WebALT 2006

First WebALT Conference and Exhibition

January 5-6, 2006
Technical University of Eindhoven, Netherlands

WebALT 2006 Proceedings

Mika Seppälä, Sebastian Xambo, Olga Caprotti
(editors)

ISBN 952-99666-0-1

Foreword

The Web Advanced Learning Technologies (WebALT) Project develops tools and content to support multilingual on-line instruction in mathematics. The project started in January 2005 and will end in the end of 2006. This volume contains the papers presented at WebALT2006: The First WebALT Conference and Exhibition held in Eindhoven on January 5-6, 2006. The final Exhibit will take place in Berlin in the end of November 2006.

The project is supported by the eContent Programme of the European Commission. Partners of the WebALT Project are Technical University of Catalonia, University of Cologne, Technical University of Eindhoven, Maths for More, and the University of Helsinki, the coordinating partner.

By limiting the linguistic domain to textual constructs typical to mathematical exercises, one can encode the content so that it is possible to generate versions of the content in many languages. This method and its implementations form the main deliverable of the WebALT Project. Besides this unique multilingual solution, the project advances best practices in on-line education in general.

Thanks to the recent introduction of various affordable and high quality on-line conferencing systems, the last barriers hindering the advance of on-line instruction are disappearing. It is now possible to offer on-line the same kind of interactivity as what one has in a regular class-room instruction. The WebALT Project will facilitate this development by offering tools and high quality content.

To sustain this activity beyond the end of the WebALT Project, the partners of the project have started the Company, Oy WebALT Inc. This company will be the owner of all the intellectual property developed by the project. Thanks to the significant initial investment by the University of Helsinki, Oy WebALT Inc. is now operational and will start commercial activities later in 2006. Electronic publication series and electronic books form a part of these commercial activities.

Mika Seppälä
Sebastian Xambó
Olga Caprotti

Program Committee

Mika Seppälä,	Conference Chair and Coordinator of WebALT
Sebastian Xambó,	Program Chair
Hans Sterk,	Exhibition Chair
Olga Caprotti,	Publicity Chair and Project Manager of WebALT
Lauri Carlson,	University of Helsinki
Hans Cuypers,	Technical University of Eindhoven
Sabina Jeschke,	Technical University of Berlin
Erica Melis,	German Research Institute for Artificial Intelligence
Robert Miner,	Design Science Inc.
Jordi Saludes,	Polytechnical University of Catalonia

Local Organization

Arjeh Cohen
Karin Poels

External Reviewers

Matti Pauna
Olivier Pfeiffer
Jordi Saludes
Chris Sangwin

Sponsors

European eContent EDC-22253-WEBALT

Table of Contents

Contributed Papers

Designing Math on-line Courses for Computer Science Students: experiences at the Open University of Catalonia

M. Antonia Huertas, Angel A. Juan, Cristina Steegmann

Open University of Catalonia (UOC), Department of Computing and Multimedia
Av. Tibidabo 39-43, 08035 Barcelona, Spain
{mhuertass, ajuanp, csteegmann}@uoc.edu
http://www.uoc.edu

Abstract. Information Technologies have produced important changes in modern societies. These changes, in turn, have a strong influence on the university environment since they imply the appearance of both new formation necessities and new methodological possibilities. As a result of this influence, new learning models arise in which the role that professors and students develop differs partly from the one established during the last century. This way, in the e-learning context, a re-definition of the teaching-learning process is taking place. This re-definition affects both the universities (as institutions) and the knowledge areas they impart. We think, however, that there is yet a lot of work to be done in the design of mathematical courses, especially when they are addressed to science and engineering students. These courses should make an effective and efficient use of the real possibilities that technology offers so that the new formative necessities of these students could be covered. In this article, we present a pedagogical model, developed by a group of maths professors at the Open University of Catalonia (UOC), which is being applied in the mathematical formation of computer engineers.

1 Introduction

In the information society, our individual knowledge must be revised and updated throughout our entire life. An university degree is not enough to provide us with all those concepts that we will need in our extensive and changing professional career. On the contrary, the university degree constitutes the starting point of a continuous formation which will last all our life. Consequently, the traditional paradigm associated to the university education is changing due, on the one hand, to the new formative necessities and, on the other hand , to the new possibilities that information technologies offer to us [6].

By formative technologies we refer to the resources derived from the application of information technologies to the educational environment, i.e.: on-line platforms for collaborative working, digital libraries, materials in electronic format (audio-visual, interactive,...), wireless LANs, Internet communities, etc... All these technological re-

2

sources are being used more and more in the on-line education (or formation via Internet), a concept which is usually called e-learning.

E-learning, which represents a new paradigm in the context of university formation, impregnates all environments of an entirely on-line university as the UOC. It is necessary to clarify, however, that the formation technologies are used also in those traditional universities based on the existence of a physical campus where students and professors meet. At present, the traditional paradigm (formation methodology centered around the figure of a masterful professor and in some perennial knowledge) continues to coexist with the new formation paradigm (centered around students and their learning process).

In this new paradigm students learn, with the help of technology, what they will potentially need in order to develop their future academic or professional activity. The professor role is more and more moving from the function of knowledge transmitter to the function of a specialist who designs the course and supervises the process of the student's formation. This paradigm is commonly known as " teaching to learn" or, as is mentioned in some references, the post-modern university paradigm [7].

Frontiers between both paradigms are not very clear since both coexist and they are interrelated. On the one hand, it happens that formation technologies (on-line courses, virtual classrooms and student support through e-mail) are used in combination with traditional methodologies and curricula. On the other hand, it is frequent to find out on-line courses that combine a student-centered methodology with materials that come from the traditional model (lecture notes, for example).

In this article, we will present a model for doing e-learning which is based on the experience of our university. The model includes the initial design (methodology and contents) of the mathematical subjects that integrate the curricula of computer science students at UOC.

The first step we took in the process of designing new math on-line courses consisted in carrying out a preliminary analysis about the current situation in our university. From this analysis, the following points should be highlighted:

(1) The number of students who passed the course was found to be significantly inferior in the math subjects than in other subjects of similar difficulty belonging to non-mathematical areas. The natural question that was derived from this observation was: why are the results in these math courses significantly lower that in the rest of subjects? [1, 4]

(2) Our computer science curricula were being completely redrawn. Taking advantage of that fact, the following question also arose: what abilities and mathematical knowledge are necessary for students of computer science at UOC? [2, 5]

3

Trying to provide answers (and solutions) to the former questions, the university staff redrew the math subjects, both in contents and methodology, and proceeded to the development of new courses. The idea was to give a clear answer to the math formative necessities of a computer science engineer.

2 Designing a new model

2.1 A model for math e-learning

The low number of students having passed the basic math courses (Linear Algebra and Calculus) was analyzed. From this analysis, we could isolate seven key points that helped to explain the obtained results:

1. Generally speaking, the students' math background was deficient. Furthermore, the students presented very different math backgrounds.

2. An important discrepancy existed between the real time necessary to obtain the academic objectives and the number of credits assigned to the subject.

3. The following facts were detected: an excess of algorithmic (repetitive) calculation methods, an excess of formalism in the material, and lack of students' motivation for the subject.

4. Several problems were also detected in the methodology as well as in the evaluation system used.

5. Many students presented serious difficulties in their conceptual modeling and abstraction capacities.

6. Scarce use of information technologies in the math curriculum.

7. Objectives for each course were not clearly detailed.

To find out solutions that we could offer to the previous problems, we decided to expose the two described teaching-learning paradigms. Table 1 compares both approaches:

4

Table 1: Comparison of traditional paradigm vs. e-learning paradigm approaches

Traditional paradigm	E-learning paradigm
Compensatory math courses which are offered to students with background problems are neither specialized nor personalized.	Detect what is the necessary math knowledge to start degree subjects. Consequently, offer the student specialized and customized initial courses. The use of mathematical software is introduced as a helpful tool.
In the assignment of credits, professors do not usually take into account the time needed to acquire the abilities and to practice them. Also, it is considered that students have full-time dedication.	Courses are redrawn considering real dedication (time that an average student employs in completing the academic objectives). The use of time is optimized employing mathematical software that takes care of repetitive calculations. The use of self-evaluation tests allows students to know their current state in the learning process without unnecessarily increasing faculty dedication.
There is an excess of formalism and algorithmic. This could be mainly due to the use of a traditional methodology which, in turn, could be explained by the fact that most of the faculty is not used to adapting the courses to the students and their context (university degree in which the math contents are integrated). Furthermore, formation of most math professors is restricted, almost exclusively, to mathematical concepts (they do not usually have formation on some important areas of computer science).	It is necessary to redraw both the contents and the methodology, re-thinking the objectives and their focus. Contents have to be close to the students' interests. Start reducing unnecessary use of repetitive algorithms and formalism. Concept understanding is reinforced by means of using mathematical software.
Traditional methodology and evaluation are used. Both are based on memorization and repetition processes. Students' creativity is not reinforced.	Use of the evaluation process as a learning methodology (not only as a way of obtaining some credentials). .
Memorization of procedures and algorithms. Conceptual modeling is not a central part of math subjects.	One basic objective is that the student will be able to choose the necessary math tools to solve real problems that can arise in their professional activity. This objective should be contemplated in every possible way (contents, methodology and evaluation).

A mathematical e-learning model for computer science students at UOC was derived from the previous analysis using the approach provided by the new paradigm.

5

First of all, the general properties of the model were established:

- Instrumental: mathematics should be considered as a tool that can help computer science engineers to solve real problems.

- Interdisciplinary: mathematics courses should be contextualized in the context of the studies to which they are associated (computer science studies in this case).

- Motivation: the main approach to use is a top-down one, that is: going from the discussion and resolution of practical cases to the theoretical concepts that constitute the fundamentals of applied methods. Also, whenever possible, case studies related to computer science problems where mathematical concepts can be helpful should be used.

- Conceptual modelling: concepts and procedures are more important than solving algorithms. The use of mathematical software can help to avoid unnecessary efforts related to solving problems manually.

- Mathematical rigor without too much formalism: an excessive use of mathematical formalism can significantly increase the difficulty of the subject as well as the time involved in achieving the academic goals.

- Personalization of the learning process: each student begins with a different mathematical background. It is highly recommended to adapt the learning process to the current status of the student.

- Adaptation of the student's dedication time: an important effort should be made in the planning phase so that the course credits correspond to the real time employed by the student.

Next, the methodological approach and the specific criteria of the model are described:

- Giving preference to learning concepts over learning calculation algorithms. The student should use mathematical software to accelerate calculations. They also should dedicate the time they save using software to deal with more cases and potential examples of applications to real-life problems.

- Formalism and algorithmic has to be softened without losing mathematical rigor, which adds value to the student's formation.

- Mathematical software should be intuitive, multilingual (Spanish, Catalan and English) and ready to be used both on-line and off-line.

6

- It is fundamental for the body of professors to work on this new paradigm. Particularly, It is fundamental to have a more instrumental conception of mathematical courses, where emphasis is put on the applications rather than on theoretical concepts.

- A continuous evaluation process should be defined. This process should integrate software use and be able to evaluate the students' acquisition of course objectives.

- Special attention should be paid to the determination of each student background.

- Finally, but not less important, faculty staff at UOC must have to take into consideration the UOC learning model which uses a detailed syllabus as the key element of the course planning. We will call this the institutional e-learning model.

2.2 Re-designing the math curriculum

At this point, it is convenient to remember that we were involved in a complete process of redrawing the computer science curricula at the UOC. Without giving too many details, we will describe the main ideas of this process directed to determine the concrete contents that every area of knowledge should cover (obviously, one of those areas is mathematics). It was a top-down process in the sense that it started with the description of the general capabilities and knowledge that a computer science professional will need to develop their activity and then it followed by determining the areas where those capabilities and knowledge could be acquired. The whole process was developed as team work where stuff from different disciplines worked together exchanging points of view. Finally the results or conclusions of this analysis were compared with results obtained by similar universities (universities with students with a profile similar to ours). After this final comparison, we obtained a new program defined by the whole computer science faculty staff.

At the following stage, math professors –with the help of professors from other disciplines like networking, data base theory, cryptography, etc., followed the top-down approach adding more details to the proposed math curricula. Math contents were concretized, giving priority to those contents which served as necessary tools for developing other areas of knowledge. A big effort was made to place specific uses and applications of math concepts and techniques among other disciplines of the degree.

This way, a map of necessary skills and contents was built. The final contents could be grouped together in the following themes :

- Logic
- Set theory

7

- Natural Numbers
- Linear Algebra and geometry
- Geometrical Transformations
- Real Functions of one variable. Derivation. Integration
- Complex Functions of one variable. Integration
- Some numerical methods
- Statistic methods
- Discrete and combinatorial Mathematics and Graphs Theory

2.3 The process of re-designing math courses

Once contents, competences and skills -as well as approach and methodology- were determined and/or selected, the following step was to completely re-design the different math courses (Fig. 1)

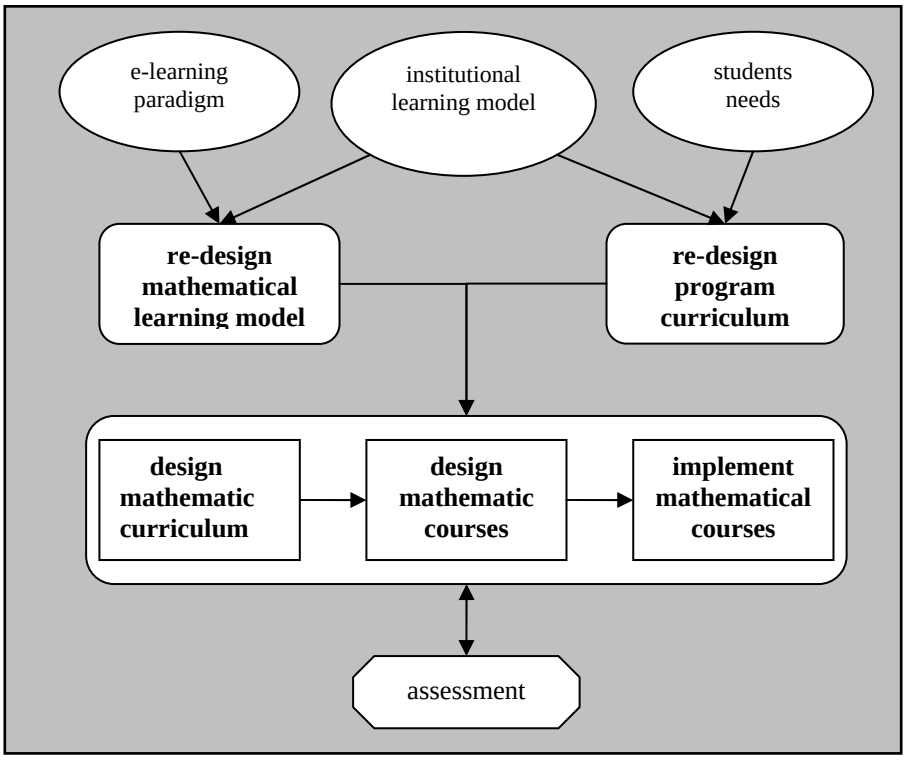


Fig. 1. Design process

In our case, as a direct consequence of the official computer science curricula, we had to design five math courses, Logic, Discrete Mathematics, Statistics, Algebra and Calculus. In the first step only the last two would be redesigned, Algebra and Calculus. At the beginning we decided to separate the contents according to their corre-

8

spondent difficulty and regardless of the subject. Nevertheless, after analyzing computer science curricula at other Spanish universities, and in order to allow the students to move to other universities, we decided to separate contents belonging to Algebra (set theory, natural numbers, linear algebra and geometry) from those belonging to Calculus.

Algebra and Calculus courses were designed simultaneously. In fact several modules of material were designed, each of them containing the following sections: goals, necessary background, an introductory practical example, body of the module with examples and activities (many of them designed to be carried out using several alternative math software such as Mathematica, Maple, Wiris or Mathcad), self-evaluation exercises, summary and glossary of terms. Contents should be put in context, algorithmic calculations would be mostly done with the assistance of math software and formalism should be softened.

The faculty staff also developed an interactive web material to introduce the use of the math software Wiris. This material was also conceived as a way to revise the necessary background during the first week of the course and to act as a training laboratory during the rest of the semester. A professor specialised in Wiris helps the students at that virtual laboratory.

The use of an integrated software in the learning and evaluation processes is coordinated by a specialist in Wiris. A module of introduction of the software, in this case Wiris, was also designed and elaborated.

We chose the software Wiris for our courses, there were some reasons which made it different from the other commercial software and that more suitable for the UOC:

- On-line and local versions. The access of our students to the virtual campus is possible from any computer, and because of that an also accessible software from any computer is important
- Multilingual. We have Spanish and Catalan versions, the two languages in which these courses are offered. Most of the other mathematical software is in English, which is an added difficulty for our students.
- Intuitive commands. The screen resembles very much a blackboard or the sheet of paper where the student could write.

The design of the didactic material, although impregnated with the use of Wiris, was designed and created to be used with any other software of similar characteristics. The essential feature is the integration of mathematical software, not one in particular. The specific software is contingent, the use of software is necessary.

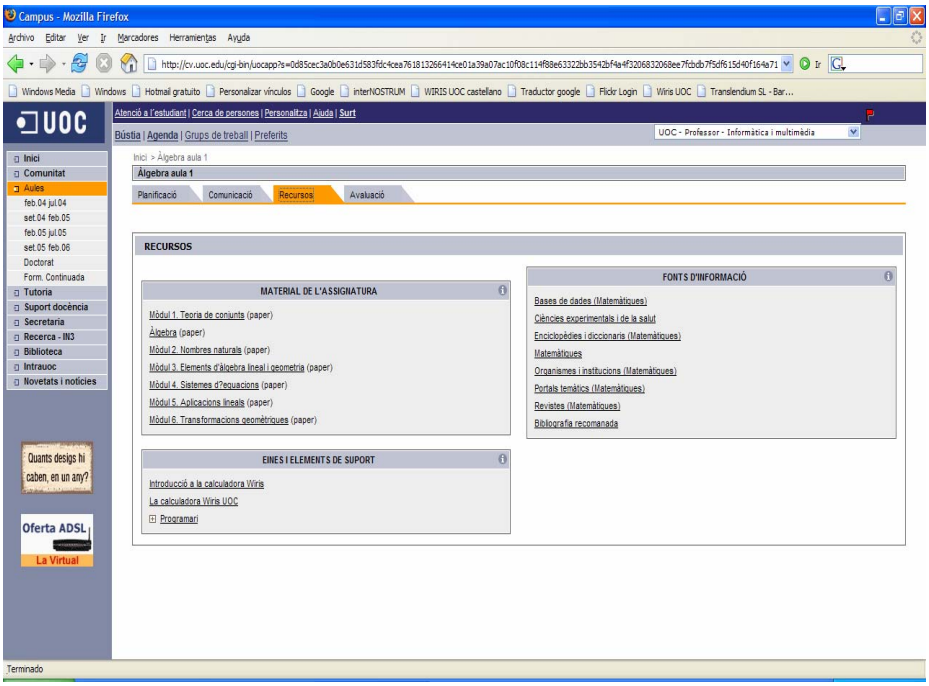


Fig. 2. Contents in the Algebra course inside the UOC Virtual Campus

The following points can synthesise the process of design and creation of the Algebra and Calculus courses:

1. A team of professors of the mathematical area (UOC) with one coordinator selected the authors among professors in the new paradigm, with experience in the use of the TIC, in the on-line teaching and in the creation of educational material. A total of seven authors (one of them specialist in Wiris) and two coordinators formed the working team.

2. Distribution of the previously selected contents in the different modules. Also distribution of the total credits among the modules.

3. The team designed the different modules. The contents of each module were created by one or two authors, working with the coordinators.

4. Integration of Wiris not only into the didactic material in paper (capture of screens), pdf or web (interactive) but into the system of evaluation. Particularly the accomplishment of personalised practices (different for every student) generated with Wiris.

10

The new courses are a methodological improvement that is based on the use of TIC (mathematical software and virtual campus). The generated digital material is a mixture of pdf documents, interactive web material and practices with Wiris. The next step should be a digital material of quality, with mathematical markup, using ontologies and semantic web technologies, and probably working with the learning objects paradigm.

3 Integrating new material in the UOC Virtual Campus

3.1 Classrooms inside the Virtual Campus

Inside the Virtual Campus, one of the most important spaces for the student, if not the most important one, is the virtual classroom (Fig. 2). There is a virtual classroom for each subject the student is registered into. Each subject classroom is a virtual space, managed by a professor, where the whole educational activity related to that subject is developed. In this space the student will find out all subject-related documentation (detailed syllabus, basic course material, mathematical software and, also, complementary material). Furthermore, each classroom includes virtual spaces that facilitate communication among students as well as between the students and the professor.

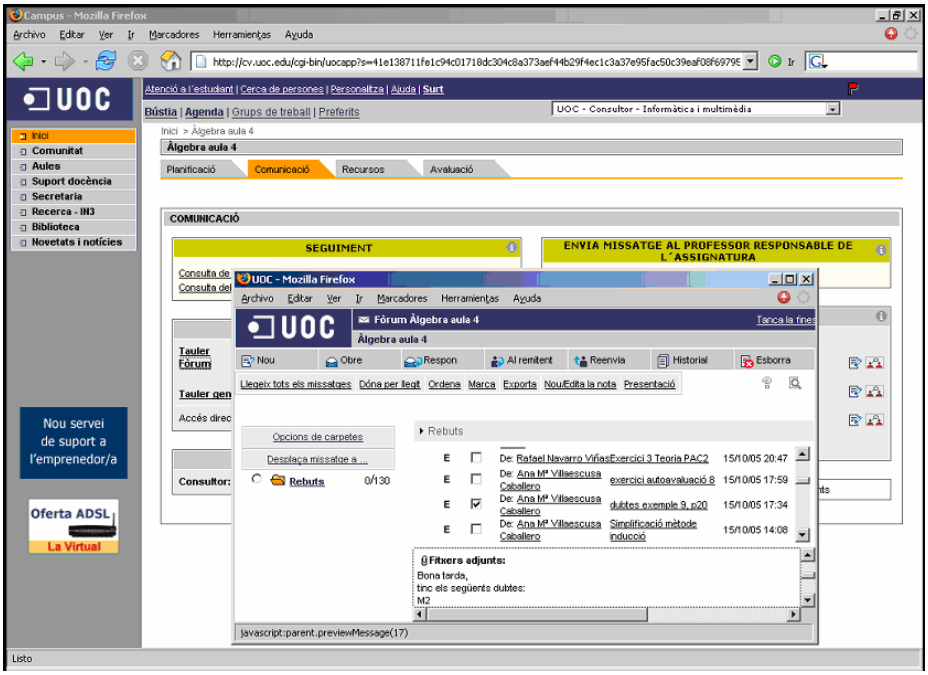


Fig. 3. A classroom inside the UOC Virtual Campus

One of these spaces acts as a notice board, its function is to facilitate the direct professor-student communication (this space is used by the professor to establish the course dynamics by means of board messages and annexed documents to these). The other communication space in the classroom is an open Forum, where students can formulate their questions about the related subject contents –which will be answered by other students or by the professor- and even start academic debates.

The classroom, as a communication space, also offers other advanced options that facilitate the synchronous communication among students (information about partners on-line, integrated chat service, etc.) and even the monitoring process of each student activity (professors can easily obtain information about which students have read a certain message, or when was the last time a student got connected to the classroom, etc.)

On the other hand, the classroom also offers other spaces that contribute to add value to the Virtual Campus, as the possibility of consulting on-line qualification reports or the existence of an incorporated academic calendar, in which each subject key dates are settled down.

3.2 Detailed subject syllabus

One of the main classroom functions is to serve as a repository for the entire course documentation. In this sense, students will find out inside the classroom a detailed syllabus of the course. This syllabus includes subject objectives, methodology, citation of basic and complementary material, information about the mathematical software to use, a clear description of the evaluation system and, finally, key dates that the student should consider. The detailed syllabus is not only the first document a student should read but also a document that they should check with relative frequency. This document is upgraded at the beginning of each semester by the faculty staff associated to the subject.

3.3 Electronic format material

The material related to each subject is divided into two parts: the basic one (a material that the student must read) and the complementary one (an optional reading material that allows to complete or to revise certain mathematical concepts or even to get inside certain contents included as basic material). Basic material is mainly composed of a series of modules which contain the concepts and mathematical techniques established by the course objectives. The focus of this material is more practical than theoretical (it is designed to show students mathematical applications to computer science rather than to offer too many abstract contents). In this sense, the material includes several solved examples relating math applications to databases, cryptography, computer networks, algorithmic and computer graphics. At the same time, this material integrates the use of mathematical software in problem solving (the main objective here is to automate, by using software, the execution of calculation algorithms, so

12

that students can dedicate more time to concentrate in mathematical concepts and their applications). To avoid an excessive dependence of any math software, the material usually solves each problem with several alternative programs (Wiris, Maple, Mathcad and Mathematica). In this sense, it could be said that this material integrates the use of mathematical software and, at the same time, it is software independent. This basic material is available, in a PDF version, from the classroom (Fig. 3). Additionally, at the beginning of each course, a hard copy of it is distributed to each student [3]. As part of the course basic material, students also have access to an electronic (web) material that introduces them in the use of the selected mathematical software.

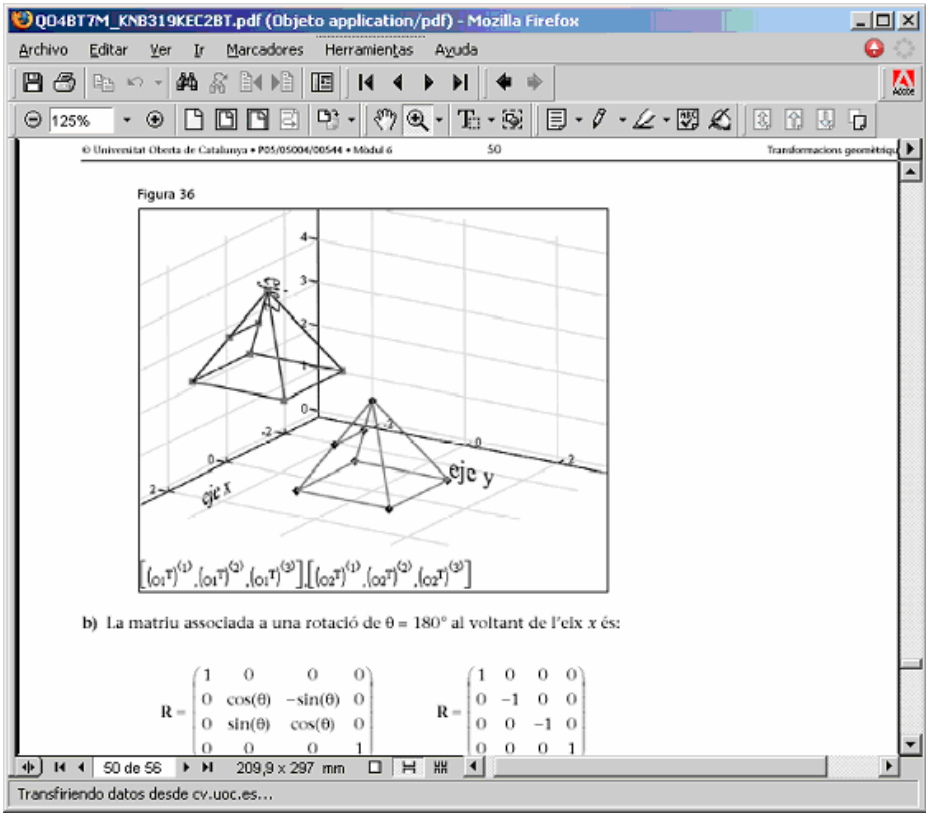


Fig. 4. PDF material showing use of software

Finally, students also have access to diverse complementary materials that allow them:

a) to reinforce some math core concepts, which are necessary for the correct understanding of the subject,
b) to consult additional collections of solved problems, or
c) to acquire deeper knowledge in some aspect of the basic material

Some of these complementary materials are available at the beginning of the course, while others are introduced by professors according to the course development.

3.4 Use of Wiris: software integration and laboratory

Currently, the UOC uses Wiris as the mathematical software for its courses. Wiris is an advanced numeric and symbolic software which can be used both on-line and off-line (local version). As Wiris is an on-line software based on Java, integration with our Virtual Campus is easy.

In order to reduce, as much as possible, the learning curve for this software, an introductory web material has been developed (Fig. 4). This basic material introduces students in the use and applications of Wiris.

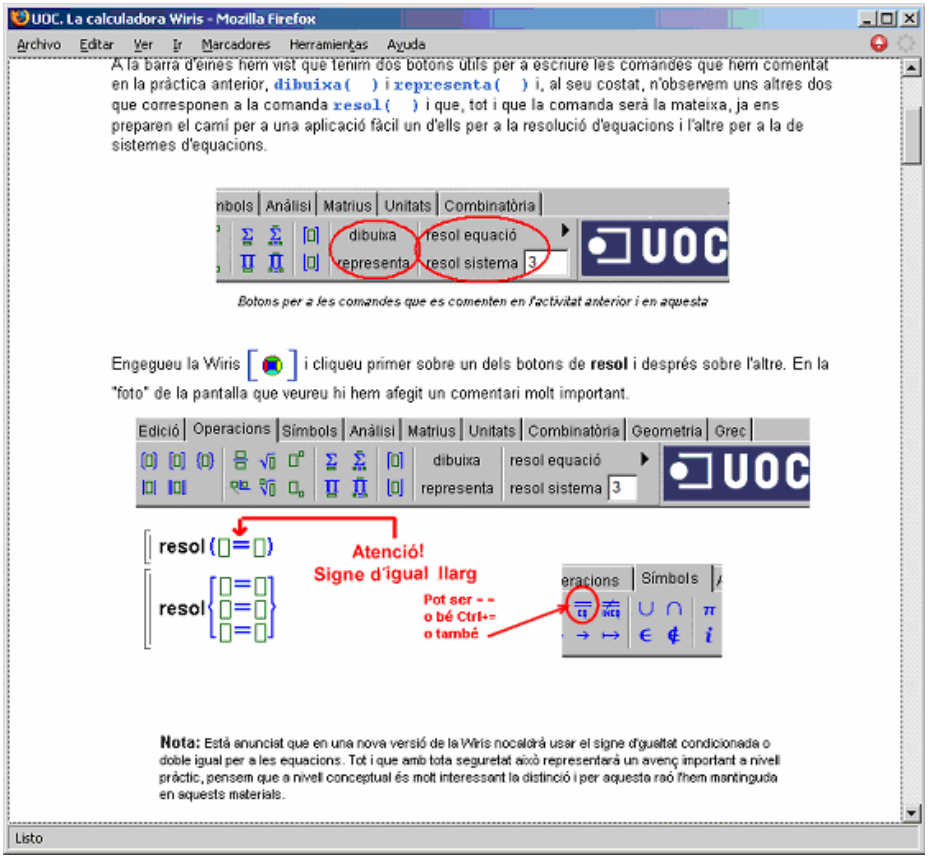


Fig. 5. Web material explaining basic use of Wiris

14

As it has already been noted, the integrated use of mathematical software is a main objective of the curriculum. For that reason, basic materials contain several solved examples using Wiris. Furthermore, the use of software is promoted by professors and it can even be necessary in the resolution of certain activities. The software, if conveniently used, can constitute not only a helpful computer tool for performing calculations but also an excellent learning tool that allows students to experiment with the math techniques and concepts introduced throughout the material.

In order to (a) motivate the use of software in the realisation of the proposed activities, and (b) answer the possible doubts about its use, each subject has a common classroom called Wiris laboratory, which is supported by an specialised-software professor.

4 Conclusions

In this paper, we have explained and discussed the e-learning model that we use, at the Open University of Catalonia (UOC), for those subjects related to the mathematical knowledge area. Key points of our methodology are: (a) importance of performing a top-down course design, taking special care of the context in which the course has to be developed (a computer science degree, in our case), (b) course focus centered around math engineering applications instead of math theory, (c) integrated use of software throughout the course, (d) development of electronic materials according to points mentioned above, and (e) special attention paid to each student's background in order to provide them with the already mentioned necessary concepts. We think that our ten years experience as an e-learning institution has helped us to develop a complete pedagogical model that can be helpful to other universities interested in offering math courses on-line.

References

1. Álvarez, M.; et al (2002): *Matemáticas, software y e-learning: La experiencia de la UOC*. Fórum Tecnológico, número 3. ISSN: 1579-3818.

2. Berger, T.; Pollatsek, H. (2001): "Mathematics and Mathematical Sciences in 2010: What Should Students Know?". Available from www.maa.org/news/students2010.html

3. Fortuny, G.; et al (2005): *Álgebra*. FUOC. Eureka Media. ISBN 84-9707-658-3. Barcelona.

4. Huertas, M., et al (2003): *Uso e integración de las TIC en asignaturas cuantitativas aplicadas*. XI Proceedings Congreso Universitario en Innovación Educativa en las Enseñanzas Técnicas. Vilanova i la Geltrú, Barcelona

5. Langtangen, H.; Tveito, A. (2000): "How Should We Prepare the Students of Science and Technology for a Life in the Computer Age?". Mathematics Unlimited - 2001 and Beyond. Springer Verlag.

6. Monereo, C.; Pozo, J. I. (2003): *La Universidad ante la nueva cultura educativa*. Editorial Síntesis. Madrid.

7. Raschke, C. (2003): *The digital revolution and the coming of the postmodern university*. RoutledgeFalmer. London.

An online summer course for prospective international students to remediate deficiencies in Math prior knowledge: the case of ALEKS

Dirk Tempelaar, Bart Rienties, Martin Rehm,
Joost Dijkstra, Mark Arts, and Geke Blok

Maastricht University, Faculty of Economics and Business Administration
PO Box 616, 6200 MD Maastricht, the Netherlands
D.Tempelaar@KE.UNIMAAS.NL

Abstract. This paper is based on the experiences with remedial online learning from a national collaboration initiative of University of Amsterdam, Erasmus University and Maastricht University in the Netherlands, called 'Web-spijkeren' (http://www.web-spijkeren.nl/). This project, supported by grants for educational innovation of SURF, the Dutch higher education and research partnership organisation for ICT, aims at developing electronic tests and tutorials for prospective students. Increased heterogeneity of prospective students causes study planning problems. Adaptive testing and remedial teaching might mitigate these problems. The summer course described in this contribution is based on the tool ALEKS: Assessment and Learning in Knowledge Spaces. Knowledge space theory is a branch of mathematical cognitive science that is especially suitable in the design of adaptive diagnostic tests and tutorials for remedial learning in highly structured domains as mathematics. This contribution describes both the foundations of the tool ALEKS, and the usage of ALEKS in our summer course.

1. Introduction

Acceptance to a bachelor or master program has traditionally been based on a required (combination of) degree(s), experience and/or skills. However, due to increasing internationalization of students (e.g. 60 per cent of enrolment at Faculty of Economics and Business Administration at Maastricht University is by international students), the introduction of the Bachelor-Master structure in Europe, and the new accreditation procedures by the Treaty of Bologna, heterogeneity of enrolled students has increased. Although a foreign student formally should be accepted to a study program according to the Bologna Treaty, the actual level of knowledge and skills can be below the level of regular Dutch students. Moreover, for some (international) students, the lack of prior knowledge is too large and remedial teaching before entering a programme is necessary (Brouwer et al., 2004). In addition, most students are unable to judge for themselves whether they possess sufficient prior-knowledge and/or experience to start a bachelor or master program (Prins, 1997).

In the past, several remedial teaching programs have been developed in the Netherlands. Van Leijen et al. (2005) argues that remedial teaching programs are delivered

with varying degrees of success in terms of students completing the program, depending on motivation of students, involvement of teachers and learning environments. In addition, since higher education institutions now have to compete on a European or even global market (Dittrich et al., 2005), regional/national remedial courses in a fixed (physical) location with traditional teaching methods seem inadequate to meet these new challenges. Recent developments in the area of e-learning reduce the limitations of time and place (Vrasidas & Zembylas, 2003) and could therefore help to face these problems.

Based on the experiences from the past and anticipated future changes in higher education in Europe, the University of Amsterdam, the Erasmus University Rotterdam and Maastricht University in collaboration with SURF-foundation, have decided to accept these challenges. The goal of this collaboration is to find successful educational formats focused on flexible remedial teaching in case of a high level of heterogeneity of student intake (Brouwer et al., 2004).

The goals of this paper can be split up into three interconnected parts. First, how can students assess their current level of mastery before joining a (bachelor) program? Second, if the level of mastery of individual students appears to be low, how can online summer courses help to tackle these potential deficiencies? And finally, how can online summer courses be designed to enhance motivation and increase completion rates of students? In this paper, the aspects of successful online remedial teaching are described in a model for online remedial teaching. Afterwards, one of the online summer courses offered at Maastricht University will be used as case-study to show how the online remedial teaching model can be implemented in practice.

2 The online remedial teaching model

According to Bryant et al. (2005), there are many definitions about online and/or distance education. Distance education encompasses two important elements, namely distance teaching and distance learning. Distance teaching regards mainly the way in which instruction is provided, whereas distance learning concerns optimizing student learning behaviour (Keegan, 2002). Various definitions are used for online education. Although in most of the definitions terms like web-enabled and online point at the way instruction is provided, it does not automatically lead to distance education (Bryant et al., 2005). However, in this article the term online (education) is used instead of distance education as the element of distance education is only related to the short time period before students start (physically) at a the regular curriculum.

Van Leijen et al. (2004) conducted research in various remedial teaching programs in the Netherlands. A program offered during the summer period induces an incentive problem as most graduated high school students have a strong preference to do other things besides studying. Hence, the challenge arises to construct a programme that achieves a balance between study time and time for summer activities in such a way that it provides sufficient motivation to keep students engaged in the course. Therefore, an institute planning to design and implement an online remedial summer course should consider the following aspects:

1. Accessible & Available 24/7 online: Use of internet helps to overcome the barriers of time and place, and thus enables participants to work anywhere they like and at times that suit them most (Vrasidas & Zembylas, 2003).
2. Adaptive: Each student is in a sense unique. Hence, the programme should ideally allow for an individualized learning path based on prior knowledge, learning style and preferences of the student (Falmange et al., 2004, Abdullah, 2003).
3. Rapid feedback: Besides the fact that it is pedagogically better to provide rapid feedback on performance (Draaijer, 2004), it is also important because the period before the course starts is short and often fully planned with other activities. Furthermore, rapid feedback stimulates interaction in an online course (Vrasidas & Zembylas, 2003).
4. Interactive: "A fundamental component of distance education is the communication medium" (Bryant et al., 2003, p. 257). Being solely available online, the course and learning environment should stimulate interpersonal contact in order to motivate participants to remain engaged (Ronteltap & Van der Veen, 2002).
5. Flexible learning methods and assessment: Given the fact that learning and assessment methods are subject to change, the program should be flexible enough (Segers, 2004).

3 Prior knowledge tests and Online Summer Courses in Practice

As most students in the bachelor curriculum of the Faculty of Economics and Business Administration at Maastricht University have (some) problems with Mathematics, the first prior knowledge tests and online summer courses were specifically developed for tackling these problems.

Prospective students of the bachelor studies 'International Economics' and 'International Business' at Maastricht University were given the opportunity to make an online entry (diagnostic) test in order to assess their prior knowledge of mathematics. The online test was available and accessible 24/7 on the Internet. The test was composed of exercises in open-question type form (Tempelaar & de Gruijter, 2004, Sclater & Howie, 2003). Anyone who filled in the online entry test received elaborate feedback via E-mail from an expert. Overall, the test was viewed 524 times and 230 students made a serious effort to complete the whole test.

Of the 230 students who participated in the entry test, 44% (101) registered for one of the bachelor programs at the Faculty of Economics and Business Administration. Of these 101 students, 55 were willing to invest 60 to 80 hours to remediate his/her deficiency. Those 55 constituted one group of participants of the summer course. A second group of participants of the summer course constituted of prospective students of the Master of Science in Social Protection Financing of the Maastricht Graduate School of Governance. These 18 students were required to take the summer course in order to be allowed in the master program.

The online summer course is based on the server-based tool ALEKS. Students can approach ALEKS through internet; the summer course was delivered in a pure dis-

tance format, without any physical contact with lecturers or peers. Lecturers had a limited role: they were available by mail for organizational, tool-related and mathematical issues. The majority of questions sent in appeared to be of the first two categories; the coverage of content material in ALEKS proved to be satisfactory for most students. Participation by BA prospective students was completely voluntary and in no way related to the official admission procedure of the university. The only reward was a (unrecognized) certificate at a graduation ceremony. Moreover, participation in the summer course was free of charge and costs of the course were funded by Maastricht University and SURF-foundation. For the MA prospective students, participation was required.

4 ALEKS

4.1 General Information

The ALEKS system combines adaptive, diagnostic testing with an electronic learning & practice tutorial in several domains relevant for higher education. In addition, it provides lecturers an instructor module where students' progress can be monitored, educational standards can be adapted to a particular college, or course, and other administrative tasks can be carried out. ALEKS is a commercial software product.

4.2 Assessment

The ALEKS assessment module starts with an entry assessment in order to evaluate precisely a student's knowledge state for the given domain (eg. College Algebra). Following this assessment, ALEKS delivers a graphic report analyzing the student's knowledge within all curricular areas for the relevant course, based on specified standards. The report also recommends concepts on which the student can begin working; by clicking on any of these concepts or items the student gains immediate access to the learning module.

