobat 5* to display them.[5] Although the latter product already has built-in features to realise 3, we extended its functionality by some extra plug-ins to simplify the use and speed up the access. For the fourth constraint we used the professional screen recording software *Camtasia* by *TechSmith*, which was originally designed to produce software tutorials.[6] To edit and cut the raw material and to create the video as described above, we used the video editing software *Adobe Premiere 6*.[7] We produced two different videos: one used the video codec *TSCC* (designed by *TechSmith*) and the other the common codec *DivX*.[8] The audio stream for both formats was encoded by the standard audio codec *MPEG Layer-3* which offers a sufficient compression rate.[9] The file size of the produced videos could be reduced to about 10 MB for a 45 minutes lecture. Therefore we were able to offer the video files of a complete semester course on a standard compact disc.

To give the opportunity to study the lecture material from the slides only, we additionally offered them for download without a video / audio stream – but together with the presented annotations and illustrations. The slides were available as *PostScript* files and as *PDF Documents*, to ensure platform independent access.[10]

## THE TUTORING ENVIRONMENT

The purpose of the tutorial system is to enable the student to check his / her understanding of the concepts by solving simple problems about them. The degree of difficulty in such exercises is clearly above the level of multiple-choice questions, but below the level of exercises

[5] See http://www.adobe.com/products/acrobat/main.html and http://www.adobe.com/products/acrobat/adobepdf.html
[6] See http://www.techsmith.com/products/studio/default.asp
[7] See http://www.adobe.com/products/premiere/main.html
[8] See http://www.techsmith.com/products/studio/codec.asp and http://www.divx.com/about/
[9] See http://www.iis.fraunhofer.de/amm/techinf/layer3/
[10] For examples of the material of the currently running lecture see http://www-i7.informatik.rwth-aachen.de/d/teaching/ws0203/ars/index.html

where some kind of invention is needed for the solution. The system was only offered as a self-test, and after download a student could see in a synopsis the scores he / she reached so far. The system does not support a central storage of scores so far, as would be needed to use it for examination purposes.

A main requirement in designing the system was that the solutions should be developed interactively, with hints about errors and the possibility to correct them. Since the contents of the whole course had to be covered, a large number of data types had to be taken into account for input and for processing (like words, regular expressions, automata, grammars, derivations, etc.).

In the system *GUSTAF*,[11] developed in diploma thesis work of the first author (Giani 2002), these functions were realized, and an interface for the docent was included allowing him to create new exercises. This development of an exercise includes the feedback, which the system provides in the process of solution. In order not to overload the designer of an exercise, a simple and natural dialogue structure was adopted, which includes two levels of feedback. The first feedback informs the user about wrong format of input or of wrong items in the first answer. This is a built-in function about which the docent does not have to care. In case of wrong format a repetition of the first round is allowed. If an item is syntactically correct but constitutes a wrong answer, the user is told about the mistake (commonly with more information than just "wrong"), and the user has a second try. If the answer is wrong again, the dialogue is ended and the correct answer and some extra hints are supplied.

The system offers two interfaces: a student interface, which belongs to, a module to be downloaded and executed locally, and a docent interface for the preparation of the exercises. The tutorial system allows the students to invoke support by appropriate links and to call the corresponding material (definitions, statements etc.) from the lecture.

In the present version, more than 40 prototype-exercises have been developed, covering the whole spectrum of topics in basic automata theory.[12] Some of the exercises involve somewhat nontrivial aspects; for example, in one exercise the appropriate format of an induction proof has to be developed.

## EXPERIENCES AND OUTLOOK

There was a detailed evaluation of the course and of the tutorial system based on questionnaires handed out to the students. In the responses, there was unanimous agreement that the recording and the tutorial system were a highly useful supplementary component of the course. The vast majority of students voted for a combination of real and e-lectures.[13]

Two specific effects are worth mentioning. First, there were students who used e-lecture meetings as a social event (in small groups of three or four), listening to the lecture offline together instead of visiting the real lectures. Secondly, when at the end of the semester the pressure by several exams increased, many students postponed their participation in the automata theory course in order to concentrate on other courses with exams ahead. Nevertheless, the success rate in the exam (six weeks after the end of the course) was higher than in previ-

[11] "*G*rundstudium-*U*nterstützendes *S*ystem zur *T*heorie der *A*utomaten und *F*ormalen Sprachen"
[12] See http://www.rwth-aachen.de/i7/atfs/practice/selbsttestaufgaben.html
[13] For the evaluation and its results see
http://www-i7.informatik.rwth-aachen.de/d/teaching/ss02/atfs/auswertung.html

ous years. So the recordings and tutorial system were essential in enabling the students to pursue individual schedules for the exams and thus achieve a higher personal success rate.

**Further developments**

The technical infrastructure described in section 2 was developed prior to the course. It turned out to be completely stable; there was no failure of any component. But our portable case is still a prototype: the dimensions and the weight should be decreased. An improved solution could be the recently developed Tablet PCs, which are a compact combination of a laptop and a pen-sensitive display. But then one has to put up with a smaller screen, and it is not clear whether the currently offered models are suitable for the necessary functions.

A useful extension of the current infrastructure would be to use a tutorial system also for examination purposes. This would require integrating central server-based functionalities (registration and storage of results considering the corresponding security aspects).

At the end of the course we plan to offer an electronic script (as a PDF document) consisting of the slides with annotations and the usual elements of an e-book: links, a table of contents, an index, references etc. Furthermore the reader will be able to jump directly to the scenes within the corresponding videos of the lecture where the current contents are treated as well as to the appropriate exercises within the tutoring environment. Further improvements towards a "meta document" (e.g. a XML document) with more structure are conceivable. For example there are some nice tools for the scope of automata theory, which are worth to be integrated (e.g. *AMoRE* developed by our research group, or *Exorciser*, see Tscherter et. al. 2002).[14]

**REFERENCES**

GIANI, E. (2002). Konzeption und Implementierung eines interaktiven Lernsystems für die Grundvorlesung über Automatentheorie. Diploma thesis, Lehrstuhl für Informatik VII, RWTH Aachen, Germany.

KANDZIA, P.-T.; MAASS, G. (2001). Course Production – Quick and Effective. In: Proceedings of the 3rd International Conference on New Learning Technologies (NLT / NETTIES), Fribourg 2001.

LAUER, T.; TRAHASCH, S.; ZUPANCIC, B. (2002). Virtualizing University Courses: From Registration till Examination. In: Proceedings of the 4th ICNEE, Lugano 2002.

MÜLLER, R.; OTTMANN, T. (2000): The "Authoring on the Fly"-System for Automated Recording and Replay of (Tele)presentations. ACM / Springer Multimedia Systems, Vol. 8, No. 3, 2000.

POMM, C.; WIDMAYER, P. (2002). Developing Course Material with the "Authoring on the Fly" Concept – An Evaluation. In: Proceedings of the 4th ICNEE, Lugano 2002.

TSCHERTER, V.; LAMPRECHT, R.; NIEVERGELT, J. (2002). Exorciser: Automatic Generation and Interactive Grading of Exercises in the Theory of Computation. In: Proceedings of the 4th ICNEE, Lugano 2002.

[14] For AMoRE see http://www-i7.informatik.rwth-aachen.de/d/research/amore.html