UNIVERSITY OF JOENSUU
DEPARTMENT OF COMPUTER SCIENCE
Report Series A

Kolin Kolistelut – Koli Calling 2001

PROCEEDINGS OF THE FIRST ANNUAL FINNISH / BALTIC SEA CONFERENCE ON COMPUTER SCIENCE EDUCATION

October 19–21, 2001
Koli, Finland

Report A-2002-1

ACM K.3
UDK 681.3
ISSN 0789-7316
ISBN 952-458-089-6

Foreword

University of Joensuu organized the first annual Finnish / Baltic Sea Conference on Computer Science Education in Koli, Finland, on October 19 – 21, 2001.

The main reason for arranging the conference was to attract interested scholars and educational technologists within the universities both in Finland and in the Baltic Sea and Nordic countries to join their efforts to figure out the future prospects on the field of Computer Science Education. The goal of the conference was to develop the exchange of revelant information between colleagues working within the same discipline. Topics of interest included, but were not limited to:

- Teaching Practices
- Educational Technology
- Distance Education
- Virtual Universities
- Computer Science Education Research
- Educational Software
- Tools for Visualization or Concretization

The conference program consisted of short presentations, open discussion sessions and social events. The Kolin Kolistelut - Koli Calling was a relaxed and unformal event and creative new ideas were emerged.

We wish that the first Kolin Kolistelut - Koli Calling was a starting point for a deep collaboration between people who share the same interest in the field of Computer Science education. To promote the collaboration, the conference will be organized annually in future.

Joensuu, January 28, 2002

Erkki Sutinen *Marja Kuittinen*

Program committee

Erkki Sutinen (Chairman)	University of Joensuu, Finland
Jaakko Kurhila	University of Helsinki, Finland
Martti Penttonen	University of Kuopio, Finland
Tapio Salakoski	University of Turku, Finland
Jorma Tarhio	Helsinki University of Technology, Finland
Marja Kuittinen (Publication editor)	University of Joensuu, Finland

Organizing committee

Jarkko Suhonen (Head)	University of Joensuu, Finland
Jaana Lukkarinen (Local arrangements)	University of Joensuu, Finland
Niko Myller (Webmaster and technical support)	University of Joensuu, Finland

Contents

SESSION 1

Concretising tools for Computer Science Education

G. La Russa
University of Joensuu, Department of Computer Science
P.O. Box 111, FIN-80101 Joensuu, FINLAND

larussa@cs.joensuu.fi

1 Abstract

A new educational, if not cultural revolution must take place to help kids, and in particular teenagers, to master the ABC of Computer Science: programming skills. This paper deals with the creation of advanced tools that let kids learn in their own way, which is playing, what adults have created to shape their own and their own children's cybernetic spaces: computer programming.

2 Introduction

Third millennium has started and technology is developing at giant steps. Our societies are challenged to reshape their educational structures to cope with the ever-growing need for computer based systems. Computer programming (as well as Computer Science in general) can not be left confined to the world of adults while societal and marketing needs pose their demands on our new generations' education. Moreover, there are many fields of education that can benefit from a proper integration of concretising tools to newer technologies. The approach of computer science to cognitive and skills' development processes of novices has been often considered a problem-solving, thinking and understanding in the domain of mathematics from the perspective of internal mental structures. Researchers have often focused their attention upon experts understanding and how it differs from that of novices (behaviour/skills' comparison and research of solutions for existing gaps). Recently cognitive science has extended the concept of internal mental structures [1] to include the interactions [2] that take place between a person and its environment, combination of both virtual and real one.[3]

In this paper I provide a highlight on researches and discoveries that do constitute a background to my own research agenda.

3 Physical domains and physical conveners

Certain physical representations can be used not only to describe objects but also to convey concepts belonging to the non-physical environment (hidden cognitive elements). These representations are capable of keeping a sort of events' history of the environment and to interact with users stimulating the development of new concepts. Thinking and understanding have a natural concrete basis in the realistic and/or imaginary perception of the cognitive data. Representations are nothing less than concrete mental models and are fundamental to human thought processes. The mental construction and manipulation of concrete mental models is always influenced by already existing cognitive models.

A computer-based learning environment can facilitate the construction and use of mental models but in many cases it requires a physical convener to bridge abstract concepts to *concrete* mental models [4]. This is proven by the fact that cognition is not a purely mental process but a system that includes the individual perception of own context and the available tools.

Computer-based learning environments taking into account the role of the physical domain facilitate the learning process because of the concretisation of abstract thinking. This brings to the obvious conclusion that visually concrete objects can be linked to more formal and abstract concepts triggering a more dynamic learning process.

4 Constructionism

Constructionism is a developing theory of learning and education and is based on the idea that people learn by actively constructing new knowledge, rather than having information "poured" into their heads. Moreover, constructionism asserts that people learn with particular effectiveness when they are engaged in constructing personally meaningful artefacts, such as computer programmes, animations, or robots. The theory of constustionism can be defined as a technique for learning and furthermore a strategy for education. People learn when they are engaged in building a "personally meaningful" project. Also, students may become more deeply entwined in their own learning process if they are forming or building something that others may view, evaluate, or even implement into their own lives. Ultimately, through that development process students will run into various problems or hurdles and because they are motivated by what they are creating they will be more apt to learn and fix the problems themselves. Seymour Papert used the term "contructionism" for his favoured approach to learning. He wrote that constructionism is built on the assumption that children will do best by finding ("fishing") for themselves the specific knowledge they need. Organized or informal education can help most by making sure they are supported morally, psychologically, materially, and intellectually in their efforts. In research fields there is some talking about the use of the words "constructionism" and "constructivism". It is worthy therefore to highlight here that constructionism differs from constructivism [5] in the fact that it is more closely than other educational -isms at the idea of mental construction. Constructionism attaches special importance to the role of constructions in the world as a support for those constructions in the head, thereby becoming less of a purely mentalist doctrine. As examples of constructionist learning activities we can refer to measuring quantities while making a cake, building with Lego or working with the robotics programming language Logo (LEGO Mindstorms Robotics System). [6]

5 Things that teach

Things that teach is a new theory of continuous learning. A paradigm for this concept is learning to juggle using balls that know where they are at any time giving continuous feedback to the learner.

Things that teach is somehow an extension of the inferential principle to the world of interactivity: the gaming/playing produces new understanding thanks to the "teaching" ability of the objects.

6 Things that learn and then teach

One of the most obvious consequences of this approach is that wireless devices can be connected to and controllable from remote locations. Such devices may or may not have their own processors (which, in some cases, would render the device partially "independent" from remote control). The concept of Application Service Provider involves devices that appear to compute but in reality they are using the computing resources of a service provider through a wireless connection to the Internet. These sorts of devices hold huge potential for educational technologies. Toys with variable behaviour, books with changeable contents are examples of "things that learn and then teach"[7]. These devices could become the eyes, ears and mouths of a seamless, limitless, wireless world.

7 Building-block programming

Building-Block Programming is a new approach to computer programming. It is based on a visual interface composed of blocks that allow the creation of new programmes with low thresholds but high ceilings. As in the case of the Logo system adopted by LEGO (LogoBlocks) for Mindstorms Robotics System, Building-Block Programming enables kids (i.e. novice programmers) to make a smooth transition from using simple commands to the creation of complex programs. Using LogoBlocks"[8] languages, kids create programs by snapping virtual LEGO bricks together on the screen.

8 Building block programming and graphical representation

It is obvious that graphical programming has advantages and disadvantages against textual programming. The use of graphical representations of objects allows a more concrete view of object orientations (with a simple double click on an object suitcase it can be seen what is inside it), eliminates annoying programming syntax and better visualises the pathways that the program is following. Parallelism between programmes can be made more explicit and all of the different program clusters on the screen can run at the same time. One of the graphical programming advantages is the use metaphors or symbolism from real life to make programming easier. For example, to render more intuitively the programming of a light switch that must be turned on and off can be done by setting a time on a clock and so on.

Graphical programming allows easy sharing of programs. Own programs can be defined in particular blocks and given as blocks to other people for use and testing. It is worthy to mention the easy way of block programming to move in and around its blocks. In fact it is more easy to look at a picture of a program to discern its meaning than looking at a large textual program that is composed of many code files.

Probably one of the best advantages in block programming is the use of visual cues in graphical languages. Connections between various objects can be made more clear through the design and graphical representation of the elements.

Unfortunately in block programming there are also disadvantages. Some graphical languages are so "graphical" that often lead sophisticated programmers (used to express statements in a concise textual way) to frustration. Desktop environment is also a limiting factor. The problem with visual programming is that it is not possible to have more than 50 visual primitives on the screen at the same time making a bit difficult writing an operating system. But in order for icons and graphics to be understandable they need to be enough big to be seen or otherwise have a textual label. Some languages also represent function calls by lines between clusters of graphics. But still, if there are too many functions on a screen the code becomes confusing and hard to be understood.

Language extendibility is also a problem. Graphical languages usually are limited to author's design without thought of possible additional features.

In block programming there is a strong appeal for children because they like to manipulate blocks and put together collections of objects. Due to the fact that a major problem in teaching children programming is found in the syntax understanding and that graphical programs tend to eliminate syntax problems, children find it easier to create programs with block programming. Many interesting projects can be made using very short and funny programs. With time children grow more technologically experienced and this allow them to shift to textual languages and use their greater complexity.

9 Programmable bricks

Child's construction kit and building computational power are extended directly into LEGO bricks. Children use these Programmable Bricks to build everything from robotic creatures to interactive kinetic sculptures, and in the process, learning about engineering and design. A family of Programmable Bricks, known as Crickets, is currently under development that is especially well suited for rapid prototyping of computerised constructions. The LEGO company now sells a product (part of its Mindstorms product line) based on a Programmable Brick research.

10 Creation processes of concretising tools

The existing and developing collection of specific *scientific literature* and *articles*, represented graphically in the Fig. 1 as elements Lit1, Lit2, ... , Lit$n$ and Ar1, ... , Arn do constitute the *theoretical framework* for concretising tools. Most of this scientific literature does belong to *Research Branches*, such as *constructionism* and *things that teach*. On the basis of the so formed theoretical framework some *preliminary models* (depicted in Figure 1 as A1, A2, ... , An) will be constructed. These models consist of visual programming, robotic devices, interactive tools and intelligent learning environments. The *test* field will be constituted at its core of a children club (called *Kids' Club*) acting as a live laboratory for learning by gaming. The testing phase is thought to reshape or add to the theoretical framework each time a preliminary model does not fit as a proper learning tool"[9] (which means that the model has not passed its test). It is expected that the testing of a model will probably fail, at least in part, requiring its re-elaboration in its theoretical framework. This recursive process is eventually going to lead to the formulation of a good model that will pass the test phase. At that moment a concretising tool will be ready.

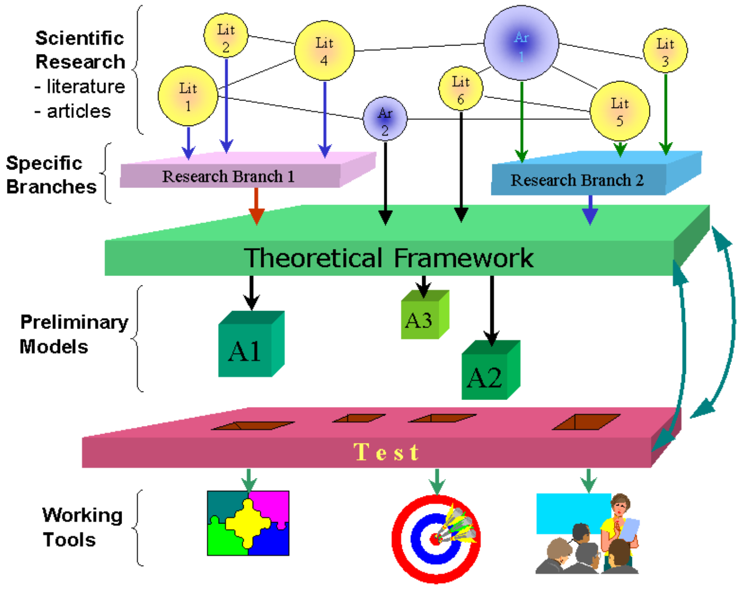

Figure 1: Creation processes of concretising tools

11 Conclusions

Concretising tools will focus on the following the goals:

- To provide computer-based *concrete learning tools* to aid programmers' mental models and to develop programming skills by using the existing interaction between the concrete tools (i.e. real world) and the abstract (i.e. scientific concepts).
- To incorporate *direct manipulation and workability* within and between the concrete tools and the abstract concepts to support the transition from novice to expert thinking. Its aim is that computer programming would enter a new phase becoming a more easy matter than traditionally thought.
- To create a *constructionist approach* (in which novices learn through a process of designing, inventing, and experimenting) to open new learning opportunities. The re-

search results could be integrated into other existing computerised tools such as "Woven Stories"[10] and "Virtual Approbatur"[11].

It has been already demonstrated that learning environments are useful tools to learn principal elements of programming (like control structures) with minimal teaching effort. During the test of the *Empirica Control* software tool intended for technology education a problem was highlighted: more complex structures did still require teacher intervention to achieve advanced outcomes. Students could construct with *Empirica Control* programs to control technological processes or systems. Each program structure corresponded to a factual event in the learner's physical environment and not only to a visual presentation on a screen. This showed that there was a need to further investigate where is the balance between student-centered exploration and teacher-guided instruction in open learning environments"[12]. Defining this balance point in children's learning skills would allow to move forward on how to increase children' interactive learning.

As a consequence of the argumentation presented in this paper it is expected that the creation of bridging systems (concretising tools) where interactions between tools (i.e. robots) and mental abstract concept (i.e. use of robot programming languages such as ANSI C and NQC) would lead children to a faster acquisition of programming skills.

References

[1] M. Dbor, "Support for problem formalisation in engineering design: An enquiry into the role of knowledge level models," *The 10th International DAAAM Symposium - Vienna University of Technology*, Vienna, Austria, 21-23 October (1999).

[2] V. Sauter, "Intuitive decision-making," *Communications of the ACM* **Volume 42**, 109–115 (1999).

[3] R. Beer, H. Chiel, and R. Drushel, "Using autonomous robotics to teach science and engineering," *Communications of the ACM* **Volume 42**, 85–92 (1999).

[4] J. Carroll and A. Aaronson, "Learning by doing with simulated intelligent help," *Communications of the ACM* **Volume 32**, 1064–1079 (1988).

[5] M. Ben-Ari, *Constructivism in computer science education* (SIGSCE, Atlanta GA USA, 1998).

[6] S. Papert, *The children's machine - Rethinking school in the age of the computer* (Basic Books, New York USA, 1993).

[7] S. Mitra, "Children and the internet: New paradigms for development in the 21st Century," *Basis for the keynote speech at the Asian Science and Technology Conference in the Year 2000, Tokyo, Japan, June 6*, 14–16 (2000).

[8] A. Begel, "LogoBlocks: A graphical programming language for interacting with the world," *Epistemology and Learning Group, MIT Media Laboratory, May 24*, 16–33 (1996).

[9] M. Martinez, *Designing international learning environments* (SIGDOC, Snowbird, Utah USA, 1997).

[10] P. Gerdt, P. Kommers, C.-K. Looi, E. Sutinen, "Woven stories as a cognitive tool," *Proceedings of The Fourth International Conference on Cognitive Technology, CT2001, Warwick, University of Warwick, August 6-9*, 233–247 (2001).

[11] A. Haataja, J. Suhonen, S. Kontkanen, E. Sutinen, "Teaching University-level computer science to high school students over the web," *ED-MEDIA 2001, World Conference on Educational Multimedia, Hypermedia & Telecommunication, Tampere, Finland, June 25-30*, (2001).

[12] J. Lavonen, V. Meisalo, M. Lattu, E. Sutinen, "Concretizing the programming task: A case study on icon-based visual programming in a secondary school," *Manuscript*, (2001).

Lego-Compiler

I. Anttila, I. Jormanainen, O. Kannusmäki, and J. Lehtonen
University of Joensuu, Department of Computer Science
P.O. Box 111, FIN-80101 Joensuu, FINLAND

ilkka.jormanainen@cs.joensuu.fi

1 Introduction

Main function of our project is examine how programming could be concretised. It is possible to use e.g. robotics to make programming more interesting and closer to a programmer. This is very important especially for children. Lego robotics [1] is one way to concretise programming. There is some existing programming environment for Lego's but usually they are very complicated for children today; Languages are similar to English and a programmer has to learn a syntax before he can learn programming. There is for example programming language called NQC (Not Quite C) [2]. Because it is quite similar to a traditional C-language, it is quite complex for the first programming language to learn. In other hand, there are a lot of "drag-n-drop" systems for programming Lego Robots. Those systems are not so good for learning because programmer can not see his program with real commands. Programmer usually sees a program as bricks etc. During our project we have developed a new kind of programming environment for Lego robots. Main features of this environment are:

- A programmer creates commands with a toolbar that has different options in programmers own language.
- The commands created are simple, understandable and similar to programmers own language.
- A programmer uses these commands to build a program.
- A programmer sees his program with real commands and can compile it in a Lego Robot.
- A programmer can not make any syntax errors.

2 Using Lego-Compiler

There is three main part when using this environment. First, a programmer build a robot. Then programmer creates some commands to robot, and finally a program will be built with these commands. Lego-Compiler translates this program and send it to RCX-unit of robot.

It does not matter what kind of robot a programmer builds. It might be a car or some kind of alien with legs for example. There is only some limits for robot: it is very important to remember that there is three input and three output on RCX-unit. Nowadays our environment can handle a light sensor or a touch sensor in each input of RCX-unit. In each output it is possible to use a motor or a lamp. There could be used more sensors or motors/lamps than three of each when they are instructed to same input/output. In this situation motors/sensors on same output/input works as one and environment handles them as one. This means that robot react at the same time when one of them.

When robot has been built, programmer has initialize robot to a program. This is possible by clicking a picture of a sensor, a motor or a lamp and then clicking output/input where programmer want to connect the device (Fig. 1). Then programmer gives a name to the program. Main window of a program (Fig. 2) shows what kind of devices is connected to the RCX-unit. Programmer selects what kind of command he/she want to create.

There is three types of command: Commands for output, command for sensors and break/loop-commands. Programmer is able to make as many commands he/she wants. The command for motor may look like this (in Finnish):

AC: Päällä Myötäpäivään. Nopeus 3. Toista 3 sek.

This tells that motors A and C are set on and direction is clockwise. Speed of motors is three and they are on three seconds. A command for sensors may look like this:

JOS KS=1 JA V2 $\geqq$ *40*
[some commands for outputs]
LOPPU JOS

This means that if a touch sensor (connected to input 1) got a value one (it is pressed) and a light sensor (connected to input 2) get a value which is equal or greater than 40, then program enters to conditional statement. There could be some other conditional statements between *JOS* and *LOPPU JOS* too.

It is possible to create some breaks and/or some loops to a program. The most important loop command is *TOISTA IKUISESTI*, which means "loop forever". This should be used if a programmer wants make a program which continue until it will be stopped by a user.

When a programmer has make as many commands is needed by a program, it is time to start combine them as a program. A programmer select a command and click "Lisaa ohjelmaan" (add to program) button. This add selected command to program window. When program is ready, it is time to translate it and send it to RCX-unit. This happens when programmer click "Vie robottiin". In Fig. 3 is one simple example of ready code. When the program is ready, programmer clicks "Vie robottiin" (send to a robot). Environment translates a program and send it to a robot. A translated command could be for example
*Form1.PBrickCtrl.Wait CON, Ohjelmalista(i).aika * 100*

Figure 1: Connection to RCX-unit.

3 Technical background

Our program has been made with Lego Mindstorms Robotics Invention System 1.5 and Microsoft Visual Basic 6.0 with Spirit.ocx ActiveX-control. Communication between program and RXC-unit happens with that ActiveX-control. RCX-unit has following technical properties:

- Three input for sensors
- Three output for motors or/and lamps
- LCD-display
- Processor

The most important feature of our environment is that a programmer could program with his/her own language. There is a multi-language support too in newest version of environment. Language could be selected with argument given in a programs command line. Environment reads all of texts from an external file. Because of that it is very easy to add some new languages to environment.