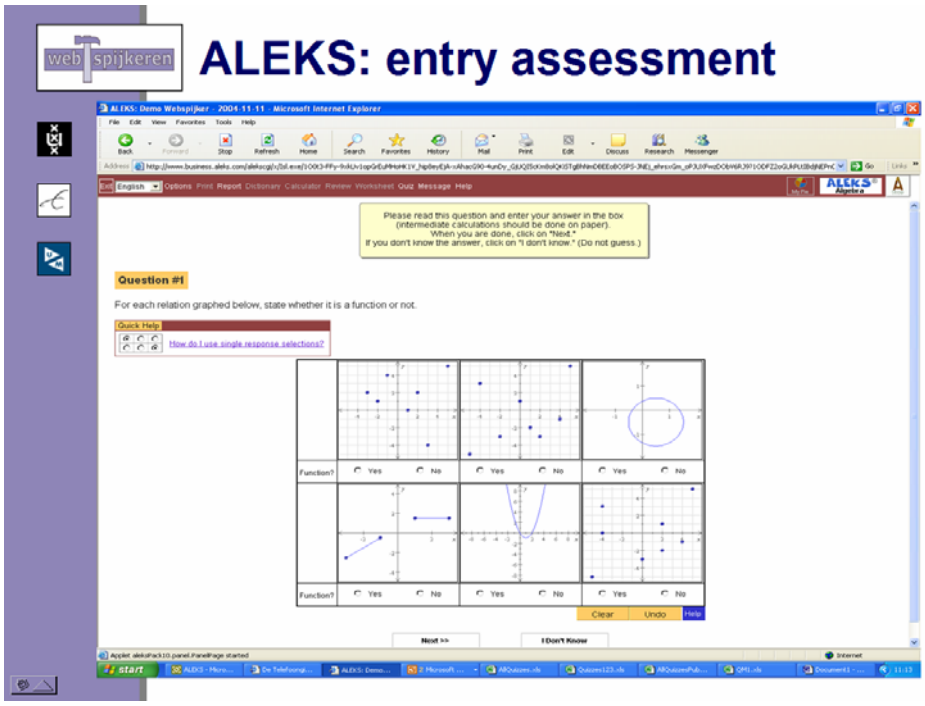


Fig. 1. Sample of an assessment item

Some key features of the assessment module are:

- All problems require that the student produce authentic input (that is, there are no multiple-choice questions).
- All problems are algorithmically generated.
- Assessment questions are generated from a carefully-designed repertoire of items ensuring comprehensive coverage of the domain.
- The assessment is adaptive: the choice of each new question is based on the aggregate of responses to all previous questions. As a result, the student's knowledge state can be found by asking only a small subset of the possible questions (typically 15-25).
- Assessment results are always framed relative to specified educational standards. The system allows instructors to customize the standards used by their courses with a syllabus editor (part of the instructor module). Both the assessment and learning modules are automatically adapted to the chosen standards.

4.3 Report

The report provides a detailed, graphic representation of the student's knowledge state by means of pie-charts divided into slices, each of which corresponds to an area of

the syllabus. In the ALEKS system, the student's progress is shown by the proportion of the slice that is filled in by solid color: if the slice is entirely filled in, the student has mastered that area, if it is two-thirds filled in, the student has mastered two thirds of the material, and so forth. Also, as the mouse is held over a given slice, a list is displayed of items within that area that the student is currently "ready to learn," as determined by the assessment. Clicking on any of these items gives access to the learning mode (beginning with the item chosen).

4.4 Learning Mode

At the conclusion of the assessment ALEKS determines the concepts that the student is currently ready to learn, based on that student's current knowledge state. These new concepts are listed in the report, and the learning mode is initiated by clicking on any highlighted phrase representing a concept in the list. The focus of the learning mode is a sequence of problems to be solved by the student, representing a series of concepts to be mastered. The facilities offered by the learning mode are as follows:

- Practice (that is, the problems themselves);
- Explanations of concepts and procedures;
- Dictionary of technical terms;
- Calculator (adapted to the topic studied, e.g. in statistical items, a special "statistics calculator" is provided).

For example, a student working on a particular problem may "ask for" an explanation of that problem (by clicking on the button marked "Explain"). The explanation typically provides a short solution of the problem, with commentary.

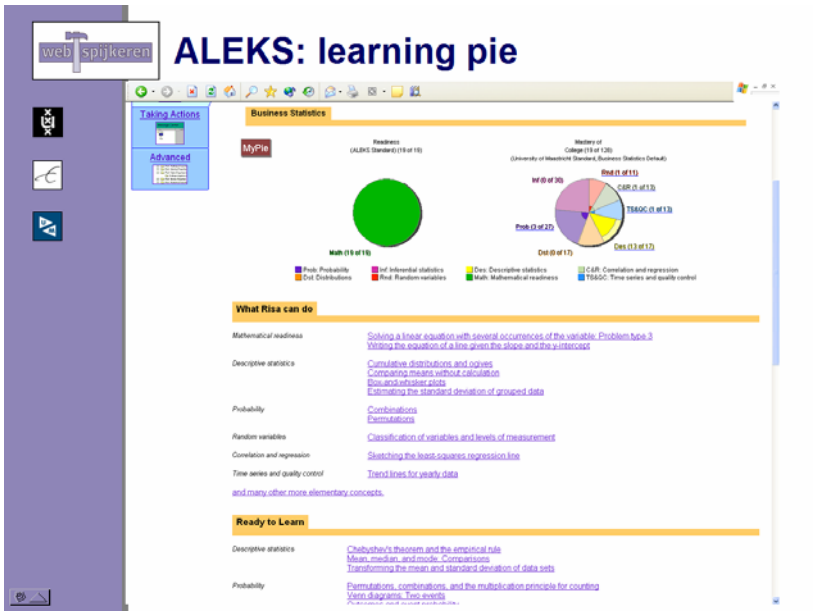


Fig. 2. Sample of a learning report

After reading the explanation(s), the student may return to "Practice" (by clicking on the button marked "Practice"), where she or he will be presented with another problem exemplifying the item or concept just illustrated. If the student is successful in solving the problem, the system will offer (usually) two or three more instances of the same item to make sure the student has mastered it.

In the text of problems and explanations, certain technical terms such as "addition", "factor" and "square root" are highlighted. Clicking on any highlighted word or phrase will open the dictionary to a definition of the corresponding concept. The dictionary can also be used independently of the current problem to look up any term the student may be curious about.

A graphing calculator is available for computing and displaying geometrical figures in analytical geometry and calculus. Other, related features of the learning mode are Feedback, Progress monitoring, and Practice. Whenever the student attempts to solve a problem in the learning mode, the system responds to the input by saying whether or not the answer is correct and, if it is incorrect, what the student's error might have been.

More generally, ALEKS follows the student's progress during each learning sequence, and will at times offer advice. For example, if a student has read the explanation of a problem a couple of times and yet continues to provide incorrect responses, ALEKS may suggest -- depending on the circumstances -- that the student looks up the definition of a certain word in the dictionary. ALEKS may also propose that the student temporarily abandon the problem at hand and work instead on a related but easier problem. The capacities of the ALEKS system to monitor and guide student learning is flexible and multi-faceted.

When a student has demonstrated mastery of a particular item by repeatedly solving problems based on it, ALEKS will encourage the student to proceed to a new item.

4.5 Instructor Module

The instructor module enables lecturers to monitor student progress and achievement; to view and change the standards applied in the generation of assessment reports; and to carry out other administrative tasks. In detail, the lecturer can:

- View and print reports for individual students;
- View and print a list of students, with a summary of information for each student including assessment results, progress in the learning mode, and total time spent in the system;
- View and print synthetic reports for entire courses, giving an overview of the class's strengths and weaknesses;
- View the standards used by default for a course, with the option of editing standards pertaining to that course only;
- Edit student registration data or retrieve forgotten passwords.

5 Structure of online summer courses, motivation and impact on regular courses

The most natural way to assess the success of the Summer course is provided by an analysis of the outcomes of the first regular course: Quantitative Methods 1, or QM1.

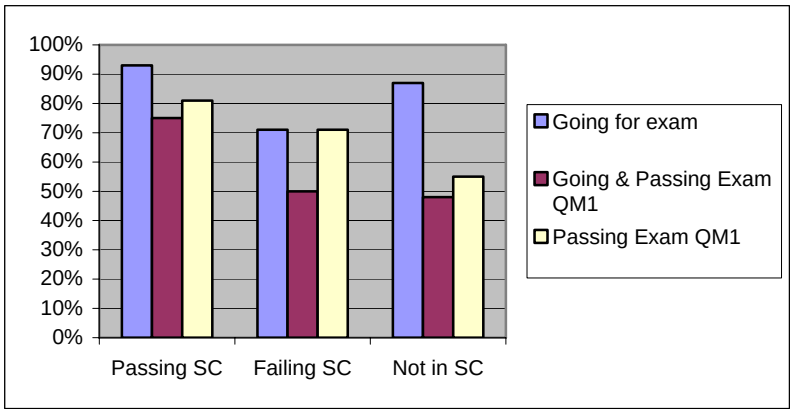


Fig. 3. Course performance of QM1, decomposed according Summer course participation

The QM1 course is composed of Mathematics, Statistics, and a third but minor part introducing applied computer science. Succession rates amongst Summer course participants are high, and differ significantly from those amongst the non-participants. Of the 29 students who successfully completed the Summer course, 81% pass the QM1 course. For the 26 participants who did not manage to fully complete the Summer course, passing rate is 71%. For the large group of 799 non-participants, passing rate is only 55%; see Figure 3 for a summary of these data. Although students participating but not succeeding the Summer course do remarkably well, those figures are to some extent non-representative, since the proportion of students in this group actually participating in the final exam of QM1, is much lower than in the other two groups: only 7 out of 10 students show up. The combination of exam participation and passing the exam generates the last piece of data: 'Going and Passing the Exam'. Participating *and* passing the Summer course appears so to be a strong predictor of the participating *and* passing the QM1 course.

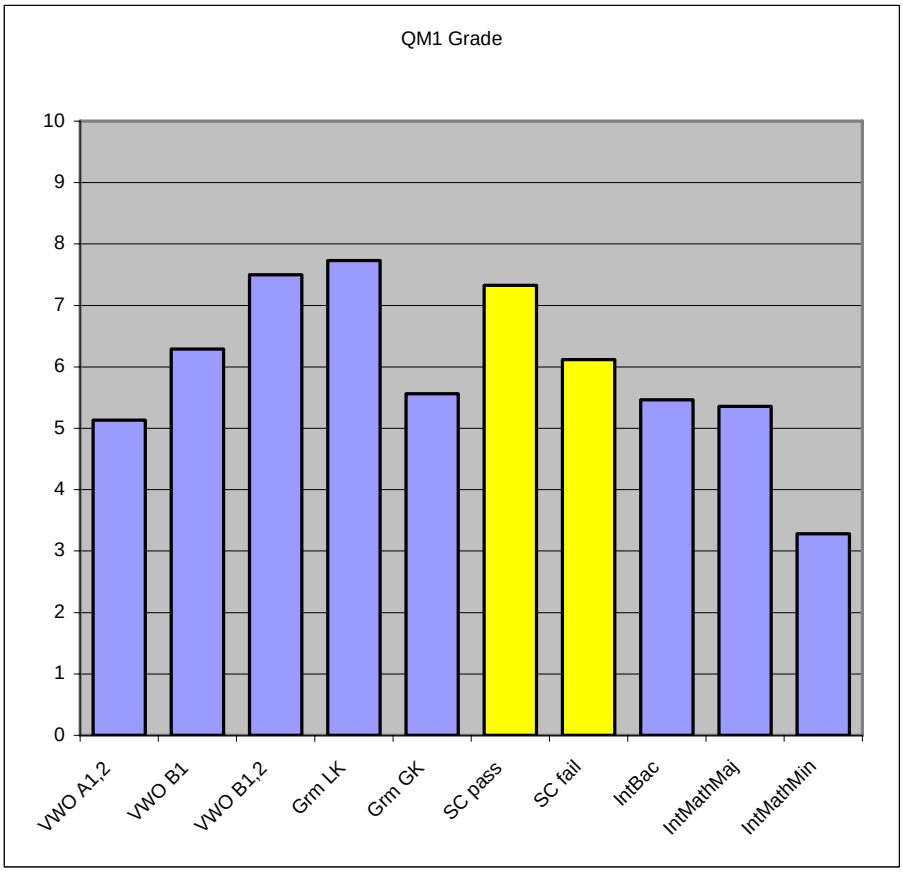


Fig. 4. QM1 grades in the final exam, decomposed according Summer course participation and prior Math education

Another expression of the success of the Summer course is found in the confrontation of QM1 course performance of students with different prior Math education, and their participation in the Summer course. As Figure 4 indicates, the heterogeneity in prior education is large: Dutch students can have taken three different types of math prior education, either of minor type (VWO A1,2), or of major type (VWO B1 or VWO B1, 2). German students, constituting about two-third of our student population, either have math taken at GK or 'Grundkurse' level, a minor program, or at LK or 'Leistungskurse' level, a major program. Further relative small groups are students with an International Baccalaureate (IntBac), and foreign students outside the Netherlands and Germany having taken a national secondary education degree with math either at major or minor level (IntMathMaj and IntMathMin, respectively). Students in our Summer course were all German students (only foreign students were invited, since these students were expected to have the greatest transfer problems), primarily of GK level (30 students), with a small minority of LK level (8 students). The LK

students did not improve their passing rates relative to the students who did not participate in the Summer course (84% for both groups); however, the GK level students did make an important jump in passing rate: from 51% to 88%. Or alternatively, expressed in terms of the grade in the final exam (with 5.5 being the floor to get a passing outcome), Figure 4 indicates that students who successfully participated in the Summer course, increased their average grade from about GK level (5.5) to a level (7.3) similar to that of students with a math major in prior education.

The finding that participants of the Summer course outperform non-participants is in itself no proof of the successfulness of the Summer course: there might be a selection bias. If the Summer course attracts a non-representative group of students, such as students being more than average motivated, the effect of participation in the Summer course might be explained by having the best motivated students in this group. To investigate the extent in which the increased success rate in the exam can be attributed to a selection bias component at the one side, and a real learning effect of the Summer course at the other side, the partial math and statistics scores in the final QM1 exam of Summer course participants and non-participants are confronted. Figure 5 provides a graphical depiction of these data. Summer course participants indeed do better than non-participants in both math and statistics, but their advantage in math is much larger than their advantage in statistics. In fact, the only statistically significant difference in partial exam scores is the math score of student passing the Summer course, and the math score of non-participants. So if any selection bias is present, the effect it causes is small, statistically insignificant and surpassed in size by a true learning effect.

A second check on the presence of a selection bias caused by motivational differences is provided by ALEKS study time data of students working in the ALEKS business Statistics module during the course QM1. Whereas a small group of students used the ALEKS College Algebra module during the Summer Course on voluntary basis, all students used the ALEKS Business Statistics module in the QM1 course in order to practice and prepare for their statistics quizzes. Study time (logon time) in ALEKS Business Statistics might therefore be regarded as a good proxy for student motivation. Figure 6 contains the study time data for students with different prior education levels, and Summer course participants.

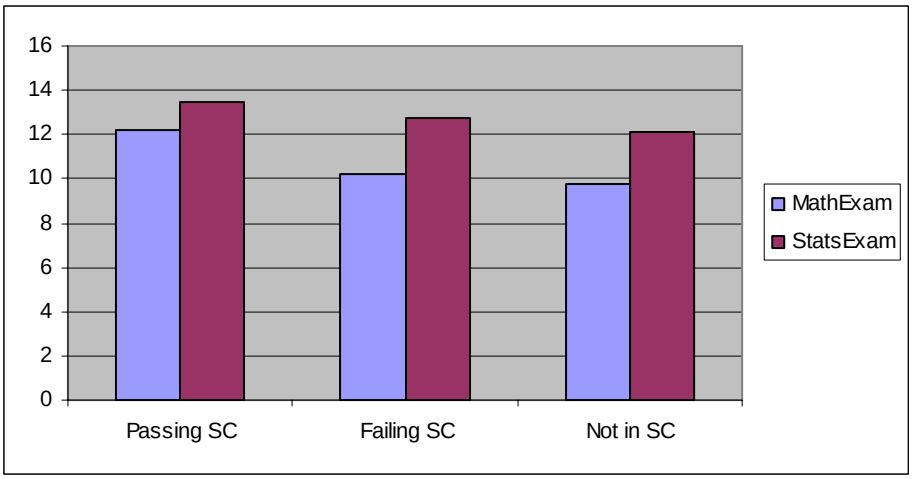


Fig. 5. Partial scores in QM1 final exam, decomposed according Summer course participation

If anything, Summercourse participants study less than students in their reference group, the students with a GK prior education. Most striking is the large difference in study time between Dutch and foreign students.

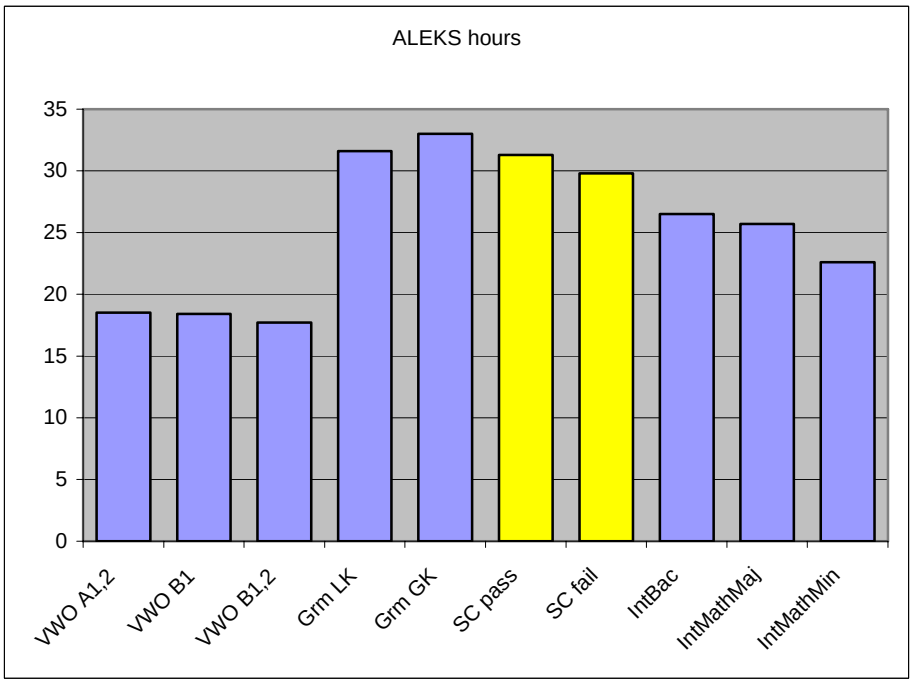


Fig. 6. ALEKS Business Statistics study hours, decomposed according Summer course participation and prior Math education

Therefore, neither partial scores, nor study time, suggest the existence of a strong selection effect, thus suggesting that the increased course performances of Summer course participants can be attributed to a true learning effect.

In total 72 students filled in the evaluation at the end of the course; see Table 1. On a scale from 1 to 10, students were very positive about both the functioning of the tool (8.6) as well as the online summer course as such (8.8). More specifically, on a Likert scale from 1-5, students felt that the course had offered them a lot (4.6) and enabled them to remediate their knowledge to such an extent that they feel ready to start in Maastricht (3.7). Remarkably, the evaluations of students failing the summer course are similar to those passing the summer course.

Table 1. Evaluation of Online Summer course Mathematics

This Summer course offered me a lot	4.6
The contents of the Summer course were inspiring	4.2
The format of the Summer course was good	4.4
The Summer course was well organized	4.6
The quality of the material in ALEKS is good	4.3
The material in ALEKS motivated me to keep up with the subject matter	4.0
Learning in an e-learning environment as ALEKS is not different from learning from a hard-copy book	2.6
It was fun that I could attend this Summer course via the internet	4.4
I gained enough knowledge and skills in mathematics to start with my study in Maastricht	3.7
It was easy to motivate myself to finish this Summer course	3.8
It was good that I could work on the subject matter at my own pace	4.6
I think that I have learned more by individually attending this course than I would have learned if I had to collaborate	3.7
Questions via e-mail were answered well by the teacher	3.7
Give an overall grade for the quality of support you were given by ALEKS in this Summer course (1 = very bad - 10 = very good)	8.6

6 Conclusion and Discussion

In this paper, the question how prior knowledge tests and online remedial teaching courses can contribute to mitigating the problems of heterogeneous enrolment of students was dealt with. First of all, an online remedial teaching model was developed. The five success factors, which an online remedial course developer should take into account, are 24/7 access and availability on Internet, adaptiveness, rapid feedback, interactive, and flexible learning methods and assessment. Next, the online remedial teaching model was implemented in practice at the online summer course of Mathematics at Maastricht University. Both learning outcomes, as evidenced in regular course performances and evaluations at the end of the course were very positive. On average, students were highly motivated to remediate their knowledge gaps and they worked approximately 70 hours for the entire course. Furthermore, students indicated that they now have more confidence of making a good start at the institute. Exactly half of the group of students that started with the course eventually received a certificate.

The case-study of the online summer course Mathematics and the experiences from other courses offered by the three institutions taking part in the project "Web-spijkeren" indicate that it is possible to construct an online course that continuously motivates students and thereby increasing passing rates. The online course was build with existing ICT-structures that are also available at other institutions. If an institute provides accurate means and expertise for establishing an online summer course program, it should be possible to mitigate the problems with increased heterogeneity of enrolments of students.

Further research is necessary to prove whether online summer courses have a temporal or structural effect on the (prior) knowledge level of students. In addition, it remains to be investigated whether the participants, in comparison to those who did not take part at the summer course, perform better in the respective courses in the curriculum. Further research efforts are focused on the relationship between student characteristics and performance in the summer course, such as students' preferred learning styles, their metacognitive abilities, and their level of achievement motivation.

References

1. Abdullah, Sophiana Chua. (2003). Student Modelling by Adaptive Testing - A Knowledge-based Approach. Dissertation. University of Kent. http://www.cs.kent.ac.uk/pubs/2003/1719/ [viewed 14th of March 2006]
2. Brouwer, N., Rienties, B.C., Engelen, A.J.M. van. (2004). Controlling Document Project Web-spijkeren. Universiteit van Amsterdam, Universiteit Maastricht, Erasmus Universiteit Rotterdam. http://www.web-spijkeren.nl/ [viewed 14th of March 2006]
3. Bryant, S., Khale, J., Schafer, B. (2005). Distance Education: A Review of the Contemporary Literature. Issues in Accounting Education. 20 (3). 255-272

4. Dittrich, K., Frederiks, M. (2005). Accreditatie in Nederland en Vlaanderen: een eerste balans. (Accreditation in the Netherlands and Flanders: a first balance). Tijdschrift van Hoger Onderwijs.Vol. 1 2005
5. Doignon, J.-P. & Falmagne J.-Cl. (1999). Knowledge Spaces. Springer. Berlin
6. Draaijer, S. (2004). Flexibilisering van toetsing. (Flexibility in Assessment). Utrecht. Digital Universiteit
7. Falmange, J., Cosyn, E., Doignon, J., & Thiéry, N. (2004). The Assessment of Knowledge, in Theory and in Practice. http://www.business.aleks.com/about/Science_Behind _ALEKS.pdf [viewed 14th of March 2006]
8. Keegan, D. (2002). Definition of distance education. Distance Education: Teaching and Learning in Higher Education. Boston, MA. Pearson Custom Publishing
9. Van Leijen, M., Wieland, A., Rienties, B. et al. (2005). Quick scan onderzoek naar de initiatieven elders. (Quick Scan research of remedial teaching initiatives). Universiteit van Amsterdam, Universiteit Maastricht, Erasmus Universiteit Rotterdam
10. Prins, J. (1997). Studieuitval in het wetenschappelijk onderwijs. (Student drop-out in Academic Education). Nijmegen: University Press
11. Ronteltap, F., Van der Veen, J. (2002). Samenwerkend leren met ICT. (Working and learning together with ICT). Universiteit Utrecht, IVLOS en Universiteit Leiden. ICLON
12. Segers, M. (2004). Assessment en leren als een twee-eenheid. (Assessment and learning as two become one). Tijdschrift van Hoger Onderwijs. Vol. 4 2004
13. Sclater, N., Howie, K. (2003). User requirements of the "ultimate" online assessment engine. Computers & Education, 40 285-306
14. Tempelaar, D. & D. de Gruijter. (2005). Computertoetsing bij de Emerge-instellingen. TU Delft
15. Vrasidas, C., Zembylas, M. (2003). The Nature of Technology-mediated Interaction in Globalised Distance Education. International Journal of Training and Development. 7(4). 271-286

Diagnostic Testing with Maple T.A.

André Heck and Leendert van Gastel

Universiteit van Amsterdam, AMSTEL Institute,
Kruislaan 404, 1098 SM Amsterdam, The Netherlands

Abstract. At the University of Amsterdam we use the Maple T.A. system for diagnostic testing of incoming mathematics and science students. Purpose of the tests is to identify students at risk of failing their mathematics and science courses in the first year and to provide students insight in their starting level of mathematical knowledge and skills. In this paper we describe our experiences with Maple T.A.

1 Background

In the Netherlands, the transition from secondary school to higher education is currently the centre of interest. Universities struggle with heterogeneous groups of students that have a disjointed body of knowledge and skills, and that are not well-prepared for the more abstract level of studies at university. Technical universities experience that incoming students lack basic mathematical knowledge and skills. They sound the alarm in national newspapers and they send a letter of advice to the Minister of Education. In reaction to the complaints from higher education and in view of the upcoming change of the examination programme, the Minister of Education questions one of the latest reforms in the education system, viz., the 'Studiehuis' (study house), in which pupils are expected to learn how to learn, are self-responsible for their learning, and work independently.

With regards to the transition from school mathematics to university mathematics, the current complaints of educators at Dutch universities about the level of mathematical knowledge and skills of incoming students are much the same as theose mentioned ten years ago by the London Mathematics Society in the report *Tackling the Mathematics Problem* [1]:

1. Students lack fluency and reliability in numerical and algebraic manipulation and simplification;
2. There is a marked decline in students' analytical powers when faced with simple two-step or multi-step problems;
3. Most students entering higher education no longer understand that mathematics is a precise discipline in which exact, reliable calculation, logical exposition and proof play essential roles.

Similar trends have been found across many European countries and many approaches to deal with the declining standard of mathematical knowledge and skills of new entrants to mathematics, science, economy, and engineering department have been proposed and implemented (see e.g. [2, 3]). The following

2

two recommendations are made in [4]: (i) use diagnostic tests to measure the mathematical competence of students upon arrival at university and (ii) provide prompt and effective support to students whose mathematical background is found wanting by the tests.

At the University of Amsterdam, an approach is being developed to cope with the mathematics problem at the threshold of the university. The approach is based on the trust that incoming students are in principle quite capable of overcoming the initial difficulties and can attain the desired level in a rather short term. Because of the heterogeneity, it is very important that both students and teachers are informed about the individual level and progress of each student. So initial and continuous assessments are an important part of the approach. For mastering the mathematical abilities, many exercise questions are envisaged together with a condensed mathematics text. A trajectory will be designed that is based on these assessments and the mastery learning of the mathematical abilities by doing many exercises. In 2005 we tried out the following four-step approach with all hundred students entering the study of chemistry, physics, astronomy, mathematics, and bio-exact sciences.

Step 1. Immediately upon arrival at the Faculty of Science, all freshmen take the one-hour diagnostic test that can be found in the appendix.

Step 2. During the first four weeks of the Calculus 1 course that all students take, they have weekly a session of two hours to practise basic mathematics.

Step 3. In the fifth week of the study year we let the students make the second diagnostic mathematics test. This digital test is made similar to the first test by randomisation of each question. Through this pretest-posttest design, students and staff can see the progress made in the meantime.

Step 4. Tutors take care of remedial teaching of mathematics using the recently published exercise book [5] and freshmen work in small self-help groups on mathematics problems under guidance of a student tutor.

An important requirement for the approach is that not much staff will be available in the long term. There are no structural funds available for this goal. So the challenge is to design a trajectory that involves only a modest staff deployment. For this reason, a pilot is undertaken with automated assessment in mathematics using the commercial package Maple T.A. [6]. This system uses the computer algebra package Maple [7] to generate and mark thousands of mathematical problem exercises from generic templates and it can deliver these exercises in various forms to students, ranging from diagnostic assessments, self-tests, and practice sessions to placement tests and summative assessments.

The development of this approach is part of two projects with other higher education institutions in the Netherlands. The project Web-spijkeren [8] has as a goal to develop didactical scenarios and assessment instrumentation for freshmen with heterogeneous mathematics levels. The project MathMatch [9] develops course materials and a database with mathematical questions for mathematics, science, and technology studies.

Our work is still in its early stages. This paper describes the instrumentation used and the results of the first try-out.

2 Overview of Maple T.A. with Blackboard Support

In this section we detail the computer algebra based assessment system Maple T.A. for Blackboard [6]. This system can be shortly described as a Blackboard-integrated system for creating tests, assignments and exercises, allowing automatic assessment of student responses and the creation of model solutions via the computer algebra package Maple. The integration with Blackboard concerns the user administration (authentication and grade book).

While some systems for computer algebra based testing and assessment already exist or are under development (for example, AiM [10], Stack [11], and WebALT [12]), the main reasons for us to choose the commercial system Maple T.A. for Blackboard are that it

- is developed far enough to have a rich set of question and assignment types;
- is based on two components, viz., Maple [7] and EDU Campus [13], which have been proved successful in educational practice;
- can be used with large groups of students without a high load on the computer environment;
- is integrated with the virtual learning environment currently in use at our university;
- offers licensed use of a familiar computer algebra system in an assessment environment.

In the following subsections we discuss the process of creating randomised questions, marking answers, and providing model solutions in Maple T.A., and the built-in facilities for students and teachers.

2.1 Authoring Questions

Initially, it is necessary to create a bank of questions that can be used in assignments. Maple T.A. questions are stored internally in Brownstone's EDU code, which is a scripting language. There exist four ways of authoring questions:

1. editing a plain-text script file in EDU code;
2. via the Question Bank Editor;
3. using the LaTeX to EDU conversion facility;
4. by way of a Maple 10 document.

Each method has its pros and cons. The first method of coding in EDU format can be done both on-line and off-line and provides the author maximum flexibility, but it is undocumented and very error prone. Use of the question bank editor can only be done while being connected to the internet. It makes heavy use of caching intermediate work and therefore the danger of loosing work and item banks is rather large. The Maple 10 document does not support all question types. We prefer the LaTeX authoring mode because it supports advanced question types, it allows off-line editing, and it supports re-use of many course materials already written in this typesetting language. It lowers the learning

4

curve of many a university teacher who wants to use the assessment system. One only has to convert via a web-facility the LaTeX code into EDU format. It is a pity that this is not made available as an off-line facility to come along with the Maple T.A. software.

Many of the question types available in Maple T.A. will be familiar to users of existing computer-aided assessment systems. Common examples of closed question types are multiple choice, multiple selection, true/false, ordering, clickable image, and matching questions. For these types answers are predefined, and can be marked automatically without difficulty. Common examples of open question types are fill-in-the-blanks (text or numerical value), essay, and graphical sketch problems. For these free-response questions it is more difficult to mark automatically.

But specific to Maple T.A. is the availability of mathematical free-response question types: (restricted) formula, multi-formula, and Maple-graded questions. The latter type is the one that we will discuss more deeply in this paper because, we believe that free-response questions are indispensable in evaluating mathematical knowledge and skills. However, there is still a purpose for multiple choice questions. For example, in Questions 22 and 25 of the pretest, the distractors make students think about the subject. Besides, a larger variety of question types in mathematical tests is attractive in itself. It is also a common misbelief to think that only standard mathematical questions leading to a clear and unique answer can be automatically assessed and that higher mathematical skills are out of reach in this approach. We refer to [14] for examples in which a computer algebra system is used to verify mathematical properties of given answers (out of an infinite number of possible, correct answers) and to assess advanced mathematical skills.

Henceforth we will use the following typical example to illustrate issues of generating randomised questions and of marking and commenting on students' responses. It is the LaTeX code for a randomised version of Question 5 in the pretest.

```
\begin{question}{Maple}
\name{powers} \type{formula}
\qutext{Simplify as much as possible: $$\frac{a^{\var{alpha}}
  (\var{npow3}b)^{\frac{1}{4}}c^{\frac{\var{gamma1}}{2}}}
  {\sqrt{a b c^{\var{gamma2}}\sqrt{\var{n}}}}\,.$$
  assuming that $a$, $b$, $c$ are positive numbers.}
\maple*{expr := $RESPONSE;
  symexpr := select(has, expr, {a,b,c});
  numexpr := eval(expr,{a=1,b=1,c=1});
  symtrue := select(has, $r, {a,b,c});
  with(StringTools, CountCharacterOccurrences);
  if simplify(symexpr-symtrue, symbolic)=0 and
     CountCharacterOccurrences("$RESPONSE", "a")=1 and
     CountCharacterOccurrences("$RESPONSE", "b")=1 and
     CountCharacterOccurrences("$RESPONSE", "c")=1
  then
     if numexpr=sqrt($n) or numexpr=($n)^(1/2)
```

5

```
     then grade := 1.0
     else grade := 0.5
     end if;
  else grade := 0.0
  end if;
  grade;}
\code{$n = switch(rint(3),2,3,5);
  $npow3  = int($n^3);
  $alpha  = range(2,5);
  $gamma1 = range(1,3,2);
  $gamma2 = range(5,9,2);
  $r = maple("simplify(a^($alpha)*(($npow3)*b)^(1/4)
    *c^(($gamma1)/2)/sqrt(sqrt($n)*a*b*c^($gamma2)), symbolic)");
  $dr = maple("printf(MathML:-ExportPresentation($r))");}
\comment{The correct answer is \var{dr}.\newline
  You get this answer by rewriting the denominator as product
  of powers with rational exponents, followed by simplification.}
\end{question}
```

From the above code listing it is clear that Maple is not only used to check the student's response, but also to calculate the answer itself, to provide feedback, and last but not least to create randomised questions. In other words, the teacher does not calculate the correct answer, mark a student's answer or provide feedback: these are done by the computer algebra system.

In the first five lines of the `\code` part, several random variables are defined in the EDU scripting language to create a mathematical formula

$$\frac{a^{\alpha}(n^3 b)^{\frac{1}{4}} c^{\gamma_1}}{\sqrt{\sqrt{n}abc^{\gamma_2}}},$$

where

$$n \in \{2,3,5\}, \quad \alpha \in \{2,3,4,5\}, \quad \gamma_1 \in \{1,3\}, \quad \gamma_2 \in \{5,7,9\}.$$

This means that we have in fact created here seventy-two exercises of similar type. This randomisation of questions makes it possible to generate many tests with questions of similar types, which is ideal for pretest and posttest designs. But also in self-assessment it is very useful that students can redo questions of similar type until they have reached the requested level of understanding.

The dollar variables defined in the `\code` part can be used in other parts of the question. For example, when the question gets instantiated, then the `\var{alpha}` in the question text (the argument of `\qutext`) is replaced by the randomly generated value of the parameter `$alpha`. In the last two lines of the `\code` fragment, Maple is used to compute the correct answer (`$r`) and a MathML expression (`$dr`) for pretty display in the feedback, using standard mathematical notation.

6

2.2 Marking and Feedback

A computer algebra system is needed to verify automatically whether the answer given by a student to a free-response mathematical question is algebraically equivalent to the correct answer as predefined by the author (in most cases a teacher). The simplest and much used Maple T.A. code for checking an answer has the following form:

`evalb( simplify( $RESPONSE -` *correct_answer* `) = 0 );`

In other words, we subtract the author's answer from the student's answer, simplify the intermediate expression, and test whether it is zero. The `maple*` code in our sample question is more complicated for two reasons:

1. We do not simply accept any answer that is algebraically equivalent with the predefined answer as fully correct.
2. We give partial credit when the symbolic part of the answer is correct, but the numerical part is wrong.

Thus, we split the student's answer in a numerical and symbolic part and we do a simple check on the number of times that the symbols a, b, and c appear in order to identify 'wrong' symbolic parts of the answer like $c^{\frac{1}{2}\gamma_1} \cdot c^{-\frac{1}{2}\gamma_2}$. If the symbolic part of the question is correct, but a wrong numerical factor is provided in the answer, the student gets half of the possible score.

It is our experience that the more precise you want to be about the acceptable form of an answer to a free-response question, the more difficult and error prone the coding of a question becomes. Working for one hour on the creation of a question is then not exceptional, even for an experienced author. Also some experience and a painstaking accuracy are needed for selecting suitable ranges of random variables that lead to questions of equal difficulty and not to impossible or trivial problems.

The feedback to the student is in our sample question just the correct answer, together with a short description of how the problem can be solved. There exist a few options in Maple T.A. to give feedback that intelligently depends on the answer given by the student. Multiple choice and multiple selection questions allow a different comment for each option. In a free-response mathematical question a given answer can be compared with some wrong answers that the teacher expects, possibly based on his or her experience in class, and feedback can provided accordingly. A simple example of tailored feedback in a Maple-graded question is the following question about integration, in which differentiation or omission of the integration constant are likely mistakes:

```
\begin{question}{Maple}
\name{integration} \type{formula}
\qutext{Compute the anti-derivative of $x^{\var{n}}$.}
\maple*{type(simplify($RESPONSE-$a2), {constant,name});}
\code{$n  = range(2,7);
  $a1 = maple("diff(x^($n),x)");
  $a2 = maple("integrate(x^($n),x)");}
```

7

```
\comment{${switch(indexof($RESPONSE, dummy, $a1, $a2),
  "If marked as incorrect, check your answer by differentiation.",
  "You did differentiation, not integration!",
  "Good, but think of integration constant next time.")}}
\end{question}
```

2.3 Facilities for Students and Teachers

Generating exercises and automatically assessing students responses in a mathematical way is important in a computer algebra based assessment system, but there is more needed for successful implementation of computer-aided assessment. As important are issues like the way students can enter mathematical expressions and see what formulas they actually type, the delivery process of tests and assessments, the administration of students results, the monitoring of students' work, and the integration with a virtual learning environment. In our case, the learning environment is Blackboard 6 and we use Maple T.A. through its Blackboard building block.

From students' point of view, a Maple T.A.-based assignment is just as like any other assignment in Blackboard: (s)he clicks on a hyperlink and the assignment appears in a separate window. One or more questions are presented on a single page, one page after another, and the student can try to answer the question(s) in any order. The student enters a response to a question, presses the button to go to the next exercise or selects the button to jump to another question. When (s)he has finished work or time is up for the assignment, the student pressed the 'grade' button. The student is warned about questions that (s)he may have forgotten to answer and is given the opportunity to repair this and submit the work again, or to force immediate marking. If the teacher has not decided differently, the student can immediately know the (provisional) mark obtained by looking it up in the Blackboard grade book and (s)he can immediately obtain via the 'Maple T.A. Tools' a detailed report, in which responses are compared with the teacher's answer and in which, at the discretion of the teacher, model solutions of the problems are presented.

The above process of doing a Maple T.A. assignment within the Blackboard environment runs smoothly and needs not much training beforehand, except possibly the way of entering mathematical expressions and the process of submitting work. Maple T.A. provides two ways to enter mathematical formulas, viz., the 'Maple syntax' mode and the 'Symbol' mode. The Maple syntax for entering mathematical expressions is only natural for students after some Maple experience through lab work; they are helped enough by the possibility to preview the formula entered in standard mathematical notation or by a reference chart to quickly look up input details. For those who lack Maple experience, like our students at the beginning of their study, the 'Symbol' mode offers a way out or a helping hand. In this mode, a student uses the built-in 2d-math editor, which is also present in Maple 10, to enter formulas in a manner that relates directly to standard mathematical notation and that makes use of templates for common formatting. Implicit multiplication ensures that `3x` and `x y` are considered the

same as `3*x` and `x*y`, respectively. Common commands are automatically formatted: for example, `^` automatically moves the cursor to superscript position, and `/` creates a division bar and moves the cursor to the denominator. Expression palettes with fill-in-the-blank templates for common mathematical objects are available and can be defined by authors. The equation editor, which is a Java applet that is automatically downloaded to the student's computer the first time it is requested, enables a student to enter responses easily and see whether they look the same way as (s)he would write them down.

All administration is done via a web interface. Using this interface, an instructor can design assignments, generate aggregate statistics and grade reports of students' results, and monitor the progress made by individual students. A teacher has at his or her disposal a variety of assignments types with different policies to choose from: anonymous practice (no need for a login, no recording of results), homework or quiz (login needed, recording of results), a proctored exam (authorisation needed), mastery session (progress control), and study session (flash card learning). Questions can be drawn from various question banks and may be tested beforehand. Note that editing of questions can only be done separately, with the help of the question bank editor or by editing a LaTeX source file of a question bank and re-converting it into EDU format. Questions in an assignment may be randomly selected from a pool of questions and the selected questions may be scrambled at the instructor's will. The teacher can also set several policies for an assignment such as visibility on the web, duration and schedule of the assignment, passing score, availability of hints and links to additional notes, the level and timing of feedback, and additional requirements for doing the assignment.

The Blackboard grade book and the Maple T.A. grade book are synchronised. When an instructor includes for the first time a Maple T.A. assignment in a Blackboard course, then a Maple T.A. class is automatically created for this course on the Maple T.A. server and the user accounts are forwarded from Blackboard to Maple T.A. In this way, all users' administration is taken care of. The instructor does not have to worry about this or bother an administrator to make student accounts available. In the Blackboard grade book, an instructor can view and modify students' grades, clear attempts, and view statistics for a particular grade book item. Via the Maple T.A. grade book a teacher can get detailed grade reports, obtain item statistics, have a look at work of individual students and comment on it, change marks, and authorise a student to start a fresh (proctored) test.

3 Our Experiences with Diagnostic Testing of Freshmen

In this section we discuss (i) the mathematical performance of the freshmen in the pretest and posttest taken in the first weeks of the study year, (ii) the students' opinions about the computer algebra based diagnostic testing, and (iii) our experiences with Maple T.A. for Blackboard.

9

3.1 Students' Results

The overall scores of the freshmen at in the pretest and posttest are shown in the two histograms of Fig. 1.

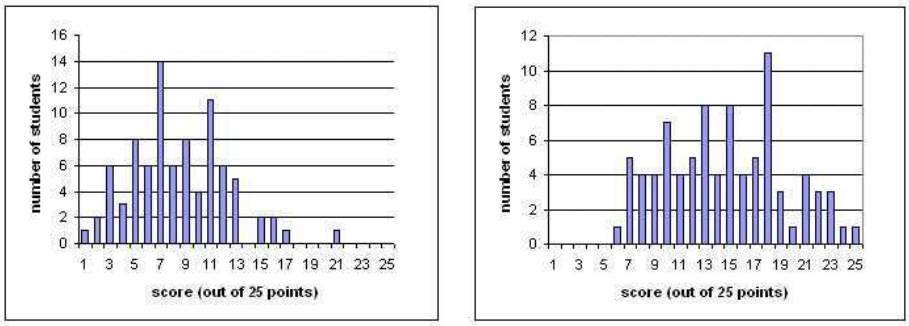


Fig. 1. Histogram of students' result in pretest (left) and posttest (right)

The progress of individual students between pretest and posttest is represented in the scatter diagram of Fig. 2: each dot represents the score of a student at the pretest and posttest.