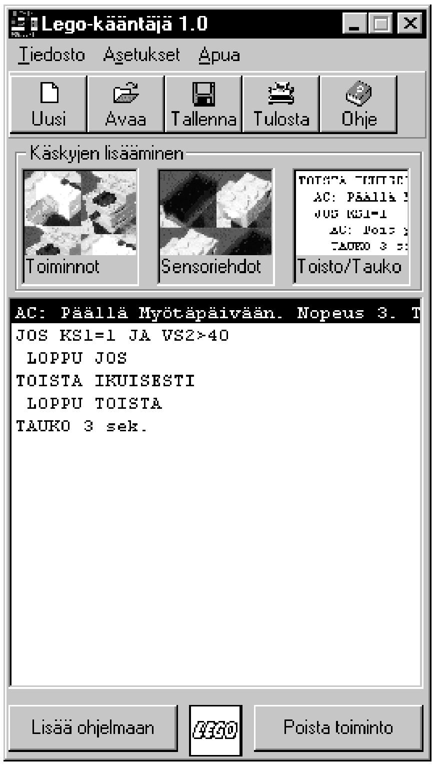

Figure 2: Main window of the program.

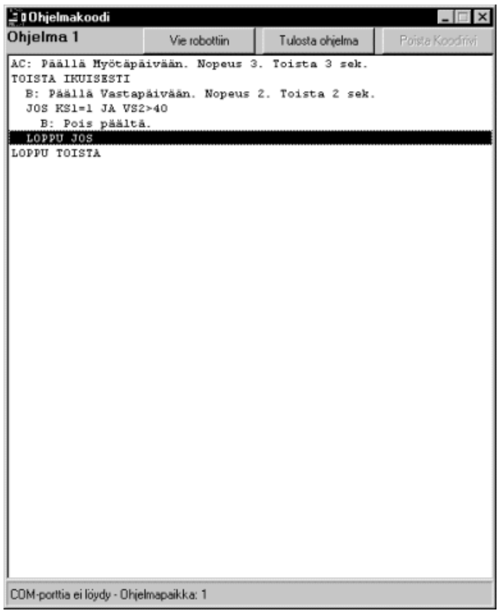

Figure 3: One simple example of code.

4 Further development

Nowadays we are testing our program with children at Kid's Club which is a project on Department of Computer Science at University of Joensuu. We collect some results and opinions about our environment. Main focus of this evaluation is how this environment makes programming more concrete. At the beginning of year 2002 we start the second part of our project. This part contains some evaluation of results we got during year 2001. Furthermore we are going to develop our environment to take hopefully advantage in learning. This new version will be written with Java, in this way it is able to make an environment which is not dependent on with a particular operating system. There will be written three Master's Theses during this second part of our project. The first one is focused on the usablility and user interface of our environment, the second one is focused on the common architecture of the environment and the third one handles a translator which is in the core of environment. The new environment and the Master's Theses will be ready by the end of 2002.

References

[1] LEGO Mindstorms: URL: http://mindstorms.lego.com/products/ris/index.asp, accessed: Nov. 28, 2001.

[2] NQC - Not Quite C: URL: http://www.enteract.com/dbaum/nqc/index.html, accessed: Nov. 28, 2001.

SQL-trainer

H. Laine
Department of Computer Science, P.O.Box 26, FIN-00014 University of Helsinki

Harri.Laine@cs.Helsinki.FI

1 Introduction

SQL database language is a basic component in the first database course. Every programmer who writes programs for database applications must know how to use SQL. Learning SQL requires practice. If the practice is with pen and paper only, the students don't know whether their solutions are correct of not. They should be able to work with a live database. SQL is the standard interface in all relational database management systems. Thus, a practice environment may be build up by installing whatever relational database management system and establishing an example database. This is a straightforward solution, but is not very good for educational use. The user interfaces that may be used without installing the client software for the database management system are typically poor and assume that students have rights to access the database server. Our experience with this kind of practice environment indicates that some students do not get their tasks done at all, and most students are satisfied with the first attempt that is able to produce a real result instead of error messages.

In the University of Helsinki the first database course is a first year course in the curriculum of Computer science. About 700 students take the course yearly, and less than half of them are Computer science major students. The course is lectured in each term. Our goals for arranging the SQL practice in this course were

- to make it possible to practice with a live database anywhere without any special client software and user accounts
- to provide a user friendly multi-lingual (Finnish, English) user interface
- to allow the use of any database as the example database
- to provide students feedback about their solutions
- to automate the bookkeeping and checking of solutions and thus to reduce the teaching resources needed for the course.

SQL-trainer is our solution to these goals. SQL-trainer is a web-based practice environment for SQL. Chapter 2 of this paper discusses the tasks, the correctness of solutions and the feedback provided. Chapter 3 outlines the architecture of SQL-trainer. Chapter 4 discusses the analysis of the solutions. Chapter 5 presents how SQL-trainer is used in our courses. Chapter 6 contains some remarks about the further development of SQL-trainer.

2 Tasks and their solutions

SQL is a database language. It may be used for defining and re-organising the database, for making queries on the database, and for maintaining the database. SQL-trainer may be used in the practice of queries and maintenance operations. The tasks to be solved as SQL-queries are expressed as natural language definitions of information needs. The goal of the practice is that the students learn to build up queries that correctly produce results that satisfy the needs specified. We could make the tasks so specific that there is only one correct result to satisfy the need given in the task. Making queries is, however, more than just technically transforming a very detailed request into SQL. In practice, one must usually interpret the request, decide what data to include in the result and how to organise the result. When this is the case, then there are usually more than one correct results. SQL-trainer allows many correct results. A result is considered to be correct if it fulfils the task specific result constraints. These include that

- the number of rows is correct
- at least the given columns are included in the result
- the order of the rows is the required one
- checksums over certain columns are correct (currently only one column is checked)
- there are not too many extra columns in the result, and
- columns with equal contents are not included unless required.

As compared to a single correct result this approach allows the columns of a result table to be ordered in alternate ways, some additional columns to be included and the freedom to choose the format of some non-essential columns.

In database maintenance the result is the subject table after the maintenance operation has been done. The correctness of a maintenance result assumes that

- the number of rows is correct, and
- the checksums over certain columns are correct.

Although the result of a query given as the solution for a task is correct, the query itself may not be correct. It may use tables that are not necessary or it may not use all the necessary tables. The selection of the rows may use information not given in the task definition. SQL-trainer uses query constrains to place requirements on the queries. These include the requirements that

- the given tables must be used
- the given tables should not be used
- certain character strings must be included, and
- certain character strings should not be present (this has been used to reject the fake queries)

SQL-trainer considers a task to be correctly solved, if both the constraints on the result and the constraints on the query are satisfied. Our constraints for correctly solved tasks are not complete. It might be possible to produce a query that is not correct but is accepted as a correct solution. Careful design of the example database and the tasks reduces this risk but does not eliminate it. Especially aggregate queries are problematic in this respect. The amount of constraints checked in SQL-trainer is considerably less that the collection of 199 constraints in the SQL-tutor learning tool [2, 3]. SQL-trainer is still able to provide useful feedback about the errors in the solutions.

There are two types of errors in the solutions: syntactic and semantic. Syntactic errors are violations against SQL syntax. Each database management system (DBMS) carries out a syntax check during query compilation and reports about the syntax errors. The quality of error messages varies according to the DBMS, but that is the facility the users have to work with, in practice. SQL-trainer lets the DBMS to take care of the syntax checking. Database management systems are not able to check the semantics of database operations. The constraints in SQL-trainer deal with the semantics. They are checked when the operation is found to be syntactically correct. Checking the constraints produces feedback. The feedback is provided according to the constraints that are violated. The feedback tries to point out the problems in the solution and to assist the student in finding the correct solution. The feedback might be, for example,

- *You should use table X*
- *You need to provide the columns Y and Z*
- *The result should be ordered according to Z*
- *The result contains too many rows. The correct amount is P. The result probably has duplicate rows*

- *The result contains too many rows. The correct amount is P. There may be a missing join condition*
- *You are not supposed to know the identifier W*
- *The content of column Q is not correct.*

3 The architecture

SQL-trainer is designed to be used connected with a database course. It provides services to keep book on the students and their solutions. It uses a centralised system database for bookkeeping. The definitions of the tasks are also stored in the system database. One SQL-trainer instance is able to handle many concurrent courses, for example a lecture course and an e-learning course. Also the example databases used in queries and in maintenance operations are centralised.

SQL-trainer consists of three tools: a training tool for the students, and a task manager and statistics tools for the teachers.

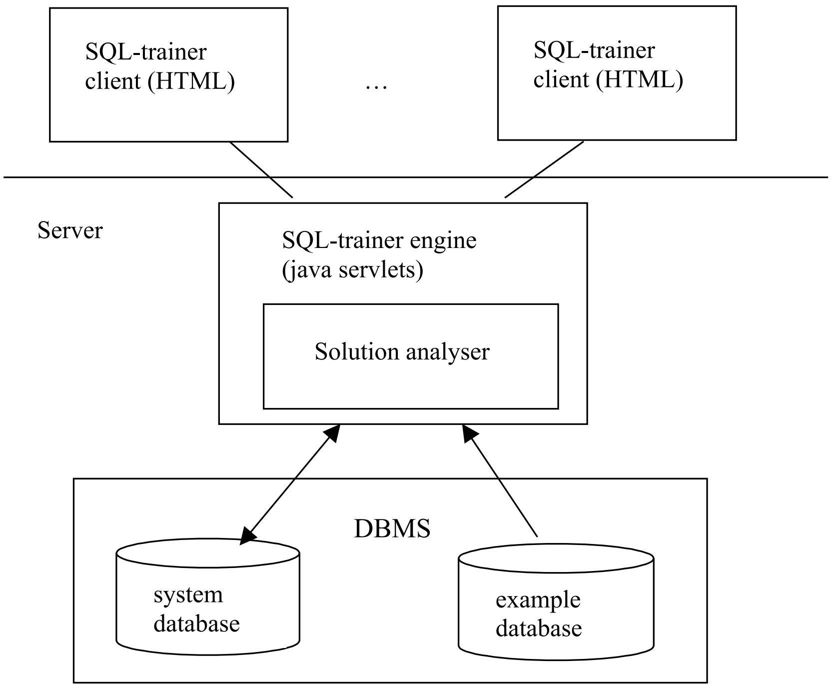

Figure 1: Components of SQL-trainer.

The training tool is basically a user interface for working with the example databases. Its components are outlined in Fig. 1. Operations may be entered and the results of the operations are presented to the user. SQL-trainer uses a real database management system, currently Oracle, for compiling and executing the SQL-operations and for handling the system database. Using a real DBMS has both advantages and disadvantages as compared to self-made SQL-interpreters used in WinRDBI [1] and query analysers used in SQL-tutor [2, 3] learning tools for SQL. A real DBMS supports multiple simultaneous users, checks the syntax of operations and executes them efficiently, which makes it possible to use the tool in large courses. However, all the error messages may not be as good as what can be provided by a self-made interpreters or analysers.

SQL-trainer stores all the solutions students provide and the results of their analysis in the system database. Statistics about the solutions may be produced on both task and student basis. SQL-trainer's statistic tools are currently implemented as SQL-scripts. The other tools are implemented as Java-servlets. Standard JDBC-API is used in communicating with the database management system. Thus, in theory, it would be possible to change the DBMS just by changing the configuration data. In practice, we have changed the DBMS once and it took us a couple of days to find out and fix the problems due to the differences in the behaviour of the JDBC API-functions.

The user interface of the SQL-trainer is implemented in standard HTML and may be run on any web browser. The user interface of the training tool contains the pages for

- *login* - Students make their own user identifiers and passwords
- *student status* - This page shows how many times the student has tried to solve each task, and which tasks he/she has solved correctly. The number of tries is unlimited. This page also acts as an index for the task pages.
- *previous solutions* - Students have access on their own solutions and may use them in building new solutions,
- *task* - This page shows the task and provides a field to write the solution. After posting the solution this page is shown again furnished with the result and the feedback on the operation. The task page provides also buttons for opening the database definition and the SQL-syntax assistant in separate windows.

The task manager makes it possible to define new tasks and to modify the existing ones. A task definition contains descriptors for the constraints introduced in Chapter 2. In addition there are descriptors needed for detecting the cause for erroneous results, for example, the number of rows in the result when the DISTINCT-clause is not included. Teachers may assign values for each descriptor separately. They may also specify an example solution that is used in determining many descriptors at a time, automatically. Typically a task is defined by using the example solution first and then modifying some of the descriptors to increase freedom in the solution.

SQL-trainer makes it possible to practice both queries and maintenance operations. Maintenance operations do not operate directly on the tables of the example database but on student specific copy rows that are made as the first step of executing a maintenance operation and deleted as the last step.

4 Answer analysis

The solutions are subjects to both pre- and post-execution analysis. The pre-execution analysis is currently very simple. Checks are made that the users are not trying to do any operations on the system database and that they are not trying to alter or re-define the example database. The pre-execution analysis may also modify the solution, for example, change the maintenance operations to work with the student specific copy rows. Students are able to produce very large results even with small example databases. Pre-execution analysis of missing join conditions in multi-table queries would be useful in preventing queries with very large results of being executed. This analysis is to be included in the next version of the trainer.

We use the DBMS to check the syntax of the operations and to execute them. Our current Oracle DBMS provides poor error messages in some common syntax error situations, for example, if column names are misspelled. We are considering post execution analysis or changing the DBMS to get rid of this problem.

Post execution analysis takes place if the query is found to be syntactically correct and has been executed without errors. This analysis evaluates the semantic constraints placed on the result and on the solution. If the operation is a query, the result of this query is analysed. If the operation is a maintenance operation, the result to be analysed is produced by a query that finds out the state of the student specific copy of the target table after the maintenance.

5 Teaching arrangements and experiences

SQL-trainer has been used since autumn 1999 in 8 lecture courses and in two e-learning courses. The task collections for the courses have contained 30-40 tasks in 2-3 groups with different deadlines. An average course with 300 students produces about 50000 answers, i.e. about 5 tries per task for each student. Our task collections contain both easy and hard tasks. Students get credit according to the number of tasks they have solved successfully. About 80 percent of tasks are needed for the maximum credit and about half of the students in lecture courses achieve it.

We have had one teaching assistant available to assist the students in solving the tasks. She has been on duty a couple of hours twice a week in a computer class during the most active practice periods just before the deadlines. She has also provided assistance by e-mail. All the classroom exercises have been substituted by practice with SQL-trainer. After each deadline there has been a lecture where the tasks that have proven to be most difficult are explained. According to the course evaluations, students prefer this way of practice to the standard classroom exercises.

We have used the SQL-trainer now for two years in large database courses. It has been found to be an easy to use interface for entering database operations. The feedback based on the semantic constraints assists the students in finding the correct solutions. The error messages produced by the DBMS are not always informative. We have tried two data base management systems and both had problems in their reporting of syntax errors. The problems were not, however, the same. SQL-trainer seems to activate the students. The exam results are at least as good as with the classroom exercises. Students that have been able to solve many trainer tasks manage well in exam.

6 Further development

Our first version of SQL-trainer supported only queries. Maintenance operations were included in 2001. We have considered new task types for SQL-trainer, for example, relational algebra tasks and simple multi-choice tasks that would be useful in a database course. We have made a prototype where relational algebra operations are transformed into SQL queries and view definitions. This would make it possible to use the same analyser for relational algebra expressions as we use for the SQL-queries. Multi-choice queries however need a different kind of analyser and different kinds of task descriptors. Our next goal is to redesign the system database and the software architecture to such that we may easily add new task types just by registering new analysers and task presenter modules. We are also analysing the solutions stored in our database to find out typical cases requiring better feedback than what we are able to provide currently.

References

[1] S. W. Dietrich, E. Eckert and K. Piscator, "A Windows-based Relational Database Educational Tool," *Proc of the 28th SIGCSE technical symposium on Computer science education*, 126–130 (1997).

[2] A. Mitrovic, "Learning SQL with a Computerized Tutor," *Proc. of the 29th SIGCSE technical symposium on Computer science education*, 307–311 (1998).

[3] A. Mitrovic, K. Hauser, "Porting SQL-Tutor on the Web," *Proc. ITS'2000 workshop on Adaptive and Intelligent Web-based Education Systems*, 37–44 (2000).

SESSION 2

Automatic Feedback and Algorithm Simulation

A. Korhonen, L. Malmi, J. Nikander, and P. Tenhunen
Department of Computer Science and Engineering
Helsinki University of Technology
P.O. Box 5400
02015 HUT
Finland

{archie,lma,jtn,ptenhune}@cs.hut.fi

Abstract

Feedback is an essential element of learning. Students need feedback on their work and their solutions to assignments regardless of whether they work manually or use a computer. A number of tools have been implemented to give automatic feedback about solutions to, *e.g.*, programming assignments. However, the problem of the feedback provided by these systems is that in most cases it is purely verbal which is not best perceived by many students. In this paper we consider how feedback should be designed to support the needs of different students. We also introduce the Matrix system, which provides both visual and verbal feedback for both active and reflective students working in the area of data structures, algorithm animation and algorithm simulation.

1 Introduction

What I hear, I forget.
What I see, I remember.
What I do, I understand.

– *Confucius*

In this paper we discuss different learning styles in the context of studying in virtual learning environments. The key idea of such enviroments is that the learner constructs his own internal models of knowledge. An essential aid for carrying out such a construction process successfully is getting feedback when the internal model is applied to solve problems.

Providing feedback is laborious on large courses and therefore a number of automatic feedback and assessment systems have been developed [2, 3, 7, 8, 10, 12]. In addition, it is evident that interactive working with such a tool providing feedback aids the learning process in small courses, as well, as has been demonstrated by Baker *et al.* [1].

The feedback given by an automatic assessment system is usually very brief, and often not very informative. Typically the system just tells the students whether their answers were correct or not, or how many points they achieved. These systems seldom give hint at what went wrong; thus they do not give the student the opportunity to track down the mistake. Though for learning purposes, it is better not to point out the exact mistake, but to point out the area where the error occured with a certain degree of accuracy. However, most automatic assessment systems do not provide such information for the student.

In this paper we introduce a method to give feedback for algorithmic assignments. The Matrix system [11] supports both algorithm animation and algorithm simulation, where the user directly manipulates data structures through a graphical user interface. An application built on the Matrix framework assesses such simulation sequences and gives feedback for the student.

We start by discussing learning models generally in Section 2. There are many learning style models, such as *Kolb's experiental learning* [9] and the *Felder–Silverman learning model* [6]. However, Felder [5] has pointed out that the choice of a model is almost irrelevant. We therefore consider more closely only the Felder–Silverman model, because it supports our purpose to classify automatic feedback.

The algorithm animation and simulation framework Matrix and the automatic assesment system based on it is introduced in Section 3. The system is classified and evaluated by using the classification of automatic feedback derived from the Felder–Silverman model in Section 4.

2 Learning Models

The purpose of learning models is to identify and classify different learning styles. A learning model can be used in designing a course to meet the needs of different students better. For example, some students prefer facts and hard data before theories. Others prefer visual information, *i.e.*, pictures and animation, before written or spoken information. Some like individual studying and others might prefer interactive learning in groups.

Students could be categorized in numerous ways. A learning model provides a framework for categorizing different learners. The model attempts to represent the whole spectrum of actual learning styles with a specified multidimensional space with named axis.

Ideally a learning environment should support students in all of these categories, and we therefore face the question which model should we choose as the basis of design. Fortunately this is not the real problem, because when a learning model is applied in order to reach all students, the results are very similar regardless of the model used [5]. We have therefore decided to use the Felder–Silverman model, since it has dimensions that are particularly relevant to science education.

2.1 Felder - Silverman Learning Model

The Felder-Silverman model characterizes students' learning styles by using five dimensions. Each of these dimensions has two extremes. The dimensions of the model are:

1. Sensory *vs.* Intuitive, *i.e.*, what type of information does the student preferentially perceive (*e.g. Sensing learners* are practical and oriented toward facts whereas *Intuitive learners* like conceptualization and prefer theories.).

2. Visual *vs.* Verbal, *i.e.*, through which modality is the sensory information most effectively perceived (*e.g. Visual learners* prefer visual information like pictures before verbal information like written or spoken language.).

3. Inductive *vs.* Deductive, *i.e.*, with which organization of information is the student most comfortable (*e.g. Inductive learners* prefer presentation from the specific to the general and *Deductive learners* prefer vice versa.).

4. Active *vs.* Reflective, *i.e.*, how does the student prefer to process information (*e.g. Active learners* are group workers who learn by doing whereas *Reflective learners* learn by thinking things through by themselves.).

5. Sequential *vs.* Global, *i.e.*, how does the student progress towards understanding (*e.g. Sequential learners* proceed linearly with small steps whereas *Global learners* learn holistically in large steps).

The dimensions are continua – not discrete categories. A student's preference on a given scale may change with time and may vary from one subject matter to another.

2.2 Classification of Automatic Feedback

When discussing automatic feedback, perhaps the most interesting dimension of the Felder–Silverman model is the second dimension; the visual / verbal pair. Some students can most effectively perceive information presented in verbal form – as written text, speech, or mathematical formulas, while others are more comfortable with pictures, animations or diagrams.

In automatic assessment it is desirable to have feedback designed towards both visual and verbal learners. Following this, we have two separate feedback categories:

1. visual feedback
2. verbal feedback

Verbal feedback is commonly given in textual form; for example, a message telling the number of points gained from an exercise. In general, verbal feedback includes all feedback represented in spoken or written form.

Visual feedback is feedback that can somehow be visualized. A diagram of the answer's data, a picture or an animation showing the state or evolution of a data structure, etc. are possible types of visual feedback.

As an example in the area of computer science education, we can consider programming tools. Ordinary debuggers where the user can view trace information and inspect variables represent verbal feedback, whereas visual debuggers which show data structures graphically represent visual feedback. Obviously visual feedback may give the information here in a way that is more easily understood by most users.

Another interesting dimension in the Felder–Silverman model is the active–reflective axis. Some students prefer learning by trying things out, while others prefer first thinking things through. For active learners we should offer continuous feedback and for reflective learners summarising feedback. We can separate the feedback into two categories:

1. active feedback
2. reflective feedback

In this context, the nature of active feedback is continuous during a simulation process. The user can actively observe and directly manipulate the view. For example, the user can modify a data structure and the representation is changed immediately as a result.

Reflective feedback, however, is a form of feedback that drives the student to rethink his solution anew. As an example, the system could provide textual descriptions of user mistakes during the simulation process.

This categorization of feedback could be driven even further by following the Felder–Silverman model. Our opinion is, however, that these two dimensions are the most relevant for automatic feedback and we have decided to concentrate on these only.

3 Matrix

Data structures and algorithms are essential topics in elementary computer science education. They include abstract concepts and processes such as data types and the procedural encoding of programs, which people often find difficult to understand. Algorithm animation and simulation can significantly help in understanding these concepts.

Matrix is an algorithm animation and simulation framework created at the Helsinki University of Technology. The primary purpose of the system is to serve as a tool for learning the basics of data structures and algorithms. Thus, it provides a framework for both algorithm animation and simulation. These terms are explained briefly below.

3.1 Algorithm Animation and Simulation

Algorithm animation is a process where an algorithm manipulates data structures and the resulting changes in the structures are visualized for the user. In general, algorithm animation is based on observing the working of an actual implemented algorithm code.

Algorithm simulation is a process where the user directly manipulates data structures by using tools available through a user interface. No user written code is needed. Instead, the user performs operations similar to which an algorithm would do, and the simulation system executes the changes in the structures.

These methods can be combined. For example, a user could experiment on the behaviour of a red-black tree by moving the keys from a key palet into the tree. The system inserts the keys and animates the implemented insertion algorithm.

3.2 The Framework

At Helsinki University of Technology we have used an automatic assessment system on our Data Structures and Algorithms course since 1991. The old system, called TRAKLA [10], had some primitive capacity of algorithm simulation, but it was very limited. Moreover, the World Wide Web front-end of TRAKLA that enabled the user to visually manipulate a “data structure” was only a simple guided drawing tool without any knowledge of the underlying structure.

This, and the limited feedback given by TRAKLA, were two main reasons to create a new more versatile system, called Matrix. It is an algorithm simulation framework where users can directly manipulate data structures on several different levels of abstractions, thus creating algorithm animations in terms of algorithm simulation.

In Matrix, all data structure manipulation operations are performed through a graphical user interface, and the results are immediately visible as a data structure visualization created by the system. Some snaphots from an algorithm simulation are presented in Fig. 1. The user has inserted a number of keys from Example Array into Example Tree by dragging keys from the array into the corresponding empty leaves of the tree. Between the first two frames keys A, N, E, X have been inserted. The rest of the keys have been inserted between the Frames 2 and 3, respectively. Finally, between Frames 3 and 4 a rotation has been performed at the node M. The user has performed the necessary pointer manipulation by moving the corresponding edges to point to new nodes, after which the tree has been redrawn.

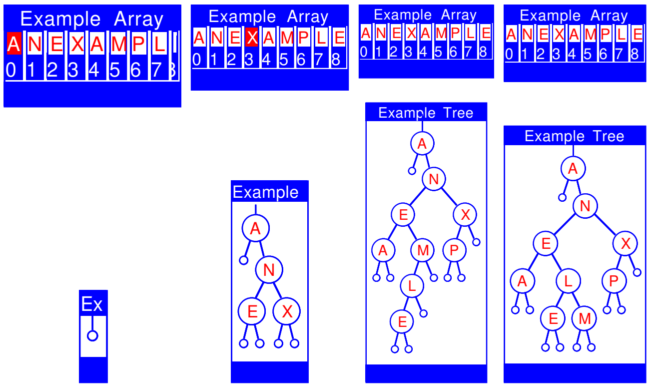

Figure 1: A Binary Search Tree Simulation.

Algorithm simulation can be performed on different levels of abstraction. First, it is possible to manipulate a data structure primitively by directly manipulating the parts, *i.e.* keys, nodes and edges, of the data structure. For example, the insert algorithm of a binary search tree can be simulated by drag–and–dropping keys into the nodes of a binary tree. Second, it is possible to raise the level of abstraction, and to use more complicated operations (for example the rotations of balanced search trees) in order to perform complex operations more easily. Third, it is possible to create algorithm animations in Matrix by using different implemented abstract data types. Thus, after understanding how and when rotations are performed in an AVL tree insertion or deletion, the user can proceed to investigate the behaviour of AVL-trees when different sequences of keys are inserted into the tree. The user can either drag arbitrary keys from a key palet into the tree, or he can insert an array of keys into the tree as a whole. Then the system inserts all keys one by one using the implemented AVL insertion algorithms. This feature generates actually an animation which can be stepped back and forth so that the user can inspect the structure of the tree between the operations.

Finally, Matrix allows advanced methods to visualize data structures implemented by the user himself. Java classes can be loaded into Matrix by using the dynamic class loader. Thereafter the simulation tools can be used to investigate the behaviour of his own algorithms.

There are different ways to implement the algorithm. It is possible to use ready-made classes available in the Matrix library, or to implement a number of interfaces. However, whichever method is used, the basic functionality for performing simulations and creating animations is available through the user interface.

3.3 The Automatic Assessment Application

We have also developed an application for automatic assessment based on the Matrix framework. This application can assess algorithm simulation exercises by comparing the user-generated simulation to the corresponding model solution generated by Matrix. As an example, the exercise on binary search trees is presented in Fig. 2.

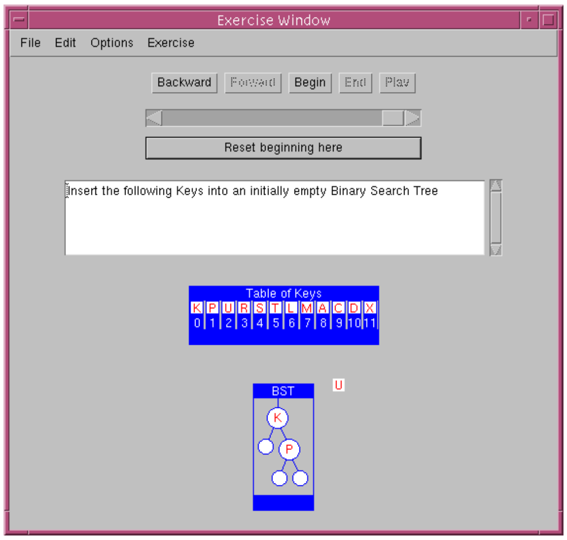

Figure 2: The Exercise on Binary Search Trees.

Each user has an individual input data structure for the exercise (Table of Keys). The user constructs a binary search tree by drag&dropping the keys from the table of keys to the leaf nodes of the tree. The visualization is redrawn after each operation. When the user has created the simulation it can be assessed with the application. Correct number of steps is returned in verbal form, for example: "*Binary Search Tree: 7 steps out of 12 correct.*" Moreover, the user can also get the individual model answer for the exercise. This model answer is an algorithm animation which can be stepped back and forth.

4 Discussion

In order to be an effective learning environment, tools applied in these systems should fit the learning style of as many students as possible. Therefore, it is important to have some learning model in mind while creating such an environment. According to Felder [4], however, the teaching style on most lecture courses tilts heavily towards the few students who are, at the same time, intuitive, verbal, deductive, reflective and sequential learners. The design of virtual learning environments should have a broader vision.

As mentioned earlier, the Felder–Silverman learning model discusses, among other things, how sensory information is perceived. The extremes of that dimension of the model are verbal and visual learners. The feedback given by most fully automatic assessment systems is verbal [2, 7, 12]. The Matrix system supplements verbal feedback with an ability to offer each student a visual model answer – an algorithm animation – of the problem they had to solve.

Moreover, the students can solve the exercises by using visual tools, when the assignment is to generate an algorithm simulation sequence.

The Matrix system also gives an environment for students where they can experiment on different data structures and observe the effect of algorithms on data structures. In other words, Matrix provides an environment where students who are active learners, people who process information by actively trying it out, can do experiments and thus broaden their understanding. At the same time Matrix is an environment where reflective learners, people processing information and learning by internal reflection, can confirm their conclusions.

These issues can be visualized as a two-dimensional plane with *visual-verbal* and *active-reflective* axis, as illustrated in Fig. 3.

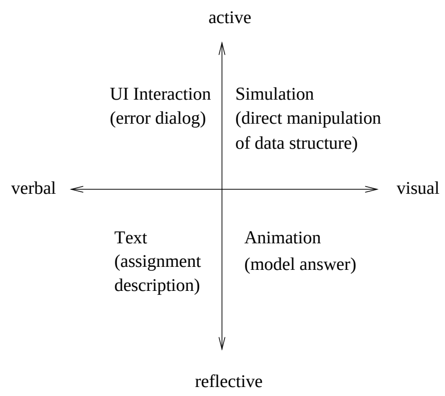

Figure 3: Examples of Visual-Verbal × Active-Reflective Categories.

4.1 Verbal – Visual

Should we prefer verbal or visual feedback when designing new automated feedback systems? It seems that adding visual components increases the value of feedback. Currently most implemented feedback systems are, however, capable of giving feedback only as written text. That type of feedback is of less use to students who are visual learners and would be more comfortable with some kind of visual feedback. It is also clear that purely visual feedback is not a good idea either.

Some information is easier to present in verbal form while some other information may clearly be visual. In the Matrix assignments the visual feedback provided is the model answer that is presented as an algorithm animation. Verbal feedback tells the correct number of steps, from the beginning, in the user's simulation. The same information could also be presented visually. Users own (*visual*) simulation sequence could be returned and the area where the user made a mistake could be highlighted visually.

Our opinion is that verbal and visual feedback should be combined in automatic assessment. Information obtained from these sources can and should be overlapping. However, if some information is easier to present in one form, this should also be notified.

4.2 Active – Reflective

Matrix supports the active learning process by giving immediate visual feedback in terms of algorithm visualization. The student can observe the effects of his manipulations on a data structure while solving an exercise as demonstrated in Figure 2. Moreover, because the system actually understands the underlying concepts, it is capable of giving feedback also in the level of the graphical user interface. If the user performs operations that are out of the scope of the data structure, the GUI can alert the user about the misconception.

All the assignments implemented in the application include both the textual description of the assignment and, finally the model solution of the exercise. Obviously the understanding of the description requires reflection. Moreover, the student may reflectively learn more about the topic by studying the model solution provided by the system.

4.3 Future directions

Learning models should be considered closely when designing virtual learning environments, since students using these systems work more on their own compared with students working mostly in traditional classrooms. The models can also be used for categorizing and evaluating automatic feedback systems, and for designing better feedback. We also need further research and results from the area of educational science to identify how the different modes of feedback reflect to the learning results. We are planning to compare the Matrix system with both visual and verbal feedback to the old TRAKLA system that provides only verbal feedback. This experiment is scheduled for the spring 2002.

Concluding remarks The Matrix framework was originally designed and implemented by Ari Korhonen. The automatic assessment application built on it has been designed and implemented by Jussi Nikander and Petri Tenhunen, who have also written most of this article.

References

[1] R. S. Baker, M. Boilen, M. T. Goodrich, *et al.*, "Testers and Visualizers for Teaching Data Structures," *Proceedings of the 30th SIGCSE Technical Symposium on Computer Science Education, New Orleans, LA, USA*, 261–265 (ACM, 1999).

[2] S. Benford, E. Burke, E. Foxley, *et al.*, "Ceilidh: A Course Administration and Marking System," *Proceedings of the International Conference of Computer Based Learning, Vienna*, (1993).

[3] J. English, P. Siviter, "Experience with an Automatically Assessed Course," *Proceedings of the 5th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, Helsinki, Finland*, 168–171 (ACM 2000).

[4] R. M. Felder, "Reaching the Second Tier," *J. College Science Teaching 23 (5)*, 286–290 (1993).

[5] R. M. Felder, "Matters of Style," *ASEE Prism* **6(4)**, 18–23 (1996).

[6] R. M. Felder, L. K. Silverman, "Learning Styles and Teaching Styles in Engineering Education," *Engr. Education* **78 (7)**, 674–681 (1988).

[7] J. Hyvönen & L. Malmi, "TRAKLA – A System for Teaching Algorithms Using Email and a Graphical Editor," *Proceedings of HYPERMEDIA in Vaasa'93*, 141–147 (1993).

[8] D. Jackson, and M. Usher, "Grading student programs using ASSYST," *Proceedings of 28th ACM SIGCSE Tech. Symposium on Computer Science Education, San Jose, California, USA*, 335–339 (ACM 1998).

[9] D. Kolb, *Experiential Learning* (Prentice-Hall, New Jersey, 1984).

[10] A. Korhonen, L. Malmi, "Algorithm Simulation with Automatic Assessment," *Proceedings of the 5th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, Helsinki, Finland*, 160–163 (ACM 2000).

[11] Matrix - Algorithm Simulation Tool: *URL: http://www.cs.hut.fi/Research/Matrix/* (2001)

[12] R. Saikkonen, L. Malmi, A. Korhonen, "Fully Automatic Assessment of Programming Exercises," *Proceedings of the 6th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, Canterbury, UK*, 133–136 (ACM 2001).

Computer Supported Learning Environment for C Programming Language

E. Vihtonen
Viope Solutions OY
Laserkatu 6
FIN-53851 Lappeenranta, Finland

S. Alaoutinen and A. Kaarna
Lappeenranta University of Technology
Department of Information Technology
P.O.Box 20, FIN-53851 Lappeenranta, Finland

Esa.Vihtonen@lut.fi

1 Introduction

Learning programming skills is a demanding task for many first-year students at the university level. The skill of computer programming is obtained by actually writing programs. The solution for the given problem, the algorithm, should be written by pen and paper and it is clearly a creative process. Transforming the algorithm to a program in a selected language requires similar skills and they can be learnt best by working with the language in problem solving [1, 2].

Many different computer supported learning environments exist, some of them provide a general environment for any learning subject, some are strictly dedicated to a certain subject. For example, WebCT as a representative of the previous provides a set of common utilities, that have no special support for learning programming [3]. Many program language specific environments check the correctness of the individual programs or larger project works, but limited features to support interactive learning of programming skills exists [4]. Another approach to programming comes from modeling, visualization and animation of the programs. The systems help to understand the solution of the problem and the data structures and objects applied, but the actual teaching of programming language has lighter importance [5, 6].

In this report we describe Viope, a computer based learning environment for the C programming language. In addition, experiments from the use of the environment during one teaching period in August 2001 are presented. The results from the questionary and the exam show that Viope learning environment supports the learning process and ends up to better grades from the exam.

2 Viope learning environment

2.1 Background

Viope is based on the holistic approach to learn programming skills [7]. It contains correctness checking as a basic feature but also other features for learning: it provides exercises with various difficulties, hints for correcting programs with errors, quizes, discussion area, lectures and manual pages for language commands. An individual student can follow her own timetable and proceed from the easier level to more difficult through working out the required exercises.

From the teachers point of view, Viope manages the most time consuming activity of the traditional teaching: it checks the correctness of a programming project. Thus, the teacher has more time for higher quality teaching and support for the students. As the environment provides learning statistics of the participating students, the teacher can give better targeted lectures and support to the students.

2.2 Essential features

During the course the students filled a questionary, where they were asked to grade a set of features for a learning environment. These features and their grading are collected into the Fig. 1. The grading was from 0 to 10, where 0 meant not essential or not important and 10 meant required, very essential or very important feature. In the questionary, we recieved 11 answers out of 16 participants, thus, 69 % returning their answers.

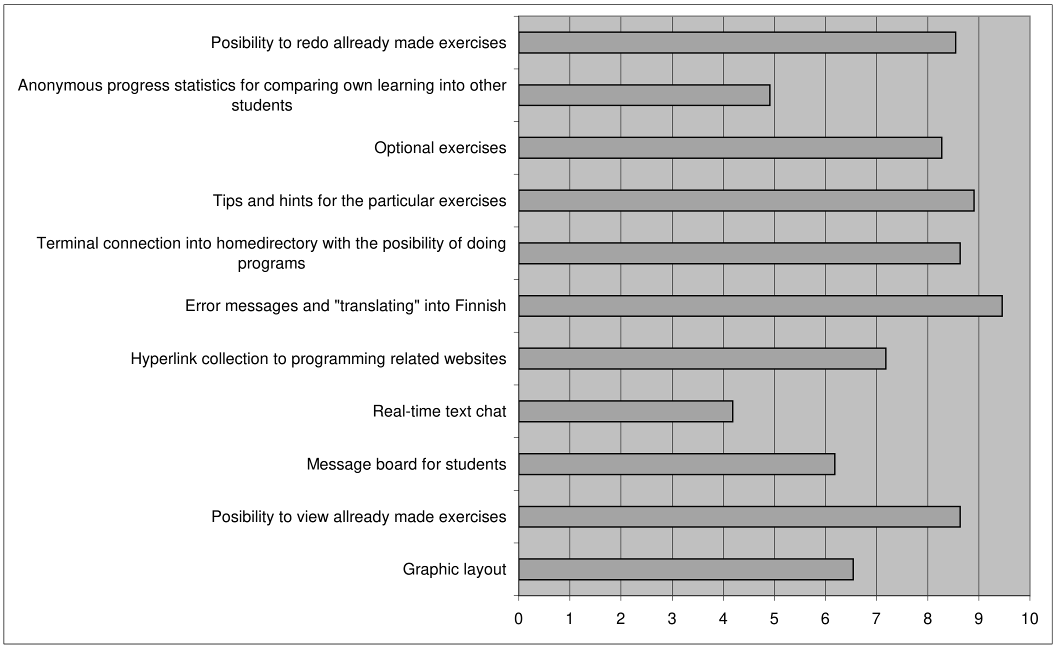

Figure 1: Features considered important in the learning environment.

According to the questionary the students were very satisfied with the Viope environment: they considered that learning programming was easy and the environment suited well for learning. The most important features of Viope were error messages, hints for correcting the errors, connection to the home directory to store the programs and the possibility to redo the exercises.

3 Testing the environment

3.1 Experiments from testing

The Viope environment was used during one teaching period in August 2001. The students were mostly second year students who haven't passed the course in their first year studies.