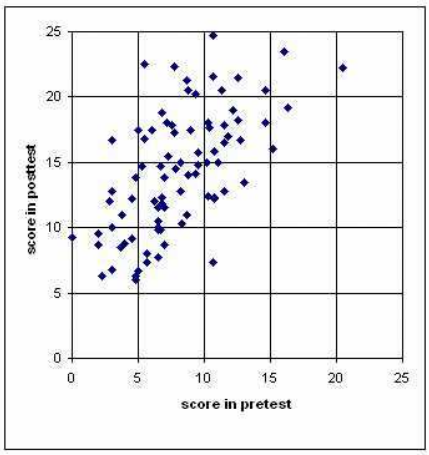


Fig. 2. Scatter diagram of students' scores in pretest and posttest

Needless to say that these low scores of beginning students worry university educators. Even though some students make remarkable progress in four weeks of remedial training, still only one-third of the student population entering the exact sciences programme has reached a desirable mathematical level.

10

3.2 Students' Opinions about the Diagnostic Testing

Just after the posttest, the students were asked to give their opinions by means of a survey. The response was sixty-five percent.

The students found that the goal of the diagnostic tests were sufficiently clear (90%). The chosen approach was appreciated by 86%. They confirmed that the tests helped to identify shortcomings in their mathematical abilities (69%), and also that the tests contributed to making progress with mathematics (68%). About 63% of the students found the tests stimulating to set to work. A minority (42%) said that they obtained more insight in the difference between school and university mathematics.

About the instrumentation, 61% of the students found that they could work well with the Maple T.A. software. This is remarkable, since there was no training beforehand. Students were permitted to use the graphing calculator and the standard formula chart. A small majority of the students (63%) thought that this influenced their score positively. It also means that a large number of students do not feel dependent on the graphing calculator and the formula chart, while teachers noticed during practice sessions that the students used them many times, even in case when it seems inappropriate.

At the end of the survey there was an open field for suggestions and remarks. About half of the students used this field. Nine students made it clear that they preferred a test on paper instead of the computer; an important reason they mentioned is that they could only fill in the end result and no intermediate steps. Maybe they felt disadvantaged by not been given clearly the opportunity to earn partial credit (although we did this in fact at a manual check of the grading). Six students made a positive comment about the usefulness of the approach. Two students made a remark about the planning of the pretest as it was planned just before the very first week of the start of first courses. This happened to be the period of the university introduction with all kind of activities and festivities. Arguably, a pretest was not appreciated at that moment.

3.3 Experiences with Maple T.A. for Blackboard

Luckily we had decided in advance that in the pretest students would use printouts of the diagnostic test and that a student after solving a problem with pencil-and-paper would fill out the answer in the Maple T.A. test form. We planned this because we were not sure whether everything would run smoothly from technical and organisational points of view. Indeed we suffered from problems with students not having a computer account or not having a Blackboard account. Although we had anticipated this, creating accounts at the beginning of the diagnostic turned out to be too hectic a job. More important was that one-third of the students had unexpected login problems with the Maple T.A. assignment. These students, having access to the Blackboard course, should not have to login for the diagnostic test anyway! But unforeseen name conflicts between user accounts with special characters and the Maple T.A. administration of users caused these problems. Because one-third of the students were allowed

to hand in the pencil-and-paper work alone, one-third of the students decided to do the same. Consequently, only one-third of the students filled out the digital test form. But this percentage gave enough insight in the process of computer algebra based assessment and the points to take care of in the next diagnostic testing. By the way, knowing the technical problem was enough to fix it for the posttest by simply changing the settings of the Maple T.A. server.

Maple T.A. uses one file per student to keep the complete history of work. The instructor cannot delete irrelevant test data and consequently most of the aggregate statistics of the pretest and posttest generated by Maple T.A. was useless.

Afterwards the instructor had great difficulty in editing the diagnostic tests because many student were registered as still having active assignments. This seemed to be caused by the following submission process: many students submitted the diagnostic tests for marking and received warning about questions not answered or input problems. But many of them did not realise that they had to press the grade button again to force submission for marking. They thought that they has already submitted their work, saved and quit the test, and logged off the computer. The only way to get rid of such active users was to identify them manually and then force marking of their tests. Of course it is a wise design choice to ensure that an instructor cannot edit an assignment because some student may still be busy with the assignment, but there are circumstances in which an instructor knows better and wants to force removal or marking of active assignments without much effort.

We found serious bugs in the software, which luckily did not affect much the students' results. We noticed while we were checking the computer marking of given answers that adding extra spaces around the multiplication symbol caused that correct answers were marked as incorrect in Symbol entry mode. This kind of errors in marking could easily be identified by us and marks could easily be changed. Another problem that luckily did not affect students results was the following: in Symbol entry mode, the equation editor parses the given formula into an expression with valid Maple-syntax. Unfortunately, it also applies automatic simplification before checking the correctness of the answer. As a consequence of this, a student could answer for example to the first question of the pretest simply `1/(1/2+1/3+1/4)` and still get full marks because Maple already does the computation. Of course, no student came to this idea, but it is a serious weakness of the system. You do not want a marking procedure that depends on the selected input mode.

Despite the ten minutes introduction into the Maple T.A. students made foreseen errors in entering mathematical formulas. We noticed the use of uppercase characters instead of lowercase symbols and sometimes an algebraic operator was lacking, like in `xy` being entered instead of `x y` or `x*y`. Luckily, an instructor quickly notices these things when he or she scans the computer marking, but it remains annoying.

The above remarks should not give the impression of failure of Maple T.A. under practical circumstances with unexperienced students. It more reflects that

12

Maple T.A. is a young product is not yet over its teething troubles, but still needs more active development. Actually it worked very well in the posttest, when we knew that we should pay more attention to the submission process. In fact, the 'Symbol entry mode' allows usage of the assessment system with little introduction to the users.

We did the diagnostic tests with fifty simultaneous users of the Maple T.A. without any significant load on the server (dual processor machine at 3 GHz speed with Linux OS). We expect that we can use the testing and assessment system with large groups of students in the computer labs at one time, without running into serious problems. The connection between Blackboard and Maple T.A. system works well and makes life of an instructor easier.

4 Conclusion

The main question to address in the conclusion is whether computer algebra based diagnostic testing of mathematical competencies of freshmen meets its goals in a staff-extensive way. As can be seen from the substantial progress of the students, the trajectory does lead to an overall considerable amelioration of the mathematical abilities. The results of the pretest and posttest also indicate clearly that much attention must be paid to bringing freshmen as soon as possible to the required starting level of mathematical knowledge and skills in order to avoid study delay or early drop-out.

On the whole, the students appreciated that they were confronted with the mathematics abilities as desired by the universities, and were informed about their own level. Although the first assessment was even before the first course day, the students did not feel uncomfortable with the fact that they were assessed. As can be seen from the questionnaire, they understood well the purpose of the testing. We believe that for this the repeated information to students, tutor students and staff is very important.

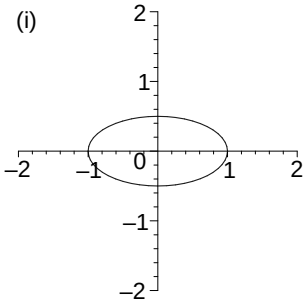
The Maple T.A. system is able to generate and assess questions from the templates in the database. So the whole trajectory can be based on a single test template: it is used for the entry assessment, the formative exercises and the diagnosis tests. This is a very strong advantage of the approach with a tremendous potential for saving time of staff. However, this version of Maple T.A. showed some peculiarities in the submission process and in the automatic assessment by Maple of formula based items. These forced us to check manually the student results, so in this pilot the time efficiency requirement was not met. Nevertheless, we believe that once the peculiarities of the system are removed, also the condition of staff-extensiveness is satisfied.

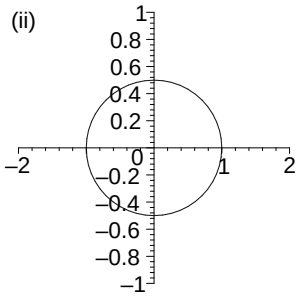
Appendix: First Diagnostic Test

1. Simplify as much as possible: $\dfrac{1}{\frac{1}{2}+\frac{1}{3}+\frac{1}{4}}$.

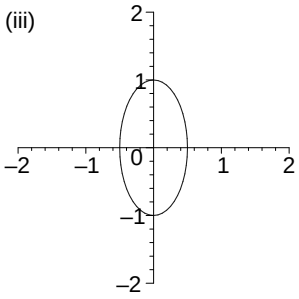
13

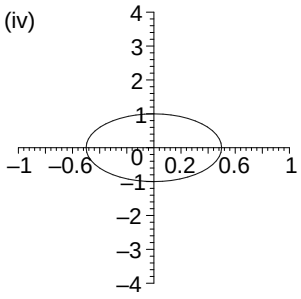
2. Simplify as much as possible: $\left(\frac{6}{5}\right)^{-3} \times \left(\frac{3}{10}\right)^{2}$.
3. For kinetic energy E of a particle with mass m and velocity v holds: $E = \frac{1}{2}mv^2$. For the momentum p holds: $p = mv$. Write the formula of kinetic energy as an expression in m and p.
4. Rewrite the formula of lenses $\frac{1}{f} = \frac{1}{b} + \frac{1}{v}$ in the simplest form: $f = \ldots$
5. Simplify as much as possible: $\frac{a^2(8b)^{\frac{1}{4}}c^{\frac{1}{2}}}{\sqrt{abc^3\sqrt{2}}}$, assuming that a, b and c are positive numbers.
6. Expand brackets and simplify: $(2a + 3b)(3a - 2b)$.
7. Expand brackets and simplify: $(2x - y + z)^2 - z^2$.
8. Simplify as much as possible: $\frac{9r^2 - 4s^2}{3r + 2s}$.
9. Determine the exact value of the sum: $\sum_{k=-10}^{10} k$
10. Given is a positive number a. Compute the exact value of the limit: $\lim_{n\to\infty} \frac{n^2 - 2^n}{an^2 + e^n}$.
11. Given is the system of two equations in two unknowns x and y and two constants a and b: $\begin{cases} 2x + 3y = a \\ 4x + 6y = b \end{cases}$
    Mark for each of the conclusions below whether it is true or false:
    (a) For some values of a and b the system has no solutions.
    (b) For some values of a and b the system has exactly one solution.
    (c) For some values of a and b the system has an infinite number of solutions.
12. Given is the point P with coordinates $(1, 2)$. We translate the coordinate system over the vector $\begin{pmatrix} -1 \\ 3 \end{pmatrix}$ and leave the point P at its current position. What are the coordinates of P in the new coordinate system?
13. Which of the following graphs belong(s) to the curve described by the equation $x^2 + 4y^2 = 1$?



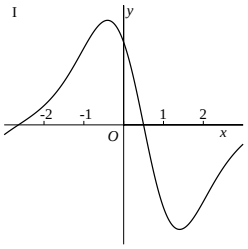


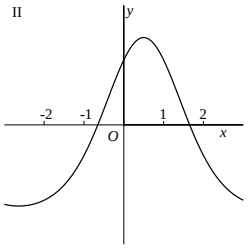
14

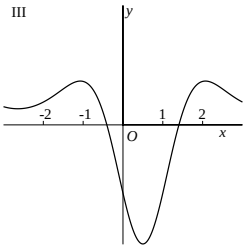




14. For $x > -2$ is the function $f(x) = \sqrt{1 + \frac{x}{2}}$. The formula of the inverse function is then: $f^{-1}(x) = \ldots$
15. We assume that the standard limit $\lim\limits_{x \to 0} \dfrac{\sin(x)}{x} = 1$ is known.
    Given is a number $a \neq 0$. Determine the exact value of the limit: $\lim\limits_{x \to 0} \dfrac{\sin(ax)}{\tan 2x}$.
16. Determine the quadratic function $f(x)$ with stationary point $(1, -4)$ and a zero in $x = 2$.
17. Determine the exact values of all solutions of the equation: $x^3 + 14x^2 = 72x$.
18. Determine the exact values of all real solutions of the equation: $e^{2x} - 2e^x - 3 = 0$.
19. Simplify as much as possible: $\dfrac{4^x - 1}{2^x - 1}$.
20. Differentiate with respect to t and simplify your answer as much as possible: $\ln(1 - t^2) + t^2$.
21. Differentiate with respect to x and simplify your answer as much as possible: $\dfrac{x^2 - 4}{x - 2}$.
22. Given are a function $f(x)$ with $f'(1) = 3.8$ and the function $g(x) = f(2x - 1)$. What is $g'(1)$.
    Select one of the following responses.
    ◯ 1.9 ◯ 2.8 ◯ 3.8 ◯ 4.8 ◯ 7.6
    ◯ For this you must know the formula of $f(x)$.
23. Below are drawn in random order the graphs of the function $f(x)$, its derivative $f'(x)$ and the second derivative $f''(x)$. Identify them.







24. Determine the exact value of $\displaystyle\int_1^2 (x^3 + \sqrt{x})\, dx$.

15

25. Given is a function $f(t)$ with $\int_1^3 f(t)\,dt = 2.4$. What is $\int_3^5 f(t-2)\,dt$?
Select one of the following responses.
◯ 0.4 ◯ 2.4 ◯ 4.4 ◯ For this you must know the formula of $f(x)$.

Acknowledgements

The authors would like to thank their colleagues Jan van de Craats, Peter de Paepe, and Lidy Wesker for their help in the development of the pretest and posttest, either by making fruitful comments on the exercises, or by suggesting or providing new exercises. We much appreciated the help of our colleague Wolter Kaper in setting up the Maple T.A. for Blackboard in short time and in running the tests. We also thank Louise Krmpotic and Robin Coe of Maplesoft Inc. for providing us, before official release, with the version 2.5 of Maple T.A. with Blackboard support and for helping us out with technical problems when needed.

References

1. London Mathematical Society: Tackling the Mathematics Problem. Advisory report, 1995. `www.lms.ac.uk/policy`
2. SEFI Mathematics Working Group; Mustoe, L., and Lawson, D. (eds): Mathematics for the European Engineer — a Curriculum for the twenty-first Century. Report, 2002. `http://learn.lboro.ac.uk/mwg/core/latest/sefimarch2002.pdf`
3. Kent, P., and Noss, T.: Mathematics in the University Education of Engineers. A Report to The Ove Arup Foundation, 2003. `www.theovearupfoundation.org/arupfoundation/pages/download25.pdf`
4. Engineering Council: Measuring the Mathematics Problem. Report, 2000. `www.engc.org.uk/documents/Measuring_the_Maths_Problems.pdf`
5. van de Craats, J., and Bosch, J.: Basisboek Wiskunde (in Dutch, Basic Mathematics). Pearson Education Benelux, 2005. Most parts of the book are electronically available at `www.science.uva.nl/~craats`
6. Maple T.A.: Automated Online Testing & Assessment. Developer's website `www.maplesoft.com/products/mapleta`
7. Maple: `www.maplesoft.com/products/maple`
8. Web-spijkeren: SURF Project, information (in Dutch) electronically available at `www.web-spijkeren.nl`
9. MathMatch: Project of the Digital University, information (in Dutch) electronically available at `www.du.nl/mathmatch`
10. AIM: Assessment in Mathematics. `http://aiminfo.net`
11. Sangwin, C.J., and Grove, M.: STACK: addressing the needs of the"neglected learners". WebALT 2006 Proceedings.
12. WebALT, Web Advanced Technologies: `www.webalt.net`
13. Brownstone's EDU Campus: `www.brownstone.net/products/ecampus.asp`
14. Sangwin, C.J: Assessing mathematics automatically using computer algebra and internet. *Teaching Mathematics and it Applications* **23** (2004) 813–829.

Feedback in an interactive equation solver

Harrie Passier
Open University
Heerlen, the Netherlands

and

Johan Jeuring
Open University
Heerlen, the Netherlands
and
Department of Computer Science
Utrecht University, the Netherlands

Abstract. E-learning tools for mathematical problem solving such as solving linear equations should be interactive. As with pen and paper, a student constructs a solution stepwise. E-learning tools provide the capability to give feedback to a student at each step. Feedback is essential for effective learning and hence crucial for interactive e-learning tools. This paper describes a framework for providing feedback in interactive e-learning tools. The framework is particularly useful for domains with hierarchically structured terms, a set of rewrite rules to rewrite the terms from the domain into other terms, and a well-described goal. The framework is used to give feedback about syntactical errors, about several kinds of semantic errors, and about progression towards a solution. The framework explicitly uses the structure in the data to produce feedback. We discuss an e-learning tool for solving linear equations in which the framework for feedback is used. The techniques for providing feedback are taken from compiler technology and rewriting theory.

1 Introduction

Mathematics is constructive in nature: mathematics students learn to *construct* solutions to mathematical problems. Solving mathematical problems is often done with pen and paper, but e-learning tools offer great possibilities. Interactive e-learning tools that support learning mathematics should provide the capability to give feedback to a student at each step. To illustrate our approach, we will use an e-learning tool for solving a system of linear equations. We call this tool the Equation Solver. Figure 1 shows a screenshot of our tool.

The Equation Solver consists of three text fields. The top text field is the working area, in which a student can edit a system of equations stepwise to a solution. The current system of equations is

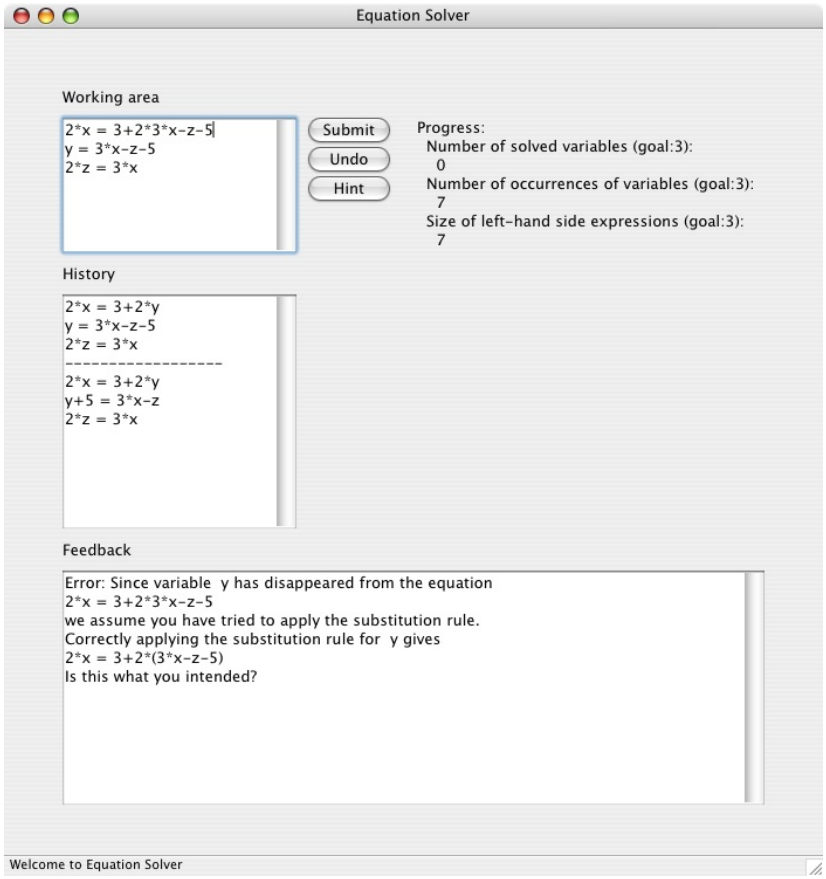


Fig. 1. The Equation Solver

```
2*x = 3+2*3*x-z-5
y = 3*x-z-5
2*z = 3*x
```

The second text field displays the history of equations. Apparently the previous system of equations was

```
2*x = 3+2*y
y = 3*x-z-5
2*z = 3*x
```

and the student replaced `y` by `3*x-z-5`, forgetting to parenthesize the result. The third text field displays the feedback. In the figure it explains why the last step is incorrect.

The Equation Solver presents a system of equations to a student, for example

```
2*x = 3+2*y
y+5 = 3*x-z
2*z = 3*x
```

and the student has to rewrite these equations into a form with a variable to the left of the equals symbol, and a constant to the right of the equals symbol, for example `x = 7;y = 11/2;z = 21/2`. The student presses the Submit button to submit an edited system of equations, and the Undo button to undo the last step (or any amount of steps). If a student wants help, he or she presses the Hint button to get a suggestion about how to proceed. Finally, the Equation Solver gives information about progress towards a solution by showing how many variables have been solved, and several other kinds of information.

The Equation Solver gives feedback about two kinds of mistakes:

- syntactical mistakes, for example when a student writes `y+5 = 3*x-` instead of `y+5 = 3*x-z`,
- semantical mistakes, usually mistakes in applying a step towards a solution, for example when a student rewrites `y = 3+1` by `y = 5`,

and it gives feedback about (lack of) progression towards a solution.

The Equation Solver consists of a *solver*, which performs symbolic calculations, an *analyser*, which analyses the submitted equations of a student based on a set of rewrite rules for the domain of a system of equations, and several *indicators*, which indicate the progression of a (series of) rewrite step(s).

Note that we try to mimic the pen-and-paper situation as closely as possible, by letting students enter and rewrite equations in a text field. An other approach is to offer the possible rewrite steps to the student, and let the student select a rewrite step, which is then applied to the system of equations by the Equation Solver, as propagated by Beeson [1] in MathXpert. In such a situation, it is impossible to make a syntactical mistake, or to rewrite an equation incorrectly. The former approach has the advantage that a student also learns to enter correct equations, and to choose and apply rewrite steps correctly. Furthermore, it is

closer to the pen-and-paper situation. The latter approach has the advantage that a student can concentrate solely on solving a system of equations. The only feedback that needs to be provided in the latter approach is feedback about progression. Although we do not support the latter approach, it is orthogonal to our approach, and easily added to our tool.

This paper discusses a framework for providing feedback, in which feedback about syntactical mistakes, semantical mistakes, and (lack of) progression in the solving process is produced. The framework assumes a structured domain (like linear equations), for which a set of rewrite rules (or transformations) is defined (like `x+0 = x` for all `x`), a goal is specified (like rewrite all equations to a form where there is a single variable to the left, and a constant to the right of the equality symbol), and one or more measures can be defined with which we can (possibly partly) determine the distance to the goal.

The main results of our work are:

- We show how results from theoretical Computer Science, in particular from the term-rewriting and compiler technology (and in particular parsing) fields, can be used to develop tools that provide semantically rich feedback to students.
- We show how using structural information in data for feedback improves the feedback a tool can give.

We think our framework is useful for several purposes. Developing a tool in our framework forces the developer (a teacher) to be explicit about *all* aspects of a particular domain, and it helps developers of e-learning tools to set up a well-structured feedback component that gives better feedback than existing tools.

This paper is organised as follows. Section 2 discusses the framework for feedback. Section 3 briefly discusses our approach to syntactic feedback. Section 4 discusses how we provide feedback on individual rewriting steps towards a solution. Section 5 discusses the indicators we have defined and how we help students that seem to be stuck. In Section 6 we draw our conclusions, describe related work, and list planned future work.

2 The feedback framework

Our framework for providing feedback assumes we have the following components:

1. A domain with a semantics.
2. A set of rewrite rules for the domain.
3. A goal that can be reached by applying the rewrite rules in a certain order.
4. A set of progression indicators to determine the distance between the goal and the current situation.

Our framework provides feedback about syntactic errors, semantic errors (incorrectly applied rewrite rules), and about progression, using the progression indicators.

To illustrate our framework, we will use the Equation Solver introduced in the Introduction. Solving a system of n linear equations with n variables $x_1, \ldots, x_n$ amounts to finding constants $c_1, \ldots, c_n$ such that $x_1 = c_1, \ldots, x_n = c_n$ is a solution to the system of equations.

We describe the components of our framework for the Equation Solver.

Domain and semantics of the Equation Solver. The domain of the Equation Solver consists of a system of linear equations. We use Haskell [9] types to describe this domain. The top-level type is a list of equations:

```
type Equations      = [Equation]
```

Each equation consists of a left and a right hand expression separated by a '=' (in Haskell denoted by the infix constructor `:=:`) symbol.

```
data Equation = Expr :=: Expr
```

An expression is either zero, a constant, a variable, or two expressions separated by an operator '+', '-', '*', or '/'.

```
data Expr     =  Zero
              |  Con Rational
              |  Var String
              |  Expr :+: Expr
              |  Expr :-: Expr
              |  Expr :*: Expr
              |  Expr :/: Expr
```

The semantics describes how the domain should be interpreted. For the Equation Solver, the semantics is the solution to the system of linear equations.

Rewrite rules for the Equation Solver. A domain has a set of rewrite rules with which terms in the domain can be rewritten.

A rewrite rule rewrites a term of a particular domain to another term of that domain. For example, we have the following rewrite rule for expressions: $(a + b)c \rightarrow ac + bc$, which says that we can rewrite the expression $(a + b)c$ to the expression $ac + bc$ (distribute multiplication over addition) in any context in which this expression appears. For a general introduction to rewrite systems, see Dershowitz et al. [4].

Using rewrite rules, we rewrite terms in the domain to some desired form. For the Equation Solver, the goal is to rewrite the given system of equations to a solution. We now informally present the rewrite rules for the domain of the Equation Solver.

We follow the data representation of the domain and distinguish between rules on the level of a system of linear equations, an equation, and an expression. In these rules a, b, and c are rational numbers, x, y, and z are variables, and e is an expression.

For a system of linear equations we have a single rewrite rule: substitution. If we have an equation $x = e_1$, we may replace occurrences of x in another equation by e_1. Informally:

$$[x = e_1, \ldots x \ldots = e_2, \ldots] \rightarrow [x = e_1, \ldots e_1 \ldots = e_2, \ldots]$$

For an equation we have four rewrite rules:

$$e_1 = e_2 \rightarrow e_1 \oplus e = e_2 \oplus e$$

where $\oplus$ may be any of $+$, $-$, $*$, or $/$.

For an expression we have a large number of rewrite rules. First, constants may be added, multiplied, etc:

$$a \oplus b \rightarrow c,$$

where c is the rational number sum of a and b if $\oplus$ is $+$, and similarly for $-$, $*$, and $/$. Coefficients of the same variable are summed using the inverse rule of distributing multiplication over addition.

$$a * x + b * x \rightarrow (a + b) * x$$

Furthermore, multiplication (division) distributes over addition (subtraction):

$$(e_1 \oplus e_2) \otimes e_3 \rightarrow e_1 \otimes e_3 \oplus e_2 \otimes e_3,$$

where $\oplus$ may be $+$ or $-$, and $\otimes$ may be $*$ or $/$.

As argued by Beeson [1], many mathematical operations cannot be expressed by rewrite rules, because they take an arbitrary number of arguments and because other arguments can come in between. Moreover, associativity and commutativity cause problems in rewrite rules. Hence, applying rewrite rules or recognising applications of rewrite rules in user-supplied equations is not a trivial application of pattern matching, but requires more sophisticated programs.

A *normal form* of a term in the domain of a term-rewriting system is a term which cannot be rewritten anymore. The solution of a system of linear equations is a kind of normal form of a system of linear equations, but since we can always add terms to a term, our system does not have normal forms. A term-rewriting system *terminates* if for every term t, we can only rewrite t a finite number of steps. Since we can distribute multiplication over addition and vice versa, our term-rewriting system is clearly not terminating. A term t' is *reachable* from a term t, if there exists a sequence of term-rewriting steps with which we can rewrite t into t'. Clearly, given a solvable system of linear equations, the solution of this system is reachable. In a situation in which terms have normal forms, and the rewriting system is terminating it is much easier to give useful feedback, but for most domains about which we want to give feedback these properties do not hold.

The goal of the Equation Solver. The goal of the Equation Solver is to find constants $c_1, \ldots, c_n$ such that $x_1 = c_1, \ldots x_n = c_n$ is a solution to the system of equations. We assume that all systems of equations set as exercises by the Equation Solver are solvable, a property that is easily verified. The goal is reachable by applying the rewrite rules to the system of equations in a certain order.

Progress indicators for the Equation Solver. To inform a student about the progress in solving a problem, we have defined indicators. An indicator is a measure which (partly) describes the distance from the current system of equations to the solution (the goal). There are several ways to indicate the distance between the currents system of equations and the solution. A possibility is to determine the minimum number of rewriting steps needed to rewrite the current system of equations to the solution. In this paper we investigate indicators that follow the structure of the data. Thus we can provide more specific feedback than just about the distance to the final solution. We have indicators that indicate progress on the level of a system of equations, on the level of a single equation, and on the level of an expression.

In the next sections, we describe how we provide feedback about syntactical errors, semantical errors, and about progression using this frame work.

3 Syntax analysis

A student enters an expression in a text field in the Equation Solver. We have to parse this expression in order to analyse it. We use error recovery parser combinators [12] to collect as many errors as possible (not just the first), and to suggest possible solutions to the errors we encounter. For example, when a student enters `2*x = 3+2*y;y+5 = 3*x-;2*z = 3*x`, the tool reports an error, and says it expects a lower case identifier or an integer in the equation `y+5 = 3*x-`. Furthermore, it proceeds with parsing `2*z = 3*x`, assuming the expression `y+5 = 3*x-<identifier>` has been entered. The parser combinators are very similar to the context-free grammar for the domain of equations. We have tuned the parser such that common errors, such as writing `2y` for `2*y`, are automatically repaired (and reported).

After parsing we perform several syntactic checks, such that the set of variables that occurs in the system of equations hasn't changed, and that all equations are still linear equations, and not for example quadratic equations, which would happen if the student would multiply both sides of the equation `2*z = 3*x` by `x`. If such an error occurs, it is reported.

4 Rewriting terms

When a student submits a system of equations to the Equation Solver, the analyser checks if something has changed. If something has changed, the solver checks that the submitted system of equations has the same solution as the original system, and the analyser tries to infer the rewrite rule applied by the

student. If the solution of the system has not changed, it is not necessary to determine the rewrite rule that has been applied, but it might still be useful for the student to see the name of the applied rule. If the solution *has* changed, the student has made an error, and it is important to try to report the likely cause of the error.

An important assumption (restriction) we apply here is that we assume a student applies only one rewrite rule per submitted system of equations. In practice, this will not always be the case, but the added complexity of recognising multiple rewrite rules is left to future work.

In the rest of this section, we discuss the feedback produced by the Equation Solver by means of examples on each of the three levels of our domain. To determine which rule a student intended to apply, we follow a hierarchical approach.

Determining a rewrite on the system of equations level. The analyser starts with trying to find out if the student intended to apply a rule on the level of the system of equations: the substitution rule. The analyser can determine whether or not the substitution rule has been applied by collecting the variables that appear in the different equations. If one variable has disappeared from the set of variables that appear in an equation a substitution step has been applied. Here we assume that an expression such as $x - x$ is internally represented as 0, so that replacing $x - x$ by 0 does not lead to the false conclusion that substitution has been applied. The internal representation is some normal form of the expressions and equations, where occurrences of the same variable are combined. The normalised form of an expression is an expression of the form $a_1 * x_1 + ... + a_n x_n + c$, where each variable occurs once, and all constants have been added in a single constant c. The Equation Solver determines which variable has disappeared, and checks that applying the substitution using that variable leads to the submitted expression. If this is not the case, the Equation Solver reports an error, and shows the correct equation that results from the substitution. For example, if the system of equations `2*x+2*y = 6;y = 4-2*x` is rewritten to `2*x+2*(4+2*x) = 6; y = 4-2*x`, the analyser produces the following error message:

```
Error: Since variable y has disappeared from the equation

                  2*x+2*(4+2*x) = 6

we assume you have tried to apply the substitution rule.
Correctly applying the substitution rule for y results in

                  2*x+2*(4-2*x) = 6

Is this what you meant?
```

There are several things to note about this message: it is only about the equation that contains an error, it tells why it thinks a certain rewrite rule has been applied, and it shows how the correct application of that rule looks.

Determining a rewrite on the equation level. If the analyser has detected a change in the system of equations and in no equation the set of variables that occur has changed, the analyser tries to find out if there exists an equation that has been rewritten. An equation has been rewritten if both the left-hand side and the right-hand side expression of an equation have changed. On the level of an equation, four rewrite rules may be applied: $e_1 = e_2 \rightarrow e_1 \oplus e = e_2 \oplus e$, where $\oplus$ may be any of $+$, $-$, $*$, or $/$. The analyser determines whether or not these rules have been applied by comparing the new equation with the old equation. For example, if the previous system is `2*x+2*y = 5; x-y = 2` and the submitted system is `2*x+2*y = 5; x-y+y = 2+y`, the analyser concludes that the rule: $e_1 = e_2 \rightarrow e_1 + e = e_2 + e$ has been applied on the second equation of the system. In general, the analyser can infer an application of the addition (and subtraction) rule on the level of an equation by calculating the value of the expression `(l-l')-(r-r')`, where `l = r` is the equation in the previous system, and `l' = r'` is the submitted equation. If this value equals 0, then it is likely that the student has performed an addition (or subtraction) step with value `l-l'` on both sides of the equation. If the value equals a constant unequal 0 or a variable (possibly multiplied by a constant), then it is likely that a student has performed an addition step, but has made an error in doing so. This error is reported. Finally, if the value is not a constant or a variable, it is likely that the student has performed a multiplication (or division). To determine if a multiplication step has been performed, the analyser calculates the value of `(l/l') - (r/r')`. If this value equals 0, then it is likely that the student has performed a multiplication (or division) step with value `l/l'` on both sides of the equation. If the value equals a constant, then it is likely that a student has performed a multiplication step, but has made an error in doing so. This error is reported. Finally, if the value is not a constant, something serious is wrong.

Determining a rewrite on the expression level. If no rewrite on the level of a system of equations or on the level of an equation has taken place, the analyser tries to determine if a rewrite on the level of an expression has taken place. It is easy to determine which expression in the system of equations has been changed.

For example, suppose the previous system is `2*(2+y)+2*y = 5; x = 2+y` and the submitted system `2*2+2*y+2*y = 5; x = 2+y`. The analyser infers that the left-hand side expression of the first equation has changed. The analyser checks that the normalised form of the new expression and the previous expression are the same. Furthermore, the analyser tries to infer which expression rewrite rule has been applied. It does this by determining the expression difference between the old expression and the new expression. The expression difference of two expressions consists of the subexpressions that have disappeared from the old expression in the new expression, and the subexpressions that have appeared in the new expression. In the above example, the expression difference is `2*(2+y)` (disappeared) and `2*2+2*y` (appeared). These expressions match the rewrite rule for distributing multiplication over addition. If the normalised form of the new expression and the old expression are different, an error is reported,

and the analyser shows all possible correct rewrites of the subexpression that has disappeared from the expression.

The hierarchical approach to determining which rewrite rule has been applied allows us to pinpoint precisely, in many cases, which mistake (likely) has been made.

5 Progression and hints

An indicator gives a distance from the current system of equations to the solution (the goal). It is used to inform a student about progression towards a solution. Before calculating the value of the different indicators, the Equation Solver detects whether or not a student has completed the problem. In that case, the system of equations has the form of $x_1 = c_1, \ldots, x_n = c_n$. This is easily detected.

We have defined a number of indicators in the Equation Solver.

The first indicator calculates the number of variables for which a student has found a solution. If this number increases the student makes progression.

If the number of variables for which the student has found a solution has not increased, the Equation Solver uses the second indicator, which calculates the number of occurrences of variables in a system of equations. Progression is made if this number decreases. For example, in the system `4 + 2*y + 2*x = 5; x = 2 + y` there are four occurrences of variables. Substituting `2+y` for `x` in the first equation reduces the value of this indicator by one. Sometimes the value of this indicator increases due to a substitution, so we do not enforce that the value of this indicator decreases or stays the same at each step.

If the previous indicators do not notice a change, we check if the expression size of the left-hand side expression of an equation has decreased. Since in our solution we want the left-hand side expression of an equation to be a single variable, a reduction in the size of the left-hand side expression (without removing all variables, since in the end a single variable should remain) indicates progression. For example, rewriting the expression `y+3-1` to `y+2` reduces the size of the expression from 5 to 3 (where operators, constants, and variables all count for 1).

The indicators are independent of the rewrite rules in the Equation Solver. So if a student performs a transformation on the system of equations that doesn't change the semantics of the system of equations, but for which the analyser cannot find a corresponding rewrite rule, the indicators can still inform the student about his or her progress.

If a student is stuck, he or she can press the hint button. The Equation Solver will then give a next step, or a hint to help the student to produce a next step. The next step depends on the solving strategy used. We have only implemented the Gaussian solving method in the Equation Solver. The Equation Solver produces a next step or a hint based on the previous system of equations submitted by the student, the set of rewrite rules and the solving strategy. For

example, if the previous system submitted by the student is `4+2*y+2*x = 5; x = 2+y`, the system will suggest:

```
Try to substitute 2+y for x in the first equation.
```

We intend to offer various levels of help. In the above situation we can think of the following, increasingly detailed, messages:

```
Try to apply the substitution rule.

Try to substitute 2+y for x in the first equation.

Substitute 2+y for x in the first equation, resulting in
4 + 2*y + 2*(2+y) = 5
x = 2 + y
```

6 Discussion

Conclusions. We have introduced a framework for providing feedback in an e-learning tool for a structured domain. Using the structure in the domain, we can provide more detailed feedback. The framework consists of a domain with a certain semantics, a set of rewrite rules for the domain, a goal that can be reached by applying the rewrite rules in a certain order, and finally a set of indicators to determine the distance between the desired solution and the current situation. We have used our framework in a prototype e-learning tool for solving a system of linear equations. All of our ideas have been implemented, but we have yet to add the bells and whistles to turn the Equation Solver into a mature tool.

Feedback is crucial in education and is used in many learning paradigms. It is an essential element needed for effective learning. Nevertheless, electronic learning environment courses frequently lack effective feedback [7]. Almost all feedback is related to question-answer situations, is hard coded, and does not use a structural approach. In general, this situation also holds for the field of learning mathematics. If feedback is a crucial element in education and electronic environments increasingly support mathematics education, this gap needs to be filled.

We think our framework will be useful for students, teachers, and e-learning tool developers that build interactive tools in which students have to construct solutions stepwise. It forces a teacher to be explicit about all terms and the semantics of a particular domain, the goal that has to be reached, how progression of the solving process can be measured, and which rewrite rules may be used. It helps the (software) developer to build the feedback component in a structural way. The steps a student can take in such a tool are not limited by a predefined set of rewrite rules provided by the tool, but can be any combination of correct steps, or erroneous steps. The tool tries to recover the rewrite steps taken by the student in order to provide detailed feedback about possible errors. If the tool cannot recover the rewrite rules, the indicators can still help a student in determining whether or not he or she is on the right track. This is important because

it is hard if not impossible to always determine which sequence of rewrite rules has been applied by a student.

We have said little about the form and content of the feedback messages. We have shown two messages: one error and one advice message. The error message not only tells that an error has been made, but also which equation contains the error and additional information based on the rewrite analysis. This additional information is important because information about the nature of the error and the way it can be corrected is much more effective for learning than simply being informed that an error has been made without any further guidance [6]. This is especially important when students are working in a 'pen-and-paper' like environment, instead of an environment where rewrite rules can be selected from a menu.

Our Equation Solver satisfies most of Beesons [1] eight criteria that must be met if we are to provide successful computer support for education in algebra, trig, and calculus. The first two are *cognitive fidelity*, which means that the software solves the problem in the same way as the student should solve it, and *the glass box principle*, which means that a student can see how the computer solves the problem. To produce feedback and advice about which step to take next, the Equation Solver uses a well-known set of rewrite rules and a solving strategy. The feedback and advice are based on the application of a single rule. Applying multiple rules in a single rewrite step is not supported yet, in the sense that it is allowed, but no feedback is given if an error is made, other than the fact that an error has been made. It follows that our Equation Solver does not completely meet the *customised to the level of the user* criterion. Indicators still help advanced users that apply multiple rules in one step though. The fourth criterion is *the correctness principle*, which mean that a student cannot perform incorrect operations. The Equation Solver calculates after each submission the solution of the system of linear equations. If the solution is changed, the analyser will inform the student and, if possible, point out the erroneously applied rule. However, the student can still enter an incorrect equation. We think this has an added advantage: the student becomes aware of the syntax of systems of equations, and learns how to apply rewrite rules. The fifth criterion, *user in control*, means that the student decides what steps should be taken and the computer can help a student when he or she is stuck. As mentioned in the introduction, we try to stay as close as possible to the pen-and-paper situation. Instead of selecting rewrite steps from a menu, a student rewrites equations in a text field. When a student is stuck he/she press on the help button and a next step is produced. This also shows that *the computer can take over if the user is lost* criterion is satisfied. We think our Equation Solver is *easy to use*: no unnecessary typing is required, an infinite Undo is provided, and no unnecessary distractions have been added to the Equation Solver. Finally, the Equation Solver goes beyond the answer-only approach and is thus *usable with a standard curriculum*: it supports a standard curriculum in mathematics, which emphasizes step-by-step solutions.

We have implemented the Equation Solver in Haskell using standard tools from compiler technology, such as parser combinators, and pretty-printing combinators, and using the standard compiler architecture, consisting of a parsing phase, an analysis phase, and a code-generation phase. In our case the code-generation phase is the feedback-generation phase. We intend to make our tool publicly available in the future. The latest version of the tool can be obtained via email from the authors.

Related work. We have found little literature on structured feedback in e-learning tools, and the way feedback is produced. Most intelligent tutoring systems that have a feedback component use techniques from Artificial Intelligence to report feedback. We think that using the structure in the data and the rewrite rules, we can give more precise and detailed feedback. Of course, there will still be situations where our feedback is insufficient: the amount of possible errors is infinite.