During the first two days, the lectures given contained general items of programming, like why and where programming is needed, how to produce algorithms to solve problems and how to present algorithms in a formal way. Also, simple examples of the C programming language were presented.

For the two exercise days, the students were divided into two groups, one was participating in the Viope exercises, the other group has traditional exercises in a microcomputer classroom. Both groups had a teacher in the classroom. The students had no or very little experience on programming before participating the course, see Fig. 2.

Before the course the students considered learning programming quite difficult, see Fig. 3.

In the previous studies the students have participated a regular course with lectures and

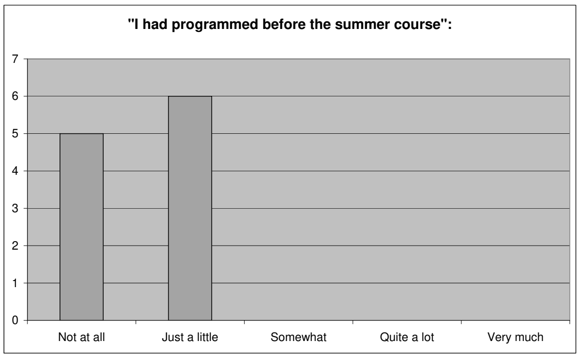

Figure 2: Programming experience before the summer course.

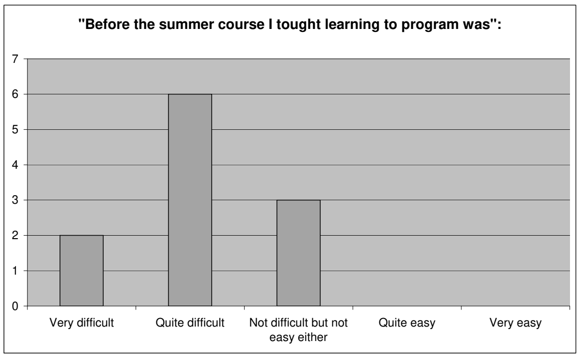

Figure 3: Difficulty of learning programming, before the course.

traditional exercises in the computer classroom. None of the students have passed any programming course before but some of the students had some experience in programming.

After the course the students thought that learning programming was not so difficult as they thought before the course, Fig. 4.

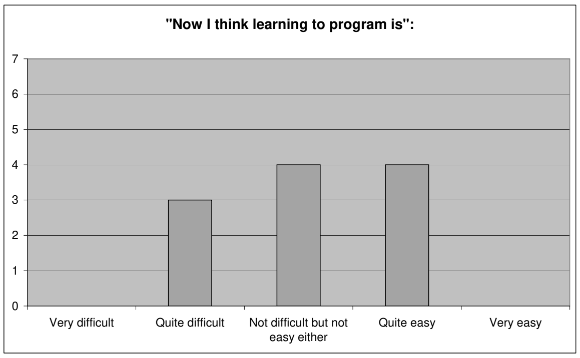

Figure 4: Difficulty of learning programming, after the course.

The change of attitude in learning programming was mostly affected by the learning environment. The environment gave new possibilities for learning, each student can proceed with her own timetable and more time could be dedicated to the items considered difficult. In Fig. 5 the opinions on the usability of the environment in exercises are displayed.

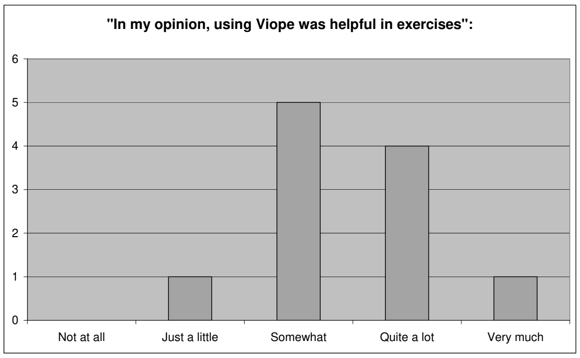

Figure 5: Usability of the environment in exercises.

The overall satisfaction to the learning environment was high, see Fig. 6. This group of students couldn't exploit the traditional teaching methods, but the new environment was more suitable for their learning practices.

Even though the environment was supporting learning well, there are still shortcomings in the environment. Fro example, the following items were mentioned:

- the selection of the programming problems was too narrow. A larger variance in difficulty would make the environment equally challanging for students with different background in programming skills.
- the correctness of the programming problems were checked too strictly.
- the error messages should be more instructive.
- the hints how to correct an erroneous program should be more instructive.
- more exercises for each item were requested.
- practical examples on the items, when it is too difficult to solve the problem at hand.

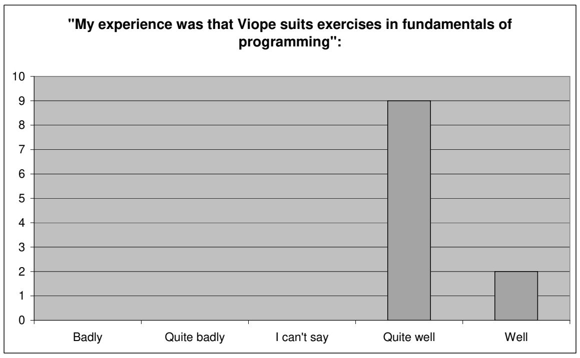

Figure 6: The overall satisfaction to the environment.

A traditional exercise in a computer classroom requires more experience in gathering information. Normally, the error messages from compilers are short, difficult to understand and possible the actual error has happened earlier than the sentence that the error message points to. In the learning environment these difficulties can be avoided, and thus, the environment was found better for this target group.

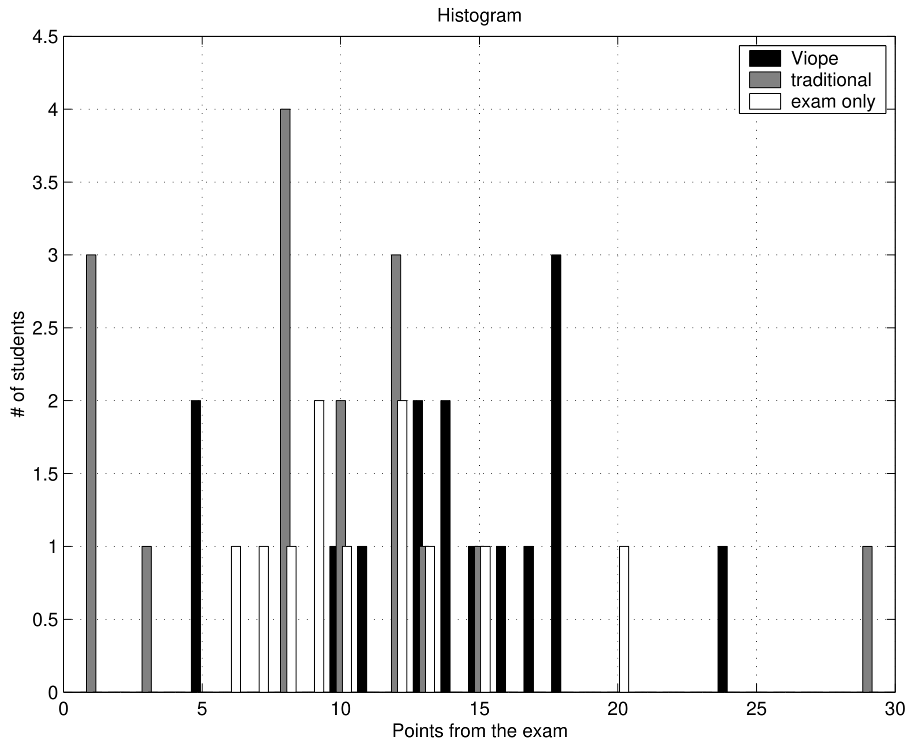

Figure 7: The three groups, that participated the exam.

3.2 Results from the exam

After the course the students had the exam, that contained both questions with essay answers and program coding answers.

The histograms for the three groups are illustrated in Fig. 7. The black bars describe the results for the Viope group, the gray bars for the traditional exercise group, and the white bars describe results for the students, who participated the exam through self-study.

Fifteen students from the Viope exercise group participated the exam and eleven (73 %) received the grade passed. From the other exercise group, sixteen students participated the exam and six (38 %) received the grade passed. Thus, there is a significant difference between the two groups, see Fig. 8.

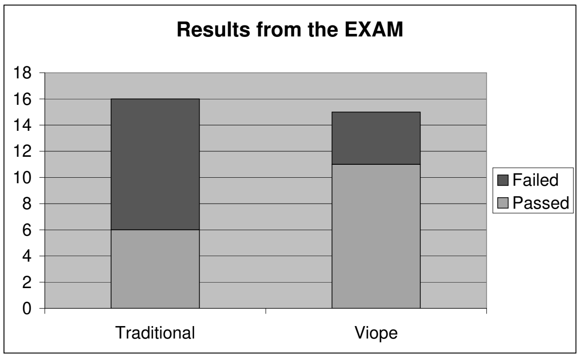

Figure 8: Results from the exam.

After removing the extreme values from the sets, the Viope group had a higher mean value of 13.7 than the traditional group with mean value of 7.64. According to the t-test, $T = 3.42$, the mean values were clearly different with significance level 0.05.

During the exercises, the students answered a questionary where the total number of questions was 28. One part of the questions were ment for the development of the environment, one part was used to find out how useful the environment was for learning the programming skills.

4 Conclusions

The students participating the Viope exercices were very motivated for learning, they participated voluntarily to this experiment and testing activity. During the exercise, they found the environment very inspiring and helpful. But, the environment is better utilized if a teacher also attends the exercises. The students should be guided to use the learning environment, later they can work independently.

The Viope system still has some shortcomings, for example the checking of correctness of a program was too strict and the hints for correcting the program were not found satisfactory. New features of the Viope system will include more flexible checking procedure and hint generation. The previous will allow formulation of the answers more freely. The latter will be managed with more versatile language teaching aids and with more detailed, profound and better targeted hints.

The Viope learning environment was very effective for learning the C programming language. It provides time- and place-independent environment for learning and it is also very suitable for students who prefer learning-by-doing to lectures and traditional exercises.

References

[1] R. Rothwell, J. Edgar, M. Spink, D. Duret, *Computer Programming* (Blackwell Publishers, Singapore, 1996)

[2] H. Abelson, G. J. Sussman, J. Sussman, *Structure and Interpretation of Computer Programs* (The MIT Press, Boston, 1996)

[3] WebCT Learning Environment. Available: http://www.webct.com/, accessed: Sep. 13, 2001

[4] Ceilidh - On Line C Tutoring System, available: http://www.cs.cf.ac.uk/Dave/C-/chapter2_19.html, accessed Nov. 16, 2001.

[5] BlueJ, The Interactive Java Environment, available: http://www.bluej.org, accessed: Nov. 16, 2001.

[6] Jeliot–Interactive Algorithm Theater, available: http://verosaari.cs.helsinki.fi:8807/Jeliot/, accessed Nov. 16, 2001.

[7] E. Vihtonen, "Raportti Viope C:n testikäytöstä LTKK:n ohjelmoinnin perusteiden intensiivikurssilla," Internal report, Viope Solutions OY, 2001.

Automatic assessment of exercises

P. Nieminen
Lappeenranta University of Technolgy
Department of Information Technology
P.O.Box 20, FIN-53851 Lappeenranta, FINLAND

pekka.nieminen@lut.fi

1 Introduction

Basic skills in computing must be taught for all new students, so the number of students is big. The skills vary and teaching will easily be too hard or too easy. Putting hundreds of students through few computer rooms in a short time is hasty, time consuming and boring. A self-paced computer-aided learning system can be used for expanding the time and rooms where exercises can happen and give some additional value to learning. According to Bennett [1] computers will not only be able to present and assess educational material and its absorption by the student, but also that the computer will be able to analyse and compensate for the difficulties a student has in the comprehension of this material. In other words, to be able to provide a meta-understanding of the processes that the student is undergoing, in order to effectively tutor that student.

Many studies like Brad and Humphreys [2] show that active working on net-material like on-line computer graded practice quizzes and posting are positively correlated with student performance on the quizzes and exams, but use of web-based content and passive reading of posts is not. So the focus should reside in activation of the student.

General and specific automatic teaching and testing systems are exists to tackle these problems. Specific systems like WWW-TRAKLA [3], Ceilidh [4] and Test-Pilot [5] are dedicated to one or more purpose. General learning environment like WebCT [6] offers a range of tools, but it's assessment tools are slow to use and can't make variations easily. Because no specific tool were available for our purpose and our general tool (WebCT) could do something but not all, we decided to make a new add-on to create variations of exercises and assess them automatically.

2 Goals

The basic skills of computing are taught in the course "Introduction to Information Technology". Exercises consist of Windows, Unix-terminal, mathematical program, CAD, word-processing, spreadsheet and database. The paperwork for 300 students for most of these exercises is big and should be minimized. CAD-exercise is taught as a normal classroom exercises, but others could be automated in some level and allowed to execute as distant execution.

Our university had unlimited license to WebCT and it's services would be used. Users can use browsers to use services over Internet. Students can book themselves in to the course and download each exercise. WebCT can gather submissions and produce a report about them. This will save paper and multiple booking enormously. The new system could concentrate details, which are new or difficult to handle otherwise. The main principles are to offer variation, immediate help and feedback and prevent cheating and copied answers.

The system should activate the student and give her/him adequate information and hints for each task. The interface should be informative and easy to learn. Each student would get an individual but equal exercise by randomly selected variations of tasks. After making all tasks the student's accomplishment should be recorded. This would assure the unity of exercises and assessment and decrease booking errors.

The execution of exercises should be possible any time on different computers to allow a student make exercises in her/his own time, place and pace. This possibility will break the crowds in exercises to different times and computer rooms, distant working allow a student make exercises in her/his own pace with home computers could be possible. Cheating, which is easy when the execution is not controlled, would be minimized by the amount of variation and intensity of exercises, making it easier to accomplish the exercise than cheat.

3 Structure

First exercise is to learn to make these exercises and it is offered for all. Computer rooms are allocated to these exercises and a course-assistant is present to start exercises and help in problems. Much less is allocated to the second exercise, but those who need it, a dedicated time and room with an assistant is available. These services will diminish when the course goes on.

The exercises are published each on its time. A student has two week to complete the exercise. Each exercise will tell its contents and ask randomized questions. A student must answer all questions to get the exercise done. When it's done an exercise will send the ID-number of student to an intranet database. For a confirmation a student must send a checking number from the exercise to WebCT, where they are manually checked.

4 Application

As a compromise of time needed build the exercises and broad range of fitting computers the exercises are made and executed with MS-Access database. Thus a student will need a computer with Windows operating system, a rather new version of MS-Access in addition to Internet-connection.

The data structure of the system is shown in Figure 1. An exercise consists of tasks that are similar to all. Every task has one or more variations with each different right solution. When a student opens such task, a variation is randomly selected. The student must give an answer, which is compared with the solution. Only giving an acceptable answer a student can take another task.

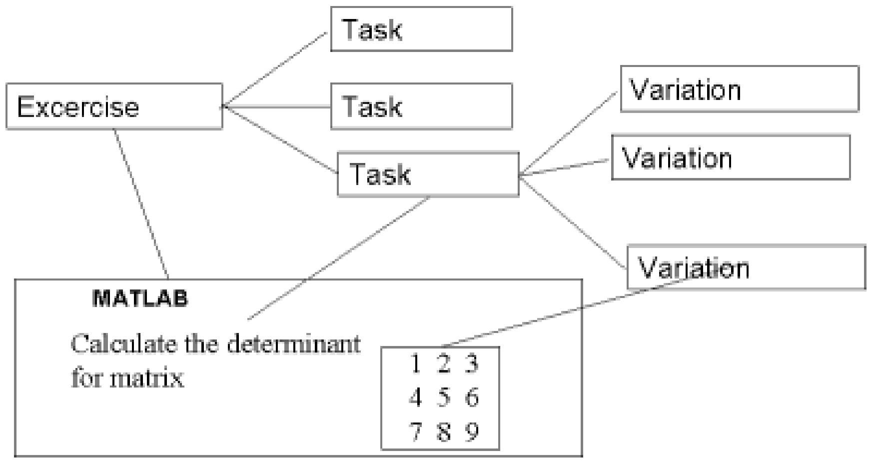

Figure 1: Data model.

The need to validate the answer automatically determines the type of questions. An answer can be a number, which is subtracted from the solution and accepted within a tolerance. It can be a text, which is looked for from solutions or the solution is looked for from the answer. This limits textual answers to clear short answers. To make tasks more versatile, answering to a matrix is possible and of course multiple-choice questions are utilized. A sample of questions is presented in Figure 2.

4.1 Data types

Each task is written in HTML and will be read from the www-pages of the course. This solution helps to format tasks and add easily pictures. An Access-form calculates the address from the fields of the task-table and shows the task in an ActiveX-control. The variations of one task are text and saved in child-tables, each type of task has an own child-table and a child-form to show it. When a new task is loaded, a proper child-form is loaded too and a randomizer picks one variation from the child-table and it's solution, which is naturally invisible. Each child-form has own fields to get answers and the logic how to accept or reject the answer. If the answer is rejected a helping system can be triggered. If the answer is correct a message is sent to main-form, that shows that the task is accepted and allows loading the next task. The last task of each exercise is to collect an individual secure number that is loaded from a hidden table. There is a secure number for each student's ID-number and for each exercise.

4.2 Booking

Students send their secure numbers to WebCT and to each exercise-assignment. The only booking-task that a teacher must do is checking those numbers and put a mark to WebCT. A student can see the mark immediately and follow her/his success in the course. When the course is over a teacher makes WebCT prepare a list of students who have all exercises accepted.

5 Conclusions

The system is in use for the first time and final evaluation can't be done yet, but experiences seem promising. Most of the students are working in schedule and crowds in computer-rooms are avoided. Complaints and questions are reasonable though frequent. Thus it seems that: Self-paced learning helps students and they accept this system. Learning is distributed to different times and places. Distant execution is possible in most of the exercises. Exceptions

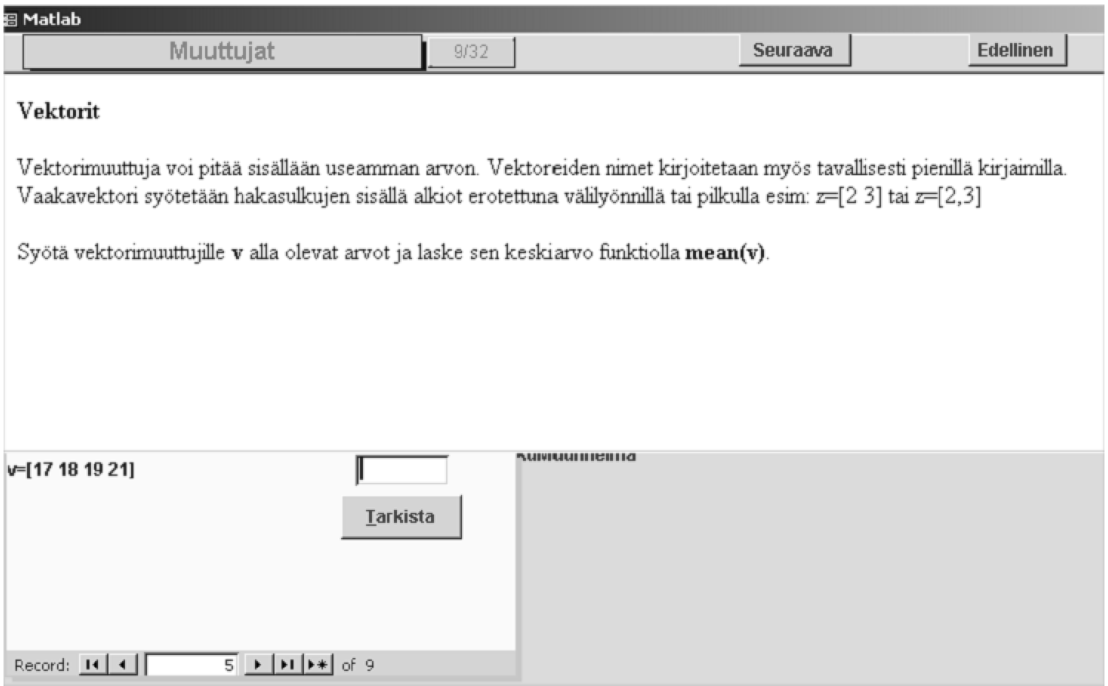

Figure 2: A student's view to tasks.

are connected to applications that are available only in certain computer-rooms. Booking is easier though not yet avoided and less mistakes happens.

As a drawback personal meeting with the students is decreased and some new booking errors exists, but put together some improvement has been gained. The lack of face-to-face teaching is compensated with dedicated exercise-times with an assistant. She/he can hear of difficulties and change instructions immediately if necessary, because the tasks are in editable www-pages. Variations and solutions are harder to edit, but it's possible to build real-time editing for them, too.

6 Future possibilities

Booking in the system should further simplified for instance accomplishments in the campus-net could be saved into an intranet database, or the interface could be a web-form attached to the database. The services of WebCT could be integrate to the database and leave WebCT out. The internet-connection must be on during the whole exercise. Some packaging could be developed a student to download exercises as a whole and disconnect. Student's activities could be logged to find out the difficulties of exercises.

7 Acknowledgment

The project was funded by LUT Virtual University-project and supervised by Dr. Arto Kaarna. It is a part of my postgraduate studies supervised by Dr. Jan Voracek.

References

[1] F. Bennett, *Solving the Crisis in Education,* Faben, Inc., Sarasota USA, 2000.

[2] D. Brad & C. Humphreys, "Evaluation of Computer-Assisted Instruction in Principles of Economics," *Educational Technology & Society 4* **2**, (2001).

[3] A. Korhonen & L. Malmi, "Algorithm Simulation with Automatic Assessment," *The proceedings of the 5th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education – ITiCSE 2000, Helsinki, Finland*, (2000).

[4] S. Benford, E. Burke, E. Foxley, "A System to Teach Programming in a Quality Controlled Environment," *The Software Quality Journal*, **2**, pp 177-197, (1993).

[5] Test Pilot *http://www.clearlearning.com/about.html*

[6] WebCT *www.webct.com*

SESSION 3

Applying Computer Supported Collaborative Writing In Education

P. Gerdt
University of Joensuu, Department of Computer Science
P.O. Box 111, FIN-80101 Joensuu, Finland

pgerdt@cs.joensuu.fi

1 Introduction

A big part of documents produced in commercial and academic circles are written collaboratively, up to 85% of commercial and academic documents are written jointly as group work [2]. Today joint writing of documents is often a distributed effort as many commercial and academic organizations are global, dispersed all over the world. In the information techonolgy business many documents, such as requirements specifications and other product documentation, are very large artifacts that are developed by groups of professionals. In the academic world research is mainly conducted by writing scientific articles describing research and getting them published.

Supprisingly collaborative document production and collaboration in general is not emphasised or even aknowledged as something important in our educational system. With the highly developed information sharing infrastructure of today collaboration and collaborative work in general have very promising prospects. But collaborative writing skills do not emerge out of nothing, the would be collaborators must be educated in collaboration skills in order to use them naturally.

Distance education is a natural area for applying collaborative writing as a way for geographically separated members of a group to do group work and process ideas. Different platforms that facilitate distance education, such as WebCT, have usually some kind of mechanisms for sharing documents with others. These mechanisms are quite inflexible tools of collaboration, they only give the members of a group means to share information, not to manipulate it together.

This paper is structured as follows. In Section 2 we survey some of the basic ideas of computer supported collaborative writing. Then we proceed to take a look at collaborative writing in educational settings in Section 3. In Section 4 we introduce our collaborative writing prototype system, the Woven Stories 2, along with future plans.

2 Computers Supporting Collaborative Writing

Computer supported collaborative work (CSCW) is a multidisciplinary study area, which aims to make groupwork easier or even facilitate groupwork in a situation where it would not be possible without information technology. Software that support the principles of CSCW are called groupware. Co-authoring tools are groupware that support a group of people in collaboratively producing a document [13]. Computer supported co-authoring has been quite widely researched as far as co-authoring [21], basic system requirements [18, 1] and prototype systems [8, 5, 6, 2] are concerned.

Next we are going to take a look at what actually happens in a collaborative writing process in order to get an idea about what the co-authoring systems are trying to support.

2.1 The collaborative writing process

A collaborative writing process consist of a sequence of phases, that are generally identified as the planning phase, drafting phase and the reviewing phase [20]. The participants of the joint writing project plan the outline of the document to be written in the planning phase. When the participants of the group agree on a plan they shift over to the drafting phase and start to produce text. When the group has produced certain amount of text it must be reviewed and revised. These activities take place in the reviewing phase. The whole process in recursive, or cyclic, the group may always go back to drafting after deciding that the document is lacking in content in the reviewing phase. Members of the group may be in different phases at the same time [20, 1], one member may for example be engaged in writing her contribution as another one already reviews a part of the document.

The persons participating in group work tend to have different roles that tell them about their responsibilities in the joint work. A person may have more than one role, and the configuration of roles may change during the writing process as the members of the team get more experience. Posner and Baecker [21] identified four different roles that occur in collaborative writing: writer, editor, reviewer and consultant. Writers are members of a co-authoring team, who produce text actively. Editors edit the text that others have produced and reviewers evaluate and comment the produced text. Consultants participate actively in the production of the text, but do not actually write anything themselves.

When several persons make modifications to the same artifact some kind of a control method is needed in order to prevent total chaos. Posner and Baecker [21] identified four different strategies for document control. In the single writer method one person writes the document based on discussions within the group. In the scribe approach one person writes the document based on notes from meetings, where the group members have gathered. Separate writers is a method, where each writer controls a part of the document. Finally, in the document may be jointly written, the whole group write and administer every part of the document together.

In this section we have presented three different aspects of collaborative writing: the writing process, the participants' roles and document control methods. Next we will take look at the possibilities that co-authoring systems offer to support the before mentioned aspects of collaborative writing.

2.2 Some features of co-authoring systems

The starting point for a computer supported collaborative writing tool is good basic word processing capabilities and good compatibility with other related software [18, 1]. A co-authoring tool should work as a basic text processor system. This is important, because one of the main reasons for failures in computer supported collaborative work are caused by the simple fact that they do not make things easier for the group [11].

A co-authoring tool must address the question of roles within the group. The definition of roles can be explicit, where the system has pre-defined roles or supports user definable roles. For example the co-authoring tool Quilt [8, 1] supports explicit roles. Most collaborative writing systems do not use explicit roles, they have various sets of access rights regarding the shared artifact that their users can use to back up different roles. This approach leaves the notion of roles outside the system. The co-authoring tool can support document control by a version control mechanism and by letting the users define who has the right to read and modify the document.

A co-authoring tool may support the writing process by defining different mechanisms, that let the users to define where they are in the writing process. For example the co-authoring tools PREP [19] and Quilt [8] have notifications of user-definable deadlines. Another important feature of co-authoring tools that is related to the writing process is the support for changing intensity of work. For example a planning session may be a very intensive brainstorming (virtual)meeting, where the users need a synchronized view of the shared artifact. On the other hand writing is mostly done at individual pace and the users do not need a fully synchronized view of the shared document. Distributed groups may be separated by timezones in addition to distances, so asynchronous work must be an option in a co-authoring tool.

In a co-authoring tool the actual work around the shared artifact takes place in a virtual working environment [13]. The virtual working place must provide the users with similar information about other users as we get in real-world offices and classrooms etc. This information is called awareness information about what others are doing (in synchronous work) or have been doing (in asynchronous work) in the environment [12]. Communication tools, such as the possibility to attach annotations and comments to the document (see [20] for more information about annotations in collaborative writing) and chat programs provide the users with some awareness of others. There are special user interface components, such as multi user telepointers and views with information about the users' activities, that facilitate awareness, that is mediated by gestures, pose and eyesight in the real world environments.

3 Collaborative writing in education

Educational reformists have called forth social constructivism and learner-centered environments instead of teacher-centered instructional methods [4]. Writing is an essential skill in our information-oriented society that can be used to achieve active, student-driven learning experiences. Writing gives the learners the opportunity to rather communicate with readers what they want to say or accomplish rather than be judged by teachers [23]. Furthermore, co-writing improves collaboration skills as students engage in creative group work. Until recently the use of computer supported co-authoring in educational settings has received much less attention than basic computer supported collaborative writing (see for example [16]).

In this section we will first take a look at the pedagogical background of co-authoring in the educational context and then make a survey of selected applications of collaborative writing in education.

3.1 Pedagogical background

Writing is active communication between readers and writers, a relationship which can be used to enrich learning [23]. Collaborative writing involves sharing ones views with other writers and the creation of a common understanding of the topic. The recursive nature of the writing process make the learners revise their texts and the ideas behind them repeatedly, which is good for learning.

There are some conditions that collaboration needs to exist in the co-writing context [15]. A writer needs a well defined community in which she can form an identity through written text. In order for this to happen the writer needs an active community. Finally the writer must value the opinions of her colleagues toward her text. It may be hard to create these ideal conditions in a classroom and even harder to enforce them on students.

The teacher encounters new challenges in a classroom, where computers and networked writing situations are involved [17]. The models of teaching used in traditional classroom teaching are not sufficient, new models are needed. For example the classroom presence of the teacher must become on-line presence that is enforced by on-line authority. It is not surprising that many teachers report feeling discomfort in computerized classrooms, they may not have any educational background for the situation.

3.2 Applications of co-writing in education

Tanikawa *et al.* [23] report their system for creating a wall newspaper collaboratively in LANs. The system lets a group of students on separate workstations to share a virtual wall, where they can write and put graphics on. The wall is divided into segments, which are allotted to the writers. The system is intended to be used with all participants of the collaborating group in the same room. Observations of the students working with the wall newspaper system show, that students interact with each other during the writing process and repeatedly revise their contributions based on this interaction.

The TC3 (Text Composer, Computer supported & Collaborative) groupware environment [7] is intended to be used in Dutch high schools to support collaborative argumentative writing. The typical scenario of use is two students choosing a position pro or contra a topic and then using TC3 to write a document that supports their point of view.

The both systems mentioned in this subsection support synchronous writing in a LAN. An occurrence of asynchronous collaborative writing can be found in the WebCT environment, that can not be regarded as a real collaborative writing tool as it is rather a advanced virtual bulletin board.

4 The Woven Stories 2 prototype system

We are developing a collaborative writing tool, the Woven Stories 2 (WS2) [10, 9]. As the name implies our aim is that WS2 facilitates collaborative thought processing through co-authoring in different scenarios of use. WS2 differs from mainstream collaborative writing software in that it is intended to be used over the whole Internet rather than in a LAN. This requirement is important as we want WS2 to support distance education as well as computer enhanced classroom-based education. Roughly one could say that WS2 is a marriage of the availability of WebCT boosted with the functionality found in collaborative writing systems. We intend to support these features with mechanisms, that lets a teacher set up different learning scenarios with the WS2 system ranging from a one hour brainstorming session in a classroom to a six month project with a geographically distributed team of students.

WS2 supports both synchronous and asynchronous writing, although there are few awareness mechanisms in the current implementation. The WS2 supports any number of documents, which consist of graphically linked text passages, or sections, that the users may input. The text passages are represented by rectangles and the links between them as arrows (see Fig. 1). The content of a text passage can be viewed and edited by clicking on a section. There is no limits for the length of text passages, or for the count of text passages and links. The WS2 interface includes a chat program, that the users can use to communicate in real time.

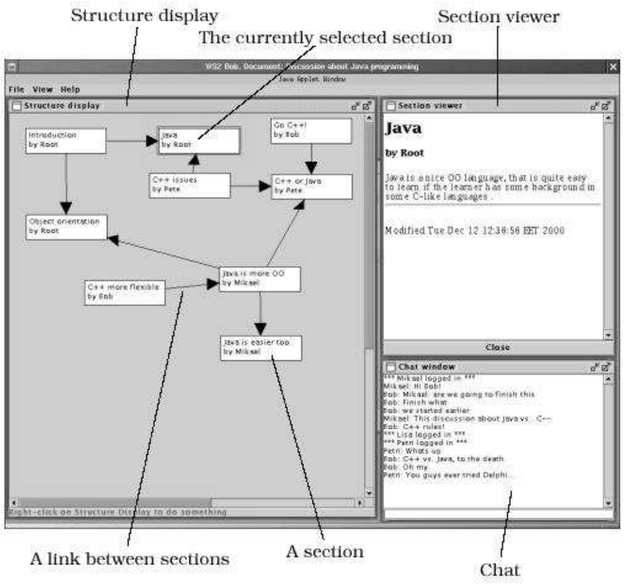

Figure 1: Screen dump of the WS2 client user interface.

The user interface works in WYSIWIS mode for the Structure display and Chat application (see Fig. 1), all users working on the same document see similar views of the components. Strictly speaking the Structure display uses relaxed WYSIWIS [22], because user selections that it shows are not projected in other users' Structure displays. The Section viewer works in non-WYSIWIS mode, as users use it to browse the text passages individually.

WS2 includes a simple user management system, which protects the system with user passwords and access control rules. Users of three different levels can be specified: super users, administrators and users. Users may read text passages and insert new text passages in documents. Administrators may in addition create and modify new documents and super users may in addition manage users. There can be any number of users in the WS2 system simultaneously working on an arbitrary number of documents.

The main contribution of the current prototype is the underlying system architecture, which we will describe in the next subsection.

4.1 Technical features

The WS2 is a stand-alone Internet-based collaborative writing software system for Wan use. The WS2 system is coded with Java 2 for maximal platform independence and it can be run on any platform that support Java 2. The architecture of the current implementation consist of a centralized with a server that distributed clients contact. The server stores the data that the system manipulates and facilitates communication between the clients. The clients contact the server to obtain the latest version of the data at startup, and then receive updates from the server during runtime. The clients do not store any data locally. The server uses a database to store the data. The database engine is implemented in Java too, so it does not complicate the system requirements beyond that of the Java 2 support.

Communication between clients and server is implemented by HTTP-tunneling. In HTTP-tunneling all communication is done by sending HTTP-packets between the clients and the server. This way the clients can communicate with the server even if they are behind firewalls, as most firewall configurations permit access to the WWW. In other words the communication between a WS2 client and the server looks like ordinary web-surfing from the firewall's point of view. The server must be installed on a web-server which routes the clients' HTTP-requests to it.

The installation of the WS2 client programs is a very straightforward process. In addition to the installation of Java 2 support only one file containing the WS2 client executable is copied on the hard disk of the computer. The installation of the server is similarly simple, but may involve the opening of a communication socket if a firewall is in use.

4.2 Pedagogical aims

WS2 is intended to be an open learning environment that can be used to a wide variety of different learning scenarios. The basic idea behind the system is that it lets users to engage in collaborative document production to achieve different learning goals.

The simplest form of WS2 use is with only a few users. For example two students may jointly write a presentation or a argumentation as with the TC3 system [7]. In an another case a teacher and a student may share a document, the student writes her portfolio there and the teacher can access and comment it.

WS2 may be used to facilitate collaborative learning through writing, thus facilitating the learning of collaboration skills. For example software engineering projects in computer science education typically include 3-5 students working on a software project under the supervision of one or two teachers. Much of the work done in these projects are related to writing: requirements specification and other documentation, code writing. WS2 can serve as a platform for a project of this kind, allowing students and teachers to view, modify and comment a set of documents.

The ability of the WS2 system to operate all over the Internet can be used to facilitate collaboration between students from different countries to promote cultural exchange and cross cultural projects in general to encourage interaction between cultures to diminish prejudices.

For example combining a course of academic english for non-english speakers to a scientific writing or communication course for native english speakers might be very interesting.

Distance education is an area, where the WS2 system can really make a contribution. If a group of students are located in different areas, then a virtual environment may provide them with possibilities for group work and a sense of a 'class' with classmates. As writing is a form of communication it can be used to make interaction between the students, and to make the connection between the teacher and student a living one.

4.3 Future plans

A new prototype system, Woven Stories 3 (WS3), is in the making. The WS3 is a major rework of the WS2 system both from the users' and software architecturers' point of views, althought the new solutions draw heavily from the WS2 system.

The most apparent change is a new user interface, that lets users to view the document in different ways. One of the views will be similar to the WYSIWIS found in most mainstream word processor systems. A other view will be a zoomable interface, where the users may zoom out from the point where the text fills the whole screen to see the whole set of sections and subsections of the whole document (Fig. 2). The zoomable user interface has been used in a co-writing and co-drawing system for kids called KidPad [3]. The users of KidPad found the zoomable interface very useful and exiting.

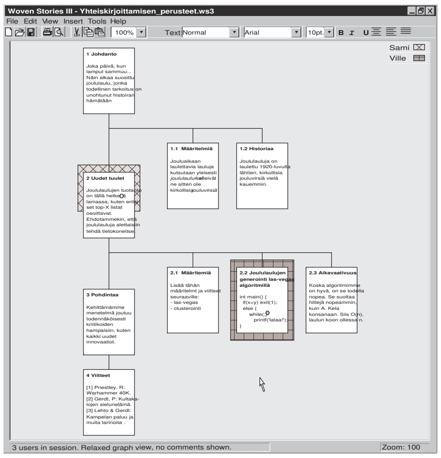

Figure 2: Concept art: The new zoomable interface of the WS3 prototype system.

The WS3 needs a better support for awareness. The users need for example information about whether other users are logged and active in the system, what has happened since the last time they were logged into the system and what the other users think of their contributions. One way to provide information about logged users is to show where they are working by highlighting their viewport, as has been done in Fig. 2, where a user has zoomed out as far as he can and sees two shaded rectangles which represent the views of two other users working on the same document. Another way to improve awareness is to improve communication. WS3 will include annotation tools, that let the users attach comments to the text in documents (in WS2 a user may attach a text passage as a comment to another text passage one, but WS3 will be more explicit about the comments). This way asynchronous communication is possible.

The user management of WS2 will be improved by introducing configurable rights management in WS3 with the possibility to tag a collection of rights with a role label. This way it is possible to create and support different learning situations. The rights management will be extended to work principally similar to the Unix rights management system, where users may change the rights to objects they own anytime.

WS3 will include reports about the usage of the system that can be used by the teacher to get a overview about the evolving learning situation. For example a report can tell the teacher how many words different students have written, how many comments they have attached to other students' work and what is the frequency of use of the system for a group of students. The teacher can use these reports as indicators of how the learning situation is going and to find situations where possible intervention is needed.

5 Final remarks

We are convinced that computer supported collaborative authoring could be used in education to gain collaboration skills and to make learning an richer and more social experience. Computer supported collaborative authoring could for example be used in distance education to bring the students together in joint projects. We have implemented a prototype system [10, 9] that we intend to use and develop to explore the use of computer supported collaborative writing as a integral part of education. The system facilitates co-writing over the Internet both synchronously and asynchronously overcoming geographical barriers and limitations set by timezones. The development of technology is a two sided process, where new technology creates new procedures of work and vice versa. We have started to find uses and relevant contexts for the application of co-writing in education, but we are still in the early stages of research. In the near future we intend to develop our prototype system further and to test it as a part of a course to find out how students react to the system and the application of computer supported collaborative writing in general.

References

[1] R. Baecker, D. Nastos, I. Posner and K. Mawby, "The User-centered Iterative Design Of Collaborative Writing Software," *Conference proceedings on Human factors in computing systems*, 399–405 (1993).

[2] R. Baecker, G. Glass, A. Mitchell, and I. Posner, "SASSE: The Collaborative Editor," *Proceedings of the CHI '94 conference companion on Human factors in computing systems*, 459–462 (1994).

[3] S. Benford, S. B. Bederson, K. Akesson, V. Bayon, A. Druin, P. Hansson, J. Hourcade, R. Ingram, H. Neale, C. O'Malley, K. Simsarian, D. Stanton, Y. Sundblad, and G. Taxn, "Designing storytelling technologies to encouraging collaboration between young children," *Proceedings of the CHI 2000 conference on Human factors in computing systems*, 556–563 (2000).

[4] C. Bonk and K. King, "Computer Conferencing and Collaborative Writing Tools: Starting a Dialogue About Student Dialogue," *Proceedings of CSCL '95*, (1995).

[5] T. Cavalier, R. Chandhok, D. Kaufer, J. Morris, C. Neuwirth, "A visual design for collaborative work: columns for commenting and annotation," *Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences*, 729–738 (1991).