Other tools for solving systems of linear equations pay little or no attention to feedback. For example, the Linear System solver (using determinant) [2] returns 'ERROR in perl script on line 23: Illegal division by zero at (eval 129) line 37', if an unsolvable system of equations is entered. Commercial tools such as Algebrator [11] and MathXpert [1] do give feedback on the syntactical level, and hints about making progression, but do not use a structural approach to providing feedback about rewriting steps entered by the student

In [3] an implementation of a generic exercise for computing the derivative of elementary functions is presented. The system uses rewrite rules called domain rules and decomposes the original problem into sub-problems obtaining a multi-step exercise based on a solution graph. An interactive exercise is than seen as a collection of problems together with the order in which they are solved. According to the students answer and a predefined strategy, a next step is selected. The correctness of a students answer is evaluated by a computer algebra system. In this way, a student is guided in solving the initial exercise. A similar approach is taken by Goguadze et al. in ActiveMath [5]. Our approach is not based on a solution graph, but uses indicators to inform the student about progression of the solving process and a rewrite analysis module to determine which rewrite rule has been (correctly or incorrectly) applied. As a result, the steps a student can take are not limited by a predefined set of rewrite rules or solving strategy, but can be any combination of correct or erroneous rewrite steps. If a step is erroneous, the tool of course complains, but it also tries to give a helpful error message to the student. If the tool cannot recover which rewrite rules have been used, the indicators can still help a student in determining whether or not he or she is on the right track.

Future work. The work reported on in this paper is part of our work about feedback, see [8] for a general description of our research. In the future we want to work on several things:

- We want to investigate if we can recognise the application of more than one rewrite rule when comparing a submitted system of equations with the previous system. This probably requires some advanced reachability analysis.
- We want to apply our framework in more domains taken from mathematics.
- We want to allow for nice presentations in our tool, so that for example `(3/2)*x` can be displayed as $\frac{3x}{2}$. The Proxima editor [10], or Amaya [13], might be useful here.
- We want to do more research on types of feedback and the way these can be incorporated in the framework. For example, feedback about the goal structure leads to better performance than feedback about the reasons for the error [6]. Goal related feedback allows students to correct the incorrect actions more often than other types of feedback. Our framework explicitly incorporates the goal and the indicators relate the current system of equations with the goal.
- We want to perform experiments with our tool, in order to obtain experience with different levels of detail in feedback and progression indicators.
- We want to apply our framework on a rather different domain such as UML diagrams: can we apply the framework to provide feedback in an e-learning tool that supports the construction of UML diagrams?

Applying our framework to other mathematical domains, and to more or less structured domains outside mathematics will help us in evaluating the applicability of our ideas.

On a different note: we would like to analyse the behaviour of a student using the Equation Solver to generate new problems, in which rules a student has not applied yet have to be used, or in which the student has to practice rules in applying which he or she made errors solving the last problem.

Acknowledgements. Bert Zwaneveld gave feedback on drafts of this paper. Evert van de Vrie pointed us to the right information. Doaitse Swierstra and Atze Dijkstra helped us in using the Utrecht parsing, scanning, and attribute grammar tools.

References

1. M. Beeson. Design principles of mathpert: Software to support education in algebra and calculus. In N. Kajler, editor, *Computer-Human Interaction in Symbolic Computation*, pages 89–115. Springer-Verlag, 1998.
2. Igor Chudov. Linear system solver (using determinant), 2004. See `http://www.algebra.com/algebra/homework/coordinate/linear.solver`.
3. Arjeh Cohen, Hans Cuypers, Dorina Jibetean, and Mark Spanbroek. Interactive learning and mathematical calculus. In *Mathematical Knowledge Management*, 2005.
4. Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In *Handbook of theoretical computer science (vol. B): formal models and semantics*, pages 243–320, Cambridge, MA, USA, 1990. MIT Press.

5. G.Goguadze, A.González Palomo, and E.Melis. Interactivity of exercises in activemath. 2005.
6. J. McKendree. Effective feedback content for tutoring complex skills. *Human computer interaction*, 5:381 – 413, 1990.
7. E. Mory. Feedback research revisited. In D.H. Jonassen, editor, *Handbook of research for educational communications and technology*, 2003.
8. Harrie Passier and Johan Jeuring. Ontology based feedback generation in design-oriented e-learning systems. In P.Isaias, P. Kommers, and Maggie McPherson, editors, *Proceedings of the IADIS International conference, e-Society*, volume II, pages 992–996, 2004.
9. Simon Peyton Jones et al. *Haskell 98, Language and Libraries. The Revised Report*. Cambridge University Press, 2003. A special issue of the Journal of Functional Programming, see also `http://www.haskell.org/`.
10. Martijn M. Schrage. *Proxima – a presentation-oriented editor for structured documents*. PhD thesis, Utrecht University, The Netherlands, Oct 2004.
11. Sofmath. Algebrator – algebra help software, 2005. See `http://www.algebra-help.com/g2-solve-x.html`.
12. S. D. Swierstra and L. Duponcheel. Deterministic, error-correcting combinator parsers. In John Launchbury, Erik Meijer, and Tim Sheard, editors, *Advanced Functional Programming*, volume 1129 of *LNCS-Tutorial*, pages 184–207. Springer-Verlag, 1996.
13. World Wide Web Consortium. Amaya web editor/browser. `http://www.w3.org/Amaya`, 2004.

Interoperability Issues between Markup formats for Mathematical Exercises

Giorgi Goguadze[1], Manolis Mavrikis[2], Alberto González Palomo[3]

[1] University Of Saarland,D-66123, Saarbrücken, Germany
george@activemath.org
[2] The SCRE Centre, University of Glasgow,
G3 6NH Glasgow, UK
M.Mavrikis@ed.ac.uk
[3] DFKI Saarbrücken, D-66123, Saarbrücken, Germany,
alberto@activemath.org

Abstract. In this paper we describe several existing knowledge representation formats for interactive exercises and how these address the representational needs of mathematical exercises. The paper also provides an overview of existing tools that support these formats such as players, rendering exercises and the role of mathematical services assisting these players. Since most of the developers of this family of languages have the same goal, it is now more possible than before, to reach a common representation format. Having this in mind, we discuss features, limitations and the interoperability between them.

1 Introduction

Authoring exercises, and particularly interactive and intelligent ones for mathematics, is one of the most time consuming processes in the e-learning field. In addition, as [10] describes, authoring is often conducted in a proprietary format of particular systems that hinders reusability, sharing and exchange between interested parties. Efforts are underway to establish standards and specifications that would facilitate authoring, reusability, exchange between educators, and interoperability among systems. These efforts often have different perspectives but at least share some common goals making the transformation between data formats more possible than ever before. On the other hand there are still problems, since some formats are not generic enough, the representation is often bound to the underlying technology of the system, and the efforts of transforming such representations fail at these system specific parts.

This paper reviews briefly these approaches and tools that support them, and describes some important interoperability and transformation problems among them in an attempt to reveal basic common subset of those formats.

2 Data formats for interactive exercises

2.1 QTI and related formats

The IMS Question & Test Interoperability (QTI) specification provides a data model for the representation of questions (and tests), the way to process the user's response and their corresponding results. It has been developed by the non-profit organization IMS Global Learning Consortium, whose mission is to promote the adoption of open technical specifications for interoperable learning technologies worldwide.

QTI's very first version appeared in early 2000. After a number of issues that were raised by implementers and other interested parties the specification was extended and updated a couple of times. Other issues uncovered more fundamental aspects of the specification that would require extensive clarification or significant extension in order to be addressed. As the QTI overview describes, "since the QTI specification was first conceived, the breadth of IMS specifications has grown and work on Content Packaging, Simple Sequencing and most recently Learning Design created the need for a cross-specification review".This review process led to a release of the second version which focuses only on the individual items and does not update those parts of the specification that dealt with the aggregation of items into assessments.

In this second version several question types are supported. MCQs, fill-in-blanks, even matching and other types of graphical interactions. A major change was the introduction of the concept of cloning and question templates which allow the creation of a set of questions based on randomly selected values which can be used in the text and response processing of the question. The new specification supports templates that define common marking schemes that processing engines can support in advance in a way that enables further interoperability. Of course, other than the default response processing templates, authors can declare their own processing instructions. The response processing model has been changed from the quite declarative one of the first version to one that supports more complicated response processing, that consists of a sequence of rules that are carried in order by the response processing engine. Based on some flag-like variables, which are set during the response processing, conditional feedback can be presented to the learner. Although version 2 also allows multiple parts of a question and adaptive items which change according to the response processing of previous steps, these parts are not reusable but bound to the specific item that includes them. In addition, authoring the several conditions and paths becomes very complicated due to the programming-like (yet XML-based) language that the response processing is based on. On the other hand, one could argue that this provides enough flexibility to encode complex and elaborate questions. However, as [1] observes, this greater flexibility inevitably provides multiple ways of implementing a response processing. This way one can use a more complex coding than what is required. Such redundancy does not introduce ambiguity, however, it can hurt interoperability as some engines will not be able to support the more complicated questions.

2.2 Mathematical exercises and QTI

It is worth noting here that QTI (especially its first version) was not explicitly developed with mathematics in mind [4] and ignores many significant problems that authors of mathematical activities face. These have been discussed in several meetings and workshops of the community [12,14,13] and mailing lists[5]. The main issues are summarized here:

- Answers to mathematical questions, more often than not, involve mathematical entities which need to be understood by the system, processed and evaluated appropriately.
- Mathematical questions especially for Science and Engineering often test precision and accuracy of real values and therefore feedback and scoring needs to be adapted to the students' answers.
- It is not possible to have mathematics in all of the parts of the question (e.g title, statement, feedback) in an interoperable, platform-independent and system-understandable way.
- It is often necessary and faster to author mathematical questions, describe their response processing and the feedback they provide, by using mathematical constructs like variables that can also be randomized.
- QTI is weighted towards "teacher provided" response questions in which a student is required to make a certain kind of selection from responses provided by a teacher (for example MCQs). Actually, a teacher more usually would prefer to establish that the student's own answer satisfies certain (mathematical) properties.
- Mathematical questions often require the learner to provide an answer for which the form (e.g matrix, fraction, parts of a formula) is already provided (in order, for example, to test specific aspects or even to make it easier for the student to answer).
- There are many examples of question structures which research has shown to be useful and effective. These include such things as conditional hints, multiple parts and partial marking. Such features are now reasonably well established in mathematical Computer-Assisted Asessment (CAA) and their absence provides less flexibility during the question processing and leads to less effective feedback.

2.3 MathQTI as an extensions to QTI

The aforementioned limitations were some of the inspiration behind the JISC-funded "Serving Mathematics in a distributed e-learning environment" [6] project which aimed to develop open-source software and tools to address the special

[4] see discussions in http://ltsn.mathstore.ac.uk/articles/maths-caa-series/ and http://assessment.cetis.ac.uk/minutes

[5] http://lists.ucles.org.uk/public/ims-qti/ and http://mantis.york.ac.uk/moodle/mod/forum/view.php?id=88

[6] http://maths.york.ac.uk/serving_maths

requirements of Mathematics in the context of e-learning and e-assessment. Part of the project looked into extending the IMS Question & Test Interoperability (QTI) specification in an way as compatible and convenient as possible, in an attempt to enable the exchange of questions with mathematical content between question engines and authoring tools.

A result of this process was a first draft of the MathQTI specification[9] and the decisions to:

- use presentation MathML for questions statements and feedback.
- employ OpenMath[18] with a restricted content dictionary for the mathematical expressions in the template and response processing. An attribute determines how the OpenMath expression is dealt with (is it evaluated to a number, to an expression, simplified, or left unevaluated).
- introduce a new tag to test for syntactical equivalence in response processing.
- allow an alternative way of including interaction elements in question statements and feedback, which works via id's and the "for" attribute, further separating content from presentation and allowing mathematical expressions to be substituted by interactive elements.

Most of these aspects were inspired from other formats (like the ones in WaLLiS and ActiveMath, which are discussed in the following sections) in an attempt to bring them close and enable interoperability. On the other hand, as the format is based on QTI it benefits from its features but also suffers some of its limitations in relation to multiple parts, help or hint requests and adaptive response processing. Although it is appreciated that there will always be a tension between writing a specification which includes interesting "features" and the complexity of writing platforms which comply with emerging standards, features such as conditional hints, multiple parts and partial marking are now reasonably well established in mathematical CAA and their absence from a common specification would hinder its widespread approval.

2.4 ActiveMath Exercise Format

The interactive learning environment ActiveMath [15] possesses its own representation format for interactive exercises, which extends the OMDoc [17] knowledge representation. As in OMDoc, OpenMath standard [18] for representation of mathematical formulas is used. This format was first defined in [5], then some refactoring was made in cooperation with the WaLLiS project [10].

Some further changes were made over the last few years, based on experience gained from testing the capabilities of such a representation with authors in realistic conditions. The expressivity of the exercise language was improved based on real demands from an extensive test and study at a school (70 6th grade pupils) in Saarbrücken, a customisation as Matheführerschein[7], and a large study with 250 students at the Institute of Education of the University of London.

[7] http://mfonline.activemath.org:8080/MatheOnline

These changes were mainly fuelled by the goal to reuse exercise encodings for several learning situations and different tutorial strategies.

The knowledge representation of ACTIVEMATH, as described below, was the basis for defining the less expressive knowledge representation for interactive exercises in the LEACTIVEMATH project. The current version of the ACTIVEMATH exercise knowledge representation is described in [6]. In this format, a fully authored exercise is a finite-state machine consisting of a graph of `interactions`, which represent the nodes of the exercise graph and transitions that connect different nodes in this graph. The basic structure of an interaction element is shown in Figure 1. An `interaction` contains one or more `feedback` elements providing textual[8] or graphical feedback to the learner. Each feedback is followed by an `interactivity assigment` specifying which interactive elements should be used in the `interaction` (e.g. fill-in-blanks, multiple-choice-questions). Each `interaction` can have one or more interactive elements of different types.

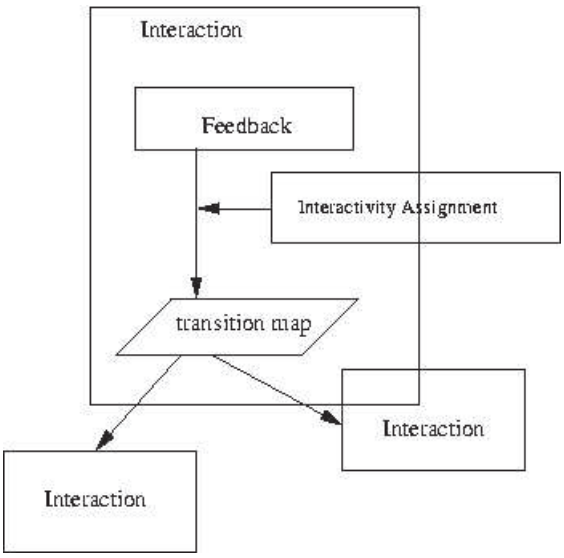


Fig. 1. ACTIVEMATH Exercise Subsystem architecture

Finally a `transition map` element follows, containing transitions. Each transition connects two `interaction` nodes of the exercise. It contains a condition that has to be satisfied in order to jump from the current `interaction` to the target of the transition. Conditions contain comparisons of different types, that have to be applied to the user's input. A comparison can be syntactic — without

[8] A recent development allows ACTIVEMATH to read the feedback text aloud.

simplifying the terms in the user's input, numeric — comparing numeric expressions, and semantic — where the external mathematical services are asked to perform the calculations in the given context.

Each `interaction` node can have metadata, specifying mathematical and educational properties of an `interaction`, such as *difficulty* of a step w.r.t. the particular *learning context*, *competency* to be trained in the step, *concepts* that this step is elaborating, specifying whether the *hint* is given, and so on.

Each transition can have diagnosis information in case the user's input satisfies the condition of a given transition, such as the *achievement* of the learner w.r.t. the task, *relevance* of the answer and/or *misconceptions* the learner had in the step.

Importantly, due to its compactness and extensibility, the ActiveMath exercise format makes it easy to build more complex representations. This has often been requested over the last few years of the application of ActiveMath by authors and users. Therefore a property of the system is that exercises can also be fully or partially generated and the graph of interactions in such exercises can be dynamically changed. Tutorial strategies applied to a compactly encoded exercise will enhance the exercise by introducing additional nodes, cloning existing ones, and adding more connections between them.

2.5 WaLLiS format

WaLLiS is a web-based system developed at the School of Mathematics of the University of Edinburgh. A detailed description can be read at [10]. WaLLiS employs interactive exercises that in a way have a predetermined context as they are designed for a specific learning situation. Yet they are adaptive in the sense of what kind of feedback they provide. These interactive exercises are authored in an XML format which is very similar to the initial ActiveMath one [5], whose main characteristic was that it also employs OpenMath to represent the formulas needed for representing the response processing conditions. This format was a direct consequence of early experience from the system's use and authoring content and needs of the context where it was used. Several of the issues mentioned in section 2.1 actually arose during this period.

In addition, in an attempt to address the reusability and ease of maintenance of multiple parts, it was decided that it was better to link rather than nest items and their parts. Thanks to the collobaration between the authors of this paper the two formats are now isomorphic yet their players follow complete different architectures, demonstrating the power of employing generic formats that are not bound to implementation. ActiveMath has a clear client-server architecture (described at section 5.1) and the WaLLiS system follows a series of XSLT transformations that result to an XHTML interactive document which is presented to the learner. Briefly, the transformation results appear on a browser page that includes a form for the interaction, hidden elements (eg. the solution or the hints) and embedded JavaScript that takes care of the interaction and the communication with the server to evaluate the learner's response, without the learner ever leaving or submitting the page. More details are provided at

[11,?]. The main point here is that despite the differences the isomorphic formats have been used to share exercises by transforming from one to the other and the fact that both were adequate for two complete different implementations provides positive indicators towards realising the ideal situation of having a common format or at least translating automatically between them without loss of functionality.

2.6 LeActiveMath Exercise Language

The MathDox language [8] is developed in RIACA [9] (TUE) and is based on the former representation, influenced by DocBook [10] and OMDoc languages, called MathBook (RIACA) [7]. OpenMath is used for mathematical formulas here as well. The language is mainly closer to the QTI one with more flexibility (but also increased complexity) in the response processing where specialised tags ("if", "then") are used to represent the response processing.

The LeActiveMath exercise language is developed in RIACA within the LeActiveMath project and is based on the ActiveMath exercise language with some additions, inspired by MathDox. For instance, it makes heavy use of state variables, called "dynamic context" in MathDox. In addition the language allows MONET queries [16], sent to the external mathematical systems.

3 Tools for Authoring and Interpretation of the Exercise Languages

3.1 The ActiveMath Tools

The main component of the ActiveMath learning environment is the exercise subsystem, developed by two of the authors. It comprises several components communicating within a modular architecture. Describing this in detail is beyond the scope of this paper. Therefore we describe only those parts of the exercise subsystem architecture which deal with interpretation of the knowledge representation of the exercise steps and their generation.

Briefly, the core of the exercise subsystem consists of an *Exercise Manager*, *Interaction Generator* and *Evaluator/Diagnoser* components. The Exercise Manager is responsible for communicating with the database and the client (typically a web browser). It receives the problem statement of the step and the user input, forwards them further to the Interaction Generator and asks it to return the next `interaction`. The Interaction Generator queries the Evaluator/Diagnoser for diagnosis of the user's action, and builds a new `interaction` based on the result of this diagnosis.

The Interaction Generator base class provides an architecture for defining different Interaction Generators. The default Interaction Generator used in ActiveMath is called *Static Interaction Generator*. It is designed for retrieving

[9] http://www.riaca.win.tue.nl

[10] http://www.docbook.org

`interactions` in manually authored exercises. In this case, the Evaluator has to check whether the user's input satisfies one of the manually authored conditions provided for the current `interaction`, and the conditions point to the `interactions` to be executed in the next step in case they are satisfied. The benefit of this approach is that, instead of the Static Interaction Generator, other specific generators can be employed. Each Interaction Generator can define its own custom Evaluator/Diagnoser class, extending the basic Evaluator/Diagnoser for more advanced queries of the mathematical services.

Since the exercise subsystem of ActiveMath allows for so-called virtual exercises, in which the `interactions` are generated, writing such interaction generators can give birth to whole classes of automatically generated exercises. There already exist several custom interaction generators such as the "Randomizer", which extends the static one by randomizing variable values (which can be whole sub-formulas, not just numbers) in the manually authored exercise, using a CAS for computing those values when necessary. Another generator defines a custom Evaluator that queries the available services such as Domain Reasoners or CAS in each step, in order to receive the diagnosis on the user's actions. Based on the result of this diagnosis, the next `interaction` is generated.

In order to facilitate reusability of exercises in different learning situations, additional generators can be applied to an authored or generated exercise, in a similar way as the randomizer mentioned before. These generators implement different tutorial strategies which define the sequence in which the steps are rendered, the frequency of hints and other parameters that may vary in different learning situations. Such a flexible system of cascading interaction generators is the result of a careful design of an exercise language and the corresponding engine, interpreting and rendering this language in a modular and extensible way.

3.2 RIACA Exercise Repository Tools

The Exercise repository, developed within the LeActiveMath project, provides an authoring tool for exercises that can be encoded in the LeActiveMath exercise language. This authoring tool allows for constructing exercises with fill-in-blanks and multiple-choice-questions. Exercises can have multiple steps and connect to CAS for checking the answers of the learner. Certain randomization of exercises is also possible. The repository itself is provided as a web-service that serves interactive exercises that can be used directly in the web or to be sent and rendered in another system (for example LeActiveMath).

3.3 The MathQTI engine

The MathQTI engine[11] is a module of WaLLiS that interprets a MathQTI exercise in a similar way as described in section 2.5 but in way that is separated from the rest of the system, with the hope of reusing it as an engine for other

[11] `http://sourceforge.net/projects/wallis`

systems. It is worth mentioning here that for interpreting the OpenMath formulas and handling the response processing with the Yacas system (which is used at the background for evaluating the learner's answer) the RIACA JSP (Java Server Pages) tag libraries[12] are used. With simple extensions and some additional in-house coded libraries the math-qti engine provided a proof-of-concept system for further developments of WaLLiS and other systems. This demonstrates, once again, that once based on common standards (such as OpenMath) at least a partial interoperability and code reusability is possible.

3.4 Protocols and web services

Within the LeActiveMath project, generic web-services support is developed in the ActiveMath system. It consists of an OpenMath broker component, which receives mathematical queries from the ActiveMath exercise system, and distributes these among available web services, capable of providing an answer. Among the services connected to such a broker, are Computer Algebra systems (in our case Yacas, WIRIS and Maxima) and other mathematical systems such as Domain Reasoners, written in Prolog, that all communicate with the ActiveMath broker in a variety of ways, including complete SOAP (Simple Object Access Protocol)[13] queries, XML-RPC (Remote Procedure Call), and local system processes via pipes, all of them transparent to the rest of the system and to the content authors and learners. The queries defined have the following parameters:

- the name of the query
- one or more formulas in OpenMath format
- context of the query defined by a set of concepts in OpenMath format
- depth of the rule application

On one hand such queries can be used for semantic comparison of the user's input within the condition in the transition map of the authored exercise. On the other hand, several other queries can be sent for receiving more detailed diagnosis of the learner's actions and, based on such diagnosis, construct the next node in the generated exercise.

It is worth noting that the LeActiveMath exercise language of RIACA, used for the separate repository of interactive exercises outside the ActiveMath system, directly defines MONET queries inside the content of the exercises. These queries can specify the name of the concrete CAS to be used for evaluation. In the opposite case the available CAS is selected automatically. Therefore, the exercises which specify the concrete CAS to be used are not guaranteed to be reusable with other CAS. On the other hand, it can be argued that this way authors of the exercise can be sure that the result of their query will be as expected and therefore pedagogically more sound.

[12] http://www.riaca.win.tue.nl/products/taglib/

[13] http://webservices.xml.com/pub/a/ws/2001/04/04/webservices/index.html#soaphead

Finally, it should be mentioned that efforts are underway to define common protocols for exercise players. For example, the Remote Question Protocol (RQP[14]) that allows e-learning systems to use external services for presenting a question and evaluating the student's response. Other query languages like the MONET Mathematical Query Ontology, as well as the developments of mathematical web services will enable more and more Interactive Learning Environments (ILE) to employ facilities such as elaborate question players, Computer Algebra systems or other processing engines, thus enhancing their functionalities and allowing more elaborate questions not limited by the features of an specific system.

4 Transforming Between Formats

We have mentioned before that several transformations are possible between formats. For example. because of the close collaboration between the authors, back-and-forth transformations between the WaLLiS and the ActiveMath formats is possible. At least for now we are interested in the main framework of the exercise rather than all the medatata and additional information that are included in it. It is more important, at least at this phase, to be able to automatically convert the questions statement and response processing even if that does not entail the full fledged result reporting and metadata annotations that are still perhaps dependent on the context and use of the exercises.

Similarly, QTI exercises have been transformed to the WALLIS format. This was certainly easier for the first version of QTI (v1.2) which was more descriptive in nature than v2. Since we are able to transform between the WALLIS format and the ACTIVEMATH and LEACTIVEMATH one it is hoped that in the future transformations from QTI to these formats will also be possible. It has to be said here that some exercise types (such as matching and puzzles) are worth investigating further as until now they were not our primary target. On the other hand, we have found that only simple exercises that do not have multiple steps and interactions are transformable from the WaLLiS formats to the QTI one. In addition, exercises from QTI v2 are easily transformed only when these exploit the predefined response processing templates. Otherwise, we have found that exercises that were authored using the full flexibility of QTI v2 were impossible to transform in a generic way (e.g. an all-purpose XSLT) mainly due to the prescriptive nature of QTI v2. Once extra knowledge on the rationale behind the exercise, the flags it uses, and the naming conventions are included to the transformation process the task is simplified. Unfortunately the fact that human intervention is required complicates and makes the process time consuming. This is certainly a matter worth investigating further.

Some mappings that are certainly planned and partially already implemented include the back-and-forth mapping between MathDox, ACTIVEMATH, WALLIS and MathQTI. For example, one of the expected outcomes of the LEACTIVE-

[14] `http://mantis.york.ac.uk/moodle/course/view.php?id=14`

MATH project is the interoperability of ACTIVEMATH and LEACTIVEMATH exercise languages. The mappings are mostly straightforward, but not all of the formats are isomorphic. Some of them are bound to a particular implementation rather than designed with reusability in mind. It is certain that the notion of dynamic states and the variable flags used in QTI and MathQTI require additional knowledge of the context making the automatic translation even harder. To avoid these problems it seems necessary to avoid encoding information which provide additional knowledge such as pedagogical knowledge rather than just the exercise itself. It is in the benefit of everyone who is involved in the process of authoring and deploying content in real situations, to be able to author and use exercises in different educational situations and goals. We believe that as with other cases where interoperablity was achieved mainly due to the separation of layers (such as the early days of XML and the separation of content and presentation) the dream of interoperablity will be achieved if more formats avoid as much as possible to mix pedagogical knowledge or other information with the response processing.

5 Conclusion

It is clear from this paper that further work is necessary in order to identify the most generic exercise format suitable for most of the applications and try to join the efforts of the involved communities and produce such a uniform format. This seems to be reasonable, since all the formats considered in this paper (except QTI) not only share common properties, but are not far from being isomorphic. By reaching this common agreement, all the involved communities would gain variety of tools and players, developed by each side, that will become reusable for others but more importantly a common pool of exercises and content that can be reused and shared between them.

Acknowledgement

Part of research for investigating the knowledge representation of interactive exercises was supported by the `iClass` project, funded under the FP6 Framework Program of the European Community – (Contract IST-507922).

This publication is also partly a result of work in the context of the LEACTIVEMATH project, funded under the 6th Framework Program of the European Community – (Contract IST-507826).

References

1. D. Bacon. Review of QTI v2.0. Available online:
<http://support.imsglobal.org/question/qtiSupportIndex.html>
2. A. Cohen, H. Cuypers, D. Jibetian, M. Spanbroek, LeActiveMath Exercise Language, Deliverable D7., LeActiveMath Project, FP6 Framework, 2005.

3. A. Cohen, H. Cuypers, E.R. Barreiro, H. Sterk Interactive Mathematical Documents on the Web. Algebra, Geometry, and Software Systems 2003, pages 289–307, 2003.
4. Global Learning Consortium. IMS Question & Test Interoperability Specification: A Review <http://www.imsglobal.org/question/whitepaper.pdf>
5. G. Goguadze, E. Melis, C. Ullrich and P. Cairns, Problems and Solutions for Markup for Mathematical Examples and Exercises. In Proceedings of the Second International Conference on Mathematical Knowledge Management, MKM03, Andrea Asperti (ed.). <http://www.ags.uni-sb.de/~ilo/articles/EncodingExoExa.pdf>
6. G. Goguadze, A.G. Palomo, E. Melis, Interactivity of Exercises in ActiveMath. In Proceedings of the 13th International Conference on Computers in Education (ICCE2005), 2005.
7. MathBook Standard, Research Institute for Applications of Computer Algebra <http://www.riaca.win.tue.nl/products/mathbook/>
8. MathDox Website <http://www.mathdox.org>
9. M. Mavrikis. MathQTI draft specification overview. Available online at <http://www.maths.ed.ac.uk/mathqti/docs/v0p3/mathqti\protect\unhbox\voidb@x\kern.06em\vbox{\hrulewidth.3em}v0p3.pdf>
10. M. Mavrikis and A. Maciocia. Wallis: a web-based ILE for science and engineering students studying mathematics. *Supplement Proc. of the International Conference on AIED*, 2003.
11. M. Mavrikis and A. González Palomo. Mathematical, Interactive Exercise Generation from Static Documents. *Electronic Notes in Theoretical Computer Science*, 93:183–201, 2004.
12. C. Miligan. Question and test interoperability (QTI): Extending the specification for mathematics and numerical disciplines. *LTSN maths-CAA series*, Nov 2003. Available online at <http://ltsn.mathstore.ac.uk/articles/maths-caa-series/nov2003/>
13. M. Mavrikis. Thoughts on MathQTI (see "documents on MathQTI" section of the "serving maths" project or: <http://www.maths.ed.ac.uk/mathqti/documents.shtml>. Technical report.
14. Minutes of 1st MathQTI workshop. Available online: <http://maths.york.ac.uk/serving\protect\unhbox\voidb@x\kern.06em\vbox{\hrulewidth.3em}maths/mod/book/view.php?id=154>
15. E. Melis, E. Andrès, J. Büdenbender, A. Frischauf, G. Goguadze, P. Libbrecht, M. Pollet, and C. Ullrich. ActiveMath: A generic and adaptive web-based learning environment. *International Journal of Artificial Intelligence in Education*, 12(4):385–407, 2001.
16. MONET: Mathematics on the Net, EU Project <http://monet.nag.co.uk>
17. OMDoc: An Open Markup Format for Mathematical Documents, <http://www.mathweb.org/omdoc>
18. OpenMath Society Website, <http://www.openmath.org>

STACK: addressing the needs of the "neglected learners"

Christopher J Sangwin[1] and Michael Grove[2]

[1] Maths Stats and OR Network, School of Mathematics, University of Birmingham, Birmingham B15 2TT, United Kingdom
C.J.Sangwin@bham.ac.uk,
[2] Maths Stats and OR Network, School of Mathematics, University of Birmingham, Birmingham B15 2TT, United Kingdom
M.J.Grove@bham.ac.uk,

Abstract. This paper concerns the implementation of a computer aided assessment system for mathematics known as STACK. STACK makes use of the computer algebra system Maxima for a variety of tasks, the most important of which is establishing mathematical properties of student's answers. We discuss STACK with a focus on the data structure used to represent questions, and how teachers write questions of their own.

1 Introduction

This paper concerns the implementation of a computer aided assessment (CAA) system for mathematics known as STACK: a *System for Teaching and Assessment using a Computer algebra Kernel*. In this paper we pay particular attention to the needs of the *teacher* in authoring questions of their own.

As the names implies, STACK relies on a computer algebra system (CAS) at its heart to support a variety of tasks. The application of CAS to support an online assessment system is quite different from the roles to which a CAS is traditionally put. These include a reduction in the computational load, automation of graphical representations, and the ability to perform rapid re-calculation to facilitate explorations. As an illustration, consider the situation in which a student enters his or her response to a mathematical question into a CAA system. The system then uses a CAS to subtract the student's response from the teacher's response and to simplify the resulting expression algebraically. If the result is zero an algebraic equivalence between the student's answer and the teacher's answer has been established. Note that the CAA system evaluates the *student's answer* which contains mathematical content, rather than allow selection from a list of *teacher provided answers*, such as in multiple choice or multiple response questions. The system then takes appropriate action, such as providing feedback, assigning a mark and storing these outcomes in a database. Of course, establishing algebraic equivalence is only a prototype test and other more sophisticated response processing can be achieved with the support of a CAS.

2 Design goals

The primary design goal was to support the evaluation of student provided answers entered as algebraic expressions, for use in the teaching and learning of mathematics in higher education. Many existing generic CAA systems provide only question types in which *teacher provided answers* are selected by students. This occurs in multiple choice questions (MCQ) or similar (multiple response etc). While a well constructed multiple choice question presents a list of plausible *distracters*, which ideally will be constructed from knowledge and understanding of common student errors, the teacher is essentially forced to 'give the game away' by presenting these choices up front. There is also the possibility that the student will remember the distracter and not the correct answer, therby the testing process could well lead to incorrect learning. In any case, the purpose of many questions is grotesquely distorted by using a MCQ.

For example, solving an equation from scratch is significantly different than checking whether each potential response is indeed a solution. Another problem particular to mathematics is where the difficulty of a reversible process is markedly altered in different directions. For example, expansion versus factorization of algebraic expressions, or integration versus differentiation. The strategic student does not answer the question as set, but checks each answer in reverse. We preferred a system which evaluates student provided answers.

A parallel and equally significant goal, and one upon which we choose to dwell here, was to provide a means for teachers to mark up their own questions and learning materials as CAA. While students are the traditional focus of research into learning and teaching, the needs of teachers as learners of the art of CAA are rarely, if ever, considered explicitly. An underlying assumption seems to be that teachers are experts in both the theory and practice. While this may be a safe assumption for the enthusiasts who have traditionally pioneered CAA use, this cannot be sustained as CAA use continues to gain momentum and become mainstream. Furthermore, it may be possible to master the technical details of traditional multiple choice based systems in a short training period, but this is certainly not the case of a CAS: a highly technical and specialized piece of software. Previous systems, in particular the AiM system of [14], forced the question author to become a computer programmer, in effect writing substantial pieces of code to generate the required response processing commands for each question.

While the attention of the CAA community is predominantly, and quite rightly, focused on the needs of the students, our experiences in running CAA staff development sessions for colleagues in Higher Education Mathematics Departments suggests that teachers are often *neglected learners*. In order for a CAA system to be desirable for adoption by teachers there needs to be immediate progress in getting their questions online, and a steady and incremental learning curve as increasing sophistication is sought by the teacher as confidence and familiarity grows. Furthermore, the system should be adaptable by staff in a variety of ways to suit their student groups. In this way they have more control over their teaching, are more likely to consider carefully what CAA can and can-

not support, and hence in a better position to adopt effective pedagogic practice that would be the case with an off-the-shelf CAA system. Certainly in higher education there is an explicit desire for intellectual ownership over the learning materials used. Experience suggests that the majority of staff consider using a particular system only if their own questions can be developed, or existing questions easily modified.

One of the design goals of STACK was to provide mathematical sophistication in an environment of gradually increasing complexity. Routine tasks should be simple, while the maximum flexibility should be available to the expert.

3 Background

Systems in which the processing of student's answers is supported by computer algebra has gradually gained ground in higher education over the last five years. Perhaps the first system to make CAS a central feature was the AiM system, described by [5], with subsequent technical developments described in [14]. This system operates using Maple, as does the Wallis system of [6]. Other systems have access to a different CAS, such as CalMath which uses Mathematica, CABLE, see [8], which uses Axiom and the STACK system which uses the CAS Maxima. From private correspondence, the authors are also aware of systems which use Derive in a similar way.

Actually a notion such as "simplification" is not universal across different CAS implementations and there is a surprising variety amongst CAS's; the differences between which have been discussed elsewhere, for example [4], or [16]. These comparisons, however, are from the point of view of the research mathematician. The functionality required for the application of CAA is quite different from the needs of the research mathematician or the role of "super-calculator" to which CAS has traditionally been put when used with students. AiM, Wallis, CABLE and the other systems using CAS for CAA, including STACK, all accept the design decisions of their chosen CAS. Clearly, accepting design decisions 'as is' is somewhat unsatisfactory, but specifying the characteristics desirable in a CAS for our application lies outside the scope of this paper. Indeed, this is a substantial and separate topic and is an area of current research. Note that we are still using an objective test, since the notion of algebraic equivalence is well defined within each CAS.

In practice teachers use current deficiencies as "learning opportunities", much the same way as when discussing unexpected output from a CAS during classroom use. For example, students often express confusion at the 'loss' of the minus sign when $\cos(-x)$ is automatically simplified by the CAS to $\cos(x)$.

It should be noted that a CAS is not *required* for response processing of a student's answer and there are many examples of CAA and computer based learning systems in which a student is required to enter an answer. However, the authors of such CAA systems often replicate libraries of CAS-like functions. Examples of such systems are the CALM system of [1] and the Metric system of [9]. Aspects of this paper are just as relevant to these systems.

In evaluating student provided answers an understanding of students' mistakes is still important, since it is possible to compare a student's response with that arising from a known common mistake. However, there is no requirement to construct a particular number of such artificial mistakes for each question, or that these be displayed up front to the student.

In this new setting the student–machine interface takes on a new importance. The student must now communicate their answer unambiguously to the computer. Computer algebra systems already have a linear syntax, and one approach is to adopt this for students. A graphical input tool or pen-based entry system (see for example [3]) could also be used. Traditionally in an interactive CAS session the student can edit and alter their input at will, correcting inevitable mistakes and, where needed, consulting staff or online help files. We are designing a computer aided assessment system where the stakes are higher: the student is being evaluated on their input. We certainly do not wish to mix an evaluation of their ability to express themselves using the correct syntax with their ability to actually solve the problem in hand. If so, the user risks being penalized on a technicality which has important implications for high stakes assessment. Entry of answers by students is a quite separate topic from that of question authoring dealt with here, and has been addressed in, for example [10].

The immediacy of feedback is seen as one of the most important benefits of CAA. STACK separates two kinds of feedback. The first is associated with the *syntax* of the student's answer, the second to its interpretation *semantics*. To provide a consistent interface, feedback based on the syntax should not depend on the context in which a question is taken. So feedback about syntax errors is the same in a formative learning context as a summative test. Feedback based on the semantics of the answer should be at the discretion of the teacher.

To separate the syntax check from the request to mark a response STACK adopts the validate/mark feedback model used successfully by the AiM system of [5]. In this, the student enters their answer as a typed linear expression and then requests for this to be either validated or marked. The validate request checks the expression is syntactically correct and if so gives the feedback *"your last answer was interpreted as* $\cdots$*"*. The system then displays the student's answer in a traditional two dimensional form, as it might be printed on a page.

It should be noted that the CAS interprets the student's expression and represents it internally so that various "simplifications" take place, including gathering of like terms and ordering in polynomials. Hence, the resulting displayed expression may differ from that actually entered by the student which causes some difficulties when assessing very elementary mathematics. This is perhaps the most significant drawback of using a pre-existing CAS, over a suite of bespoke functions for this application.

Being able to implement semantic-based feedback is predicated on the ability to automatically establish mathematical properties of an expression. Our prototype 'Answer Test' was establishing the property of algebraic equivalence of the student's response with the answer of the teacher. However, other properties

are also important in learning and teaching, including the algebraic form of an expression (e. g. factored, expanded, partial fraction).

An example of a more sophisticated answer test is required to process the responses to the following question, which occurs in the context of a first year calculus class. "*Give an example of a function with a stationary point at* $x = 1$." To mark this, the CAS differentiates the student's answer with respect to x and substitutes $x = 1$. If the result simplifies to zero the student's answer is correct. This is a symbolic and mathematical manipulation of a student's answer. Note, that the term "stationary point" was defined explicitly and carefully in the notes, although this term is not universal. As one student comments after using AiM for a related question:

> Recognising [...] the functions produced in question 2 was impressive, as there are a lot of functions [...] and it would be difficult to simply input all possibilities to be recognised as answers.

The pedagogic potential for this style of question is well documented in the educational literature, for example [15] or [7]. The practicalities of using CAS to assess them is considered in [13].

4 Implementation

Fundamental to the implementation is the data structure that encapsulates our question type. To illustrate this, and hence the functionality of STACK more directly, we use the following question which is ubiquitous in first year calculus courses:

$$\text{Integrate } p \text{ with respect to } x. \tag{1}$$

CAS allows mathematical expressions to be generated in a carefully structured but pseudo-random way from a specific seed value. STACK then creates the HTML that the browser uses to display the question. This can be used to provide each student with a similar, but distinct question version. Of course attention must still be paid to the choice of random parameters to prevent the generation of impossible or trivial questions. In turn, random versions of an individual question will be created to fit within a carefully designed and structured scheme of work by taking a specific value for p. For example, we might choose a question space consisting of $p := nx^m$ where $n \in \{2, ..., 9\}$ and $m \in \{3, 4, 5\}$. We begin by briefly examining how a student would use STACK.

4.1 The students' view

The student interface with STACK is a simple quiz structure which presents a list of questions on a single page, or one at a time, at the discretion of the student, thereby allowing questions to be tried in any order. Repeated attempts are encouraged and the quizzes may have have a "due date". We concentrate on interactions with a single question, rather than with the content management system.

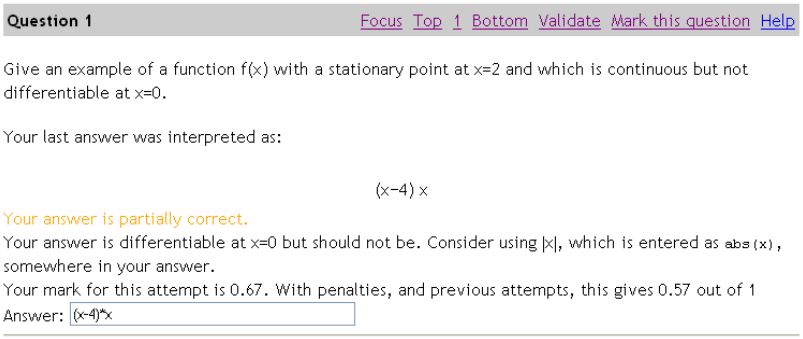


Fig. 1. An example STACK question

The student enters a response to the question in the form provided on the web page, and selects a button to indicate whether the answer is to be "validated" or "marked". The validate step essentially performs a syntax check to establish that the expression is syntactically correct and displays the student's one dimensional typed expression in a two dimensional display format that is easier to read and more in keeping with traditional printed mathematics. Feedback in the form of a displayed version of the student's answer, or related to problems with the syntax, is provided at this stage.