[6] C. Ellis, S. Gibbs and G. Rein, "Groupware: some issues and experiences," *Communications of the ACM*, **34(1)**, 39–58 (1991).

[7] G. Erkens, J. Jaspers, H. Tabachneck-Schijf and M. Prangsma, "Computer-supported collaboration in argumentative writing," *Proceedings of the EuroCSCL '01*, Internet WWW-page, URL: http://www.mmi.unimaas.nl/euro-cscl/venue.htm (2001).

[8] R. Fish, R. Kraut and M. Leland, "Quilt: a collaborative tool for cooperative writing," *Proceedings of Conference Sponsored by ACM SIGOIS and IEEECS TC-OA on Office information systems (COIS'88)*, 30–37 (1988).

[9] P. Gerdt, "Woven Stories 2 Documentation Index," Internet WWW-page, URL: http://cs.joensuu.fi/ ws2/docs/ (2001).

[10] P. Gerdt, P. Kommers, C. Looi and E. Sutinen, "Woven Stories as a Cognitive Tool," *Cognitive Technology 2001*, LNAI 2117, Springer-Verlag, 233–247 (2001).

[11] J. Grudin, "Why CSCW applications fail: Problems in the Design and evaluation of organizational interfaces," *Proceedings of the conference on Computer-supported cooperative work*, 85–93 (1988).

[12] C. Gutwin, G. Stark and S. Greenberg, "Support for Workspace Awareness in Educational Groupware," *Proceedings of the Conference on Computer Supported Collaborative Learning*, 147–156 (1995).

[13] H. ter Hofte, "Generic service features of CSCW applications," *Technical Report RS/96002*, Telematics Research Centre, (2001).

[14] U. Hoppe, A. Lingnau, I. Machado, A. Paiva, R. Prada and F. Tewissen, "Supporting collaborative activities in computer integrated classrooms-the NIMIS approach," *Proceedings of the 6th International Workshop on Groupware (CRIWG'00)*, 94–101 (2000).

[15] P. LeBlanc, "Social Constructivist Theory and the Design of Writing Software: The Race Between Theory and Technology," *ACM SIGCUE Outlook - Special issue on computer supported collaborative learning* **21(3)**, 53–54 (1992).

[16] A. Mitchell, I. Posner and R. Baecker, "Learning to write together using groupware," *Conference proceedings on Human factors in computing systems*, 288–295 (1995).

[17] C. Moran and E. Klem, "The Situation of the Teacher in the Networked Writing Classroom," *ACM SIGCUE Outlook - Special issue on computer supported collaborative learning* **21(3)**, 59–61 (1992).

[18] C. Neuwirth, D. Kaufer, R. Chandhok and J. Morris, "Issues in the design of computer support for co-authoring and commenting," *Proceedings of the conference on computer-supported cooperative work*, 183–195 (1990).

[19] C. Neuwirth, D. Kaufer, R. Chandhok and J. Morris, "Computer Support for Distributed Collaborative Writing: Defining Parameters of interaction," *Proceedings of the conference on Computer supported cooperative work*, 145–152 (1994).

[20] C. Neuwirth, D. Kaufer, R. Chandhok and J. Morris, "Computer Support for Distributed Collaborative Writing: A Coordination Science Perspective," *Coordination Theory and Collaboration Technology*, Computers, Cognition and Work, Lawrence Erlbaum Associates, 2001.

[21] I. Posner and R. Baecker, "How people write together," *Proceedings of the Twenty-Fifth International Conference on the System Sciences*, 127–138 (1992).

[22] M. Stefik, D. Bobrow, G. Foster, S. Lanning and D. Tatar, "WYSIWIS revised: early experiences with multiuser interfaces". *ACM Transactions on Information Systems*, **5(2)**, 147–167 (1987).

[23] Y. Tanikawa, H. Suzuki and H. Kato, "A Synchronous Collaborative Editing System for Learning to Write," *Proceedings of the CSCL'99*, 621–630 (1999).

Teaching High-School Mathematics with Structured Derivations in Hypertext Format

T. Kavander and T. Salakoski
University of Turku and TUCS
Lemminkäisenkatu 14-18A 20520 Turku

M. Nylund and J. von Wright
Åbo Akademi and TUCS
Lemminkäisenkatu 14-18A 20520 Turku

M. Peltomäki
Kupittaa High School and TUCS
Sireenikuja 1 20720 Turku

tanja.kavander@cs.utu.fi

1 Introduction

Structured derivation is a calculational proof paradigm developed for the formal refinement of computer programs and reasoning about their correctness. Using structured derivations as a semi-formal representation in teaching high-school mathematics, part of the learning material is presented to the students in hypertext format browsable on the Internet. These structured derivations are applied as a new, more structured way of deriving and presenting solutions to typical problems. Every step of the derived solution to an exercise is presented with facts justifying the step. Furthermore, when solutions are very long, parts of them (subderivations) can be hidden and replaced with a hypertext link giving a more detailed view of the partial solution. In this way, the concept of mathematical proof, or derivation, is informally introduced to the students to enhance their mathematical thinking beyond that achieved by the standard high-school approach.

This method of teaching mathematics at high school level is being examined and tested in Kupittaa High School, in Turku, with a group of students who take their first courses in high-school mathematics. In addition to the standard mathematical background, basic propositional logic with a few notations have been taught at the beginning of the course. In addition to the test group, another group of students is simultaneously studying the same course contents in mathematics the traditional way.

In the calculational proof paradigm, which has been attributed to Feijen and is described in detail by van Gasteren [1], the mathematical expressions are transformed step by step. In Computer Science community, this proof style has, in addition to mathematical proofs, been used for proving the correctness of programs and refining program specifications. Based on their work with program specifications, Back and von Wright have developed the idea of structured derivations [7, 8]. Structured derivations enlarge the basic calculational proof style with subderivations. The structured proof style is also an alternative presentation of Gentzen's natural deduction. With structured derivations, the benefits of natural deduction and calculational proof are grouped together: the hierarchy of natural deduction and the easy way of reading calculational proofs.

When applying this method in high-school mathematics, writing solutions to typical expression reduction problems means that a new version of an expression is written on a new line and between the lines is written a semi-formal justification for the step. Thus, problem solving is seen as a rather formal and incremental transformation with well-defined steps from the initial problem description to a final irreducible statement yielding the solution. With subderivations, auxiliary details can be added to the solution in an organized way. The idea of subderivations is especially suitable for presentation in hypertext format with a web browser, because the hierarchical nature of structured derivations can be shown or hidden dynamically at will.

2 Structured derivations in teaching

The structured derivations method is applied in teaching the first two courses in high school mathematics, which contain less than 30 hours for face-to-face teaching in a classroom for one course. These courses contain basic algebra, calculus, and some function theory. A major part of the matter have been taught in the comprehensive school, so presenting a new way of resolving problems is quite safe. Before applying this method a basic knowledge of logical connectives, equivalence, and truth values had to be taught to the students. Within well-defined time limits, only four or five hours could be used for teaching the foundations of logic.

Structured derivations are used by the teacher when presenting the basic theory face-to-face and writing additional examples on board. The method is also applied to all the pre-made examples of the study material. Whenever the schedule allows, examples are shown using a web browser. Only the teacher has the access to the web pages, because students can't be allowed to see the correct solutions for the home exercises beforehand. Students, on the other hand, use structured derivations in doing their homework and as they represent their solutions to the exercises on board. Subderivations are used as a way of binding partial solution together.

As an example of applying this method in the courses in detail, the solving of an inequality with absolute values is shown next. It is shown in fully expanded form as written on board. When using a browser, subderivations corresponding to the underlined explanations can be hidden and replaced with a hypertext link giving a more detailed view of the partial solutions.

$|3x - 6| \geq 2$

$\equiv$ { property of absolute values }

$3x - 6 \geq 2 \vee 3x - 6 \leq -2$

$\equiv$ { <u>solve the inequalities</u> }

- $3x - 6 \geq 2$

  $\equiv$ { add 6 to both sides of the inequality }

  $3x \geq 8$

  $\equiv$ { divide both sides of the inequality with 3 }

  $x \geq \frac{8}{3}$

- $3x - 6 \leq -2$

  $\equiv$ { add 6 to both sides of the inequality }

  $3x \leq 4$

  $\equiv$ { divide both sides of the inequality with 3 }

  $x \leq \frac{4}{3}$

$x \leq \frac{4}{3} \vee x \geq \frac{8}{3}$

Besides the first courses of basic algebra, calculus, and function theory, there is good potential for using this method also in forthcoming courses dealing with, e.g., geometry and trigonometry. As an example is shown an exercise from the Finnish matriculation examination from the advanced level test in 1995. This exercise deals with a barrel shaped like a right circular cylinder, with a volume of 200 liters and the length of it's lateral edge of 1.12m. There is 32cm of fuel in the barrel when it is lying on it's side. According to the exercice, a customer paid 600 marks for the barrel and the fuel together. The question then is, was that a good buy, if the fuel costs 4.67 marks per liter and the price of the barrel is 100 marks. The problem solution also refers to a figure (Fig. 1).

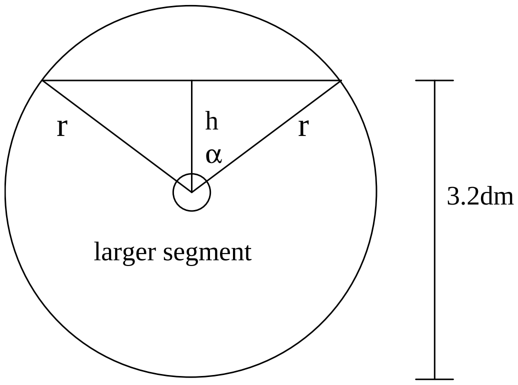

Figure 1: A picture of the fuel exercise.

The man makes a good buy

$\equiv$ { set up condition }

"price of fuel and barrel each sold separately" > 600 mk

$\equiv$ { price of barrel is 100 mk, one liter of fuel costs 4.67 mk }

$4.67\ \text{mk/dm}^3 \cdot$ "amount of fuel" $+ 100$ mk > 600 mk

$\equiv$ { <u>determine amount of fuel</u> }

- "amount of fuel"

  $=$ { based on the shape of the cylinder }

  "area of larger segment" $\cdot$ 11.2 dm

  $=$ { <u>determine larger segment</u> }

  - "larger segment"

    $=$ { definition of segment }

    $(360^\circ - 2\alpha)/360^\circ \cdot \pi r^2 +$ "central triangle"

    $=$ { <u>determine r, α and central triangle</u> }

- r

$= \{\ V = \pi r^2 \cdot h,\ \text{isolate } r\ \}$

$\sqrt{V/(\pi h)}$

$= \{\ V = 200\ \text{dm}^3, h = 11.2\ \text{dm}\ \}$

$\sqrt{200\ \text{dm}^3/(\pi \cdot 11.2\ \text{dm})}$

$= \{\ \text{compute}\ \}$

$2.3841\ \text{dm}$

- α

$= \{\ \text{Fig. 1}\ \}$

$arccos\left(\frac{3.2\ \text{dm} - r}{r}\right)$

$= \{\ r = 2.3841\ \text{dm}\ \}$

$69.987°$

- "central triangle"

$= \{\ A = 1/2(r^2\ sin(2\alpha))\ \}$

$1/2 \cdot (2.3841^2\ \text{dm}^2 \cdot\ \sin\ (2 \cdot 69.987°))$

$= \{\ \text{compute}\ \}$

$1.828\ \text{dm}^2$

$(360° - 2 \cdot 69.987°)/360° \cdot \pi \cdot 2.3841^2\ \text{dm}^2 + 1.828\ \text{dm}^2$

$= \{\ \text{compute}\ \}$

$12.742\ \text{dm}^2$

$12.742\ \text{dm}^2 \cdot 11.2\ \text{dm}$

$= \{\ \text{compute}\ \}$

$142.7\ \text{dm}^3$

$4.67\ \text{mk}/\text{dm}^3 \cdot 142.7\ \text{dm}^3 + 100\ \text{mk} > 600\ \text{mk}$

$\equiv \{\ \text{compute}\ \}$

$766.43\ \text{mk} > 600\ \text{mk}$

$\equiv \{\ \text{simplify}\ \}$

T

Besides high-school mathematics, the new method can be used also at university level. The last example of real analysis shows how the structured derivations are used in proving the uniform continuity of the function f(x)=2x. Here, the underlying ideas of mathematical proof are reflected by the fact that the derivation is not an expression reduction but a transitive series of statements connected by conditionals, or backward implications. The details of this as well as of the use of first order quantification and the focusing mechanism can be found in [8].

$$
\begin{array}{lll}
 & f \text{ uniformly continuous} & \\
\equiv & \{ \text{ definition of continuity } \} & \\
 & \forall \epsilon : \epsilon > 0 \Rightarrow (\exists \delta : \delta > 0 \wedge (\forall x\ y : x - y < \delta \wedge x > y \Rightarrow |f(x) - f(y)| < \epsilon)) & \\
\Leftarrow & \{ \underline{\text{focus on consequent, replace in monotonic context}} \} & \\
 & \quad \bullet \quad [\epsilon > 0] & \\
 & \quad\quad \exists \delta : \delta > 0 \wedge (\forall x\ y : x - y < \delta \wedge x > y \Rightarrow |f(x) - f(y)| < \epsilon) & \\
 & \quad \Leftarrow \{ \underline{\text{focus on innermost consequent, replace in monotonic context}} \} & \\
 & \quad\quad\quad \bullet \quad [\delta > 0, x - y < \delta, x > y] & \\
 & \quad\quad\quad\quad |f(x) - f(y)| < \epsilon & \\
 & \quad\quad\quad \equiv \{ \text{ definition of } f \ \} & \\
 & \quad\quad\quad\quad |2x - 2y| < \epsilon & \\
 & \quad\quad\quad \equiv \{ \text{ simplify using assumptions } \} & \\
 & \quad\quad\quad\quad x - y < \epsilon/2 & \\
 & \quad\quad\quad \Leftarrow \{ \text{ transitivity using assumptions } \} & \\
 & \quad\quad\quad\quad \delta \leq \epsilon/2 & \\
 & \quad\quad \exists \delta : \delta > 0 \wedge (\forall x\ y : x - y < \delta \wedge x > y \Rightarrow \delta < \epsilon/2) & \\
 & \quad \Leftarrow \{ \exists - \text{introduction rule, witness for } \delta \text{ is } \epsilon/2 \ \} & \\
 & \quad\quad \epsilon/2 > 0 \wedge (\forall x\ y : x - y < \epsilon/2 \wedge x > y \Rightarrow \epsilon/2 < \epsilon/2) & \\
 & \quad \equiv \{ \text{ simplify using assumptions } \} & \\
 & \quad\quad \mathsf{T} & \\
 & \forall \epsilon : \epsilon > 0 \Rightarrow \mathsf{T} & \\
\equiv & \{ \text{ simplify using basic rules of predicate logic } \} & \\
 & \mathsf{T} &
\end{array}
$$

3 Technical considerations

The web pages are basically constructed using standard html augmented with images for representing complex mathematical expressions and Java scripts introducing the collapsible lists feature. Every line of the derivation is written in a TeX file, which is then converted into html [3, 4]. As the mathematical expressions are images, they can be inserted in an appropriate way in the html-document.

Structured derivations are presented as collapsible lists based on the Java scripts [2], slightly modified for enhanced appearance. Every derivation starts out as an empty list, and items are then added to the list. Every other line in the derivation is an image converted from TeX files and the other line is a simple plain text comment. When subderivations are used, a sublist is created and added to the main list. The items in the lists are html code, either comments or references to images, all written in the html file representing the solution. Having generated necessary images and written the corresponding html file, the solution can be rendered and browsed using any browser with support for Java scripts.

4 Evaluation and discussion

At first, the students were disappointed with the new method, because they felt that they were forced to write auxiliary text when solving the exercises. The use of logic, on the other hand, has not been any kind of burden, but especially in solving inequalities, logic proved out to be a very effective tool. The use of logic has cleared the students' solutions and unified their structure. Also, using this method makes it easier for the students to follow the teaching process. The most important difficulties experienced have been related to the hasty course schedules; e.g., too few examples can be shown on the web. There has also been some technical problems regarding the complex process of preparing the web pages.

The method is being tested with two groups of students. Students' attitudes towards mathematics and their mathematical skills were tested before the courses began. The test group consists mainly of students that are selected to a special group focused on computer science and media subjects in their high school studies. The mathematical talents of the students in the test group are at least as good as those of the students in the control group. When enrolling to high school, the average grade in mathematics was 9.7 (on a scale from 4 to 10) for the test group and 8.9 for the control group. The difference between former school grades in other studies is not significant. Students came from several schools, so their grades are not directly comparable. Differences can be found, however, in the attitudes toward mathematics. As contrast to the control group, students in the test group rely on themselves when doing mathematical exercises. They are more confident about applying their mathematical skills also for other studies, not only for mathematics.

The test group and the control group are taught by different teachers. The primary study material used by the test group is done by their teacher, while the control group used a normal mathematics course book [3]. Both groups used the same exercise book related to the course book [2].

The examination for the first course was the same for both groups. There were eight problems from which seven had to be solved. The answers of the test group were very clear, written in an organized way, and easy to check. The time spent in correcting exam was remarkably diminished, because it was easy to immediately notice where the student lost the idea of the solution. There were a few students whose answers were somewhat confusing, but even they were quite understandable. It became clear, that if a student can't write his thoughts on a paper, then his answers are not correct either. Only one student in the test group did not use this method in general. However, also he used this method, when he had to tell the property of the imaginary unit; i.e., when the topic was difficult enough and he felt he concretely gained something using the method.

Comparing the results of the two groups is rather difficult, because teachers teach somewhat different things, and even with the same course content, they concentrate on different things in their teaching. Therefore the biggest differences found in correct and incorrect answers cannot be analyzed in detail. The lengths of the answers differs only due to the explanations of the method; especially, there were no differences with the lengths of right and wrong solutions. The steps in the solutions were pretty much the same with both groups.

Only a few students succeeded in applying this method as a way of shortening their answers to exercises. The tests show, that the most talented ones could enhance the use of this method more than what was noticed during the lessons. As a drawback it is also obvious, that if students try to write as short explanations as possible, it also makes checking their answers harder. Especially while doing an exam, good explanations would help many students to double-check their answers themselves.

Although it became evident that the control group did not succeed as well as the test group in the exam, we can't yet conclude whether it is related to the new method or not relying only on this knowledge. The results must be examined in more detail, and we are currently developing a statistical model based on the prior knowledge of the history of the students and the teachers for eliminating the various other factors. However, it can already be said that in the exam, the students in the test group explained their answers much more precisely than the students in the control group.

The second course is now in progress. The students have been tested by a formative test, which does not affect the total grade of the course. The results of this test resembled the exam results: the control group did not succeed as well as the test group. To summarize, it seems rather certain at this point, that this method does not at least impair the performance of the students.

The results of the exam can't be generalized because different teachers vary in their teaching methods and thoughts about the importance of various subjects. The two first courses in advanced level mathematics contain basic things and that is why applying this method is working in a very basic manner. The main point in these courses is to enhance the calculus skills. When exercises become more difficult the applying of the method binds together more things and thoughts. Due to the difficulties in quantitative analysis of the results, it would be interesting in the future to analyze also the qualitative results of the students. Furthermore, the differences in the students' other skills, e.g., in programming, could be analyzed as they will have the same programming courses. Students in these courses are from both of the groups. Then, the structure and logic of the programs written by the students from the two groups could be examined in more detail.

In the future we have to concentrate on the good qualities of this method. It is essential that the normative explanations should be kept very short and informative. Talented students can apply this method more freely. In forthcoming courses, a deeper analysis of the criteria for using this method is also important; in what kind of exercises students want to use this method and why.

References

[1] A. J. M. van Gasteren, *On the Shape of Mathematical Arguments* Lecture Notes in Computer Science. (Springer-Verlag, Berlin, 1990).

[2] *JavaScript DHTML Collapsible Lists*, 23 November 2001, `http://devedge.netscape.com/evangelism/tools/xbCollapsibleLists/`.

[3] *LaTeX: A document preparation system*, 23 November 2001, `http://www.latex-project.org/`.

[4] *LateX2HTML Development Repository*, 23 November 2001, `http://www.latex2html.org/`.

[5] P. Kontkanen, R. Liira, K. Luosto, J. Nurmi, R. Nurmiainen, A. Ronkainen and S. Savolainen. *Pyramidi. Matematiikan harjoituskirja 1.* (Tammi Hämeenlinna 2001).

[6] P. Kontkanen, R. Liira, K. Luosto, J. Nurmi, R. Nurmiainen, A. Ronkainen and S. Savolainen. *Pyramidi. Matematiikan tietokirja 1.* (Tammi Hämeenlinna 2001).

[7] R. J. Back, J. Grundy and J. von Wright, "Structured Calculational Proof," Tech. Rpt. 65, Turku Centre for Computer Science. (November 1996).

[8] R. J. Back and J. von Wright, "Structured Derivations: a Method for Doing High-School Mathematics Carefully," Tech. Rpt. 246, Turku Centre for Computer Science, (March 1999).

Instructional Intervention In a Web-Based Programming Course

E. Sutinen and S. Torvinen
Department of Computer Science
University of Joensuu, Finland
P.O. Box 111
FIN-80101, Joensuu, Finland

sirpa.torvinen@cs.joensuu.fi

1 Introduction

Difficulties in learning to program among the first year Computer Science students are well known. In distance education these problems become even more evident: there is a wide experience that far more effort is needed in designing courses than in contact-lessons in classroom [6]. In distance learning student's responsibility for his own studies is even larger than in face-to-face teaching. That is why the role of support measures is becoming more important, especially at the beginning of the distance studies.

In fall 2000 University of Joensuu offered for the first time a Web-Based University level Computer Science program to the high school students in the surrounding rural region of North Karelia. Roughly half of the contents dealt with programming. To get a common pedagogical framework for the course design, we developed the Candle model [1]. The main design principles in the Candle model are:

- To offer students *real human contacts* (online instructors and tutor-teachers at the schools);
- To *link the printed learning material with an activating learning environment on the web*;
- To utilize the principle of *learning-by-doing*, e.g. apply visualization tools; and
- To *keep the user interface simple* and use an otherwise light technology.

During fall semester some of the students decided to drop out, part of them even without passing any one of the courses. High school students are studying their matriculation examinations at the same time with the Virtual Approbatur studies and for example exam periods in high schools are even without any extra studies quite urgent. Students had difficulties with scheduling their high school studies and university studies.

Most of the dropped gave up during the first programming course and we focused to find out the problems with learning basics of programming and the most difficult topics on it. We also analyzed needs for support measures and found out the main reasons to drop out. Based on the analysis of the feedback given by students who dropped out, students mainly did not benefit from supports given via Web, partly because of inexperience to use Web-Based support measures like discussing forum, chat or e-mail to asking help for learning difficulties. Instructional intervention proved to be necessary in Web-Based courses focused to the high school students.

2 Virtual Approbatur

In Finland, the Ministry of Education is funding a three-year project to establish the Virtual University of Finland, during years 2001-2003. One of the particular goals in the project is to develop new methods for science education. The three universities in eastern Finland, University of Joensuu, University of Kuopio, and Lappeenranta University of Technology, work jointly in the virtual university project. One of the concrete objectives is to create a Web-Based learning environment for an introductory Computer Science course, intended for high school students [2].

On this project, called *Virtual Approbatur (Virtual Certificate)*, high school students can take 15 credits of Computer Science studies in one and a half year via the Internet. Courses will give students basic knowledge of three main domains:

- Introduction to Computer Science (5 credits),
- Basics of Programming with Java (7 credits), and
- Preliminaries of Computers (3 credits).

In the Finnish University system, each credit equals 40 hours of studying; 120 credits are required for the Bachelor's degree. One of these credit point equals two credit points in the International Credit Transfer System. Thus, after passing all the offered 15 credits a student has the first year Computer Science studies completed. Moreover, if the student passes the program with grade 2/3, she will be free to enter the university as a Computer Science major.

3 Method

We evaluated the second completed course, Programming, part 1 (2 credits), which covers the introduction to the concepts of programming in Java. We analyzed the course design from two viewpoints:

- *Learning outcomes:* the examination papers and exercises of all the students in the course were inspected.
- *Students' attitudes:* students who *dropped out* from the curriculum either during this course or after passing it were interviewed.

The questionnaire sent to the dropouts, or the *test group*, included ten questions.

- The first two questions were focused on the learning environment and the course materials on it.
- The following questions dealt with the study process: which fields of programming the students regarded as difficult, are there any needs for support measures like on-line lessons, videoconferencing or chat in WebCT.Students were also asked the main reason for dropping out.
- The last two questions were open questions: what would the students like to change in the course, are there any ideas to improve the quality of the materials or arrangements of the course etc.

A total of 25 students completed the query form. A more detailed follow-up questionnaire was sent to six volunteers by e-mail.

4 Results

4.1 Learning Outcomes

To participate in the examination, student had to complete 1/3 of the given 50 exercises. Out of 79 students, 76 submitted at least one exercise and 62 of them returned the compulsory amount of exercises. Out of 62 students allowed to participate 56 took part in the exam; a total of 20 in this group dropped out after the course. The exam included four questions on the following themes:

- **Question No.1:** Declare the meaning of the if-statement
- **Question No.2:** Write a Java program with an if-statement and a loop
- **Question No.3:** Concept of randomized numbers and using of array
- **Question No.4:** Code a Java applet (loops are needed)

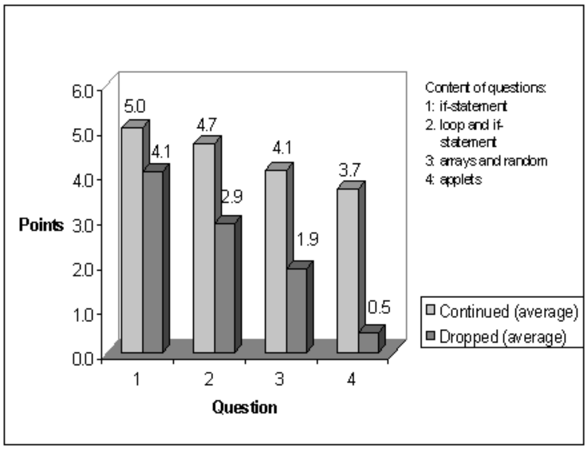

Figure 1: Results of exam by passed and dropped out students.

Figure 1 shows the results of the exam by those students who continued and those who dropped out and Table 1 shows the relative value of dropped out students' points comparing with the points of passed students.

Table 1: Dropped out students points relative part from passed students points.

Question	Relative part of passed students points
1: if-statement	81 %
2: loop and if-statement	62 %
3: random numbers	46 %
4: applets	12 %
Total	78 %

One can clearly see from Figure 1 and Table 1 that on easy questions dropped out succeeded relatively well comparing with passed students but when we look at for example question 4 which proved to be difficult one, we can see that dropped out students got only 12% of passed student's points. So the more difficult topics the more larger are differences between higher and mid-performers' success in exam.

In Candle-model *learning by doing* is one of the main principles [1]. On Figure 2 we can see how dropped out students and continued students submitted exercises.

Students had difficulties with scheduling and it might have interfered to the amount of submitted exercises. If a student does not have time enough to do exercises, she seems to meet serious problems with the examination. In Figure 2 we can see that the main portion of dropped out students submitted less than half of the exercises. On the other hand, slightly over 90% of passed students (31 out of 34) returned at least 50% of exercises. The benefit of doing for learning is obvious.

Some of the exercises were and will be hard by nature and there will always be students who cannot make use of these exercises to improve their learning outcomes. However, the

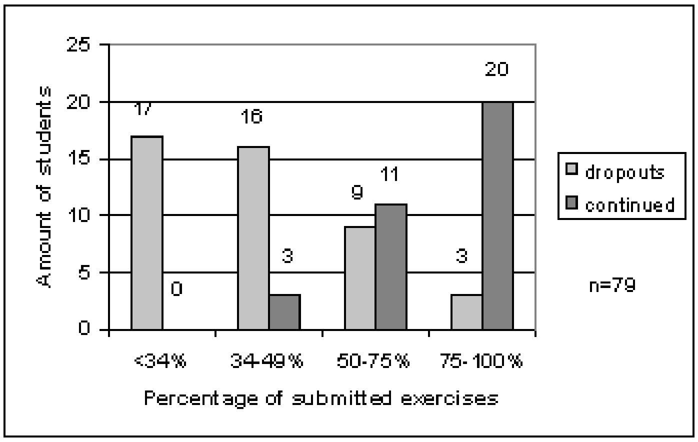

Figure 2: Submitted exercises.

clear correlation between the number of submitted exercises and passing the course illustrates the need for a more careful content analysis of the exercises. In particular, the course material should have a continuum of assignments, in terms of difficulty. Any way exercises and tasks are considered important because they require students to apply their knowledge to new situations [5].

4.2 Students' Attitudes Towards the Course

As we can see in Figure 3 most of the dropouts gave up during the first programming course or right after it on January 2001. On January there was the retake of the first programming course. The bar on March 2001 has a very natural explanation: there were difficult topics to learn on the second programming course on that time.

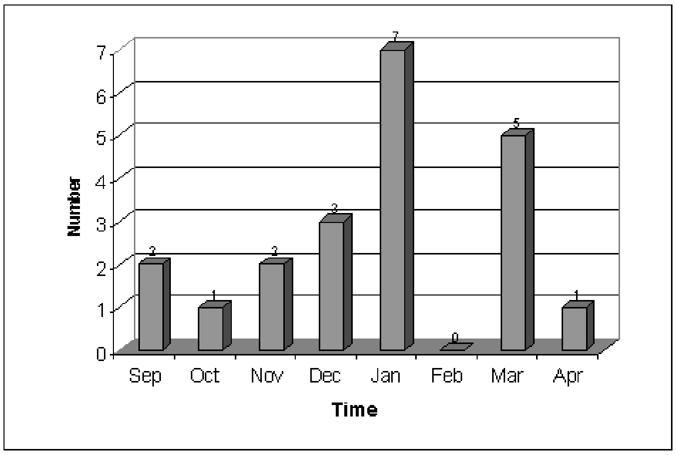

Figure 3: Schedule of dropouts.

Figure 4 shows the main reason to drop out. A total of 44% of students quitted because of the lack of time. 20% of students founded exercises too difficult to do and 8% did not pass the retake exam.

Lack of time reflects reducing amount of passed exercises and on that way it reflect even to the learning results of the students. Especially in programming courses learning by doing is the most efficient way to adopt new topics and when there is no time enough to do exercises it will very probably occur serious problems in exam also. On the other hand there will always be students who put more time and energy into socialization than academics or students who

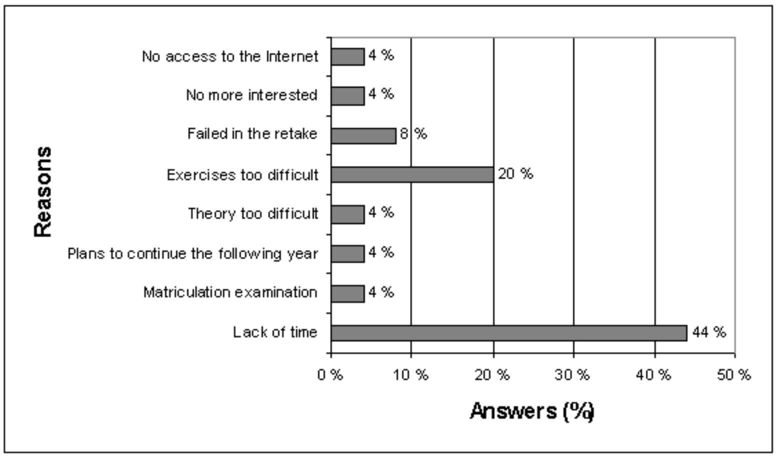

Figure 4: Main reason to drop out.

do not see the connection between what is being taught via the Web and his or her goals, needs or desires [7].

Table 2 shows the topics of programming that were considered as difficult by test-group. Quite unexpectedly, as many as 48% of students regarded conditional-statements as hard to learn. Surprisingly, a total of 72% considered arrays difficult.

Table 2: Topics of programming feeling as difficult.

Topic of programming	n=25
Variables and symbols	24 %
Assignment statement & input/output	32 %
If-statement and logical operations	48 %
Loops: for, while and do-while	24 %
Arrays	72 %
Methods	60 %
Applets and graphics	60 %
Making animations	56 %

When we compare information between Table 2 and Figure 1, we can see that the difficult topics are partly correlating with the results of exam but with loops we can see clear differences. Mainly the dropped out students did not use loops correctly in the exam so loops have been much more difficult for students than one might assume by the information shown in Table 2.

Using Java as an introductory programming language has some weaknesses like large size of libraries and difficulties with standard input [9]. The most difficult problem that students face with Java syntax is that there are so many concepts to grasp at the same time [3]. This makes the learning process even more difficult for those students who have very little programming experience or no programming experience at all because at the same time students have to learn the concepts of programming like variables, symbols, loops, conditional statements, arrays, methods etc.

4.3 Support measures

Although there was tutor-teacher in high schools and supervisors at the university, who aimed students with the problems, many of the dropped out students wish for more support especially in learning programming.

Table 3 shows that dropped out students were not very satisfy to the support measures given via Internet. We have to remember that frustration of not succeeding in the course can interfere the feedback on this question. If the passed students had also answered, the feedback might be more positive.

Table 3: Utility of the support measures.

Support	Disagree	Agree
Advises from supervisor made it easier to do exercises	52 %	20 %
Advises from supervisor helped to get over on next week's exercises	60 %	16 %
Comments of submitted exercises made it easier to understand difficult topics	56 %	20 %
Used WebCT's tools (like discussing forum or mail) to get help from other students	68 %	20 %
Found the answers to the difficult topics from WebCT's discussing forum	56 %	28 %
Discussing with other students (same high school) who also participated in virtual courses	16 %	76 %
Example solutions after exercises helped to understand the problems with own solutions	32 %	16 %
Average	49 %	28 %

But any way: the interesting point in Table 3 is that dropped out students seem to have no obvious benefit from advises or comments given via Web. Surprising is that even 32% of students found that example-solutions of exercises did not help in learning process. Table 3 shows also that dropped out students did not use tools of learning environment (like WebCT's discussing forum) to get help from each other but instead the face-to-face discussing with students at the own high school were fruitful for learning process.

Table 4: Needs for contact lessons, videoconferencing or chat in learning environment.

Topic of programming	Contact lessons	Video Conference	Chat in WebCT	Average
Variables and symbols	44 %	20 %	24 %	29 %
Assignment, input/output	52 %	12 %	20 %	28 %
Loops	40 %	16 %	24 %	27 %
If and logical operations	56 %	8 %	20 %	28 %
Arrays	52 %	16 %	40 %	36 %
Methods	56 %	20 %	28 %	35 %
Applets and graphics	56 %	16 %	24 %	32 %
Animations (thread)	64 %	20 %	24 %	36 %
Average	53 %	16 %	26 %	

On questionnaire students estimated the usefulness of contact lessons, videoconferencing or chat in learning environment as an optional method for a Web-Based programming course. As we can see in Table 4 even 53 % of students wished for contact-lessons (face-to-face) besides of Web-Based lessons. When we compare the information in Table 3 and Table 4 we can clearly see that dropped out students need **real human support** in their studies.

Surprisingly new methods like videoconferencing or chat in learning environment proved to be quite uninterested. Only exception is arrays: even 40% of students believed that chat in learning environment might help to learn arrays.

Quite expectedly the needs for support mainly correlates the feelings of difficulties (comparing Table 2 and Table 4). But even students found loops not so bad to learn, still 40% of them want contact-lessons to support the learning of loops. On the other hand, one might assume by the information shown in Table 2 that needs for supporting with arrays would be larger than it is now.

Many of the students found especially learning programming difficult to study via Internet. Students had opportunity to use the chat rooms or discussing forum in WebCT but the interesting towards them was quite poor. In future it will be very important to direct consciously students in Web-based courses to use the support tools offered by learning environment as a channel of help and support. It is known that without encouraging using these discussing tools, students seems not use them at all [8].

Well-succeeded students used these support tools already at school year 2000-2001 but dropped outs mainly did not use any of them at all. We found it important that not only the best students but also those students who own problems with learning process also realize the benefit of support tools like chat or discussing forum in Web-Based learning environment and start to use them as a natural way of learning. As we can see in Table 4 students can learn very much from each other and in distance education contacts with other students via Web can be as fruitful as discussing in face-to-face.

5 Conclusions

As we can clearly see high school students had difficulties with studying independent via Internet. One reason for that is naturally the age: Virtual Approbatur students are 16-19 - years old and some of them might own difficulties to carry out the responsibility of her own studies.

Secondly the whole Virtual Approbatur was a new way to learn: on the study it is clearly seen that students did not efficiently use the support measures offered in learning environment. On the other hand it was the first time to the University to offer so large amount of courses in distance. With the next Year's Virtual Approbatur we can avoid major of the pitfalls and experience will be one good tool for developing the virtual courses.

Based on the analyzed feedback, the curriculum has been modified to fit better to students' potential. Scheduling of topics has been modified so that there is more time to the difficult topics and on the other hand amount of requisite exercises is cut down. At the same time amount of optional exercises and easier examples has been increased.

Based on the study there is a clearly need for instructional intervention in Web-Based courses and we found out some ways to do it:

- **Giving advices for scheduling:** many of the students aged 16-19 years had difficulties to do their own timetable rationally. Some of them want to study even without having time enough to do it.

- **Giving individually feedback:** Especially mid-performers benefit on more details comments of submitted exercises.

- **Using examples and visualizations:** For example animations made by Jeliot is proved to help mid-performers to learn introduction programming [4].

- **Encourage using the support measures** offered in learning environment: encourage students to communicate with each other e.g. using team activities [8]. It is important to get students to know better each others and on that way make it easier to use discussing forum or e-mail for asking help.

- **Meaningful of exercises:** Easy optional exercises and examples will increase the motivation of mid-performances and succeeding in programming is going to help with learning process. That is why exercises should be considered as a vehicle that can direct learning behaviors in students [5].
- **Motivation in distance education:** Motivation is an important part of succeeded of Web-Based course. It is easy to give up in Web-Based courses because even 30-50% of all students who start a distance education course drop out before finishing [7].

At the virtual courses students' own responsibility of learning is more important than in contact-lessons but we still found the role of instructional intervention important. In the Web-Based programming course students need advises and encouragement to go on with studies. On the other hand there will always be situations where it would be more sensible to give up but the student still wants to carry on. What kind of indicator could tell us when it is rational to encourage students to go on even when he or she frequently failures with his studies? Especially in Web-based courses instructional intervention will be challenge and we just have to accept that no matter what intervention strategies are undertaken, there will always be those students whom the instructor will not be able to reach.

References

[1] A. Haataja, S. Kontkanen, J. Suhonen and E. Sutinen, "Teaching University-Level Computer Science to High School Students over the Web," *In Proceedings of World Conference on Educational Multimedia, Hypermedia & Telecommunication* (June 2001), Tampere, Finland, 2001.

[2] A. Haataja, J. Suhonen and E. Sutinen, "How to Learn Introductory Programming over the Web," *In Proceedings of EUNIS 2001, The 7th International Conference of European University Information Systems* (March 2001), Humboldt-University, Berlin, 2001.

[3] J. Hong, "The Use of Java as an Introductory Programming Language," In ACM Crossroads *The ACM's First Electronic Publications* Internet Web-site. URL: http://www.acm.org/crossroads/xrds4-4/introjava.html (2.11.2001)

[4] R. Ben-Bassat Levy, M. Ben-Ari and P. A. Uronen, "An Extended Experiment with Jeliot 2000," *In Proceedings of the First Program Visualization Workshop* (July 2000), pages 131-140, University of Joensuu.

[5] A. Carbone, J. Hurst and D. Mitchell, "Principles for Designing Programming Exercises to Minimize Poor Learning Behaviors in Students," *In Proceedings of the on Australasian Computing Education Conference* (December 2000), pages 26-33, Melbourne, Australia.