The mark step performs a validation and then also the semantic evaluation to actually establish the mathematical properties of the student's answer. A student selecting this step is indicating he or she wishes to have the answer assessed. If invalid, feedback is provided to guide the student. If the attempt is valid a displayed version of the student's answer is shown together with any semantic feedback designed by the teacher. A mark for this attempt is also assigned. The student's raw answer, action requested and all outcomes are stored in the database during both steps. The student is usually able to make an unlimited number of attempts at each question. How marks for repeated attempts are calculated is at the discretion of the teacher.

4.2 STACK's learning objects

We shall now examine in more detail the implementation of question (1). As with any CAA learning object it naturally includes more than just the "question" posed to the student. The minimum information which needs to be entered to create a viable question is the Question Stem, the Teacher's Answer and which Answer Test is to be applied. For many simple questions this markup process is entirely straight forward. Of course, once this has been mastered teachers soon seek to include randomly generated parameters, more sophisticated response processing and consequent outcomes, including feedback. We use the word "question" to be synonymous with the entire data structure. The practical developments of STACK connect abstract mathematical questions intimately

to the data structure representing their implementation. Hence, it seems most expedient to detail the more important data structure fields explicitly in this section to illustrate the authoring of question (1).

Question Variables are an optional list of the form `key := value`. In our example we have the assignments

```
n := 2+rand(8);     m := 3+rand(3);     p := n*x^m
```

When a version of the question is created this list is passed to the CAS and evaluated in order. Internally, both displayed and content forms are generated for each `key`. These variables are available to all parts of the question, including the text fields (into which the displayed form may be inserted) and the response processing (which makes use of a content form). Question Variables can be used to generate the question, the answer, incorrect responses worthy of partial credit and steps in a worked solution.

Clearly the question author will need to know some CAS commands, but in practice the number of algebra, calculus and other commands is rather modest. To provide flexibility, the teacher can access most of the underlying CAS functions, though it should be noted that this field is optional and can be ignored in simpler questions.

Question Stem is the actual "question" posed to the student and is one of only two compulsory fields which the teacher must enter for a valid question object.

Teacher's Answer is the "correct answer", as the name implies, and is the only other compulsory field which the teacher must enter. This may include CAS instructions using any of the Question Variables. In our case we need to perform the integration so we have a number of choices for the answer. The first consists of `n/(m+1)*x^(m+1)+c` in which we perform the integration ourselves. The second is simply `int(p,x)+c` which takes advantage of the CAS's integration command. The content form of the instantiated versions of the variables are used to calculate the actual value of the answer in each case. Notice we have chosen to use the letter `c` as a constant of integration.

Student's Answer Key denotes the name of the variable to which the student's answer is assigned. This variable name may be used in the response processing algorithm to manipulate the student's answer and process it using the CAS, it is a compulsory field but automatically assigned a default value.

Feedback Variables (optional) are similar to Question Variables but may depend on Question Variables and/or the Student's Answer Key. The values of these variables are calculated after each valid response of the student, using the instantiated versions of the Question Variables, and are used exclusively for response processing, and generation of feedback.

Worked Solution is optional and is treated internally in exactly the same way as the Question Stem. This may contain the full solution to the question, although how and when this is displayed to the student depends on the context in which the question is set. Typically it would be available after a quiz due date. It may not depend on any of the students' attempts.

Question Note is used by the teacher to leave an intelligent "note to self" about the specific values of Question Variables a student has been given. It is optional but very useful for statistical grouping of identical question versions, providing hints or analysing incorrect answers. A question note in our example might simply be `@p@`.

Questions also include various metadata fields, selected from Dublin Core.

4.3 CAS usage

The design of STACK attempts to minimize the requirement that the question author become a programmer in the CAS language. A question author will certainly need facility with some commands in the CAS language. We did not want to require knowledge of loops, conditional statements and branching. Other existing systems do require the question author to become a programmer or even system developer and experience suggests this is a significant barrier for teachers. For example, as AiM is written almost entirely using Maple's programming language, the question author needs to write fragments of Maple code for anything other than a simple algebraic equivalence check. Examples of such code is given in [12].

To facilitate interoperability of questions between CAA systems, we used as little CAS-specific code as possible, reserving it for the mathematical tasks for which it was designed. Ultimately we determined that all functions requiring CAS support reduce to two operations. The first is the evaluation of a list of assignments of the form `key := value`, for example in creating versions of the Question Variables. The second involves applying a named 'Answer Test'. This takes either one or two expressions and seeks to establish some property. We nominally think of the first of which as being the student's answer, the second (if used) as being the teacher's. However, the two expressions could be a CAS manipulation involving the student's answer or another expression connected to the question. The teacher's answer could be an incorrect expression derived from a common mistake. The result of applying an Answer Test is a data structure comprising:

1. A boolean value, denoting whether the property has been established.
2. Feedback for the student, both numeric and textual.
3. An answer note, used for statistical analysis.

The following are examples of Answer Tests.

Algebraic Equivalence performs full algebraic simplification of the difference between the two expressions and compares the result to zero. Hence this test indicates, as far as is possible within the computer algebra system, whether two expressions are considered to be algebraically equivalent. This is the default test and it copes with a variety of expressions including sets, lists, matrices, equations and single inequalities.

Factored Form is a test which considers the syntactic form of an answer. This test establishes that the expression is factored over the rational field. Note that

establishing that an expression "is factored" is significantly more subtle than a simple comparison with the result of applying the CAS's `factor` command to the Teacher's Answer. Related tests establish that an expression is *expanded* or in *partial fraction* form.

Calculus Tests are provided for convenience with certain commonly occurring questions. These check for generic mistakes such as missing constants of integration, or the wrong process used (integration vs differentiation). Such tests can be supplemented by checking for errors specific to the given question.

Numerical Tests are designed to work with floating point numbers.

Each Answer Test is equipped with a test suite to illustrate the exact behaviour of the test through carefully selected examples. These cover the typical, boundary and "difficult" cases with which the test must cope. This is available to the teacher through the online documentation. STACK allows new tests to be created in a reasonably straight forward way, although this requires knowledge of system development.

4.4 Tension between flexibility and consistency

There is a tension between flexibility and the simplicity of the question authoring process. At one extreme it should be possible in each question to alter every system feature. At the other, questions need to be authored with a maximum efficiency. The question author should enter only the minimum information, default values should be assumed where left unspecified. Our solution is a cascading options structure. Options assume a default value at the system level which can be overridden at the subject level, then quiz level, or again at the question level. Although we have implemented a three level "subject", "quiz", "question" hierarchy, the cascading options structure is flexible enough to cope with any navigation system which is essentially a tree structure. When authoring a question or quiz, teachers can use the default values to create new questions and quizzes quickly and consistently. Allowing teachers to change option values at a subject or quiz level provides maximum flexibility and control. Hence, the options force a particular question to behave in quite different ways depending on the values set in the context in which a student sees it. Options include the following.

The *input method* can be selected from combinations of a strict CAS syntax, more liberal syntax (e. g. [10]) or a graphical input tool. A *syntax hint* can also be given which allows the teacher to provide part of the answer. For example `x^2+?*x+1` or `matrix([?,?],[?,?])`.

Display can be controlled through an option, allowing for later integration of MathML, with fine tuning of whether $\sqrt{-1}$ is an i or j. The langauge can be specified, although STACK has only been partially translated into Spanish and Dutch, the mechanism for this is fully in place.

Various options exist to control response processing, including the number of marks available for a question and how marks are calculated for repeated attempts.

Options can be added by the developers in response to the needs of users.

4.5 Response processing

Response processing is the means by which the student's answer is evaluated and outcomes such as feedback assigned. Whilst we have a two phase mark/validate protocol we reserve the phrase *response processing* for the semantic evaluation only.

The simplest response processing scheme compares the student's answer to the teacher's answer using a single Answer Test (e.g. algebraic equivalence). The outcomes are simply a numerical mark, together with generic feedback such as "correct answer, well done", which is a STACK option. This allows the teacher to rapidly create questions with access to the documented answer tests. However, the benefit of CAS-enabled CAA is the immediate feedback which can result from more sophisticated distinctions which are established by using the various answer tests. To minimize the need for the teacher to write CAS code we have developed an abstraction layer termed the *response processing tree*, consisting of an arbitrary number of linked nodes we call *potential responses*.

Before we illustrate the response processing tree we discuss automatic evaluation of answers to question (1), in which students are asked to integrate nx^m for randomly chosen n and m. As before, we assume $p := nx^m$ and now assume that the student's expression is assigned to the variable sa. Even for this simple question the response processing is already quite complex. One approach is to establish equivalence of the derivative of the student's answer with the variable p. If equivalence is established we then consider in more detail whether the student has used a constant of integration. The CAS "int" command does not automatically generate such a constant, but it is also possible that the student typed a number or other expression and so care is required. If the equivalence $\mathrm{diff}(sa, x) \equiv p$ is not established in this first test we check if the student differentiated instead, by comparing sa with the derivative of p with respect to x. This is a mistake which students are apt to make, and we can provide feedback if appropriate. If we have not established that the student differentiated, we could consider other likely mistakes, such as those identified by [11] for questions of this type, where the most likely mistakes are, in order,

$$\frac{n}{m}x^{m+1}, \quad n(m+1)x^{m+1}, \quad \frac{n}{m+1}x^m.$$

In each case feedback and partial credit could be specified. If none of the tests reveal what the student has done, we might supply some "generic feedback" such as the following

> The derivative of your answer should be equal to the function you were asked to integrate. However, the derivative of your answer with respect to `x` is `\[ @diff(sa,x)@ \]` so you must have done something wrong.

Notice that this feedback is generated by operating mathematically on the student's answer: something which would be difficult if not impossible to achieve without the support of a CAS. By providing some indication as to where he or she may have gone wrong, we provide a much greater incentive for the student

to retry the question immediately, as opposed to simply stating the answer is incorrect.

While there is no claim that the feedback suggested is optimal it will serve to illustrate many of the important features of the response processing and feedback generation process. Note that to implement this feedback as code would require careful nested conditional statements which, in a traditional programming language, would be difficult and time consuming to write.

Our response processing tree consists of linked potential response nodes. Each of which may be traversed only once thus preventing infinite loops, (so that technically we have an acyclic directed graph). Each potential response node provides a mechanism by which two expressions can be compared using a specified Answer Test. Depending on the result, either the TRUE or FALSE branch is executed. Each branch has the opportunity to do any of the following.

1. Adjust the mark and penalty for this attempt.
2. Generate and add specific feedback.
3. Generate and add a specific answer note, used by the teacher.
4. Proceed to another nominated node, or end the process.

This allows a variety of different functions to be accomplished, including: (i) comparing the student's answer with the correct answer; (ii) comparing the student's answer with an incorrect answer derived from a common mistake, (iii) a test to award partial credit, (iv) to catch a common slip, and remove any penalty for this attempt.

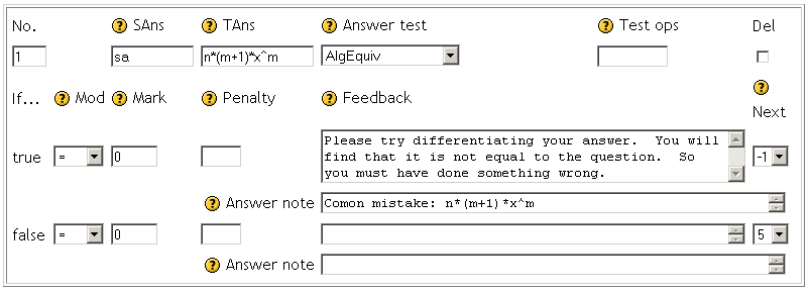


Fig. 2. The web interface for one potential response node

STACK provides a web interface for specifying the details of each node, and the corresponding TRUE/FALSE branches. An example is shown in Figure 2.

By allowing the response processing nodes to proceed to another question rather than simply another node, it would be possible to produce a sophisticated adaptive learning environment. This is the approach taken in the Wallis CAA system of [6], developed independently from STACK. This scheme could be further extended to provide CAS-based support for revealing steps in ques-

tions, such as those of [1], which allow students who have answered incorrectly to provide evidence of their intermediate steps and working.

The student's answer can also be manipulated by the CAS prior to traversing the response processing tree using the Feedback Variables. For example, we might define the variable `q:=diff(sa,x)`, and use this new variable q both in the first Answer Test, and in the generic feedback at the end of the process (if needed). This gives the question author maximum flexibility, but requires a deeper working knowledge of the CAS itself.

Notice the increasing levels of complexity:

1. Minimum, "question", "teacher's answer", "answer test".
2. Potential response tree, using more than one test or comparing the student's answer with more than one expression. Better Feedback.
3. Random versions using simple CAS commands.
4. Manipulation of the student's answer with the CAS prior to traversing the potential response tree.

We are, of course, limited in the feedback we can provide if we do not already possess a good understanding of students' common mistakes and misconceptions. In order to support the teacher in determining the misconceptions present in their group of learners STACK includes CAS supported analysis tools. These aid the teacher in tracking the progress of an individual student, analysing all responses to a particular question, and so on. Teachers use a web form to search by various combinations of Username, Question ID, Question Note, Action Requested (validate/mark), Valid (true/false), Marks Obtained, or by a specific time range. Here the Question Note and Answer Notes can be used to group answers according to categories which are relevant to the particular question. These tools are necessary to allow the teacher to track students' progress and reflect upon the learning of their group of students.

5 Interoperability

Interoperability between CAA systems is an issue that requires specific attention if their mainstream use within education is to be achieved. The Question and Test Interoperability (QTI) specification [2] is one model for representing question and test data and their resulting reports. However, the QTI standards concentrate on general teacher provided answer question types and do not cater well for the subject needs of mathematics. Various extensions of QTI, such as those of the *Serving Mathematics in a Distributed eLearning Environment* project, termed MathQTI, extend the QTI standard rather than developing a new standard. The features of STACK which make it innovative and useful in mathematics cannot be described within the QTI framework or such extensions, and so currently STACK uses a format of its own.

The IMS Question and Test Interoperability specification views an assessment item as being entirely self contained. Our option structure is at odds with this since we assume a cascading system. To maintain the teacher's intentions

it is necessary to export complete subjects or quizzes, rather than individual questions. We believe that such packages better represent the useful level of granularity in learning and teaching of mathematics, and so have implemented XML schemas for the export of quizzes in addition to single questions.

5.1 Technical details

As an authoring system we have relied on the typesetting language LaTeX so that question authors can write mathematically rich text using a reasonably efficient linear syntax, rather than a potentially cumbersome equation editor interface. Teachers write LaTeX into which CAS commands can be embedded between two `@` symbols in a way analogous to the use of `$`. LaTeX code is also produced by the CAS and then translated to HTML in a batch process using the application TtH. The HTML code does not contain graphics; it uses tables and characters from the symbol font to display the mathematics appropriately. This is still not a completely satisfactory solution, as it disadvantages students who rely on screen readers. We addressed this by providing a range of display alternatives through the options structure. These include plain text, MathML (although this is not currently fully supported) and LaTeX source code. Adopting LaTeX does require question authors to learn this syntax together with that of the CAS. In higher education mathematics, however, LaTeX is the standard typesetting system and hence is preferred.

STACK is a combination of the CAS Maxima, the web scripting language PHP and the database MySQL. STACK is available under the GPL license. The URL `http://www.stack.bham.ac.uk` points to a demonstration server.

6 Conclusion

Taking advantage of a CAS to support CAA provides access to a rich library of mathematical functions, allowing the objective evaluation of mathematical student provided answers. The CAS can be used to generate structured random question versions for each student, which supports a pedagogy of practice and can be used to encourage informal group work. Feedback, tailored to both the syntax and semantics of a student's typed answer, can be provided virtually immediately and used by the student to reflect upon his or her answer, and as an incentive to retry the question where necessary. We have developed a novel mathematical question type that encapsulates these features and implemented this as the STACK computer aided assessment system.

For many staff members, the ease of the question authoring environment is an important factor in determining whether a particular system will be adopted. We supported the question authoring procedure through (i) the options in order to achieve a balance between consistency and flexibility; (ii) the web-form for question editing, e.g. to specify answer tests and feedback; and (iii) the response processing tree. These help reduce the requirement for the question author to

become a computer programmer. A clear direction for future research is to expand the response processing scheme to support adaptive learning through the revealing of steps.

While the STACK system enables the full range of CAS features to be used, it is not in principle limited to a particular CAS implementation. We also highlighted that the areas of actual CAS usage are minimal, and confined to the tasks of instantiating pseudo-random variables and evaluating answer tests. As CAS tools have been developed to support research mathematicians an important future avenue of research will be to assess the necessary features of a CAS for CAA use and the distinctions it needs to make to support the teaching and learning of mathematics at various educational levels.

Acknowledgements

This work was part funded by the United Kingdom Higher Education Academy and a grant from the Joint Information Systems Committee.

References

1. H. Ashton, C. E. Beevers, A. A. Koraninski, and M. A. Youngson. Incorporating partial credit in computer aided assessment of mathematics in secondary education. *British Journal of Educational Technology*, 2005.
2. IMS Global Learning Consortium. IMS question & test interoperability specification. `http://www.imsglobal.org/question/`, 2004.
3. M. Fujimoto and M. Suzuki. *Infty Editor — A Mathematics Typesetting Tool with a Handwriting Interface and a Graphical Front-end to OpenXM Servers*, volume 1335 of *Computer Algebra – Algorithms, Implementations and Applications*, pages 217–226. RIMS Kokyuroku, 2003.
4. J. Grabmeir, E. Kaltofen, and V. Weispfenning. *Computer Algebra Handbook*. Springer, 2003.
5. S. Klai, T. Kolokolnikov, and N. Van den Bergh. Using Maple and the web to grade mathematics tests. In *Proceedings of the International Workshop on Advanced Learning Technologies, Palmerston North, New Zealand, 4–6 December*, 2000.
6. M. Mavrikis and A. Maciocia. Wallis: a web-based ILE for science and engineering students studying mathematics. In *Workshop of Advanced Technology for Mathematics Education in the 11th International Conference on Artificial Intelligence in Education*, pages 505–512, Sydney, Australia, 2003.
7. E. R. Michener. Understanding understanding mathematics. *Cognitive Science*, 2:361–381, 1978.
8. L. Naismith and C. J. Sangwin. Computer algebra based assessment of mathematics online. In *Proceedings of the 8th CAA Conference 2004, 6th and 7th July, The University of Loughborough, Birmingham, UK*, 2004.
9. P Ramsden. Fresh Questions, Free Expressions: METRICs Web-based Self-test Exercises. *Maths Stats and OR Network online CAA series* `http://ltsn.mathstore.ac.uk/articles/maths-caa-series/`, June 2004.
10. P. Ramsden and C. J. Sangwin. A liberalised mathematical syntax for computer-aided assessment. In *Proceedings of the International Mathematica Symposium, Perth, Australia*, 2005.

11. C. J. Sangwin. Providing Feedback to Students' Assignments. In *Proceedings of British Society for Research into Learning Mathematics, November 15, The University of Birmingham, Birmingham, UK*, pages 55–60, 2003.
12. C. J. Sangwin. Assessing mathematics automatically using computer algebra and the internet. *Teaching Mathematics and its Applications*, 23(1):1–14, 2004.
13. C. J. Sangwin. On Building Polynomials. *The Mathematical Gazette*, November 2005.
14. N. Strickland. Alice interactive mathematics. *MSOR Connections*, 2(1):27–30, 2002. http://ltsn.mathstore.ac.uk/newsletter/feb2002/pdf/aim.pdf (viewed December 2002).
15. A. Watson and J. Mason. Student-generated examples in the learning of mathematics. *Canadian Journal for Science, Mathematics and Technology Education*, 2(2):237–249, 2002.
16. M. Wester. *Computer Algebra Systems: a Practical Guide*. Wiley, 1999.

Semantic Search in LeActiveMath

Paul Libbrecht, Erica Melis
DFKI, Saarbrücken, Germany

Abstract. The web, as we experience it nowadays, is heavily based on search engines such as Google or Yahoo!. These engines are the essential step to discover web-content that would, otherwise, only be available after too many clicks. The field of computer-science which serves as their theoretical basis is information retrieval. However, the main focus of information retrieval is on textual content, that is words and sentences. Little research has been done, however, both in terms of research or tools, for information retrieval regarding *mathematical* content on the web as can be seen, for example, in the overview [M05].
In this article, we present work done for the LeActiveMath learning environment which stores and presents semantically encoded mathematical content. We have adapted information retrieval techniques to this semantic content in order to offer to learners reasonably tolerant searchability for text, metadata, and formulæ. Our efforts follow information retrieval principles stating the essential needs for fast response and easy query inputs.

1 Introduction

ActiveMath is an intelligent web-based learning environment for mathematics, it presents semantically encoded mathematical documents to learners, and allows them to practice by doing interactive exercises.[1] ActiveMath uses a fine-grained knowledge representation of mathematical documents based on the `OMDoc` encoding [OMDoc1]. Using it, the learners' competencies can be modelled and adaptive behaviours, such as the choice of content needed to achieve a learning goal, can be provided. The EU project LeActiveMath, Language Enhanced ActiveMath, developing a larger framework and content collection around ActiveMath. Within this project, the plain-text search engine available earlier in ActiveMath has been refined to bring semantic search capabilities to users which is the focus of our article.

Requirements analysis done among the partners of the project, including representative stakeholders of the publishers' and teachers' communities, has indicated that:

[1] More information about the ActiveMath learning environment can be seen from the project's home page: `http://www.activemath.org/`.

- the search tool of LeActiveMath should be very easy to use but should allow querying for text, meta-information, as well as mathematical formulæ.
- the search queries should be *reasonably tolerant* as one expects frustration to arise in a tool doing simple exact matches
- the search tool of LeActiveMath should offer access to all mathematical content thus supporting explorative learning

We expect the search tool to be used by several types of users of LeActiveMath who differ in their expectations and proficiency:

- the beginner learner uses it only for the purpose of searching quickly for a reminder or to discover new content. He will search mainly for words and for formulæ only if he can copy and paste them. The search interface and presentation should be simple and straightforward for him.
- the advanced learner may want to see details about each item and maybe search variants of presented item and search for mathematical formulæ or items' characteristics.
- an author may want to see all items, including the description of OpenMath symbols, and see details about each.

1.1 Usage of LeActiveMath Knowledge Representation

LeActiveMath stores and presents content encoded in the `OMDoc` xml-language added with metadata for educational purpose [LeAM-D6]. It is built on content split in units which are called *items*: definitions, theorems, exercises... Items are addressable by unique identifiers. Items have a granularity well-suited for our search purposes: they are easily identifiable and recognizable by a user as a separate entity and can be, partially, taken out of context.

Therefore the search engine searches for items and present results one item at a time, including relations from this item to other items.

1.2 Information Retrieval for LeActiveMath

Following classical information retrieval vocabulary [vR79], `OMDoc` items are defined as the *documents* of our search function. Based on the *resonable tolerance* requirement, we expect the search results to be often too numerous to be easily overviewed. Information retrieval indicates that a good paradigm is to provide search *ranking* where a score is computed for each matched item indicating how *relevant* the match is and to present search results from highest score on; additionally users are more likely to experiment with search and get results quickly than take the time to wait for quality search-results [vR79].

Information retrieval has been mainly focussing on the retrieval of text documents and on textual search. Two essential ingredients are put to use:

– An *analysis* or *tokenization* process which converts the text documents that will be searched in a sequence of *tokens*. Tokens are, typically, words, but these may be the result of a translation which, for example, removes the "s" character of plural words in English.
– An index which stores the occurrence of tokens in documents and can be efficiently searched for. Queries are formulated as the search for documents where tokens (results of the same analysis process) occur.

2 Prototype Description

The LeActiveMath search is a prototype that is part of the LeActiveMath learning environment. In this section, we describe the essential ingredients of the search engine.

2.1 Core Index

We have chosen the Lucene library[2] to maintain the core index. This library is a recognized open-source library and is in use in many industrial strength systems (such as Wikipedia or Technocrati). It provides storage and indexing of documents and high-speed queries on this index delivering the documents along with query-match ranking. The Lucene library is also the base of the content storage engine of the current ActiveMath *LuceneMBase*.

In our search, the documents are the items of the content. To build the index, we process the following information for each item:

– the titles
– the metadata information
– the textual content
– the mathematical formulæ in the textual content

This section describes how the index is built based on this information, that is, mostly, how texts in various languages and mathematical formulæ are tokenized. Along this description, the possible low-level queries are presented. It presents an overview of how each query is *boosted*, that is, is given a weight, in order to enter the computation of the rank of a found document, and concludes with an example illustrating all aspects.

[2] See `http://lucene.apache.org/`.

Storage of metadata information Each `OMDoc` item can have a metadata element. The metadata element is the place to store *attributes* of each item. For example, the field of study for which this item is intended (e.g. physics, law, ...) or the time one expects a learner will need to read the item. Metadata attributes as a list of name-value pairs. Documents can be queried for using the same name-value pairs.

The metadata of `OMDoc` items is also the container of relations between items, for example, the fact that a definition depends on a proof. Since the content storage of ActiveMath already has query facilities for these relations, the index does not consider them.

Tokenization of text Following classical retrieval [vR79], the tokens are made of words of the text; these words are, before, converted to lower-case, *stemmed*[3], and too common words are removed. This tokenization is language specific, we use the classical stemmers of Porter [Por05] which was written for English and generalized to many other languages. To respect the various languages, we encode the token-stream of each language in a different field of the index. To support different ranking for matches in the title compared to matches in the text, these two fields are indexed separately.

Towards Fuzzy Matching of Text In order to support learners in their searches one has to cope with words that are misspelled and to provide results that were approximately matched (fuzzy matching).

Fuzzy matching can be done with the index data described above: the Lucene library offers fuzzy matching based on the *edit distance*, that is, it allows elementary modifications of the token's characters (add or remove a letter, permute two, ...) which is then matched with lower score. The results of this form of fuzzy matching yields sensible results most of the time, but sometimes leads to surprises such matching "class" when searching for "flash", It is well suited to match words that have been misspelled, either at query or authoring time.

In order to provide an alternate form fuzzy matching of text, a phonetic tokenization of the text is stored using the metaphone phonetic algorithm. This phonetic-tokenization algorithm is an enhancement of the original Soundex algorithm of Russell and Odell typically used in spell-checkers, it translate words having the same *sound* to the same tokens. The libraries we have currently found work for English and German, and will be tested for their applicability in Spanish.

[3] The action of stemming a word is to take it to its root so that declinations of the same words end up being the same token. For example, *groups* is stemmed to *group*

Tokenization of mathematical formulæ As any information retrieval library, Lucene understands linear sequences of tokens. One wishes, however, to query the mathematical formulæ with their structure. ActiveMath uses the OpenMath standard [OM2] which organizes mathematical objects as trees of symbols and applications. For the mathematical formulæ in texts, the sibling order of xml-tree-walks produces a sequence of tokens that no sentence could produce. The anlysis process tokenizes the OpenMath-application with a depth indication, as well as the symbols, strings, floats, and integers of OpenMath: For example, the formula $\sin x^2$ becomes:

```
_(_1                <OMA>
_OMS_transc1/sin     <OMS cd="transc1" name="sin"/>
_(_2                   <OMA>
_OMS_arith1/power        <OMS cd="arith1" name="power"/>
_OMI_2                   <OMI>2</OMI>
_OMV_x                   <OMV name="x"/>
_)_2                   </OMA>
_)_1                 </OMA>
```

Using this tokenization, we can query exact formulæ by an exact phrase match, that is, a match for a sequence of tokens. As a given expression can occur at any depth of a mathematical expression, exact phrase queries have to be expanded as a disjunction of queries for each depth.

formulæ with blanks can also be queried for: the example above would be matched by a query for the following sequence of tokens expressing the search for the sine function applied to any argument would query for the tokens

```
_(_1 _OMS_transc1/sin * _)_1
```

where the * indicates a blank in the phrase query which matches anything as far as the remaining part is matched.

2.2 From User Queries to Ranked Matches

We have described how the index is built from tokens and how it can be queried and matched. Let us summarize how weight is assigned to fields so that those matches that we expect are the *most important ones* are given the highest rank. Queries will be made for text, for mathematical formulæ, and for metadata attributes. These user-level queries are translated to disjunctions of index-level queries each being given a boost-factor which influences the overall score of matches. The latter is used to order the results.

This heuristics is prepared for LeActiveMath and may be tuned depending on the results of the evaluation. The list below describes boost-factors of each query-types which get multiplied if a match of the same query occurs.

- textual and mathematical matches are expanded into queries in title and in text: a match in the title of an item count twice as much as matches in the item's text.
- exact text-matches and exact formulæ matches have factor 2.0
- metadata and keyword matches are boosted by 50
- fuzzy phonetic matches are slightly less boosted by a factor of 0.8
- formulæ matches with blanks have a boost decreasing with the *length* of the matched blanks
- fuzzy matches with edit distance are boosted depending on the amount of changes (so that a single change, which is probably a typo, yields a match with a score close to an exact match whereas a radical change yields a score close to zero)

2.3 Example Tokenization of an Item And Related Queries

We present a small example of an exercise with English text and title along with a formula:

Trigonometric exercise *Let us assume* $x < y$.

Indexing decomposes the content of this item in the fields `title-en`, `text-en`, and `text-phonetic-en`:

```
attr:              type:exercise
title-en:          trigonometr exercis
text-en:           let us assum _(_1 _OMS_relation1/lt _OMV_x _OMV_y _)_1
text-phonetic-en : LT US B ASMN
```

With this content in the index, the following queries can be performed:

- textual query: if the user enters "trigonometry", tokenization of the user-query converts this word, among others, to a query for token "trigonometr", which is exactly matched to the title yielding score 10.0.
- textual-fuzzy: user enters "asuming", tokenization converts it, among others, to a query for the token "ASMN" in the *text-phonetic-en* field which is matched to the content of the phonetic english field (with score 0.8)
- metadata query: a query of the user for the type exercise would be reformulated as an index-query for the token `type:example` in the field `attr` which is matched to our item with score 1.0.
- mathematical query: if the user queries for the formula $x < y$, the query is translated into a query for
  `_(_1 _OMS_relation1/lt _OMV_x _OMV_y _)_1`
  in the field `text-en` which is exactly matched to our formula yielding score 1.0.

2.4 The Search-Tool

The search tool uses the pre-processed content and search techniques described in the previous section. We now present how the search facility is offered to learners using ActiveMath and the ease of use of the search user-interface.

Search input The search-tool can be activated by the input of search-words in the text-field placed for this in the menu. This produces the results in the search-window. Such a search produces the default queries: fuzzy matching (both phonetic and edit-distance fuzzyness) of text in the learner's language for concepts in the current book.

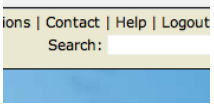


Fig. 1. search-field in the menu

The search-window enters the *plain* search mode, a mode where the search is displayed as a single string as in classical web search engines.

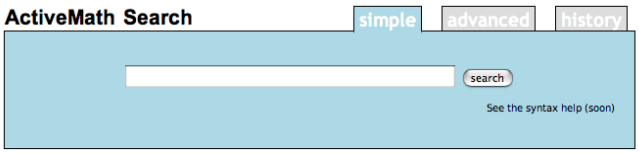


Fig. 2. The plain search, seen when first opening the LeActiveMath search

In addition, a more elaborate syntax can be used to require or exclude some word, change language, or query characteristics of the items.[4]

Searches can also be input using the *advanced search* (see figure 3) form which allows a combination of queries for:

- text queries with or without fuzzyness and exact phrases, input within a text-field
- mathematical expressions input using the Wiris input editor[5] which allows graphical input of formulæ as well as allows copy and paste of mathematical formulæ from the content.
- item characteristics as can be found in the metadata of each items, entered using pop-up menus.

[4] More details on the query-syntax, mostly intended to authors or advanced users, can be read online in the plain-text search-mode.
[5] More about the Wiris input-editor can be read from `http://www.wiris.com/`.

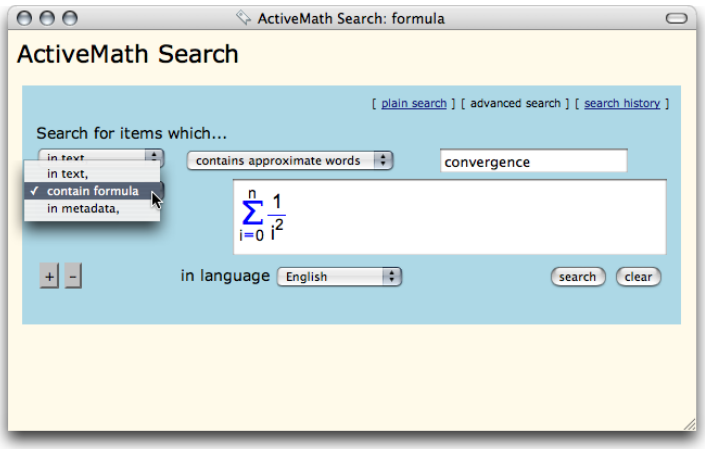


Fig. 3. The advanced search form illustrating a combinaed mathematical formula query and metadata query.

Search result presentation The results, sorted by the scores provided by the matches, are presented by title which, once clicked, show the full presentation of the item. Mastery-bullets indicate the estimated learner's *mastery* for this item. Links allow the learners to go to other search pages.

Only a first page of search results is presented if more than 20 items are found. This measure is approximately the amount of items that can fit vertically on a screen and allows a fast presentation of search results to allow further explorations of the user.

Integration of relevant web-sources In order to enable search of other mathematical web resources and to compare results of the searches, the result presentation adds links to submit the same query to search engines and content collections such as the Google engine searching the Web[6], the Wikipedia collaborative encyclopedia[7], or and the MathWorld encyclopedia[8].

Currently, these links can only be presented if the query is textual since the sources do not support metadata or semantic mathematical markup.

This integration supports exploration and, at the same time, makes the user aware of the sources which will support the user's critical thinking in terms of trust and proficiency.

[6] The Google engine is at `http://google.com`
[7] The Wikipedia encyclopedia is at `http://www.wikipedia.org`.
[8] The MathWorld repository is an enterprise of Wolfram Research Inc. and can be reached at `http://www.mathworld.com/`.

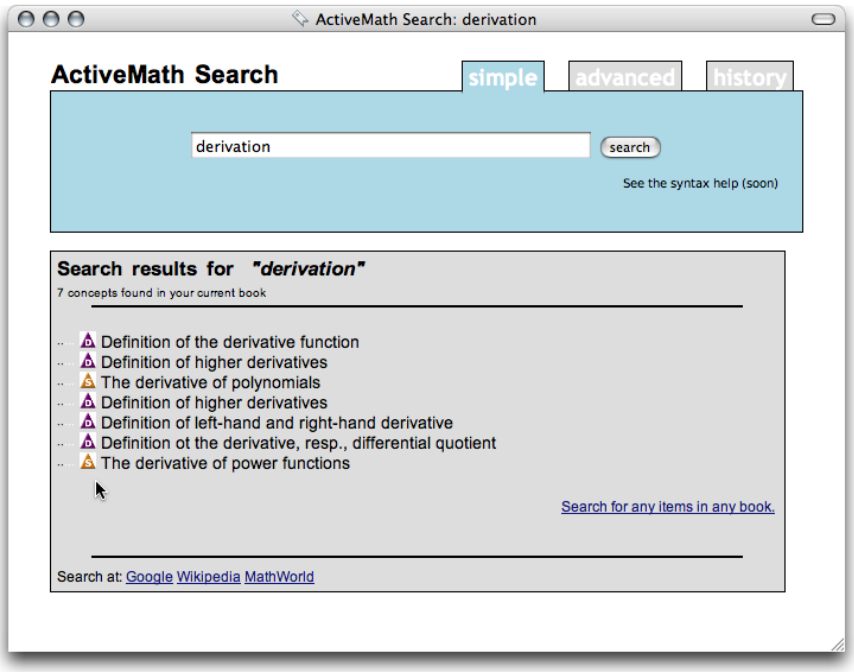


Fig. 4. Searching for derivation and selecting a concept.

Search history and search states Each search and all results obtained and browsed are stored in a search history. The LeActiveMath search engine tries to store all states of the search tool user interface. This includes the input-element under current focus in the search form, the search mode, the result page, the item being currently viewed and details about its view. Each of these states are numbered sequentially and can be re-invoked later, for example, once the search-window is closed then re-opened. The history view presents a user readable text of each query along with the titles of items shown. Each step can be restored and continued with. This contributes to make the LeActiveMath search similar in availability and memory to an opened dictionary on your desk: one can forget about it but taking it back will restore its last state.

Item display Once a link in the search result list is clicked the search tool displays single items aside of the search results. Single items display present the type, title, content of the item, the notes and *mastery* icons as well as the copyright link.

Items display is complemented with information from the metadata: in its simple form, the *for* relations from and to this items are displayed, for example *exercise*

for this item or *concept that this examples illustrates*. Pressing a *more* link opens a detailed view of the item's metadata including all relations from and to this items and all references to this item. The presentation of these relations allow the learner an explorative navigation of the rich structure of the content.

Item display LeActiveMath is using the presentation architecture of LeActiveMath [ULWM04]. That is, it uses the same rendering engine and the same notations, providing a consistent appearance of the mathematical knowledge.

3 Related Work

We briefly list the related work about the retrieval of mathematical documents in order to situate our research:

- the MBase [FK00] project has inspired both the search function and our content-storage. It offers a prototypical pattern matching of mathematical formulæ with variables being instantiated by any sub-expression: for example, querying $f(A, B) = f(B, A)$ as search for any commutativity statement. This project was based, mostly, on serial search along all OpenMath objects stored in the database which is not a very scalable approach. We have indicated in section 2.1 how our mathematical tokenization can match mathematical expressions with variables being replaced by subtrees; we have not achieved, however, the ability to compare two instances of instantiated variables (for example when querying for $f(A, B) = f(B, A)$ to match any commutativity statement requesting that the terms A and B be matched consistently). This form of query can be rephrased a query with joins and deep-equality. Experts in the field of XML databases, when asked about this form of matching seem to indicate no other strategy than serial searches.
- the MathQL project[9] who's goal is to search the CoQ-library has put forward great goals in their mathematical search such as the application of *unification* as opposed to *pattern-matching*. Such goals seem not reached yet.
- MoMM [Urb04] is a recent attempt at using formal rewriting rules to search the Mizar Mathematical Library. We have not been able to identify the feasbility of bringing such interreduction rules into the content of LeActiveMath.
- private discussions with the developers of the Digital Library of Mathematical Functions at the NIST institute indicate that search is being developed in their project. The search implementation seems to be based on a translation of the TEX sources of the library and is helped by a large number of heuristics providing a form of fuzzy matching.[10]

[9] MathQL is a subproject of the HELM project, for more information: `http://helm.cs.unibo.it/mathql/`.

[10] See `http://dlmf.nist.gov/` about the Digital Library of Mathematical Functions.

– The company Design Science Inc. has started an NSF-funded project on mathematical search and are currently running evalutations of tools where they can search MathML-presentation-encoded formulæ.[11]
– Paul Cairns has experimented with Latent Semantic Indexing in [Cai04] on the Mizar mathematical library. His study seems to carry interesting results with a very reasonable computational power.
– the thesaurus.maths.org project at Cambridge University has similar target audience than our project. It is to be noted, however, that the thesaurus only offer textual search having a TeX-based content-representation and semantic annotations only for the navigation between items.

4 Future Work

The LeActiveMath search tool provides a sturdy basis for information retrieval of `OMDoc`-encoded content. The search tool of LeActiveMath will enter in the evaluation phase of the project and will be polished and refined accordingly. Among others, the evaluation will measure the *understandability* of the search engine in comparison to other search engines.

The usage of latent-semantic-analysis as described in [Cai04] is probably worth following as much of the infrastructure that we have developed enables the models of vector-based occurrence-representation which is at the basis of this domain. The engine should measure items *close to* the queried document by using a distance based on co-occurrences of tokens, both textual and mathematical. Care has to be taken, however, as the latent-semantic-indexing process is patented and only available in a relatively impractical library.

The presentation of search results will be enhanced most probably. The simple display of mastery-bullets, item-types, and item titles may prove to be insufficient (for example, many exercises simply bare the title *Exercise*). We should consider avenues, such as the *result-gisting* approach presented in [CKS05] which display which other queries a given result-document could also match.

Graph-based navigation of the items' relations has been requested several times and seems to provide an intuitive representation of the navigation through the knowledge. We expect to embed such a navigation for learners.

[11] More about the search project of Design Science can be found `http://www.dessci.com/en/reference/searching/`.

5 Open Issues

We have not been able to assess how much interreduction of formulæ, as in [Urb04], or search-query unification as promoted by the MathQL project, could help a learner in his search. On the one hand, many rewrite-rules appear to be obviously needed (for example, applications of the associativity or commutativity). On the other hand most of these rewrites actually inject knowledge into the search which may surprise the learner. As an example, searching for the statement of the derivation of a given function could be, naturally, rephrased as the search for any function whose indefinite integral is the indicated function which may leave the learner quite perplex.

Finally we wish to stress the integration of the external search engines within the display of search results. They provide alternatives to users and also provide comparisons to them: one will be able to evaluate the quality of the search engines but also the quality of the searched content. Such resources as Wikipedia or Thesaurus.maths.org are more appropriate for the search-and-browse paradigm than the content of the LeActiveMath books since the latter were, mostly, written for a book usage. For example, in many tests we have made with single words, the corresponding search in the Wikipedia encyclopedia reached a *disambiguation* page which is a manually authored page branching to the many possible interpretations of a single word. There is no such item types within LeActiveMath knowledge representation yet. Experiments and comparisons will enable us to infer the essential differences.

References

Cai04. Paul Cairns. Informalising formal mathematics. In Andrea Asperti, Grzegorz Bancerek, and Andrzej Trybulec, editors, *Proceedings of 3rd Int. Conf on Mathematical Knowledge Management*, volume 3119 of *LNCS*, pages 58–72. Springer-Verlag, 2004. Available at `http://www.uclic.ucl.ac.uk/paul/research/MizarLSI.pdf`.

CKS05. Karen Church, Mart Kean, and Barry Smyth. Towards more intelligent mobile search. In *Proceedings of the conference IJCAI-05*, 2005. Available from `http://www.ijcai.org/papers/post-0135.pdf`.

FK00. A. Franke and M. Kohlhase. MBase: Representing mathematical knowledge in a relational data base. In F. Pfenning, editor, *Proc. 17th International Conference on Automated Deduction (CADE)*, Lecture Notes on Artificial Intelligence. Springer-Verlag, 2000.

OMDoc1.] Kohlhase, M. 2000 `OMDoc`: Towards an OPENMATH representation of mathematical documents (Seki Report SR-00-02). Fachbereich Informatik, Universität des Saarlandes. (`http://www.mathweb.org/omdoc`)

LeAM-D6. LeActiveMath Partners. D6: Leactivemath structure and metadata model. LeActiveMath Deliverable D6, The LeActiveMath Consortium, December 2004. See `http://www.leactivemath.org/index.php?id=4915`.

M05. Robert Miner. Enhancing the Searching of Mathematics, A position paper based on the proceedings of the Enhancing the Searching of Mathematics Workshop. June 8th, 2004. See http://www.dessci.com/en/reference/searching/math-searching.htm.

OM2. Stephen Buswell, O. Caprotti, D. Carlisle, M. Dewar, M. Gaëtano, and Michael Kohlhase. The OpenMath Standard, version 2.0. The OpenMath Society, 2004. Available at http://www.openmath.org/.

Por05. Martin Porter. The Porter stemming algorithm, 2005. See http://www.tartarus.org/~martin/PorterStemmer/.

ULWM04. C. Ullrich, P. Libbrecht, S. Winterstein, and M. Mühlenbrock. A flexible and efficient presentation-architecture for adaptive hypermedia: Description and technical evaluation. In Kinshuk, C. Looi, E. Sutinen, D. Sampson, I. Aedo, L. Uden, and E. Kähkönen, editors, *Proceedings of the 4th IEEE International Conference on Advanced Learning Technologies (ICALT 2004)*, pages 21–25, 2004.

Urb04. Josef Urban. MoMM - fast interreduction and retrieval in large libraries of formalized mathematics. In *Proceedings of the ESFOR 2004 workshop at IJCAR'04 available at http://www.mpi-sb.mpg.de/~baumgart/ijcar-workshops/proceedings/PDFs/WS5-final.pdf*., 2004. More information on the project at http://wiki.mizar.org/cgi-bin/twiki/view/Mizar/MoMM.

vR79. C.J. van Rijsbergen. *Information Retrieval*. Butterworths, 1979. Available at http://www.dcs.gla.ac.uk/~iain/keith/.

WaLLiS: a Web-based ILE for Science and Engineering Students Studying Mathematics.

Three years on.

Manolis Mavrikis[1] and Antony Maciocia[1]

School of Mathematics, The University of Edinburgh,
EH93JZ Edinburgh,
Scotland, UK
{M.Mavrikis, A.Maciocia}@ed.ac.uk

Abstract. This paper, after briefly describing the context and the problems faced by science and engineering students studying mathematics, presents WALLiS, a web- based Interactive Learning Environment which is being developed to address some of these problems. Keeping in mind pedagogy and the development of a learner-centred application, we first describe the iterative methodology which we employed, and theories (such as constructivism and contingent instruction) that influence the system's design. The paper also describes results from the system's pilot phases and its integration that yielded useful results in terms of adapting the system's behaviour and, more importantly, the feedback it provides.

1 Introduction

As we have described elsewhere [24] the main concern behind the development of a web-based system for mathematics was the growing concern of researchers and university teachers about the evident problem of the deficiency in mathematical skills amongst science and engineering students. As [17, 13] and other reports describe, although most of the problems have their origins at school, universities have to cope with several problems such as students' diverse backgrounds and levels and the fact that many of these students fail to recognise the importance of mathematics for their main degree. All these problems make additional support (tutorials, formative assessment etc.) difficult and, in conjunction with the increased intake of students, time consuming. In order to improve this situation our School is providing additional support by also adopting the Interactive (and intelligent) Learning Environment (ILE) WALLiS (the name of which is inspired by James Wallis; the 17th century mathematician).

Despite the fact that many developers and key players in the e-learning field realise the problems that students face, they often fail to notice the close relation of pedagogy and technology and, as [14] also suggests, that technology needs to be closely integrated with curriculum goals, related texts and teacher practices. It was along these lines that, during the design, implementation and integration of WALLiS, a specific user-centred methodology was adopted. The

following sections, before presenting the system itself, describe the methodology and theories that influenced its design so far.

2 Background

2.1 User-centred software engineering

It has long been argued that educational software and especially intelligent ones can only be effective when based on continuous refinement of system behaviour [2] and on careful user studies [14, 7]. Along this line of thought, there are several socio-technical [29] or user-centred approaches, which employ iteration phases [15] of software development instead of a more classical software engineering model. Extending this argument further, Clancey argues that 'computer systems as artifacts must be developed incrementally, with relatively quick periods of use, observation, reflection, and redesign ... in a context that includes the user's everyday adaptations' [3] implying that both students and teachers (or lecturers in our case) must participate from the very beginning of the development process.

One of the methodologies that takes all these considerations into account is Persistent Collaboration Methodology (PCM [5]). In particular, PCM provides a suitable framework for developing and using WALLIS as it is inspired by action research, which (despite the danger of producing results that are difficult to generalise) is the only transformational research method in education that looks at changes on the environment, addresses practical questions, and is appropriate when a new approach is to be integrated into an existing context. As the methodology's name suggests, it requires collaboration between students, teachers, researchers and technologists through all the phases of the 'observation, reflection, designing, action' cycle. Although, in our case, formal collaborations were not always possible, the phases were all realised with the cooperation of lecturers, experts in Human Computer Interaction or evaluation fields, and students. The other feature of PCM is persistence, which refers to a 'long lasting chance, not just in classroom materials but also in the evolving methods and beliefs of the collaborating partners' [5]. The above process is stimulated by a 'wheel' of tools (or techniques) which influence the design and development of a system, and theories of teaching and learning which influence designers' decisions with pedagogical aspects, which could otherwise be neglected. On the other hand, this stimulation can often occur both ways. This is what happened during the development of WALLIS as research in the Artificial Intelligence and Education (AI&ED) field strongly influenced its design but the initial decision to experiment with the application of such theories to computer-based applications came from previous observations (e.g. [22]) with similar technologies that sometimes lacked the appropriate feedback.

2.2 Theories influencing the development of WALLIS

Following a more constructivist view and recognising that learners must remain 'involved, active, and challenged to think about and learn the presented material' [2], we took the view that interactive activities would help bypass drawbacks

of passive learning. In addition, issues like intelligent feedback and help-seeking seem important and research has shown that, when coupled carefully, are very effective. In addition, recent results indicate the importance of emotions and motivation in human learning through educational software. Affective tutoring, in the sense of targeting student's emotional and motivational state, is an effective approach and needs to be further explored. One of the theories which addresses such issues to some extent and influences the development of WaLLiS is contingent instruction [10, 30] and consequently also scaffolding; notions proposed to describe exactly the need for a balance between the children's capacity to selectively ask for help and the teacher's effort to make her actions contingent upon activities of the individual learner. This is achieved by recruiting student's interest on the task, establishing and maintaining an orientation towards task relevant goals, highlighting critical features that a student might overlook, demonstrating how to achieve goals and helping control frustration ensuring that the student is neither left to struggle alone nor given too little scope for involvement and initiative in the task (see [30]).

An important issue that arises when one tries to teach in such a way, is to know when and how much help a student needs to complete a task. A possible solution comes from the learner's use of help-seeking to influence the tutoring process. It is exactly then that an adult (or a system) must intervene to promote further progress and this is where the students can play a role influencing the tutor's behaviour by their own help-seeking process. Of course there is an array of affective factors involved during this process and the AI&ED field is gradually recognising this and providing results. The reader is referred to [23, 25] for a more elaborate analysis of these issues.

3 The development of WaLLiS and its application

3.1 Initial Observations

During the initial observation period of the system's design, collaboration and expert consultation helped to expand on theoretical issues, investigate the technologies that the students were already using, and look into other technical problems. In addition, the designers were directly involved in the whole range of the teaching and learning activities — delivering tutorials and observing lectures – and so had an additional insight which is often not easy to acquire. This allowed exploration of common concepts that students find problematic, the kind of help that would be particularly useful for them, the problems that lecturers consider important to address as well as the notation they use, and their didactical approach. A final step of this phase was discussions about the systems' contents, its knowledge base, lecturers' attitude towards the system, and the use they would make of it.

3.2 Designing and developing WaLLiS

After the initial observation period a prototype was developed and appropriate material were authored to be able run some pilot tests. As mentioned before,

we decided that the materials would comprise, where applicable, interactive exploratory parts which would allow students to freely explore some aspects that cannot be covered in static pages. Based on Wood's theory of contingent instruction, we wanted students to be able to ask and receive help (see section 4 for details on the feedback mechanism). Previous observations made clear that pop-up windows were considered quite annoying [22, 24] and in the particular case of WALLIS, where student actions during the interactive parts would produce feedback interrupting the student's action, this approach was not feasible. Therefore we employed a feedback area at the bottom of the window where help and suggestions are provided (figure 1).

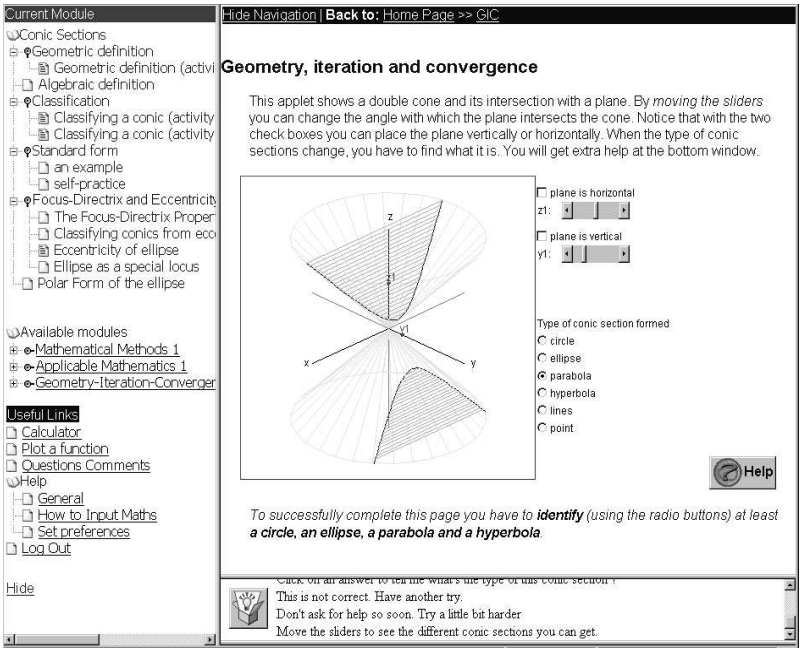


Fig. 1. The overall environment of WALLIS

Another design decision that helped us focus our research was that, since WALLIS is used in a specific context, the content has to adapt, at least for this phase, only to student preferences (colours, sizes etc.) and very broad aspects of their profile. With that we mean to the course they attend i.e. their context and their help-seeking process (see figure 2) rather than to different didactical styles or learning scenarios as is the case for other adaptive systems (such as ActiveMath [27]). Having a specific context in mind and ways of using the system in classroom, helped develop content in a specific (yet reusable) way and focus on providing more detailed feedback in the self-practice exercises. Moreover, from related literature (eg. [4]) it was evident that students are easily confused and

lost by many hyperlinks. This is a common problem in web-based environments which we currently address by having the system suggest further pages to study (see next section) and by using a a tree-like map of the contents (see figure 1) that changes according to the module students attend.

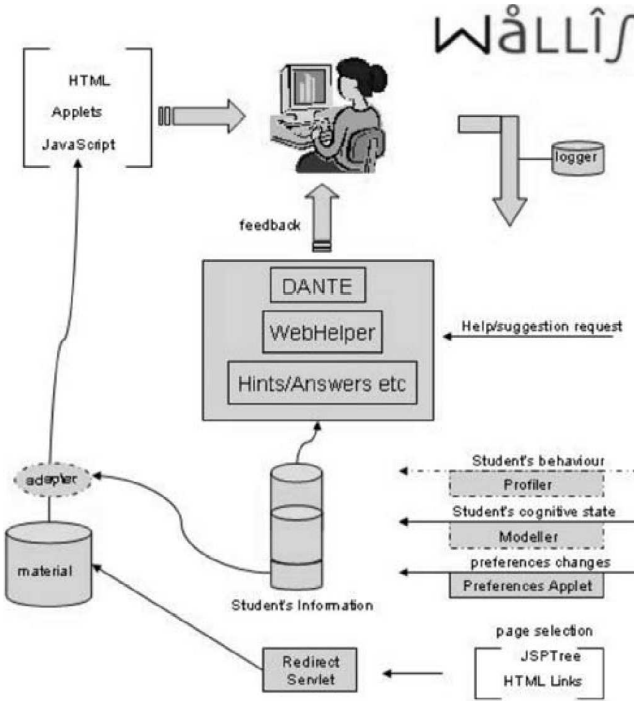


Fig. 2. The framework of WALLIS and how its components communicate

3.3 Adapting the environment to student needs

Through the iterative phases and the three years of its use we were able to adapt the feedback mechanism, set the misconceptions that it targets, and improve certain aspects of the system based on students' interactions and comments. With observers' help, we were able to conduct live observations of students' interactions. In addition, thanks to a very detailed log mechanism that records all aspects of user's interactions (see [23]), students were allowed to use the system as they wished and in their own time. The mechanism provided log files that could be statistically analysed but also replayed to facilitate recorded analysis of their interaction with the system. This (still on-going) research yields more and more valuable results. The following sections briefly describe some of these results and the approaches to target the observed problems.

3.4 Mathematical input and notation

Even from the testing of an early prototype, it was evident that students faced serious problems typing their answers in a linear format. It was considered very annoying for most of them, regardless of their level of competence and computer literacy. Similarly, they complained that too much effort was needed to quickly understand the linearly typed mathematics in the feedback frame. We believe that this factor strongly influences other problems such as the fact that some of the students, despite struggling with the chosen activity, were not paying careful attention to the feedback provided. Although one could argue that mathematics students should learn to type and understand a linear format, it is surely not the right context to do so. The cognitive load to understand and type in this format (which many mathematical learning environments have to employ) obstructs their learning of more important aspects at this stage.

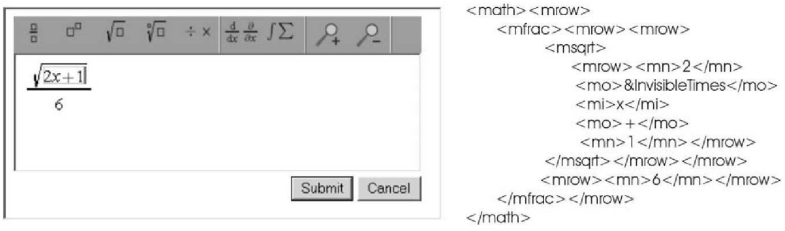


Fig. 3. Using WebEQ to input sqrt(2*x+1)/6 and its underlying MathML.

Based on the above, we made several changes at the initial prototype to deliver a more student friendly notation. For the input, WebEQ[1] was employed providing the ability to transform a student's input to MathML[2] which is then sent to a Computer Algebra System[3] where it is evaluated.

3.5 Adapting to students' interactions

Similar methodology was followed with other components of WALLIS. For instance, to make it easier for the students to use the navigation tree, after we improved some commonly reported problems, it was replaced with an adaptive

[1] WebEQ provides an equation editor applet (see http://www.dessci.com). For collecting research data we employed a modified version of WebEQ based on WebEQ SDK API adapted mainly to capture events and mouse movements in the editor. We would like to acknowledge Design Science and in particular Robert Miner for the licence, help and collaboration while developing it.

[2] http://www.w3.org/Math

[3] Over the years we have used both the commercial MAPLE (www.maplesoft.com) for which our department has a site license and the opensource YACAS (http://yacas.sourceforge.net/) as a testbed.

one that changes according to their preferences and the active concept. Several setups were presented to the students and they were asked to comment on these. In general, the qualitative comments collected [24, 1] were very fruitful, as they provided insight to different issues and proved how a user-centred methodology can help designers take decisions that are useful and effective for the students and not developed just because of the 'technology hype'. For instance, colour, size and other accessibility preferences where developed immediately while various ideas on the user interface (such as the ability to move components around) had to be dropped because they were not considered so important by our students. Similarly, an animated agent (ANA, [1]) despite being effective in other research was not adopted in the end because the relevant research (e.g. [1, 6]) did not provide appropriate evidence of an advantage (particularly for adult students).

Finally, some of the observed actions and particularly the log files helped verify certain things (such as their problems with the linear format) and other aspects that would have been otherwise neglected. In general, this process helped eliminating other factors that would interfere with research and development towards the main aim of the system; that of providing feedback. The following section covers specifically the issue of what content WaLLiS currently addresses, its feedback mechanism and how this was improved over the years.

4 Contents and feedback that WaLLiS provides

WaLLiS currently addresses various concepts such as functions (function domain, odd and even functions, slope and gradient for linear functions etc.), differentiation, integration and vectors by presenting the corresponding theory pages and employing (where appropriate and feasible) exploratory activities or interactive feedback-enabled exercises. In addition, a whole section of the 'Geometry and Iteration' course that deals with Conic Sections in detail has been authored and has been used over the years in several different setups from stand-alone to supportive and from formative to summative. As our main focus is on exercises and feedback we will not get into all the details of how content is authored and encoded. It is worth mentioning briefly that we use a data format which employs XHTML[4], MathML and OpenMath[5] and specialised elements to encode the different sections (theoretical parts and examples). Although it is true that we could have used OMDoc[6] for the internal representation of the content, OMDoc was only in its infancy when our project started and, in particular, the support for elaborate interactive exercises was very limited. In addition, because most of the available authors were finding difficult to write mathematical formulae directly in OpenMath and due to the absence of tools (such as QMath[7] that only appeared later) that could help us convert from a linear format to OpenMath, it was decided that the parts of the content, which are going to be presented to the

[4] http://www.w3.org/TR/xhtml1/
[5] http://www.openmath.org/
[6] http://www.mathweb.org/omdoc/
[7] http://www.matracas.org

student, will be encoded in content or presentation MathML. Fortunately, our context is specific and transformations to semantically richer formats is possible but not one of our immediate priorities[8].

On the other hand, choosing our own — perhaps less standard — XML format we were able to conduct research and elaborate on the representation of the interactive exercises; an issue covered in more detail in section 4.2. First, the following section describes the overall feedback mechanism that ties together exploratory activities (applets) and the rest of the web-based system

4.1 Feedback mechanism during exploratory activities

One of the components that WALLIS relies upon is its 'intelligence' that is provided by an adaptation of the feedback mechanism developed for DANTE [22]. This provides feedback during the exploratory activities presented in a microworld-like environment (built using JavaMath[9] and other in-house developed graphical objects). A threaded mechanism tracks the goals that the author of the activity sets and students have to achieve. The goals of the activity form a tree structure (see figure 4) where goals are comprised of a number of subgoals each of which has a number of completion conditions and misconceptions associated with it.

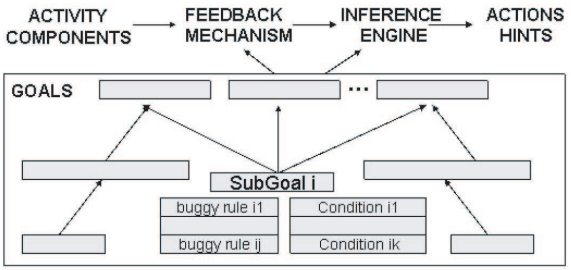


Fig. 4. The representation of DANTE's goals

The goals, depending on the activity, involve selecting an answer from a multiple choice question, putting objects into certain positions, giving numerical answers and so on. The feedback mechanism comprises production rules authored

[8] Notice that this applies to the static content that needs to be presented (hence MathML has proved enough for our needs) and not the response processing part of the interactive exercises which, as described in section 4.2, are encoded using OpenMath.

[9] JavaMath (maths.hws.edu/javamath) is a collection of graphical and other mathematical objects that help present but also validate input, calculate integrals, derivatives etc.

in Java (in old versions) and recently in JESS [10] (see [11, 19]). Along with these, there is appropriate feedback associated with each goal that students have to try, and other messages to be delivered when no action is taken or when they achieve a goal etc. More details about the feedback mechanism can be found at [22, 19, 12] but in general this follows an incremental hinting process that changes according to the time passed, the amount of help a student requested (incrementally demonstrating the answer if necessary), the current goal but also the student's achievement so far.

For example, at the simple exploratory activity seen at figure 1 students can explore conic sections. By moving the sliders they change the plane's coefficient that intersects with the double cone. Afterwards they have to choose from a multiple choice question the conic section produced. During their interaction, they receive feedback that helps them explore the activity more efficiently such as hints to move the sliders in different positions, prompts to provide an answer, to rotate the double cone etc. Similarly, when they select an answer (note that this is one of the goals for this activity) students receive feedback based on the preset rules (see figure 1 for a short example). Finally, when a student explicitly asks for help, depending on her current goal and level, the system provides feedback on how to achieve that. For example, *'Think first how many branches the conic section has'* or *'Drag and turn the cone to see it from a different position'* that would potentially, if the student keeps on failing, lead to the system providing an answer.

4.2 Self-practice interactive activities

As one can appreciate, the exploratory activities mentioned above are hard to author. Despite the script-like mechanism it requires a significant amount of expertise and understanding of the system to be able to develop and integrate the graphical components and to set the goals of the exercise. It was therefore considered necessary to have other kinds of exercises that students can use as a self practice that provide feedback and although they are not necessarily exploratory (in the sense of providing a microworld) they should still be interactive and at least provide adaptive feedback.

Therefore, after an initial template mechanism, which was used as a prototype of deciding possible and necessary types of exercises, an XML exercise language[11] was developed to facilitate the authoring process by encoding the different types of exercises, provide incremental hints and feedback on known misconceptions that the author can set in advance. Around the same time, a similar approach in the ActiveMath project was addressing the limitations of OMDoc by extending its format to represent similar kind of activities. The simultaneous efforts and similar needs lead to a redesign of the WALLiS format [26].

The latter, by separating content from presentation, is easily transformed to any desired output. In addition, the format lends itself to being completely separate from implementation and architectural issues. To present the exercise to the

[10] http://herzberg.ca.sandia.gov/jess/
[11] http://www.maths.ed.ac.uk/wallis/format/

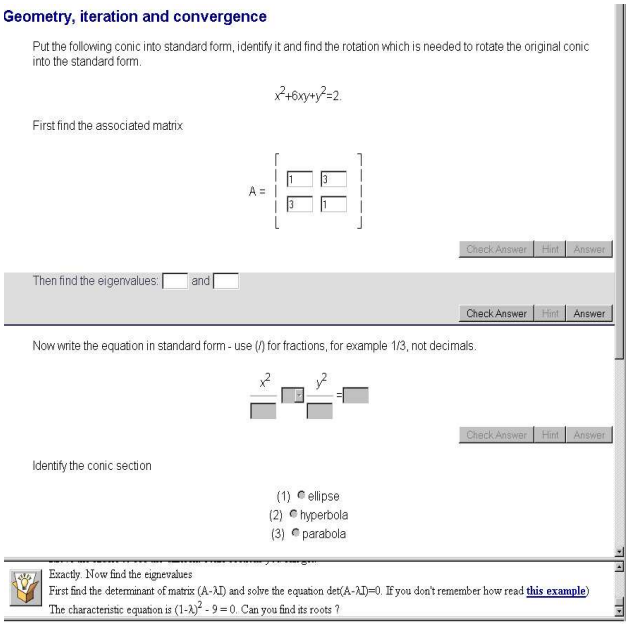


Fig. 5. A self-practice interactive exercise

learner a series of XSLT transformations result to an XHTML document which is presented to the learner. We will not elaborate here on transformation details apart from mentioning that the transformation results appear on a browser page that includes a form for the interaction (see figure 4), hidden elements (eg. the solution or the hints) and (since security and was not an important concern) embedded JavaScript that takes care of the interaction. For example, in figure 4, the learner completed one step correctly and just exhausted all possible hints that she could get, thus being permitted to get a possible answer to the question if she keeps on failing to respond correctly. In later version we investigate a more generic approach with a clear client - server separation.

4.3 Page-level suggestions

Apart from the elaborate feedback mechanism and the interactive exercises, another kind of feedback is provided to students when they are not sure what to read or which exercises to do. As mentioned before, the content is designed in a way that tries to avoid many links between items. In addition, the tree-like structure of the content and the navigation bar helps the students to progress through the content and the tree has appropriate icons to show them if they have seen and completed a page or not. Nevertheless, some students were still getting lost, following some random path in the way and order they interacted with the exercises. Therefore a similar tree structure of the goals as the one

described in section 4.1 comprises the targets that the students should achieve, the pages that they have to complete and what they should do in each one in order to complete it. These are all described in an XML file which the system uses to determine if a student has completed a goal and to know what the next goal is that they have to achieve. This way, when they are lost and they ask for help (from the help button at the left of the feedback pane of figure 1), they are given a suggestion of what they should do next.

In addition. through the log files and observations, another insight into their navigation patterns and their interactions with the activities was that a high proportion (56% on average) of cases in which the interaction with a page was not satisfactory completed. By observing more carefully the students' interactions, it was clear that some of them just wanted to see what certain pages involve and then immediately left (but some come back later), and others left the page because they often did not know the answer to a question and they were facing difficulties completing the activity. Recognising the complexity of this issue and to elicit more information from the students themselves, a simple prompt was added that was asking students to remain at the page and ask for more help or read the theory or example if they haven't done so before. This simple mechanism served the purpose of making students reconsider abandoning the page, thinking more explicitly about this process and helping us get more results during the interviews. After enabling it, the abandon rate dropped (14% in average) and we were able to separate students into several groups that manifested certain behaviour and randomly interview some of them. Interviews from a group of 126 students who used the system to study a subject that could not be covered during normal lecturers showed that students quit pages either (1) because the goal of the activity and its relevance to the assessment was not obvious to them, or (2) because they were successful in earlier parts of the activity and felt really comfortable with the rest, or (3) because they were dissatisfied with struggling without managing any of the goals and wanted either (3a) to try finish some other time, or (3b) go to the appropriate page where theory is covered and come back, or (3c) just never came back probably because they were completely demotivated.

All these reasons involve affective characteristics (see more details in [25]) and if taken into account more tailored suggestions can be found. Based even on these simple observations, it is clear that even the prompt that made students think more explicitly about their actions helped them at least complete the activities. Similarly, the suggestion mechanism for pages was also improved to make the reason behind this suggestion clearer. For example: *'Try the Classifying Conics page to practice classifying conic sections by finding out which graph best fits the given equation'.*

5 A note on data formats and interoperability

At the same time that WALLIS and the aforementioned solutions were implemented, the IMS Question & Test Interoperability (QTI) [16] specification ap-

peared as a means of providing a data model for the representation of questions and their corresponding results reports. It was soon made clear that there is a growing interest among the community to reach a common consensus, a standard that different interested parties can use to enable the exchange of items, assessments and results data between authoring tools, item banks, learning systems and assessment delivery systems.

Unfortunately, as it has been discussed in several meetings and workshops of the community [28, 18, 21] and mailing lists[12], [13], QTI (especially its first version) lacks support for mathematics. Through the first author's work in the 'Serving Maths Project'[14] QTI was extended to address some of the most important limitations. It was particularly important to take into account that most answers to mathematical questions, more often than not, involve mathematical entities which need to be understood by the system, processed and evaluated appropriately. In addition, we needed to take into account the overwhelming preference of teachers to establish that the student's own answer satisfies certain (mathematical) properties rather than just conform to some specific answer. The reader can read more about MathQTI in [20].

After some extensions of QTI we were able to transform from and to the WALLIS format and hence make a step closer to the ideal realm of sharing exercises. We have already mentioned OMDoc and the fact that that it is possible to transform both our content and the exercises to OMDOc and to its extended version as used in ActiveMath and LeActiveMath. Of course there are other data formats and similar systems like AIM (http://maths.york.ac.uk/moodle/aiminfo) and MapleTA (http://www.maplesoft.com/products/mapleta) (which unfortunately do not use an XML format) but a text based one which is largely based on Maple's language, or STACK (http://www.stack.bham.ac.uk/) and MathDox (http://www.riaca.win.tue.nl/) and many others[15], which are worth sharing and exchanging content with. A more detailed discussion of this important interoperability issue can be seen at [8].

In addition, now that the breadth of IMS specifications has grown and work on Content Packaging (CP), Simple Sequencing (SS) and most recently Learning Design (LD) is in a more stable state we are considering the possible migration to parts of these specifications. Particularly SS and LS have the ability to represent the goal structure that helps our feedback mechanism suggest pages to the students. Having the ability to fully represent what is already implemented without the loss of functionality is quite important as it provides the ability of sharing and reusing not only content but also, and perhaps more importantly, authoring tools, interpreters or other tools that could significantly reduce the time it takes to author content or even learning paths and scenarios or perform other tasks like interpretting, debugging and testing.

[12] http://www.jiscmail.ac.uk/lists/QTI-IMPLEMENT.html
[13] http://maths.york.ac.uk/serving_maths/mod/forum/view.php?id=88
[14] http://maths.york.ac.uk/serving_maths
[15] A state of the art review can be seen in [9]

5.1 Further work

WALLIS has recently been extended to support context-sensitive material and further content has been developed under the COSMaP (Contextual On-line Solution of the Mathematics Problem[16]) project supported from funding from the University of Edinburgh. As research on the help-seeking behaviour of students and their overall interaction with the system is underway, a more elaborate student model is being built to target the feedback students receive and other pedagogical aspects of the system.

Meanwhile, one of the most important issues, as with any other system, is the availability of content whether theory and examples or exploratory or other kind of interactive exercises. Content plays a major role in how the learning environment is perceived and accepted by the students but it also determines its adoption by educators and institutions. We anticipate that in the future, even if a consensus in a definite standard format is not reached, the native formats that various systems use will have some common functionalities allowing transformations between them and thus enabling sharing and exchanging of questions. Through the development of protocols such as the Remote Question Protocol (RQP[17]) that allows e-learning systems to use external services for presenting and evaluating a question, or query languages like the one the Monet Mathematical Query Ontology[18] describes, as well as the developments in mathematical web services, more and more ILEs will be able to employ facilities such as elaborate question players, computer algebra systems or other processing engines, thus enhancing their functionalities and allowing questions that are pedagogically sound and actually help the students to learn.

References

1. D. Abela. Improving a web-based ITS with ANA; an animated agent. Master's thesis, The University of Edinburgh, School of Informatics; Artificial Intelligence, 2002.
2. B.P.Woolf, J.E. Beck, C. Eliot, and M.K. Stern. Growth and maturity of its. In *[?]*. 2001.
3. W. Clancey. Guidon-manage revisited: A socio-technical systems approach. *IJAIED*, 4(1):5–34, 1993.
4. J. Conklin. Hypertext: An introduction and survey. *IEEE Computer*, pages 17–41, 1987.
5. T. Conlon and H. Pain. Persistent collaboration: a methodology for applied aied. *IJAIED*, 7, 1996.
6. D. Dehn and S. van Mulken. The impact of animated agents: a review of empirical research. *International Journal of Human-Computer Studies*, 52:1–22, 2000.
7. B. du Boulay and R Luckin. Modelling human teaching tactics and strategies. *IJAIED*, 12, 2001.

[16] www.maths.ed.ac.uk/cosmap/
[17] http://mantis.york.ac.uk/moodle/course/view.php?id=14
[18] http://monet.nag.co.uk

8. G. Goguadze, M. Mavrikis, and G.A. Palomo. Interoperability issues between markup formats for mathematical exercises. *Submitted to First WebALT Conference and Exhibition - WebALT2006*, 2006.
9. H. Cuypers, K. Poels, R. Verrijzer, O. Caprotti, J. Karhima, M. Pauna and A. Strotmann. State of the art in mathematical e-learning. Available online at http://webalt.math.helsinki.fi, 2005.
10. H., Wood. Help seeking, learning and contingent tutoring. *Computers & Education*, 33(2-3):153–169, 1999.
11. C. Hunn. Employing jess for a web-based ITS. Master's thesis, The University of Edinburgh, School of Informatics, 2003.
12. C. Hunn and M. Mavrikis. Improving knowledge representation tutoring and authoring in a component based ILE. *Poster appearing in Intelligent Tutoring Systems 2004*, 4(1):827–829, 2004.
13. D.N. Hunt and D.A. Lawson. Trends in the mathematical competency of a level students on entry to university. *Teaching Mathematics and Its Applications*, 15(4), 1996.
14. K. R. Koedinger. Cognitive tutors as modeling tool and instructional model. In *[?]*. 2001.
15. P. Krutchen. *Rational Unified Process - an introduction*. Addison - Wesley, Reading, Massachusetts, 1999.
16. S. Lay. IMS question and test interoperability overview. Available online at http://www.imsglobal.org/question/qti_item_v2p0pd/index.html.
17. LTSN. Measuring the maths problem. Technical report, The Learning and Teaching Support Network (Maths, Stats & OR), The Institute of Mathematics and its Applications, The London Mathematical Society, The Engineering Council, 2000.
18. Minutes of 1st MathQTI workshop. Available online: http://maths.york.ac.uk/serving_maths/mod/book/view.php?id=154.
19. M. Mavrikis. Improving the effectiveness of interactive open learning environments. In *3rd Hellenic Conference on Artificial Intelligence*, pages 260–269.
20. M. Mavrikis. MathQTI draft specification overview. Available online at http://www.maths.ed.ac.uk/mathqti/docs/v0p3/mathqti_v0p3.pdf.
21. M. Mavrikis. Thoughts on MathQTI (see "documents on MathQTI" section of the "serving maths" project: http://maths.york.ac.uk/serving_maths/mod/resource/view.php?id=103). Technical report.
22. M. Mavrikis. Towards more intelligent and educational Dynamic Geometry Environments. Master's thesis, The Edinburgh University, Division of Informatics; Artificial Intelligence, 2001.
23. M. Mavrikis. Logging, replaying and analysing students' interactions in a web-based ile to improve student modelling. In C. Looi, G. McCalla, B. Bredeweg, and J. Breuker, editors, *Proceedings of 12th International Conference on Artificial Intelligence in Education*, page 967, 2005.
24. M. Mavrikis and A. Maciocia. Developing WALLIS; a web-based system to enhance mathematics teaching. In *ICTM*, Crete, Greece, 2002.
25. M. Mavrikis, A. Maciocia, and J. Lee. Targeting the affective state of students studying mathematics on a web-based ILE. *Supplement Proceedings of the 11th International Conference on AIED*, 2003.
26. M. Mavrikis and A. González Palomo. Mathematical, Interactive Exercise Generation from Static Documents. *Electronic Notes in Theoretical Computer Science*, 93:183–201, 2004.

27. E. Melis, E. Andres, A. Franke, A. Frischauf, G. Goguadse, P. Libbrecht, M. Pollet, and C. Ullrich. ActiveMath: A web-based Learning Environment. *IJAIED*, 12, 2001.
28. C. Miligan. Question and test interoperability (qti): Extending the specification for mathematics and numerical disciplines. *LTSN maths-CAA series*, Nov 2003. Available online at http://ltsn.mathstore.ac.uk/articles/maths-caa-series/nov2003/.
29. M. Sharples, N. Jeffery, B. du Boulay, D. Teather, and B. Teather. Socio-cognitive engineering. In *European conference on Human Centred Processes*, 1999.
30. D. Wood and H. Wood. Contingency in tutoring and learning. *learning and Instruction*, 6(4):391–397, 1996.

WebALT Metadata = LOM + CCD

Jouni Karhima, Juha Nurmonen, and Matti Pauna

University of Helsinki, Department of Mathematics and Statistics
P.O. Box 68 (Gustaf Hällströmin katu 2b)
FI-00014 University of Helsinki
{Jouni.Karhima, Juha.Nurmonen, Matti.Pauna}@Helsinki.Fi[1]

Abstract. We explain the chosen technologies for describing and storing the multilingual mathematical e-learning content produced in the WebALT project. We use the IEEE standard LOM (Learning Object Metadata) for encoding metadata. For classifying contents we use CCD (Course Content Dictionary) which is taxonomy describing mathematical topics and their relations. We present an implementation of these technologies, called WALTER (WebALT E-resources Repository), which is a distributed repository holding contents, their metadata, and the classification structure.

1. Introduction

Metadata and classification technologies are important topics in e-learning area, because they enable better searching and sharing of digital materials. Metadata is machine understandable information to describe a resource, e.g. its author or what topic it is about. Furthermore, the format of the resource and possible system requirements can be given helping to display the content appropriately. Often metadata attributes of a resource are obtained from controlled and formally specified vocabularies which set a common framework of the concepts in a given subject matter. There are several alternative techniques in this field and in this paper we discuss some of them from the viewpoint of the Web Advanced Learning Technologies (WebALT) project [1] whose objective is to create an XML database of multilingual mathematical e-learning resources.

Classification technologies, or taxonomies, are used to classify digital contents, usually collected into a repository. Particularly, in mathematics teaching several taxonomies are used, each created for different purposes. The American Mathematical Society has developed the AMS Mathematics Subject Classification [2] which is usually used for defining the exact subject area of research publications. OpenCyc [3] is a project where ontologies for several fields, including mathematics, are being developed. LivingTaxonomy [4] is an open-source project for defining taxonomies for education in several disciplines.

Taxonomies are used in online repositories for learning objects to classify and store contents. For example, Merlot [5] has for several years collected large amount of educational materials with its own taxonomy. Similar project is the Campus of Alberta Repository of Educational Objects (CAREO) [6] which uses the standard LOM metadata (see later) but it only supports key words rather than a real taxonomy. A large repository

[1] This work is sponsored by the eContent project EDC-22253-WEBALT

collecting educational resources in the United States is Gateway to Educational Materials (GEM) [7]. The European project LeActiveMath [8] is developing an online repository of learning materials specifically for mathematics education. In many of these cases the problem is that in a repository the taxonomy is pre-defined and users have little possibilities to modify it. For example, the iLumina project observed that "How to best implement or extend standardized vocabularies and taxonomies is an open question" [9].

WebALT has chosen as a starting point for its *extendible* taxonomy the mathematics taxonomy of LivingTaxonomy since it is quite comprehensive and specifically designed to be used in teaching. We have used it as a basis of our own taxonomy called Course Content Dictionary (CCD) which was already introduced in the Helsinki Learning System project [10] and also discussed in [11]. With the CCD it is possible to define and edit the classification with topics and their relations in several languages. Our system is flexible since it is possible to enlarge taxonomies, or even add alternative taxonomies, if needed. In the WebALT system the taxonomy is integrated to the WebALT E-resources Repository (WALTER) in which it is possible to classify e-learning contents according to the CCD.

Classification information for a piece of content is written into its metadata. We use the Learning Object Metadata (LOM) [12] standard for metadata because it is very comprehensive and specifically designed for educational purposes.

In this paper we first discuss mathematical digital learning materials and their format, especially for exercises. Secondly we examine metadata standards and technologies. We shall describe our choices regarding metadata information, e.g. how and what information fields are used. Next the CCD is presented with some comparisons with existing taxonomies. Finally we discuss the WALTER repository and its implementation.

2. WebALT e-learning contents

The WebALT project started January 2005 with funding from the European Union eContent program and it uses existing standards for representing mathematics on the web and existing linguistic technologies to produce a database of language-independent mathematical problems. The final product of the project will be a showcase consisting of a variety of learning resources for mathematics education stored into a repository, in particular multi-lingual exercises.

The WebALT repository can contain many types of learning materials, such as exercises, lecture notes, and animations. Mathematical animations and interactive calculators are particularly useful for the learners as they help to experiment with and visualize mathematical concepts. Animations are implemented usually with Java applets, Maplets (applets enhanced with Maple) and WebMathematica (web pages with Mathematica interactions).

For lecture notes a variety of formats are commonly used ranging from web pages to PowerPoint slide shows and PDF documents. All of these require metadata too and can be stored in the repository. We discuss here some of the chosen formats and technologies for exercises, since they are the primary target for producing learning materials in the WebALT project.

2.1 Exercises

Digital exercises enable automatic testing and practicing of students' mathematical skills. Often mathematical exercises involve multiple steps to achieve the final result so that a student has to master several skills. This has been implemented with *problem trees* (see figure 1) in the Helsinki Learning System project [10] in which some members of the WebALT project took part. From our experience this style of exercises where the student is prompted the next exercise depending on the answer to the previous one is highly desirable.

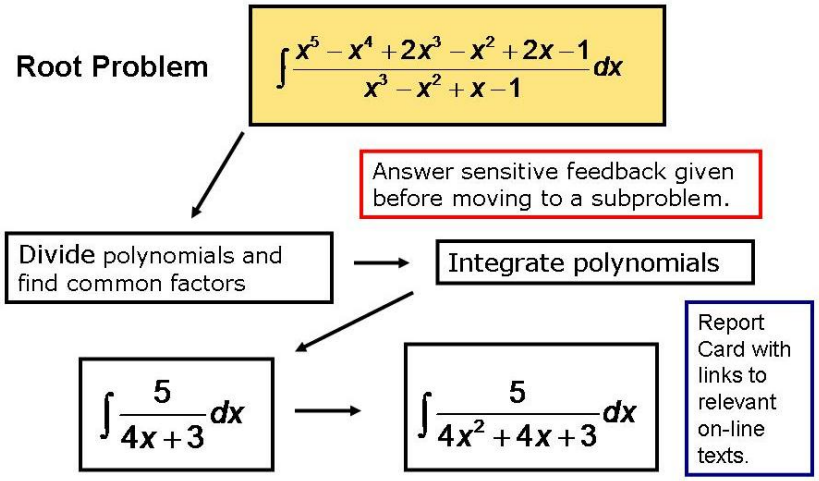


Fig. 1. A problem tree structure displaying feedbacks between steps. Also the system can produce a report card after the whole exercise showing the steps and giving links to relevant topics in the lecture notes (with the help of the CCD).

Problem trees are interesting question types since they can analyze students' skills and give specific feedback during steps. For example, an integration involving partial fraction decomposition can be first given to the student. If she cannot solve it, the system can ask whether she can do the partial fraction decomposition to see if the student had a problem there. So in this way, a rather challenging exercise can be broken into steps with hints. Notice that the individual problems occurring in the nodes of the problem tree could be reused if they were stored into a repository separately. Therefore having fine grained classification helping to find relevant sub problems for a given topic is an important requirement.

The individual problems occurring in the steps of a problem tree can be of many types, ranging from simple multiple choice problems to open problems where students answer with a formula and to graphical problem types, e.g. where the student has to select some points in a diagram. Many systems exist where students can rehearse these problems or even take exams. Moodle [13] is a popular open-source virtual learning environment with only little support for advanced mathematical question types and limited only to multiple choice because it lacks support for computer algebra system back-ends. We would also like to mention AiM (for Assessment in Mathematics) [14] and STACK (for System for Teaching

and Assessment using a Computer algebra Kernel) [15] which are computer aided assessment systems designed for mathematics with algebraic evaluation of students' answer by using an external computer algebra systems. There are also commercial systems supporting assessment, most notably MapleTA, which allows for a wide range of problem types and is powered by the Maple system. However, in all of these it is impossible or difficult to implement problem trees.

The generic computer format for mathematics exercises is XML based using MathML or OpenMath [16] to encode mathematical formulas, making them easier to be utilized and displayed in the browser. Moreover, as exercise formats are roughly equivalent in their expressivity, using XML transformations between them is feasible, and therefore one does not have to commit to one format. The most used and implemented digital encoding formats for exercises are Question and Test Interoperability (QTI) [17], which is a standard format for exercises in any subject, MathQTI [18], which is a mathematics extension of QTI, LeActiveMath exercise language [19], and EDU [20], which is a text format not encoded in XML and used in MapleTA. AiM uses Maple packages to produce questions written in AiM syntax which can include Maple commands. STACK, which is a continuation of AiM, has similar language but encoded in XML.

For exercises, WebALT favors the MathQTI format because it is based on OpenMath and QTI standards, from which it inherits the support for an extensive number of question types. In QTI an author can define random template variables producing many different problems from the same template. MathQTI extends QTI by allowing template variable definitions and students' answer processing encoded in OpenMath, enabling more sophisticated algorithmic problem types. As the WebALT exercises will be multilingual, some OpenMath encoding is also needed to express the linguistic attributes of the problem text, see [21] for details. Simple multiple choice problems in mathematics do not necessarily need the sophisticated facilities provided by MathQTI and OpenMath, so some of the WebALT problems could be imported into systems only supporting QTI, such as Moodle. This was also a reason to choose MathQTI since it is upwards compatible with QTI.

3. Metadata

Metadata is commonly understood to be a very important technique to add machine understandable information to describe contents. It enables users to classify, search, and share resources whether they are digital or non-digital. In almost every university, digital material is made available to students by the professors and seldom is the material re-used outside the university or even outside the specific class. One major reason is that the online lectures and problems are not described using metadata and thus not made available in a common framework shared by other potential users.

The production of metadata is gradually being regarded as a key issue, at institutional or national levels, and attention is paid to metadata standards. While digital learning resources develop extensively, the availability of good metadata becomes crucial as soon as those resources are to be shared.

3.1 Metadata standards

Dublin Core [22] is perhaps the most well-known metadata standard. It is designed to describe general publications and as such is inadequate for learning resources. Metadata standards particularly developed for education are for example Learning Object Metadata (LOM) [12] and Dublin Core Educational [23] which defines a set of education-specific elements, element qualifiers, and value qualifiers (controlled vocabularies) to be used with the Dublin Core. The abovementioned GEM repository project also has its own metadata which is an extension of Dublin Core.

The Learning Object Metadata, created by the IEEE, defines attributes that are needed to adequately describe a learning object. LOM is quite comprehensive and certainly more extensive than Dublin Core. A learning object is defined as an entity which can be used and referred to during technology supported learning. This entity can be digital or non-digital allowing in particular metadata descriptions of printed materials. The metadata descriptions of learning objects include content type, learning objectives, software tools, persons and organizations, among others. Learning object metadata can then be used for searching and automatic processing of learning objects and for their classification. A more recent application is metadata harvesting systems, such as Open Archives Initiative [24], providing automatic retrieval of contents from registered metadata repositories.

LOM is an information model and IEEE also provides an XML binding for it defined by an XML schema. The IMS Global Learning Consortium has also developed an XML schema (see [25]) for LOM which is the one the WebALT project uses.

The WebALT repository will be the interface to the database collecting and organizing contents, for instance, the mathematical problems in MathQTI format. LOM metadata is attached externally to every piece of content giving also relevant taxonomy classification information about it.

Although LOM metadata is already specifically designed for educational purposes, it is still necessary to further adapt it to our more restricted context consisting of only mathematics related content. LOM has more than 70 fields distributed into nine categories to cover all potential usage and hence there are also many of them not very relevant for our purposes. Applying and restricting metadata has been regarded as an important issue in the projects that deal with metadata and repositories. For example, the iLumina project has studied the filling and use of metadata in its repository and on basis of it, identified a subset of LOM elements that fit their purposes, see [9] for details. Also the ULI (University teaching network for computer science) project restricted LOM into a subset of 15 fields to be used in their project, see [26]. Therefore it is of also our concern to eliminate these in advance to avoid frustrating the metadata authors and to encourage them to fill in the more important parts as the value of metadata is only in those fields that are carefully filled in. As a consequence and to further emphasize this we have also tried to separate the most important fields from those that give additional information and we try to bring them in front when presenting the metadata input form.

3.2 WebALT's restriction of LOM

We present and motivate here some of the choices we made restricting the LOM elements. On the other hand, some information specific to our project is hard to express with the fields

LOM currently provides. Examples of these are the question type (e.g. multiple choice, formula, etc.), whether the question text is in multilingual or natural language form, and whether the question has feedback and hints. Let us go through some of the LOM categories and analyze their fields in view of WebALT's needs.

Title, Description, Keywords
These are the common fields telling about the content of the resource. Title will give the first impression and description a further glance for someone who is browsing through resources one by one. Keywords will make sure that all closely related words, that are not put in the title or description, will be explicitly mentioned and the resource can be found using keyword searches.

Author, Editors, and other contributors
The author of a resource is a very important piece of information. Not only we want to record it for copyright purposes, but also many users, searching for resources, are likely to eventually identify their favorite authors. Other contributors may also gain regard for only contributing quality content. Inside LOM, author information is encoded in vCard format.

Classification
A great effort has been made in WebALT to produce a high quality classification discipline in view also of its potential use with problem trees. The value in classification is that it creates a whole new method of searching, browsing, to compete or cooperate with keyword search. This method, being less creative and more automatic, is fundamentally different from keyword searching and may result better hits with less effort when well done. More about classifications follows in the next section.

Rights, Cost and Copyrights
This is important metadata, but it may not be up to common metadata or resource editor to provide this information. It is likely that these kinds of fields will be filled in according to strict publisher policies and will even have precompleted values.

Difficulty, Context, and other educational information
Difficulty, from very easy to very difficult, context, school or higher education, and other such fields are significantly trickier to fill in than any of the previous ones and WebALT intends not to force authors to do that. However, we might expect that when browsing a Single Variable Integration topic in the classification, the very easy exercises demonstrate the usage of very basic tools of integration and the very difficult ones present complex problems with no clear-cut solution methods written in textbooks. For this kind of fields it is necessary to create clearly written guidelines to ensure general compatibility.

Technical information, free annotations, etc
Some technical information is always needed for automatic handling and in some cases also for searching. This includes data format, physical size and possibly other things depending on final system implementation. A chance for free annotations is also provided for any notes, observations etc. that have no more suitable field for them.

4. Classification

WebALT course materials will be archived and searched primarily by means of a taxonomic index based on a classification of mathematics topics defined by the Course Content

Dictionary (CCD). The CCD was already introduced as a prototype in the Helsinki Learning System (HLS) [10] and is now implemented more completely in the WebALT project. In HLS the CCD helped to catalogue the single step problems occurring in problem trees for reuse. Since the CCD is a technique for classification, it is also applicable to other disciplines than mathematics. In the WebALT system the CCD is tightly integrated within the metadata editor and repository, where it is used in the user interface, to allow the user to link resources to topics in a tree representation of the taxonomy, as displayed in the next figure.

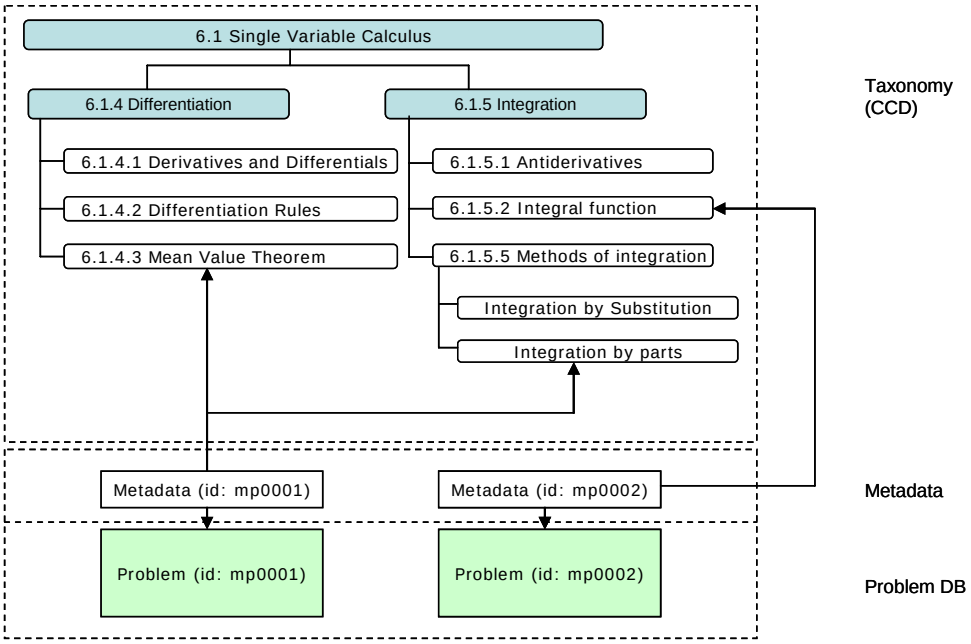


Fig. 2. WebALT repository (WALTER) structure. Metadata is the binding link between resources and the taxonomy.

With the CCD an author can classify materials and share them with other instructors who can browse through the classification to search for suitable content for their courses. It sets a unique framework for authors to develop course materials and therefore it also enables collaboration to produce a basic mathematics course, as each author can concentrate on specific topics. This makes also the lifetime of the material longer, since it is properly stored and classified for others to find and reuse.

The CCD defines a common hierarchy of topics for a course in a way similar to a table of contents in a standard textbook. But on the other hand, the CCD should be understood like taxonomy, so it just describes a collection of topics and their relations, not as a strict table of contents to be followed precisely in a specific course. Topics need not be covered in any specific order or some topics could be omitted in the lecture notes of a specific course. For this reason, the CCD must be quite general so that it covers many topics of a subject making it easy to find a natural place for resources inside it.

The CCD allows constant extending and adapting its current state, which makes it more than static taxonomy. For instance, editors might want to make it more fine grained to allow more detailed classification for their materials preventing the situation that huge amount of resources are classified under a too general topic. For this reason, the taxonomy can be edited, and if a teacher wants to use another, new taxonomies can be created to the repository. It should be noted though that having multiple taxonomies for the same course makes searching and sharing more difficult. For instance, it would be a problem if all the teachers of a course created their own private taxonomies, because then sharing is not likely to happen.

Therefore some policies and guidelines on updating the CCD have to be set. For instance, it should not be allowed to remove a topic if there are contents stored under it or if it has subtopics. On the other hand, one should be able to introduce a new topic if there are enough contents to be classified under it. As a minimum requirement, we have defined user rights levels so that users who are for example allowed to upload contents are not necessarily allowed to change the classification.

4.1 Taxonomies

There are several taxonomies used in mathematics teaching, each created for different purposes. These include American Mathematical Society's Mathematics Subject Classification [2], Merlot [5], Mathematics Content Dictionary (MCD) [27], and OpenCyc [3]. Each of the mentioned can be used as a basis for the CCD. WebALT has chosen as a basis, to expand upon in its CCD, the LivingTaxonomy.

Other taxonomies discussed here are hard to fit into educational purposes. Some of them are too detailed and are therefore suitable only for specifically designed lecture courses. These include AMS Mathematics Subject Classification, which is designed for research content, not for teaching purposes. Also a Finnish taxonomy called Mathematics Content Dictionary (MCD) has its background in a certain polytechnic training program, and therefore is only applicable for similar programs in engineering mathematics. Other taxonomies, in turn, are too general and each topic would cover all too much material. These include Merlot and also OpenCyc.

4.1.1 Living taxonomy

LivingTaxonomy is a quite comprehensive and clear taxonomy. The LivingTaxonomy Project (see [12]) currently has nine taxonomies for various disciplines and they aim at, among other things, creating global, open source, standards-based taxonomies for digital education. In mathematics the LivingTaxonomy Project relies on the Core Subject Taxonomy for Mathematical Sciences Education [28], which is an approved taxonomy for use in classifying digital resources in mathematics education by the Mathematical Sciences Conference Group on Digital Educational Resources. This taxonomy can have several levels, depending naturally on topic. The next figure shows the main topics of this taxonomy and how the main topic Calculus is expanded into subtopics.

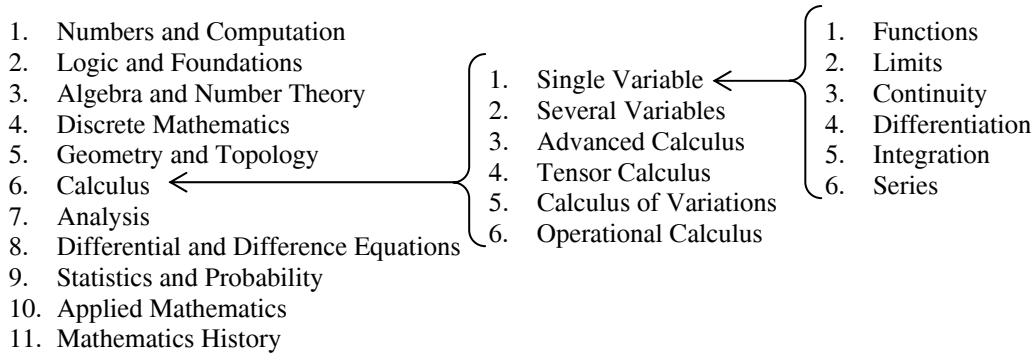


Fig. 3. Topic hierarchy of LivingTaxonomy for Single Variable Calculus

We find that the classification criteria used in LivingTaxonomy are adequate and reasonably detailed. On the other hand, it is still flexible for high school and university level teaching, and it also supports several tables of contents used in standard textbooks for mathematics courses.

4.1.2 Extending taxonomy

As mentioned earlier, the CCD allows extensions and adapting. The LivingTaxonomy is a very good taxonomy for general purposes but still too general to be used in a single course. Our experience showed that in many cases it is difficult to find sufficiently narrow topics for materials e.g. in a Calculus university course. For that reason, we wish to extend parts of LivingTaxonomy so that each subtopic covers a reasonable amount of material. This has been implemented for Single Variable Calculus part. To mention one subtopic that has been extended, the subtopic Integration of Single Variable Calculus (see above) looks like follows:

6.1.5 Integration:
- 6.1.5.1 Antiderivatives
- 6.1.5.2 Integral function
- 6.1.5.3 Riemann Sums and Definite Integrals
- 6.1.5.4 Fundamental Theorem of Calculus
- 6.1.5.5 Methods of integration
- 6.1.5.6 Improper integrals
- 6.1.5.7 Numerical Integration
- 6.1.5.8 Areas and volumes
- 6.1.5.9 Further applications of integration

4.2 Representation of the CCD

Conceptually, the CCD is a tree as a data structure, where the relation is the subtopic relation. For encoding this structure with XML, WebALT has developed an XML schema under the namespace `http://www.webalt.net/ccd`. In this data structure, which is

summarized in table 1, `Topic` is the main element and each topic also allows adding subtopics, related topics, related subtopics and prerequisites into it. The `subtopic` relation is naturally the most important (we have added for performance reasons also the inverse `supertopic` relation to each topic). Some of the data fields describe metadata for the topic, such as author, although topics have very little metadata as they are not really learning objects by themselves. For the metadata of topics we use LOM fields where applicable. Most important of these is the `description` field, where one can input e.g. mathematical definitions and examples. Moreover, the title and description of a topic can be given in several languages enabling the creation of a common taxonomy in a multicultural framework.

Table 1. CCD Data structure

`Title`	Title for the topic as specified in LOM
`Identifier`	Id of the topic
`Description`	Short description of the subject covered by the topic as specified in LOM
`Path`	A hierarchical path to the topic, e.g. 6.1.1
`Contribute`	Author information as specified in LOM
`Origin`	Source of the taxonomy, e.g. LivingTaxonomy
`Subtopic`	List of subtopics of this topic
`Prerequisite`	Topics that are required to understand this topic
`Related Topic`	Cross references to other topics in the CCD
`Related Subtopic`	Cross references to other subtopics of this topic in different branches in the CCD
`Supertopic`	Link to the supertopic

5. Repository implementation

The WebALT project has developed a repository implementation called WebALT E-resource Repository (WALTER), see [29]. It is a platform to store content and edit its metadata which also allows modifying the classification (CCD) for users authorized to do so. WALTER indexes its content by the CCD classification allowing one to browse the hierarchy to search, display, download, and upload resources.

WALTER is a web application running in a Java2EE environment. As the underlying database it uses an open source native XML database called eXist [30], which was chosen because it stores and indexes XML files and can query them using the XQuery language, which is similar to relational query languages, and also allows one to modify XML data with the XUpdate language. The database can also store binary files, so in particular it can contain multimedia e-learning contents. Furthermore, eXist is compatible with the XML:DB standard [31] which is a common interface for communicating with the database to perform standard database procedures, such as XQuery queries and XUpdate modifications.

WALTER is designed to be a *distributed* metadata repository in the sense that one can add metadata records for contents that are stored in remote databases, as shown in figure 4.

This enables the sharing and retrieval of relevant course materials from different locations across the web. It also means that if a new publisher wants to use WALTER for publication, he/she only needs to install a local database, which then can be connected to the centrally managed world-wide system. The different systems communicate with the XML:DB protocol so that each database can be of any compliant implementation such as Apache Xindice and the commercial X-Hive. This assures a real ease-of-use in adopting the system and also a great chance for interoperability between different publishers.

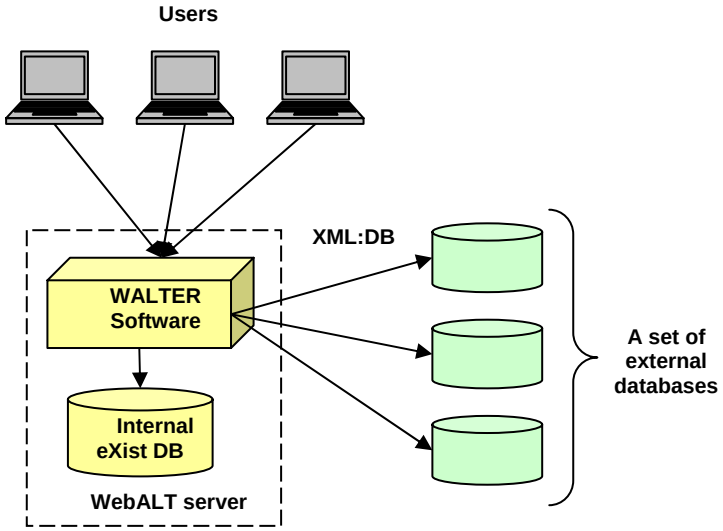


Fig. 4. WALTER software is located in the WebALT server with its internal database that stores for example the CCD classification, users and the locations of external resource databases. New external resource databases can be added simply and WALTER connects them behind the scenes with XML:DB protocol so that a common user may not even notice the distributed system.

WALTER includes a *metadata editor* for editing LOM records such as they are described in section 3.2. In the editor the metadata fields are separated into two groups based on their importance. An author creating metadata will first see the most vital fields and has an option to also view and edit the others. In the end there will at least in some contexts also be a third group of metadata fields which an author is not authorized to edit at all. Ways to encourage authors to fill in as many fields as possible are under consideration, one already implemented is the possibility to store personal *metadata templates*. As resources created by one author often share a lot of common information, and therefore it may be reasonable for him/her to fill in them carefully once and then use the same template in the future.

The searching capabilities of the repository offer an advanced way to use the specified metadata and classification either separately or combined. A user may perform a keyword search to cover the whole repository, or alternatively, first browse into a classification topic and then search only for resources that have this topic occurring in their classification path. It is also be possible to specify in which metadata fields the keywords should occur. After a search has been carried out, it is essential to offer the results in a convenient way showing

the relevant parts of the metadata in a compact form, so that the user can immediately pick the items that seem most useful for his/her purposes.

6. Conclusions

The WebALT project combines existing technologies such as OpenMath, metadata, and taxonomies to produce a repository for multilingual mathematical learning materials. WebALT is committed to supporting emerging standards, of which most interesting are the Sharable Content Object Reference Model (SCORM) [32] and ontologies, in the e-learning arena and hence it has adopted existing standards for describing the content of mathematical exercises, such as OpenMath for the mathematical fragments, QTI (extended with MathQTI) for questions and interactivity, and LOM for the metadata content. The CCD technology combined with the WALTER repository enables classifying, storing, searching, and sharing these materials.

The future development of the WebALT project will take into account semantic web technologies because they naturally fit with metadata and taxonomies. In particular, implementing the CCD using ontologies seems feasible. There are some projects already which are developing metadata using semantic web technologies, of which most useful in this setting is the Resource Description Framework (RDF) [33]. Let us mention in particular the SCAM project [34] which builds a framework for storing and handling RDF encoded metadata and closely related one, the SHAME metadata editing framework [35].

As noted earlier, there exist several standards for encoding metadata, such as LOM and Dublin Core. One important issue is how to combine metadata information of a resource described using different standards. One answer to this problem is again provided by RDF by which one can give attributes to a resource using triples of the form (resource, relation, value). The idea is that the parts of this triple are well-defined in the way that a computer can analyze its purpose, since typically the relation and the values come from well-defined ontologies and vocabularies. For instance, in [26] it is showed how to combine Dublin Core and LOM metadata attributes with RDF records. Doing so requires binding LOM to RDF, which is provided in [36]. It is also possible to have the metadata of a resource distributed across the internet since these RDF triples can co-exist in different metadata repositories describing various aspects of the resource.

As mentioned, SCORM and similar content delivery standards are important technologies gaining support from e-learning platforms. SCORM is a comprehensive set of standards for e-learning technologies, specifying also behavior of content, supported partly by many virtual learning environments, for instance, WebCT and Moodle. Therefore it is natural to add support for importing and, most importantly, exporting SCORM packages from the WALTER repository.

A future goal also for the project is making the repository accessible through web services technologies, such as SOAP messages. This would enable better distribution with systems not necessarily using exactly the same metadata format and database accessing protocol (i.e. XML:DB) as WALTER does. At the moment some components of the WebALT system are connected via web services, which enable the running of different services (e.g. language generation service) in different servers.

7. Acknowledgements

We would like to thank the referee and especially Olga Caprotti for suggestions and corrections of this paper.

References

1. *WebALT,* http://www.webalt.net.
2. *AMS Mathematics Subject Classification,* http://www.ams.org/msc/.
3. *OpenCyc,* http://www.opencyc.org/.
4. *The Living Taxonomy Project,* http://www.livingtaxonomy.org/.
5. *Merlot,* http://www.merlot.org/home/SubjectCatIndex.po?discipline=Mathematics.
6. *CAREO,* http://careo.ucalgary.ca/cgi-bin/WebObjects/CAREO.woa.
7. *GEM,* http://www.thegateway.org/.
8. *LeActiveMath,* http://www.leactivemath.org/.
9. Heath, B.P., McArthur, David J., McClelland, Marilyn K., Vetter, Ronald J., *Metadata Lessons from the iLumina Digital Library.* Communications of the ACM, 2005. **48**(7): p. 68-74.
10. *Helsinki Learning System,* http://mark.math.helsinki.fi/HLS/.
11. Karhima, J., Nurmonen, J., Pauna, M., *"Course Content Dictionary for sharing on-line educational materials",* http://mathstore.ac.uk/articles/maths-caa-series/aug2005/. 2005, Maths CAA Series.
12. *IEEE Learning Object Metadata,* http://ltsc.ieee.org/wg12/.
13. *Moodle,* http://moodle.org/.
14. *AiM,* http://maths.york.ac.uk/moodle/aiminfo/.
15. *STACK,* http://www.stack.bham.ac.uk/.
16. *OpenMath,* http://www.openmath.org/.
17. *IMS Question and Test Interoperability Specification,* http://www.imsglobal.org/question/.
18. *Mathematical Question and Test Interoperability,* http://www.maths.ed.ac.uk/Math-QTI/.
19. Cohen, A.M., Cuypers, H., Barreiro, R., *MathDox: Mathematical Documents on the Web,* http://www.win.tue.nl/~hansc/mathdox3.pdf. 2005.
20. *Brownstone EDUCampus,* http://www.brownstone.net/products/edu/.
21. Strotmann, A., Ng'ang'a, W., Caprotti, O., *Multilingual Access to Mathematical Exercise Problems.* Electronic Proceedings of the Internet Accessible Mathematical Computation Workshop, 2005.
22. *Dublin Core Metadata Initiative,* http://dublincore.org/.
23. *Dublin Core Education Working Group,* http://dublincore.org/groups/education/.
24. *Open Archives Initiative,* http://www.openarchives.org/.
25. *IMS Metadata Schema,* http://www.imsglobal.org/xsd/imsmd_v1p2p2.xsd.
26. Brase, J., Nejdl, W., *Ontologies and Metadata for eLearning*. 2004, in Handbook on Ontologies (Staab, S., and Studer, R. eds.).
27. *Mathematics Content Dictionary,* http://matta.hut.fi/mattafi/materiaalipankki/.

28. *Core Subject Taxonomy for Mathematical Sciences Education,* http://people.uncw.edu/hermanr/MathTax/.
29. *WALTER - Webalt E-resource Repository,* http://www.webalt.net/WCR/.
30. *eXist Open Source Native XML Database,* http://exist-db.org/.
31. *XML:DB Initiative,* http://xmldb-org.sourceforge.net/index.html.
32. *SCORM 2004,* http://www.adlnet.org/scorm/index.cfm.
33. *Resource Description Framework (RDF),* http://www.w3.org/RDF/.
34. Palmér, M., Naeve, A., Paulsson, F., *The SCAM Framework: Helping Semantic Web Applications to Store and Access Metadata*. 2004, European Semantic Web Symposium 2004.
35. *SHAME: Standardized Hyper Adaptable Metadata Editor,* http://kmr.nada.kth.se/shame/.
36. Nilsson, M., Palmér, M., Brase, J., *The LOM RDF Binding - Principles and Implementation*. 2003, Proceedings of the Third Annual ARIADNE conference.

WExEd - WebALT Exercise Editor
for Multilingual Mathematics Exercises

Arjeh Cohen* Hans Cuypers* Karin Poels* Mark Spanbroek* Rikko Verrijzer*
amc@win.tue.nl hansc@win.tue.nl k.j.p.m.poels@tue.nl m.j.m.spanbroek@tue.nl r.verrijzer@tue.nl

*Department of Mathematics and Computer Science
Eindhoven University of Technology
Eindhoven, The Netherlands

***Abstract*— Recently, the computer program WExEd was developed within the WebALT project. It enables one to create and edit interactive mathematics exercises that can be automatically translated to and subsequently played in different languages (currently English, Spanish, Finnish, Swedish and Italian). It makes use of OpenMath, a standard for encoding the semantics of mathematical objects in XML, and software developed within WebALT. WExEd does not only encode the mathematics of the exercise in OpenMath but also the text. This is possible because, in general, mathematics exercises are built from a well defined and relatively small set of words. In this paper we give a presentation of WExEd.**

I. Introduction

Although mathematics itself is universal, it is taught in many different languages around the world. This implies that, by restricting themselves to a specific language, publishers and authors of mathematical exercises only reach a small part of the possible audience. If the exercises could be automatically translated to different languages, then a publisher could extend his market dramatically.

Automatically translating exercises is hard because of large lexicons and complex grammars. It becomes easier when the exercises are constructed within a well defined context from only a small and specific part of a language's lexicon. In general, this is the case for mathematics exercises, which can be as simple as "Calculate the derivative of $\sin$", but also the words from more advanced exercises may come from a rather small and specific set of the language's lexicon.

OpenMath [1] is an emerging standard for representing the semantics of mathematical objects in XML. It allows mathematical objects to be exchanged between computer programs, stored in databases or published on the Worldwide Web. Each mathematical symbol (e.g. $\cos$) has its own OpenMath representation and OpenMath representations are grouped in so-called Content Dictionaries (CD's). Every combination of mathematical symbols can be written as one OpenMath object. By ways of illustration, the OpenMath representation of $\cos(1/x)$ is shown below. Note that the mathematical symbol "cosine" has the OpenMath representation "$\cos$", defined in the CD "transc1" on transcendental functions.

```
<OMOBJ>
  <OMA>
    <OMS cd="transc1" name="cos"/>
    <OMA>
      <OMS cd="arith1" name="divide"/>
      <OMI>1</OMI>
      <OMV name="x"/>
    </OMA>
  </OMA>
</OMOBJ>
```

WebALT (Web Advanced Learning Technologies), an e-learning project funded by the EU, uses OpenMath to automatically translate mathematics exercises into natural languages. To do so, not only mathematics is encoded in OpenMath, but also text so that every sentence in the exercise can be written as a single OpenMath object. To encode text in mathematics exercises into OpenMath, a mathematical lexicon and OpenMath representations for all words in this lexicon were created within WebALT. This resulted in the CD named "nlg"; it contains OpenMath symbols for e.g. "calculate", "determine", "prove", etc. In order to translate the OpenMath object representing a sentence into a natural language, grammar generation rules corresponding to the newly created OpenMath representations were formulated and

an existing grammar formalism was used. The tool that automatically translates exercise sentences is the Natural Language Generator. The tool that creates multilingual exercise sentences (in the form of OpenMath objects) is the TextMathEditor. These tools will be further described in the following section.

For creating and editing interactive multilingual mathematics exercises we developed the editor WExEd - WebALT Exercise Editor. Exercises created with WExEd can be automatically translated to various natural languages because the source code of the exercise entries consist only of OpenMath expressions. Within WExEd this code is generated by the TextMathEditor. Moreover, exercises created with WExEd are highly interactive; the exercises can be multistep, allow for random input variables and offer the possibility to check and interpret the student's answer with the help of computational services like a Computer Algebra System (CAS) such as Mathematica [2] or Maple [3]. With random variables, a student can solve multiple instances of the same exercise. Because of the capability of reading and understanding the student's answer, WExEd can prompt the student with a following exercise or task depending on the student's answer. The student can therefore be guided through the various steps of the exercise.

In this document we speak about *sentences* or *exercise sentences* when we mean sentences in an exercise such as e.g. the problem statement or the predefined answers in a multiple choice exercise.

In Section II we indicate the software of which WExEd makes use and in Section III we describe its functionality.

II. Software Components included in WExEd

The components developed within WebALT that are integrated with WExEd are described in Sections II-A and II-B. In Section II-C we describe tools developed by Technische Universiteit Eindhoven that are integrated with WExEd.

A. TextMathEditor

The TextMathEditor is a Java application that is developed for WebALT by Maths for More [6]. With this editor, one can create and edit multilingual mathematics sentences. These sentences are encoded as a single OpenMath object, i.e., both mathematics and text are encoded in OpenMath, where the text must follow a restricted grammar. Such a grammar defines sentence constructs corresponding to OpenMath representations that can be automatically translated to various natural languages. The editor stores the OpenMath representation of the created sentence, with natural language generation hints as (OpenMath-defined) attributes.

When a user (e.g., an author) creates an exercise sentence, all possible words with which he can start the sentence are shown to him (see the lower left part of Fig. 1).

From then on, the user is given a list of the possible words/mathematical objects with which his sentence might be continued (see the lower left part of Fig. 2). This list is adapted to the previously selected words. The TextMathEditor offers a completion grammar: the editor displays the different ways in which a sentence may end. This is illustrated in the lower right part of Figs. 1 and 2. It also offers error highlighting; it highlights at which part of the sentence there is an error.

The output of the TextMathEditor is an OpenMath object. Suppose that the user created the sentence "Calculate the gcd of 18 and 24". Then the output of the TextMathEditor is as follows. Note that all linguistical symbols are from the newly created CD "nlg".

```
<OMOBJ>
  <OMATTR>
    <OMATP>
      <OMS cd="nlg" name="mood"/>
      <OMS cd="nlg" name="imperative"/>
      <OMS cd="nlg" name="tense"/>
      <OMS cd="nlg" name="present"/>
      <OMS cd="nlg" name="directive"/>
      <OMS cd="nlg" name="calculate"/>
    </OMATP>
    <OMA>
      <OMS cd="arith1" name="gcd"/>
      <OMATTR>
        <OMATP>
          <OMS cd="nlg" name="render"/>
          <OMS cd="nlg" name="formula"/>
        </OMATP>
        <OMI>18</OMI>
      </OMATTR>
      <OMATTR>
        <OMATP>
          <OMS cd="nlg" name="render"/>
          <OMS cd="nlg" name="formula"/>
        </OMATP>
        <OMI>24</OMI>
      </OMATTR>
    </OMA>
  </OMATTR>
</OMOBJ>
```

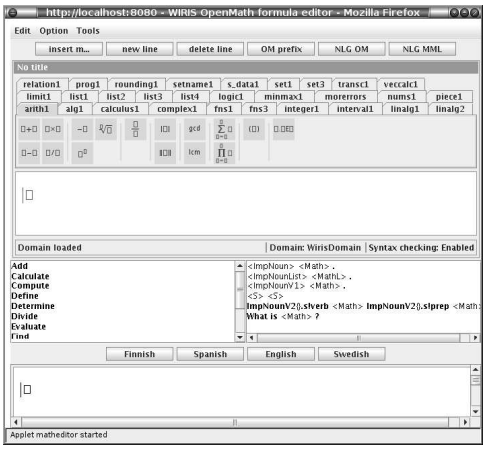


Fig. 1. TextMathEditor: a user can choose a verb to start with.

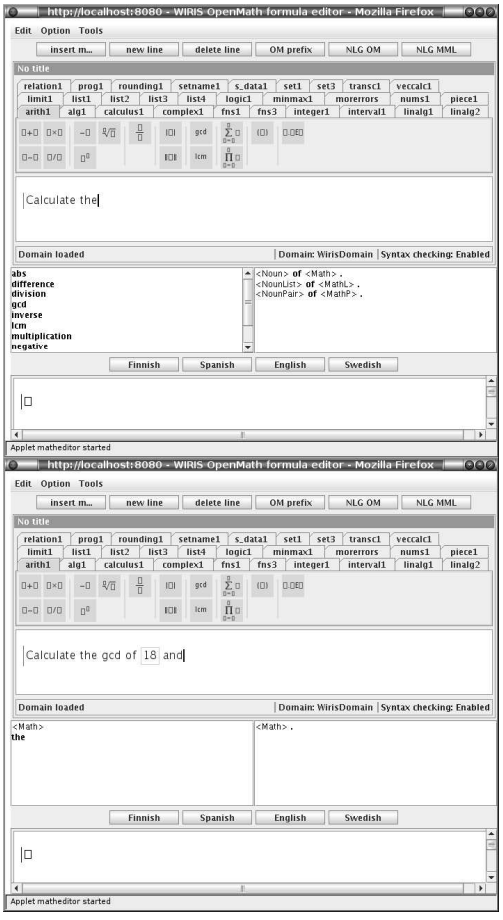


Fig. 2. TextMathEditor: a user can choose the word or mathematical object to continue.

The TextMathEditor is developed in such a way that the user can choose the language in which to edit the exercise sentence.

B. Natural Language Generator

The Natural Language Generator is being developed for WebALT by Lauri Carlson and his team at the University of Helsinki [4]. It makes use of the Grammatical Framework (GF) [5], developed at Chalmers university (Gothenburg) by Aarne Ranta. The GF grammar environment and compiler have been implemented in Haskell. There is support for a Java runtime engine.

The Natural Language Generator generates natural language for mathematics sentences that are encoded in OpenMath. Therefore, the input of the generator is an OpenMath object representing the sentence together with a set of natural language generation hints (the output of the TextMathEditor). The output of the generator is the natural language rendering of the given sentence. The target languages for WebALT include English, Finnish, Swedish, French, Italian, Spanish, Catalan, Dutch and German.

Let us illustrate the Natural Language Generator. Suppose that the input of the Natural Language Generator is the OpenMath object illustrated above. The language generation hints say that the mood is ”imperative”, the tense is ”present” and the main directive is ”calculate”. We also see that the mathematical symbol ”gcd” is rendered as text (this means that the symbol ”gcd” is verbalized) and the integers 18 and 24 should be rendered as a formula (if not, the Natural Language Generator would generate ”eighteen” and ”twentyfour” respectively in English). This leads to the following output if the generator translates to English.

”Calculate the greatest common divisor of 18 and 24.”

In Spanish it is automatically translated to

”Calcula el máximo común divisor de 18 y de 24.”

C. Answer Processing Tools Authored at TU/e

When a student gives an answer to an exercise, he is presented with a mathematics editor that outputs an OpenMath object; the answer of the student is presented to the system in OpenMath format. To verify whether the student’s answer is correct, e.g., whether the answer is equal to the correct answer as predefined by the author of the exercise, the answer has to be sent and processed by a CAS. Because a CAS cannot read OpenMath, the OpenMath should be translated into something that the CAS can read, e.g., the CAS’ own language. This translation is done by so-called *phrasebooks*.

Technische Universiteit Eindhoven developed phrasebooks for various CAS’s like Mathematica, Maxima, Wiris and Maple. WExEd uses these phrasebooks to translate the student’s OpenMath answer to something the specified CAS can read. To illustrate this, suppose that the student’s answer to our example exercise is 5. The OpenMath object that we want to have verified by a CAS is the following:

```
<om:OMOBJ>
   <om:OMA>
      <om:OMS cd="relation1" name="eq"/>
      <om:OMI>5</om:OMI>
      <om:OMA>
         <om:OMS cd="arith1" name="gcd"/>
         <om:OMI>18</om:OMI>
         <om:OMI>24</om:OMI>
      </om:OMA>
   </om:OMA>
</om:OMOBJ>
```

Suppose that we want Mathematica to evaluate this OpenMath object. Then we use the Mathematica phrasebook which translates the OpenMath object into the following Mathematica code.

```
Equal[5, GCD[18, 24]]
```

Mathematica evaluates this code and produces the "False" commando which is translated back to OpenMath by the Mathematica phrasebook. The resulting OpenMath object is given by

```
<om:OMOBJ>
   <om:OMS cd="logic1" name="false"/>
</om:OMOBJ>.
```

WExEd reads this OpenMath object and draws the conclusion that the student's answer is false.

We note that phrasebooks are not only used by WExEd for processing the student's answer but for instance also for generating the values of random variables or mathematical expressions.

III. Functionalities of WExEd

A. General Functionalities

The formal language underlying WExEd is the LeActiveMath Exercise Language [7] which is a collection of XML tags together with restrictions imposed on their use. This language is such that exercises are:

- Interactive: have means of exchanging information with the student by being able to read the student's answer and to give feedback and/or hints.
- Automatized: have means of checking whether the student gives the correct answer by connecting for example to a CAS. The author gives a query in OpenMath which determines the correct answer. This query is translated to a CAS' language by using phrasebooks, as explained in Section II-C. The CAS can then calculate the correct answer and can subsequently compare the student's answer with the correct answer.
- Adaptive: takes appropriate action depending on its knowledge of the student and/or the student's answer to a particular question; exercises are made up of *interactions* which consist of information given to the student (e.g. a question or feedback to the student's answer) and possibly information provided by the student (e.g. an answer to a question). Depending on the student's answer he is directed to a following interaction (which could be the same question, a question that will teach the student how to solve the previous question, a new question, feedback, etc.). For illustration, if a student answered our example exercise incorrectly, the student could be directed to the interaction that contains the question "What are the common prime factors of 18 and 24?".
- Reusable: (random) variables can be defined in an exercise so that students can do multiple instances of the same exercise. The values of the variables are processed by a CAS using phrasebooks.

The exercise language, as any XML-language, is extensible in the sense that new tags can be added at any moment to facilitate new features.

WExEd supports multiple choice exercises, open exercises and multistep exercises combining these exercise types.

B. Editing Multilingual Exercises

The exercises created with WExEd are multilingual because all exercise sentences are created with the TextMathEditor. When editing an exercise with WExEd, the author is presented with the TextMathEditor every time the input of a sentence is required (e.g. the exercise problem statement, the choices in a multiple choice exercise, feedback or hints). In this way, all entries of the exercise consist of OpenMath objects which could be sentences containing mathematics or completely mathematical expressions. The exercise is therefore multilingual; it does not contain text in a natural language.

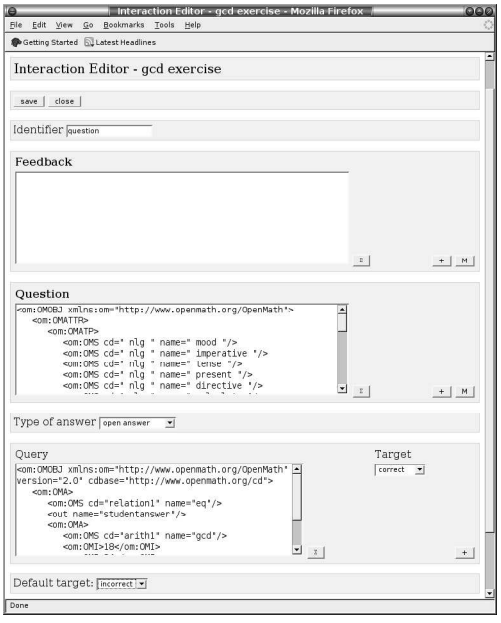


Fig. 3. WExEd: editing a multilingual exercise.

In Fig. 3 it is illustrated how the question interaction of our example exercise is created with WExEd. In the Question field, the multilingual question is inserted using the TextMathEditor. The query that determines the correct answer is created with the WIRIS OpenMath editor [6] and is inserted in the Query field. If the student's answer satisfies the query, he is directed to the interaction specified in the Target field. Otherwise he is directed to the interaction specified in the Default target field.

C. Repository

The editor is integrated with an exercise repository (an XML database containing interactive exercises) and a presentation layer; exercises in the repository can be played with the MathDox player developed by RIACA [8]. Before a multilingual exercise can be actually played, the exercise containing only OpenMath objects should be translated to some natural language. This is done by sending the exercise to the Natural Language Generator. The resulting exercise contains both mathematics (as OpenMath objects) and natural language (in the language predefined by the user).

In Fig. 4 it is shown how WExEd plays the example exercise in English and how the exercise can be played in other languages.

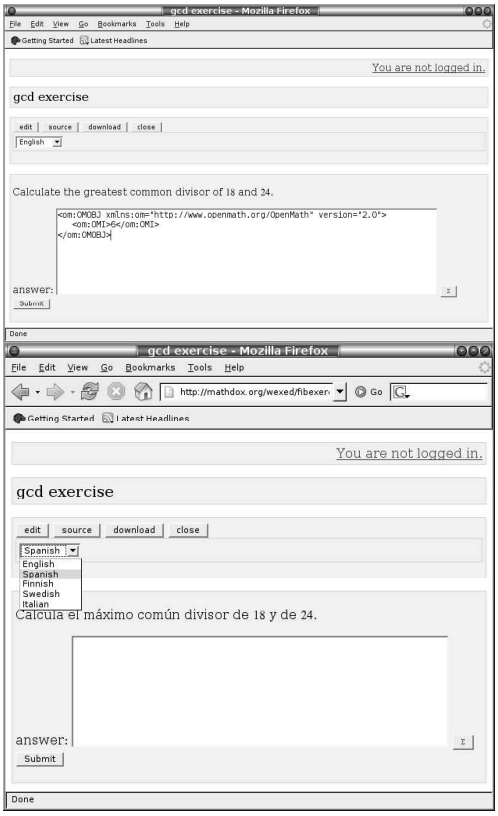


Fig. 4. WExEd: a user can play the exercise in multiple languages.

D. Current Status WExEd

WExEd is still under development, which means that not all functionality has been implemented yet. At this moment, the following features still need to be worked on:

- (Random) variables: an author cannot yet set variables for an exercise using the editor. At the moment, this is only possible by editing the source code of the exercise.
- Elaborate GF grammar: the GF grammar currently used in the TextMathEditor is a very simple, provisional, one. This means that the set of multilingual sentences that can be edited with the TextMathEditor is very limited; the author can only choose from a relatively small set of words. A more powerful GF grammar is being developed.
- Special GF grammars: at the moment, only exercise problem statements can be created with the TextMathEditor and therefore can be multilingual. In the future, special grammars for e.g. feedback and hints will be developed such that these can be multilingual as well.
- Natural languages: although the target languages are English, Finnish, Swedish, French, Italian, Spanish, Catalan, Dutch and German, WExEd is currently able to play the exercises in English, Spanish, Finnish, Swedish and Italian.

The software developed is based on free software, namely on the XML database eXist [9] and the Orbeon Presentation-Server [10] which handles the presentation.

IV. Final Remarks

In this document we described WExEd, the WebALT Exercise Editor with which interactive multilingual exercises can be created and edited, and software integrated with WExEd. This is software which will be further developed in 2006, the second year of the WebALT project.

We note that the example exercise used throughout this paper to illustrate WExEd is a very simple one. Also mathematically advanced multilingual exercises can be created with WExEd. For this, we refer to the live version of WExEd at http://www.mathdox.org/wexed where one can try out the editor.

The work concerning WExEd is sponsored by the eContent project EDC-22253-WEBALT.

References

[1] http://www.openmath.org/overview/
[2] http://www.wolfram.com/products/mathematica/
[3] http://www.maplesoft.com/products/maple/
[4] http://www.rosetta.helsinki.fi/english/index.htm
[5] http://www.cs.chalmers.se/∼aarne/GF/
[6] http://www.mathsformore.com/
[7] http://www.leactivemath.org/
[8] http://www.mathdox.org/
[9] http://exist.sourceforge.net/
[10] http://www.orbeon.com/

Contributed Posters and Demonstrations

CATS - Computer Assessable Task System

Duarte P., Nunes I., Neto J.P., and Chambel T.

Faculty of Sciences, Lisbon University
pduarte@ptmat.fc.ul.pt , in@di.fc.ul.pt, jpn@di.fc.ul.pt, tc@di.fc.ul.pt

Abstract. We present a model of a computer assessment system — CATS (Computer Assessable Task System) — which is based on the rich concept of task. Tasks are designed to assist and assess a student through the resolution of a mathematical problem. While solving a many step problem, students have their answers evaluated and get feedback on wrong answers. Tasks can be used as construction blocks for building other, more elaborated, tasks. One of CATS' goals is to facilitate the construction, by non-expert users, of tasks involving elaborated exercises, from an initial pool of basic tasks. The modular nature of tasks allows easy creation and management of problems with complex system-user interaction, including problems to be solved in several steps. The model structure is extensible so to allow further features such as statistical treatment of students evaluation marks, and virtual tutoring with the ability of adapting to the student's learning profile.

Introduction

We present a model of an interactive computer assessment system that allows:
- Students to solve elaborated mathematical problems, while being given feedback on their mistakes.
- Teachers to easily build new complex interactive problems from a pool of predefined modules.

Concept Model

The model we devised is based on the following main concepts:

Tasks are the building blocks of our approach. They represent mathematical challenges that students must overcome. A task is more than a simple mathematical exercise — it proposes a problem to the student, and then guides, assesses, and provides feedback, throughout the resolution process.

Contexts define a language, and a set of evaluation rules and functions. A Context consists of: *(i)* a grammar, used to decide whether expressions are well formed, or meaningful, through Type checking; *(ii)* a logic program, used to translate logic inference rules, algebraic manipulation rules, theory axioms, problem assumptions, etc; *(iii)* a set of built-in constants, functions, operations and relations which define the context semantics.

Engines are dynamical objects that are created for the evaluation of student answers to given tasks, according to the task Context rules.

Resolution is the sequence of intermediate answers given by the student.

Solution Set comprises the answer type plus a condition to be fulfilled by the answer (the *Success* condition).

Answer is a mathematical term, input by the student, which the system verifies to be in the solution set.

Subtasks are a set of tasks to be performed by the student during the resolution process. These can be either auxiliary or mandatory.

Task Resolution Process

Solving a task may involve several steps (c.f. Fig. 1). Each step corresponds to the resolution of a subtask, which itself is a task. At any stage of resolution, the next step may either be predefined in the task, or else selected by the student, among available tasks.

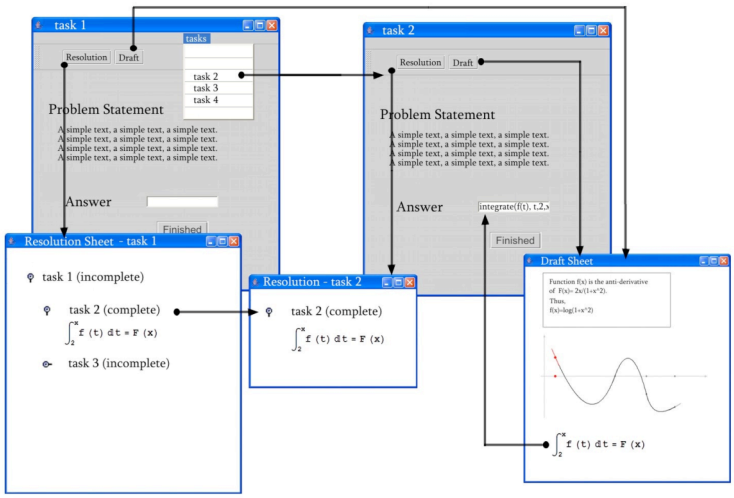


Fig. 1. Task resolution interaction.

During the task resolution process, the system: *(i)* manages the tree of all subtasks under execution; *(ii)* maintains coherence among states of all involved tasks; *(iii)* keeps track of the resolution sheet; *(iv)* provides a draft sheet (common to all subtasks).

Resolution Sheet contains the sequence of intermediate answers given by the student. More precisely, the resolution sheet is a tree formed by the resolution sheets of all undertaken and completed subtasks.

Draft Sheet is used to write and draw marginal notes, or to perform auxiliary calculations. The set of available input tools is determined by task Context. All objects written by the student in the draft sheet are terms of the task Context.

Task Construction

To build a new task, teachers must define:

- the Context in which the problem is defined; it must be chosen from a pool of predefined Contexts;
- a set of typed parameters representing terms (optional);
- a statement that describes the problem and issues the challenge; this can be a parametric statement depending on the above parameters — from it, several instances of the problem are generated;
- an extension to the task Context, e.g., the problem assumptions (optional);

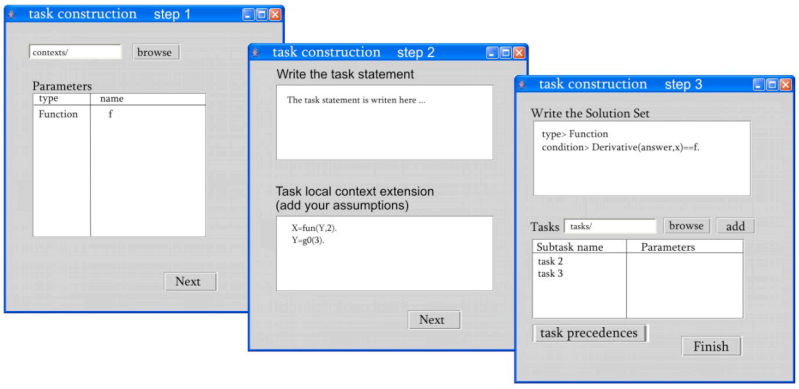


Fig. 2. Task construction interaction.

- the solution set, that is, the answer type, and the *Success* condition (the condition that any answer must fulfill);
- the set of subtasks; to be chosen from a menu of predefined tasks (optional).

By default, a task has no subtasks. In this case, the student has access to all predefined tasks sharing the same Context. This gives the student a great freedom to choose the resolution path he likes. In other words, no guidance is provided by the teacher.

By specifying subtasks, the teacher narrows the set of possible resolution paths, thus helping the student. The teacher can, furthermore, define a precedence graph among subtasks, which amounts to even more guidance through the task resolution process.

Knowledge Representation Model

Mathematical knowledge is represented as terms. Terms are formal expressions in the sense of logic programming languages like Prolog, or computer algebra systems like Maple or Mathematica. In our approach terms are typified.

Types are grammatically defined sets of terms. New types can be built from existing ones in four different ways: *(i) Union* e.g., *A::= B | C | D*, where *A* is the union of types *B*, *C* and *D*; *(ii) Recursion* e.g., *A::= fun(A) | S*, where *A* is the smallest set of terms containing *S*, which is closed under operator *fun*; *(iii) Projection* e.g., *A ::= fun(B,C)*, where *A* is the set of all terms *fun(b,c)* such that *b* and *c* are terms of types *B*

and *C*, respectively; *(iv) Restriction* e.g., *A ::= B[cond]*, where *A* is the subset of terms of type *B* which fulfill condition *cond*. This condition is defined by some expression involving built-in functions.

Thus, a Type is just a mathematically defined set of terms. The following grammar defines the set of all polynomials in variables *x*, *y*, and *z*, with degree less than four:

VAR ::= x | y | z	*(Union)*
SUM ::= POL + POL	*(Projection, Recursion)*
PROD ::= POL * POL	*(Projection, Recursion)*
POW ::= POL ^ INT	*(Projection, Recursion)*
POL ::= SUM | PROD | POW | VAR	*(Union, Recursion)*
POL3 ::= POL [deg(#) < 4]	*(Restriction* through built-in *deg)*

Answer Assessment Engine

Our system engine combines the power of the Object Oriented, Parsing, and Logic Programming (unification + backtracking) approaches.

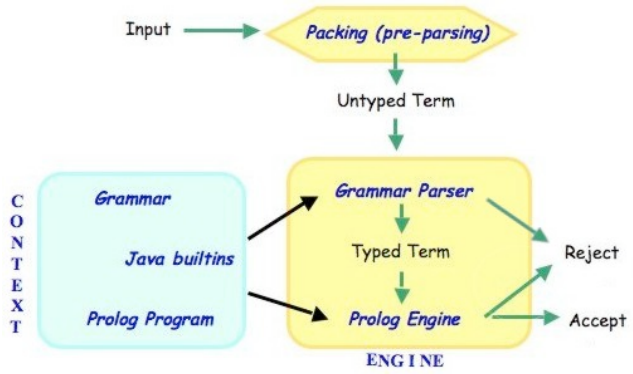


Fig. 3. Answer assessment process.

The semantics of built-in functions and operators is implemented in Java. These built-ins are called from within the Parsing and Logic Programming components of the Engine. In particular, built-ins provide the system messages that give information to the user about his mistakes.

The student's answer is first packed (and pre-parsed if necessary) into an untyped term object. Afterwards, this term object is parsed through the grammar of the task Context, where the correctness of the answer is checked. Finally, the *Success* condition (that is, the one that must be fulfilled by all right answers) is verified against the logic program of the task Context, via the corresponding Logic Programming engine.

An Example

A task, within the Calculus Context, that asks the student to sketch the graph of the anti-derivative of a given graphed function. This is to be done in five steps, from which Fig. 4 shows only the two first ones.

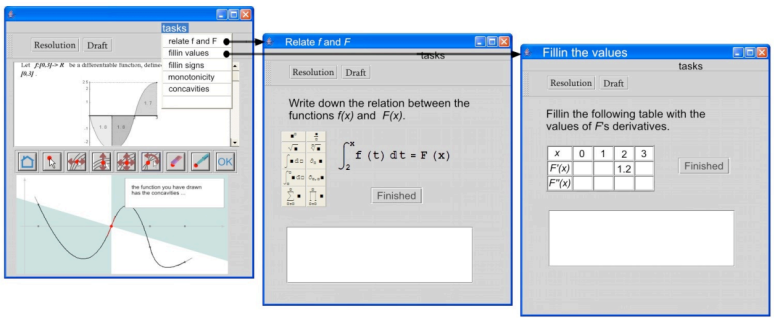


Fig. 4. A Calculus task example.

A more detailed version of this and other examples can be seen at http://perceptron.di.fc.ul.pt/labmag/~cats.

Related Work

Several software packages are available that combine a computer algebra engine/package with Computer Aided Assessment (CAA). Among them are, for example, Aim (http://maths.york.ac.uk/moodle/aiminfo/), Aleks (http://www.aleks.com/), and Calmaeth (http://calmaeth.maths.uwa.edu.au/). On the one hand, these computer algebras, for example Mathematica and Maple, are commercial products, which means that CAA users will need some kind of license; on the other hand, due to the amount of space these CAA packages get to occupy, most of them run on the server side, not the client's. CATS aims at building a CAA open source prototype that is independent from any commercial product, and which can be made accessible to end users through a CD, or through web downloading.

Conclusions and Further Work

The model we devised for CATS is general enough to allow both teachers and students to deal with basic as well as complex tasks while keeping a simple structure at concept level.

Teachers using CATS contexts and tools for task construction, will be able to provide specific libraries of problems to guide their students and, eventually, to share their libraries with a community of users.

We emphasize here that CATS system is designed to cope with a very simple task evaluation cycle, where the concept of task is actually much broader: *(i)* helping the user, or the system, to perform given actions like choosing parameters, selecting a new task, is a task; *(ii)* the draft sheet and its tools are tasks that have system automated answers; *(iii)* the act of creating new tasks, or contexts, is itself a task; *(iv)* a tutor agent that evaluates the student learning profile is a task (the student profile being its input parameter; its Context defining all needed Artificial Intelligence).

Multilingual Generation of Live Math Problems in WebALT

Wanjiku Ng'ang'a*, Anni Laine* and Lauri Carlson

Department of Linguistics
University of Helsinki
Finland
wanjiku.nganga@helsinki.fi, alaine@ling.helsinki.fi, lauri.carlson@helsinki.fi

Abstract: We describe the use of the Grammatical Framework (GF), a parsing and generation formalism, to implement grammar driven multilingual natural language generation of mathematics problems. The mathematical content is encoded in OpenMath facilitating semantically transparent, constrained multilingual parsing and generation from mathematically well-formed and interpreted input, while natural language extensions to OpenMath define linguistic hints and attributes which determine the final form of the generated natural language sentences. The generation process starts by parsing the OpenMath input into a language-independent representation which is then transformed into a language-dependent Natural Language (NL) tree. This is done by choosing corresponding lexical terms for each of the OpenMath symbols contained in the input. The GF resource grammars are then applied to the NL trees to generate the final multilingual sentences.

The WebALT Generation Task

The generation task in WebALT is to develop a Natural Language (NL) Generator that produces multilingual variants from mathematical content. By analyzing the linguistic form of different types of mathematical texts, two generation plans were adopted. Plan A targets the shorter types of texts common in numerical problems, while plan B deals with longer texts such as those found in verbal problems and instructions. The use of XSLT transformations (Wilcock, 2001) will be investigated in plan B. In this paper, we report on the generation approach adopted for plan A.

The types of mathematical problems typical in plan A can be encoded in a formal representation such as OpenMath (Caprotti et al, 2004). In plan A therefore, generation is done from an abstractly-represented object to produce multilingual NL renderings, based on generation hints as specified in the NLG content dictionary extension to OpenMath (Caprotti et. al, 2005). The Grammatical Framework (GF) (Ranta, 2004) has been used as the generation framework within WebALT as it meets the necessary generation requirements. Due to its multilingual aspects, it lends itself

* These authors were supported by the eContent project WebALT-EDC-22253.

naturally to the multilingual demands within WebALT. In addition, a GF grammar allows for the specification of an abstract representation and multiple concrete realizations, making it suitable to handle generation from a formal encoding of mathematics such as OpenMath. The GF libraries provide multilingual resource grammars which define language-specific grammatical constructs and morphological paradigms. Currently, GF resource grammars are available for English, Finnish, French, Spanish, Italian, Russian, Swedish, Danish and Norwegian with other languages such as German under development.

WebALT GF Grammars

By using the definitions provided in the GF resource libraries, application development in GF is restricted to defining only application-specific grammars. NL generation involves a three-stage process which converts an OpenMath input to its NL equivalent. The first two stages map the OpenMath input to its language-specific rendering while the final stage uses the GF resource grammar libraries to generate the final sentences. The first two stages and their implementation grammars are briefly described in the subsequent sections.

OpenMath Layer

This layer accepts the OpenMath input object and parses it into an internal tree representation. The OpenMath grammars define the surface realization of the OpenMath symbols in both the prefix and XML notations. These grammars can parse all 1067 symbols defined in 152 OpenMath content dictionaries[1].

Language Layer

The language layer provides a mapping between OpenMath and the GF resource grammar functions by augmenting the OpenMath tree (produced by the OpenMath layer), with language-specific linguistic and lexical features. The output of this layer is then transformed by the GF resource grammars into corresponding NL sentences.

The language layer produces a language-dependent representation of the OpenMath input and relies on information contained in the concept lexicon and corresponding multilingual lexicons. WebALT's concept lexicon lists words (497 nouns, 86 verbs, 120 adjectives and 14 closed class terms) that express the mathematical concepts contained in the OpenMath content dictionaries. Translations of the concept lexicon into WebALT's set of target[2] languages has been done in close collaboration with mathematicians who possess the domain knowledge in these

[1] This figure includes the 24 symbols which have been defined within WebALT as extensions to OpenMath.

[2] These languages are: English, Finnish, Spanish, French, Italian and Swedish.

languages. Figure 1 illustrates the relationship between the abstract lexicon (conceptual) and the corresponding concrete lexicons (language-specific concepts).

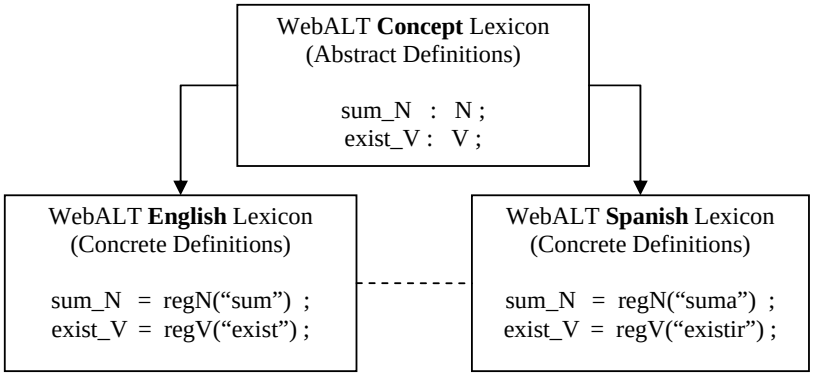


Figure 1: Lexicon Structure

The language layer also defines language-specific generation rules that determine the grammatical form and linguistic style of the NL rendering for a given symbol. The current generation approach is based on tree rewriting where the OpenMath tree is transformed into an equivalent NL tree, using the transformation mechanisms available in GF.

Defining the generation rules presupposes knowledge not only of the GF resource grammar API but also of the grammatical structure of the specific language as well as any special stylistic preferences for rendering mathematical sentences in the given language. For example, while many languages employ the imperative mood to express instructions in mathematical sentences, French prefers the infinitive form of the main verb, which serves as a polite imperative, as shown below:

EN: Calculate x + y.
FR: Calculer x + y.

Taking advantage of GF's multilingual and highly modular structure, it is possible to define language-independent generation rules that apply to any language or language family, or language-specific rules that handle exceptional cases. In the previous example, to produce the desired sentence mood, a default rule that uses the imperative mood for expressing instructions in most languages is defined, while a second rule is defined to handle the French exception.

Generation rules have been defined for a total of 508 symbols, 232 of which belong to WebALT's initial target domains[3].

[3] Calculus, Geometry and Linear Algebra.

Generation Examples

limit1:limit(0,limit1:null,fns1:lambda[x].(arith1:divide(arith1:minus(1, transc1:cos(x)), transc1:sin(arith2:times(2,x)))))

Determine the limit of $\frac{1-\cos(x)}{\sin(2x)}$ as x tends to 0.

Määritä lausekkeen $\frac{1-\cos(x)}{\sin(2x)}$ raja-arvo kun x lähestyy lukua 0.

Determina el límite de $\frac{1-\cos(x)}{\sin(2x)}$ cuando x tiende a 0.

Bestäm gränsvärdet av $\frac{1-\cos(x)}{\sin(2x)}$ då x går mot 0.

Déterminer la limite de $\frac{1-\cos(x)}{\sin(2x)}$ quand x approche de 0.

Determina il limite di $\frac{1-\cos(x)}{\sin(2x)}$ quando x tende a 0.

plangeo3:distance(plangeo1:point(x), plangeo1:point(y))

What is the distance between point x and point y?

Mikä on pisteen x ja pisteen y välinen etäisyys?

Qué es la distancia entre el punto x y el punto y?

Vad är distansen mellan punkt x och punkt y?

Quelle est la distance entre le point x et le point y?

Qual' è la distanza tra il punto x ed il punto y?

Conclusion

We have described a semantics driven, constrained language NL generator that has been implemented to support multi-lingual and multi-cultural online learning of mathematics. The current implementation of the WebALT NL generator is constantly being extended to cover more OpenMath symbols. In addition, since GF is an evolving framework, as the resource grammars are extended to include a wider variety of linguistic constructs and new languages, the generator will likewise produce more varied NL renderings of OpenMath, and for more languages. With a view to consolidate all the GF-based grammars defined for various WebALT

components into a common grammar, there is a plan to redefine the generator's abstract syntax so that it can be shared by all other concrete grammars.

Although there is a wealth of previous work on NLG, there are few truly multilingual systems on the market ready to be directly exploited for multilingual generation of mathematics problems to be used in WebALT or in any other similar online tutoring system. The GF framework has proved to be well suited for the WebALT generation task.

Credits

This document is based on joint work by the authors. We acknowledge contributions from the rest of the WebALT Team, especially Mika Seppälä, Olga Caprotti, Andreas Strotmann and Jordi Saludes. In addition, we are very grateful to Aarne Ranta for providing us with the GF tools and documentation, and for sharing his ideas with us.

References

Caprotti, O., Ng'ang'a, W. and Seppälä, M. 2005. Multilingual Technology for Teaching Mathematics. *In Proceeding of the International Conference on Engineering Education, Instructional Technology, Assessment, and E-learning (EIAE 05)*.

Caprotti, O., Buswell, S., Carlisle, D., Dewar, M., Gaëtano, M. and Kohlhase, M. (ed.) 2004. *The OpenMath Standard - Version 2.0*. Report of the OpenMath Society. <www.openmath.org>

Ranta, A. 2004. Grammatical Framework, a Type-theoretical Grammar Formalism. *Journal of Functional Programming*, 14(2):145-189.

Wilcock, G. 2001. Pipelines, Templates and Transformations: XML for Natural Language Generation. *In Proceedings of the First NLP and XML Workshop, NLPRS-2001*, Tokyo, pp. 1-8.

WExEd Software Demonstration Description

Arjeh Cohen* Hans Cuypers* Karin Poels* Mark Spanbroek* Rikko Verrijzer*
amc@win.tue.nl hansc@win.tue.nl k.j.p.m.poels@tue.nl m.j.m.spanbroek@tue.nl r.verrijzer@tue.nl

*Department of Mathematics and Computer Science
Eindhoven University of Technology
Eindhoven, The Netherlands

Abstract— **Recently, the computer program WExEd was developed within the WebALT project. It enables one to create and edit interactive mathematics exercises that can be automatically translated to and subsequently played in different languages (currently English, Spanish, Finnish, Swedish and Italian). It makes use of OpenMath, a standard for encoding the semantics of mathematical objects in XML, and software developed within WebALT. WExEd does not only encode the mathematics of the exercise in OpenMath but also the text. This is possible because, in general, mathematics exercises are built from a well defined and relatively small set of words. In this paper we give a software demonstration description of WExEd.**

I. Introduction

Although mathematics itself is universal, it is taught in many different languages around the world. This implies that, by restricting themselves to a specific language, publishers and authors of mathematical exercises only reach a small part of the possible audience. If the exercises could be automatically translated to different languages, then a publisher could extend his market dramatically.

Automatically translating exercises is hard because of large lexicons and complex grammars. It becomes easier when the exercises are constructed within a well defined context from only a small and specific part of a language's lexicon. In general, this is the case for mathematics exercises, which can be as simple as "Calculate the derivative of $\sin$", but also the words from more advanced exercises may come from a rather small and specific set of the language's lexicon.

OpenMath [1] is an emerging standard for representing the semantics of mathematical objects in XML. It allows mathematical objects to be exchanged between computer programs, stored in databases or published on the Worldwide Web. Each mathematical symbol (e.g. $\cos$) has its own OpenMath representation and OpenMath representations are grouped in so-called Content Dictionaries (CD's). Every combination of mathematical symbols can be written as one OpenMath object. By ways of illustration, the OpenMath representation of $\cos(1/x)$ is shown below. Note that the mathematical symbol "cosine" has the OpenMath representation "$\cos$", defined in the CD "transc1" on transcendental functions.

```
<OMOBJ>
  <OMA>
    <OMS cd="transc1" name="cos"/>
    <OMA>
      <OMS cd="arith1" name="divide"/>
      <OMI>1</OMI>
      <OMV name="x"/>
    </OMA>
  </OMA>
</OMOBJ>
```

WebALT (Web Advanced Learning Technologies), an e-learning project funded by the EU, uses OpenMath to automatically translate mathematics exercises into natural languages. To do so, not only mathematics is encoded in OpenMath, but also text so that every sentence in the exercise can be written as a single OpenMath object. To encode text in mathematics exercises into OpenMath, a mathematical lexicon and OpenMath representations for all words in this lexicon were created within WebALT. This resulted in the CD named "nlg"; it contains OpenMath symbols for e.g. "calculate", "determine", "prove", etc. In order to translate the OpenMath object representing a sentence into a natural language, grammar generation rules corresponding to the newly created OpenMath representations were formulated and an existing grammar formalism was used. The tool that automatically translates exercise sentences is the Natural Language Generator. The tool that creates multilingual exercise sentences (in the form of OpenMath objects) is

the TextMathEditor. These tools will be further described in the following section.

For creating and editing interactive multilingual mathematics exercises we developed the editor WExEd - WebALT Exercise Editor. Exercises created with WExEd can be automatically translated to various natural languages because the source code of the exercise entries consist only of OpenMath expressions. Within WExEd this code is generated by the TextMathEditor. Moreover, exercises created with WExEd are highly interactive; the exercises can be multistep, allow for random input variables and offer the possibility to check and interpret the student's answer with the help of computational services like a Computer Algebra System (CAS) such as Mathematica [2] or Maple [3]. With random variables, a student can solve multiple instances of the same exercise. Because of the capability of reading and understanding the student's answer, WExEd can prompt the student with a following exercise or task depending on the student's answer. The student can therefore be guided through the various steps of the exercise.

In this document we speak about *sentences* or *exercise sentences* when we mean sentences in an exercise such as e.g. the problem statement or the predefined answers in a multiple choice exercise.

In this document we describe the functionality of WExEd in Section II and in Section III we give a WExEd software demonstration description.

II. Functionalities of WExEd

The formal language underlying WExEd is the Le-ActiveMath Exercise Language [5] which is a collection of XML tags together with restrictions imposed on their use. This language is such that exercises are:

- Interactive: have means of exchanging information with the student by being able to read the student's answer and to give feedback and/or hints.
- Automatized: have means of checking whether the student gives the correct answer by connecting for example to a CAS. The author gives a query in OpenMath which determines the correct answer. This query is translated to a CAS' language by using phrasebooks. The CAS can then calculate the correct answer and can subsequently compare the student's answer with the correct answer.
- Adaptive: takes appropriate action depending on its knowledge of the student and/or the student's answer to a particular question; exercises are made up of *interactions* which consist of information given to the student (e.g. a question or feedback to the student's answer) and possibly information provided by the student (e.g. an answer to a question). Depending on the student's answer he is directed to a following interaction (which could be the same question, a question that will teach the student how to solve the previous question, a new question, feedback, etc.). For illustration, if a student answered our example exercise incorrectly, the student could be directed to the interaction that contains the question "What are the common prime factors of 18 and 24?".
- Reusable: (random) variables can be defined in an exercise so that students can do multiple instances of the same exercise. The values of the variables are processed by a CAS using phrasebooks.

The exercise language, as any XML-language, is extensible in the sense that new tags can be added at any moment to facilitate new features.

WExEd supports multiple choice exercises, open exercises and multistep exercises combining these exercise types.

The exercises created with WExEd are multilingual because all exercise sentences are created with the TextMathEditor. When editing an exercise with WExEd, the author is presented with the TextMathEditor every time the input of a sentence is required (e.g. the exercise problem statement, the choices in a multiple choice exercise, feedback or hints). In this way, all entries of the exercise consist of OpenMath objects which could be sentences containing mathematics or completely mathematical expressions. The exercise is therefore multilingual; it does not contain text in a natural language.

The editor is integrated with an exercise repository (an XML database containing interactive exercises) and a presentation layer; exercises in the repository can be played with the MathDox player developed by RIACA [6]. Before a multilingual exercise can be actually played, the exercise containing only OpenMath objects should be translated to some natural language. This is done by sending the exercise to the Natural Language Generator. The resulting exercise contains both mathematics (as OpenMath objects) and natural language (in the language predefined by the user).

The software developed is based on free software, namely on the XML database eXist [7] and the Orbeon PresentationServer [8] which handles the presentation.

WExEd is still under development, which means that not all functionality has been implemented yet.

III. Demonstration

In the following sections we describe a software demonstration of WExEd, as to be given at WebALT2006 [4], the first WebALT conference and exhibition.

During the demonstration we will navigate through the exercise repository surrounding WExEd to demonstrate the nested structure of folders and exercise files.

The demonstration is divided into the following three parts. In the first part, an exercise is created with the editor WExEd and played in various languages. In the second and the third part, we will play exercises that were created by editing the source code. This showcases the advanced future functionalities of WExEd.

A. Create and Play New Exercise

In this part, the functionality of WExEd and the TextMathEditor is demonstrated by creating a multilingual exercise with WExEd. We explain this part step by step and illustrate it with various screenshots.

1) We first create a new, empty, exercise in the exercise repository surrounding WExEd (Fig. 1).

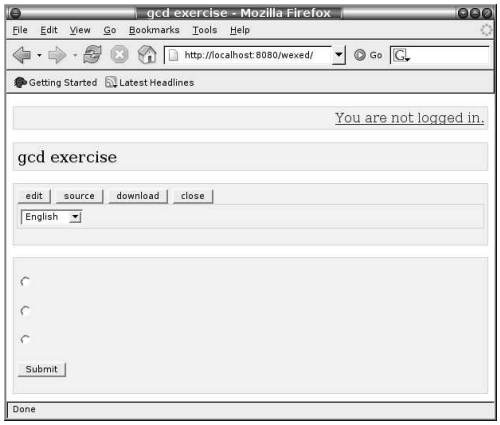


Fig. 1. WExEd: an empty exercise.

2) The exercise is opened and we choose to edit it. The editor now shows all existing interactions which is only one for an empty exercise; the default interaction *question* (Fig. 2).
3) We choose to edit this interaction.

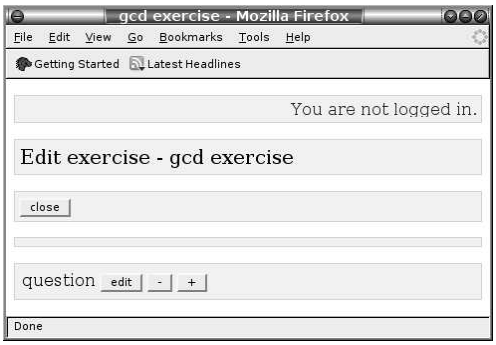


Fig. 2. WExEd: the default interaction "question".

4) We write the exercise problem statement using the TextMathEditor (Fig. 3, note that on the background the empty fields of the multiple choice exercise can be seen). This makes the exercise multilingual.

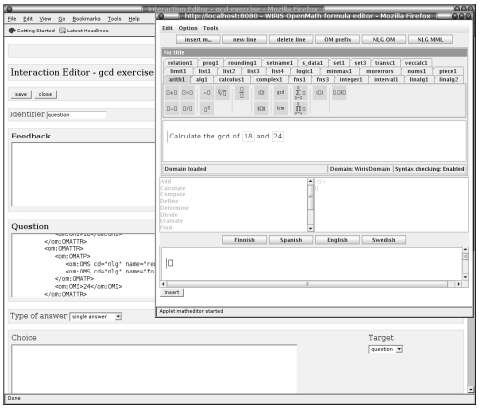


Fig. 3. WExEd: using the TextMathEditor to create the question.

The exercise we create is a multiple choice exercise and a student can be directed to different interactions for every choice. If the student answers correctly (incorrectly) then we want to send him to the "correct" ("incorrect") interaction. For this, we create new interactions (remember that we now only have the *question* interaction), see Fig. 4.

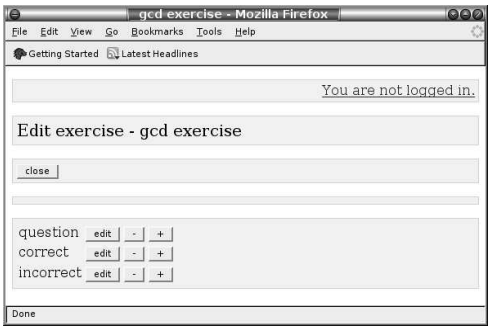


Fig. 4. WExEd: creating new interactions.

We fill in the various choices using the TextMathEditor and connect a different interaction to every choice (Fig. 5).

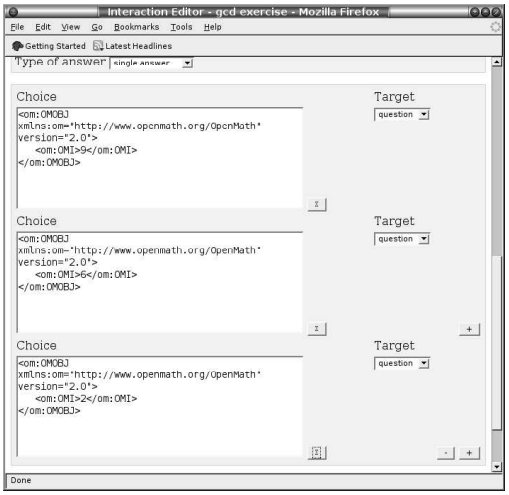


Fig. 5. WExEd: linking interactions to the multiple choices.

5) We then play the multiple choice exercise in multiple languages (Fig. 6).

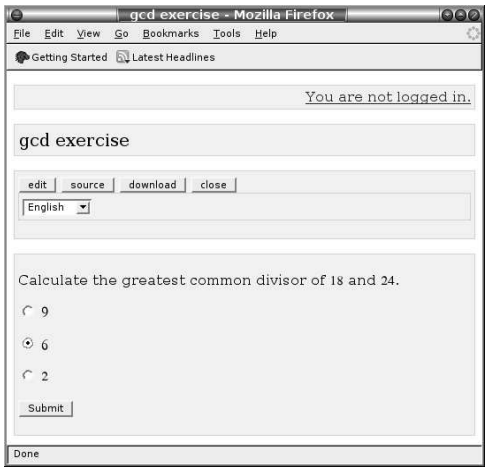


Fig. 6. WExEd: playing the multiple choice exercise.

6) We edit the exercise to make it an open exercise, which is done by changing the type of answer (Fig. 7).

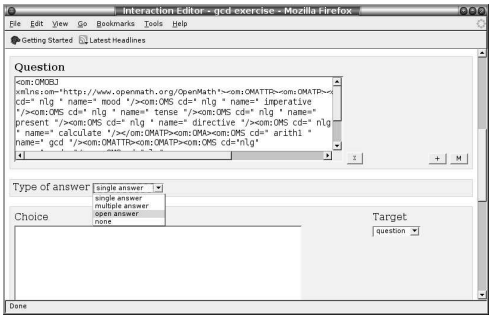


Fig. 7. WExEd: the open exercise.

Using the WIRIS math editor [**?**], we insert a query that determines the correct answer to this open exercise (Fig. 8).

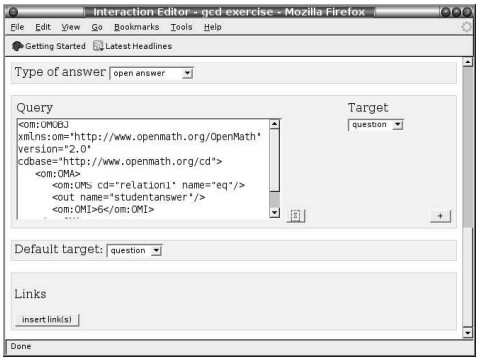


Fig. 8. WExEd: inserting a query and linking connections.

The student can give a correct or an incorrect answer to the open exercise. We connect two different interactions to these two scenarios (see Fig. 8).

7) The open exercise is played in multiple languages (Figs. 9 and 10).

Fig. 9. WExEd: playing the open exercise.

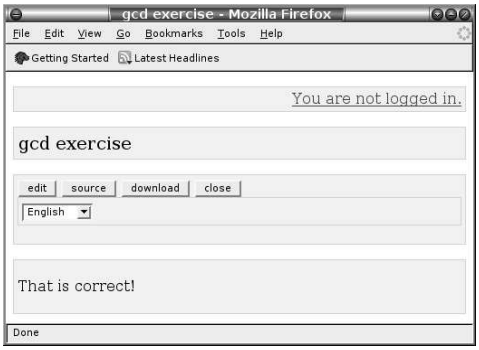


Fig. 10. WExEd: feedback to student's answer.

B. Demonstrate and Play Existing Advanced NL Exercise

In this part, the functionality of the Natural Language Generator and the future functionality of the TextMathEditor are demonstrated. This is done in the following way:

1) We start with the exercise created in the first part of the demonstration.
2) Before the conference, we change this exercise by source code editing. We only change the wording of the exercise problem statement and possibly the query defining the answer so that as a result, the exercise contains natural language that cannot be edited with the TextMathEditor but can be translated with the Natural Language Generator.
3) The exercise is played in multiple languages (at least 2).

C. Demonstrate and Play Complex Existing Advanced NL Exercise

In this last part, the richness of the LeActiveMath Exercise Language and the future functionality of WExEd are demonstrated. This is done in the following way:

1) Before the conference, we created an advanced and interactive exercise by editing the source code. The exercise will showcase future functionalities of WExEd, like random variables, media items, etc. to make the exercise richer. An example of such an exercise is one in which the student is asked to do a differentiation. If the student gives an incorrect answer then he/she is guided through the exercise step by step by many different interactions until he/she can answer the original question.
2) The exercise is played in multiple languages (at least 2).

IV. Final Remarks

In this document we described WExEd, the WebALT Exercise Editor with which interactive multilingual exercises can be created and edited, and software integrated with WExEd. This is software which will be further developed in 2006, the second year of the WebALT project.

The work concerning WExEd is sponsored by the eContent project EDC-22253-WEBALT.

References

[1] http://www.openmath.org/overview/
[2] http://www.wolfram.com/products/mathematica/
[3] http://www.maplesoft.com/products/maple/
[4] http://webalt.math.helsinki.fi/webalt2006/
[5] http://www.leactivemath.org/
[6] http://www.mathdox.org/
[7] http://exist.sourceforge.net/
[8] http://www.orbeon.com/

Author Index