[6] J. R. Cordani and R. J. Tucker, "Tools for Higher Education Distance Teaching," *In Proceedings of the 26th SIGUCCS Conference on User Services* (October 1998), pages 71-76, Bloomington, USA.

[7] R. Cornell and B. L. Martin, "The Role of Motivation in Web-Based Instruction," In Khan, B.H. (Ed.): *Web-Based Instruction* (Educational Technology Publications, pages 93-106, New Jersey, USA, 1997).

[8] V. Klemm, "Creating Online Courses: A Step-by-Step Guide," *The Techonology Source* (May/June 2001), Internet WWW-site, URL: http://horizon.unc.edu/TS/default.asp?show=article&id=861 (1.11.2001)

[9] E. Roberts, "An Overview of MiniJava," *In The Proceedings of the Thirty Second SIGCSE Technical Symposium on Computer Science Education* (February 2001), pages 1-5, Charlotte, North Carolina.

SESSION 4

Evaluation of faculty workload for various methods in computer science education

J. Kurhila
Dept. of Computer Science, P.O.Box 26, FIN-00014 University of Helsinki

kurhila@cs.helsinki.fi

1 Introduction

The purpose of the paper is to evaluate the workload for faculty when considering various modern educational methods or approaches when developing computer science education. The methods are different in the sense that they have different orientation (learner vs. teacher centeredness) and they may or may not exploit information technology facilitating the learning process. The methods are independent from the topics within the field of computer science, i.e. they can be applied to basic programming courses as well as more advanced topics, such as computer graphics or software engineering project management.

There has been a growing need to develop computer science education, since qualified teachers are a rarity and at the same time the amount of computer science students has grown rapidly. Developing education in general has also been a trend during the last few years. For example, in Finland the concepts of *problem-based learning* [11] and its more developed form *learning by research* [3] are gaining a lot of room and advocates in higher and even in elementary education, although there is still no clear consensus how to employ those particular approaches in computer science education. Several reports of different trials exist [6, 7, 8, 12], where the trials apply a slightly modified form of problem-based learning or learning by research to some single courses.

The paper discusses about four different categories of approaches for computer science education. The paper does not argue that the approaches chosen represent the most commonly used or the most suitable approaches. Instead, they are chosen because they represent conceptual differencies underlying various possible approaches, thus covering at least a part of the current and up-to-date approaches. After the discussion, the approaches presented are positioned into a dimension according to their learner-centeredness, and the faculty workload aspect is discussed further.

2 Different educational approaches to computer science education

2.1 Traditional lecture-based courses

Lecture-based computer science courses mean in this context a model where the course consists of lectures and weekly (laboratory) exercises. Typically in Finland, the lectures last for 90 minutes twice a week during some twelve week period. Once a week a laboratory exercise session is held, where students present their solutions to the weekly exercises.

This traditional lecture-based model can be argued to be cost-efficient since the lectures can host 500 students at the same time. Lab exercises are typically hosted for up to 25 students. The model has been critizied lately because it lacks learner activity; this can been seen in drop-out rates and passing grades in final exams. Typically at the University of Helsinki, 35-50% of students attending the core computer science courses fail every year. In addition, the opponents of traditional lecture course argue that even those who pass the course cannot apply the knowledge gained, at least not after few years time.

2.2 Pure web-courses

Recently one distinctive problem in computer science education has been the lack of competent teaching personnel. To address this problem, there have been various attempts to build web-courses for the most popular computer science courses. The standard approach is to transform the material into the web, add exercises and assignments, and have newsgroup-type of discussion boards or chatrooms available.

Traditional web-courses rely on self-study, where the majority of the learners remains alone unless forced to communicate with each other. The drop-out rates for web-courses have often been even more than in traditional lecture courses, even when there is some interactivity in the material.

2.3 Variants of problem-based learning

Problem-based learning. M. Savin-Baden [11] describes problem-based learning (PBL) to be "an approach to learning through which many students have been enabled to understand their own situations and frameworks so that they are able to perceive how they learn, and how they see themselves as future professionals."

There are slight variants also inside PBL, but Boud proposed characteristics of many problem-based learning courses [1]. The characteristics include crossing the boundaries between discipline, intertwining theory and practice, and a change in focus from staff assessment of outcomes of learning to student self- and peer assessment.

In practice, the curricular content is organized around problem scenarios rather than subjects or disciplines [11]. Students themselves — working as a small team — decide and evaluate what information they need to learn and what skills they need to gain when trying to deal with the presented problem.

Problem-solving learning. A common mistake about PBL is to think problem-solving learning (PSL) to be problem-based learning. Whereas PBL consists of complex, open-ended situations and students setting their own objectives to tackle the situations, PSL focuses on giving students an article or some other type of stimulus, such as a lecture, and then a set of questions based upon the informatiopn given [11]. Students are expected to find the solutions to these questions, and at least in some cases, the solutions are discussed later in groups.

The key difference between PBL and PSL is that in problem-solving learning the "problem scenarios here are set within and bounded by a discrete subject or disciplinary area" [11]. There are (a set of) right answers, which are known to the teacher, and the teacher expects the students to get them. Moreover, the solutions to these problems "are always linked to a specific curricula content, which is seen as vital for students to cover in order for them to be competent and effective practitioners" [11].

Problem-solving learning is in a way simpler for the students and the teachers compared to problem-based learning. PSL has been used for years among practitioners, often without any attempt to radically re-think their teaching process.

Learning by research. Learning by research is a method where the studens' learning process is taken yet further into a direction where learning is seen as a process of research, and the students are researchers forming research groups [3].

The iterative process of learning by research is described as follows [3]:

1. Building a context
2. Setting the research problems
3. Building a strategy for research
4. Critical evaluation
5. Searching the knowledge
6. Refining the research problems
7. Building new strategies for research

In other words, a group of learners starts with a loosely defined problem with no clear or obvious solution. Learning by research means knowledge building in a social context which exploits and leads to shared expertise. Teacher's role is that of a tutor or a facilitator. It is important that the teacher suggests no solutions or even does not ask unnecessary questions. Instead, the group itself evaluates the theories formulated during the work and sets new objectives and generates the methods to achieve them, much like the process in PBL.

Learning by research differs from PBL in the sense that learning by research can include PBL as means to achieve research-type learning process, but other methods exist and can be used as well. For example, one popular method is to pose questions, to others and to oneself. The "theory" of learning by research is not yet clearly defined but the applications are galore, and a vast amount of litterature can be found on the subject (see e.g. [3, 4] for starting points).

2.4 Peer-assisted learning

To reduce the workload for the faculty (teachers), the concept of peer-assisted learning has been suggested as one possible remedy. However, it is often the case that the learning is not *solely* based on peer-helping but uses peer- help as an additive to e.g. lectures. On the other hand, peer-help easily becomes so integral part of the learning process that using the term "peer- assisted learning" is valid.

Traditional study circles. The simplest and the oldest model for peer-assisted learning is study circles. Study circles are often complementary to the traditional lecture course. The circles can be voluntary, obligatory, or strongly supported by some external rewards such as extra credit points for final grade. Sometimes study circles have a leader who is an authority such as a teacher or a teaching assistant but still the circle can involve peer-learning.

The common knowledge about study circles says that they should be formed only if there is a true need for them and the students are motivated to form a group and stay in it. Thus the problem remains; how to get less social students to be involved?

Computer-mediated peer-learning. I-Help represents an approach where the peer-assisted learning process is heavily supported by a computer system. I- Help is a state-of-the-art system to facilitate peer-help for the learners and thus taking off some of the workload of the teachers [9]. When simplified, I-Help can be described to be a communication tool for learners which includes a public discussion area but more interestingly, one-to-one discussions as well to be used for peer-help for the course content.

I-Help is based on a multi-agent architecture where every participant has his own agent. Agents negotiate with other agents and agree on giving and getting help from their human master. The interactions between the agents and ultimately humans involve an economic system, where credits (I-Help Credit Units, ICUs) are exchanged in return for helping [9].

I-Help is in extensive use at the University of Saskatchewan in Canada where it has been built, and the evaluation is in progress. The university is using the system to accompany traditional lectures. The authors state that the system [5]:

> "... should have a number of implications for organizations that use it: to strengthen weak ties among workers or learners, to foster a sense of community and common purpose, to spread the responsibility for teaching and learning widely in the organization, to more effectively achieve just in time learning, and to seamlessly integrate humans and learning technology."

Of course there are many other approaches between the approaches of traditional study groups and agent-based intelligent systems, but the idea underlying the approach remains more or less the same. Efforts have been made in the direction of peer-assisted learning where the students have prepared the whole learning environment for their peers, but at least in some cases the learning results have been less than originally intended [2].

3 Evaluation of the educational approaches

In this section, a model for evaluating various approaches to computer science education is proposed. The evaluation model has some resemblance to the model proposed by Reeves [10], although Reeves' model aims at evaluating computer-aided learning programs instead of computer science education. Reeves' model has fourteen bi-polar dimensions, and a learning program has a position for every dimension. Reeves' dimensions range from underlying epistemology, pedagogy and psychology to cultural issues and learner collaboration. In addition, Reeves' dimensions include e.g. "teacher control" which is also one of the issues in the evaluation proposed here.

Bi-polar one-dimensional evaluation. To make the evaluation simpler than in the Reeves' model, the approaches presented in the previous section are positioned along just one bi-polar dimension. The basic underlying question for the dimension is *learner-centeredness*. In this context, learner-centeredness has to be considered broadly, to incorporate also other aspects which can be seen as ingredients for or at least closely related to learner-centeredness (Fig. 1).

On the learner-centered end of the dimension, there are also aspects such as internal motivation, learners' ability to self-control and metacognitive skills. In other words, we can talk about a learner who can take responsibility for his own learning. On the other end of the dimension, there are aspects related to teacher-centeredness. These aspects include the need for external motivation like rewards and forced obligations, and the lack of the need for self-control since the learning process is teacher-driven.

There is always room for debate when positioning different educational approaches for computer science education along such a dimension, partly because of the variation within the approaches themselves. One can argue that e.g. traditional lectures can in some cases be learner-centered. This might be the case but the evaluation presented reflects one possible scenario and the overall and general conception of the author, and should of course be subjected to further discussion.

Figure 1 shows how the different approaches are positioned on the dimension. Traditional lectures and standard web-courses are clearly on the teacher-centered side of the dimension. Study circles are more learner-oriented, but only when they are self-initiated and not part of the course requirements. "I-Help systems" help the students to form *ad hoc* study pairs or groups dynamically and benefit from the asynchronous or synchronous communication facilitated by the information technology. Problem- solving-learning is by default less learner-centered than problem-based learning [11], and if learning by research is seen as a superclass for PBL [3], it can be placed on the top end of the dimension.

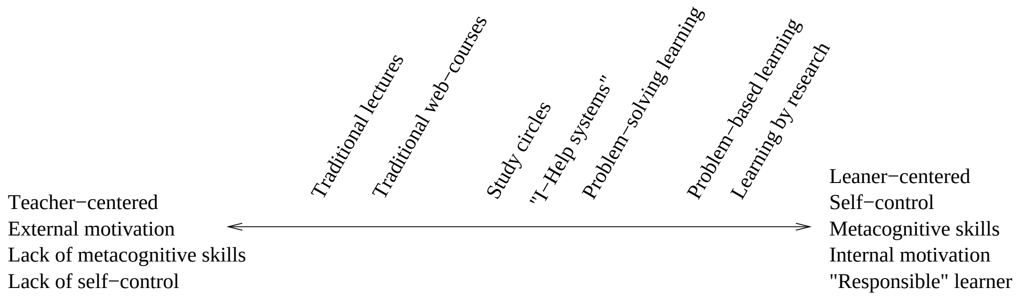

Figure 1: Various educational approaches for computer science positioned on the bi-polar dimension of learner-centeredness.

Evaluating the workload for the faculty. When considering the workload costs for the faculty for every approach discussed, we have to separate two types of costs. The first is the work needed to prepare the education (i.e. *initial costs*), and the second is the work needed to run the education (i.e. *running costs*). Typically, traditional lecture courses have some initial costs when preparing the course, but they have running costs as well when laboratory exercises are discussed in smaller groups. Traditional web-courses have high costs for preparing the course, but when the course is up and running, the need for human work is in most cases restricted to teacher tutors helping, guiding and assessing the students.

Study groups have very low initial and running costs, unless the study group is headed by a faculty member. In that case, the running costs are rather high since study groups normally consists of 6-8 students. When considering other peer-assisted learning approaches, the systems like I-Help offer very low running costs, but the initial costs to set up the system can be overwhelming, since there are not any systems readily available off-the-shelf with the support needed. Experimental systems could be used free of charge but they come with "no warranties".

Variants of problem-based learning are somewhat laborous to prepare, and guaranteed success of PBL demands qualified faculty to support the process during the whole time. Problem-solving learning demands less running support, but preparing suitable and valid problems can prove to be more laborous than initially appeared. Learning by research needs even more human and other resources than pure PBL; the tutors (or the facilitators, as they are sometimes called) should be trained properly to embrace the radically new type of education, and they need to devote time and effort to the whole process.

Table 1 summarizes the workload costs for different educational approaches.

Final remarks. From the Table 1 we can see that the approaches which are highly learner-centered (Learning by research, PBL, PSL) have all high or very high initial costs, and learning by research and PBL have also very high running costs. Indeed, transforming the curriculum to rely on problem-based learning requires training of the faculty, open attitude towards drastically different learning, and resources to be put into the process. It is no surprise that the most of the attempts to incorporate PBL or learning by research into computer science curriculum have used modified and lighter versions compared to the original approaches.

The most promising direction of research is the research on learning communities, where educational technology can be used to enable and amplify communication and collaboration. Systems like I-Help are a step to that direction. The learner becomes more active participant in the learning environment but the existing structures and attitudes need not be demolished. In addition, the the workload of the faculty does not grow.

It is clear that no evaluation can be comprehensive enough that it could serve as a directive when selecting educational approaches or developing computer science education at the highest level. However, the evaluation presented makes a point that under the given circumstances there is not necessarily a need to completely new paradigm in computer science, since computer science as a discipline is in a rare position of being truly capable of enhancing the learning with technology while reducing the resources needed. Discussion should continue.

References

[1] D. Boud (ed.), *Problem-based Learning in Education for the Professions* (Higher Education Research and Development Society of Australasia, Sydney, 1985).

[2] M. Daniels, A. Berglund and M. Petre, "Reflections on International Projects in Undergraduate CS Education," *Computer Science Education* **9**(3), 256–267 (1999).

[3] K. Hakkarainen, K. Lonka, L. Lipponen *Learning by research - the limits of intelligent activity and how to cross them (in Finnish)* (WSOY, 1999).

Table 1: Workload for different educational approaches to computer science education.

Approach	**Initial costs**	**Running costs**
Learning by research	very high when training the faculty; high when designing the process	very high since the groups are small
Problem-based learning	very high when training the faculty; high when designing the problems	very high since the groups are small
Problem-solving learning	high when designing the problems	low if there is no support during the problem solving
"I-Help systems"	very high if the system has to be developed; high if prototype system is used since there is still a need for lectures or other sources upon which the peer-lerning is built	very low when the students act without faculty help
Study circles	very low	very low if there is no need for faculty members in groups; very high if every group has a group leader from the faculty
Web-courses	very high if done properly	very low unless some group or individual activities are supported
Lectures and lab exercises	moderate if lectured for the first time; low after the first time	moderate

[4] K. Hakkarainen et al., *Information and Communication Technology as a Tool for Learning by Research (in Finnish)* (http://www.hyvan.helsinki.fi/tutkiva, 1999).

[5] I-Help Project description, http://www.cs.usask.ca/research/research_groups/aries/iHelp2.html, 2001.

[6] J. Kay et al., "Problem-Based Learning for Foundation Computer Science Courses," *Computer Science Education* **10**(2), 109–128 (2000).

[7] S-M. Kim et al., "Design and Implementation of Teaching Models in Web-Based Teacher Training," *Proceedings of ICCE/ICCAI 2000*, 591–594 (2000).

[8] L. Malmi, Learning by research in own teaching, (http://www.cs.hut.fi/~lma/memos/tutkiva_oppiminen_ja_OLO.html, 2001).

[9] G. McCalla, J. Vassileva, J. Gree, S. Bull, Active Learner Modelling, in Gautier, Frasson and VanLehn (eds.) Proceedings of ITS'2000, Springer LNCS 1839, 53-62, 2000.

[10] T. Reeves, "Evaluationg what really matters in computer-based education," (http://www.educationau.edu.au/archives/cp/reeves.htm, 1997)

[11] M. Savin-Baden, *Problem-based Learning in Higher Education: Untold Stories* (Open University Press, Buckingham, 2000).

[12] W. Yip, "Utilisation of the Problem-Based Learning approach facilitated by Information Technology to teach Information Systems Development," *Proceedings of ICCE/SchoolNet 2001*, 616–623 (2001).

Introductory course to Software Engineering: teaching by doing

J. Voracek
Department of Information Technology, Lappeenranta University of Technology
P.O. Box 20, FIN-53851, Lappeenranta, Finland

Jan.Voracek@lut.fi

1 Introduction

The increasing significance of a systematic approach to all stages of software development process escalates also the meaning of Software Engineering (SWE) education in general. Companies from information technology area are currently searching rather for project managers or software engineers than for pure programmers. From this point of view a proper and straightforward introduction of the whole subject is important and can significantly influence students' future relationship to SWE. Moreover, the content of such non-technical course is so different from the majority of those hitherto given in the technical university that the question of teaching strategy is really crucial.

The introduction to SWE in the Lappeenranta University of Technology is 3-credits course for second year and international students taught during the spring term. Its weekly teaching includes 3 hours of lecturing, 2 hours of exercises and a regular project work. Our teaching policy strives to develop student's rational thinking based on SWE principles as a first step of their future active membership in professional teams.

2 Course structure

For the successful passing of the whole course, students had to complete a project till the given deadline, actively participate in exam and collect at least the minimal amount of points from any combination from the whole set of particular tasks listed in Table 1.

From this summary it is evident, that the traditional leading role of exam is suppressed and replaced with an individual work, preferably according to student's interests. Also the total point profit from exam is top-limited and lower than the course minimum, so students were really forced to do also something more. This "more "could represent active participation in exercises, where either sample solutions could be given or home readings presented, answering weekly quizzes or simply obtaining enough project-related points. Figure 1 illustrates the outlined course structure and single non-standard activities or features are discussed in the following sections.

2.1 Exercises

Basically single students could give each exercise session by themselves. In such case the teaching assistant (TA) only focused and facilitated their discussions, suggested possible alternative solutions and classified the effort of participants. Each exercise class was composed of two sessions, the presentation of recommended articles or book chapters (readings) and the sample solution of a related set of examples from the textbook.

All references and problems to ponder were published at the beginning of the course and everybody could freely select up to two readings and sample solutions per course, including optional free topic. This eventuality means that everyone with a practical experience from any phase of SWE process related to the current class could pass this knowledge to the audience. On top of such voluntary activities project-related issues were regularly discussed within exercise sessions.

Table 1: Single parts of the course and related time estimation, including a case study of an average student.

Activity	Duration [Hours]	Amount [Times]	Maximal load [Hours]	Average student Effort[%]/Hours
Lecture	3	14	42	0.5 / 21
Exercise	2	13	26	0.5 / 13
Preparation (normal exercise)	1	13	13	0.5 / 6.5
Preparation (extra exercise)	5	4	20	0.5 / 10
Quiz	1	14	14	0.5 / 7
Regular project work	3	13	39	0.8 / 31.2
Finishing project	15	1	15	0.9 / 13.5
Preparation (exam)	15	1	15	1.1 / 16.5
Exam	3	1	3	1.0 / 3
TOTAL			**187**	**121.7**

2.2 Readings

Several reading assignments were specified for every week. One of them was always a current part of the textbook, remaining were supplementary materials. Students had to reserve single topics from web in advance and present orally during the particular class. The time for a single presentation was typically 15 minutes including a discussion. Every presentation had fixed structure consisting of overview, introduction, problem description, conclusions and references. A written summary on it was delivered to TA beforehand.

2.3 Sample solutions

Every participating student had to be prepared for any exercise. At the beginning of each session the TA sent a list of actual problems and everybody had to mark those, which is

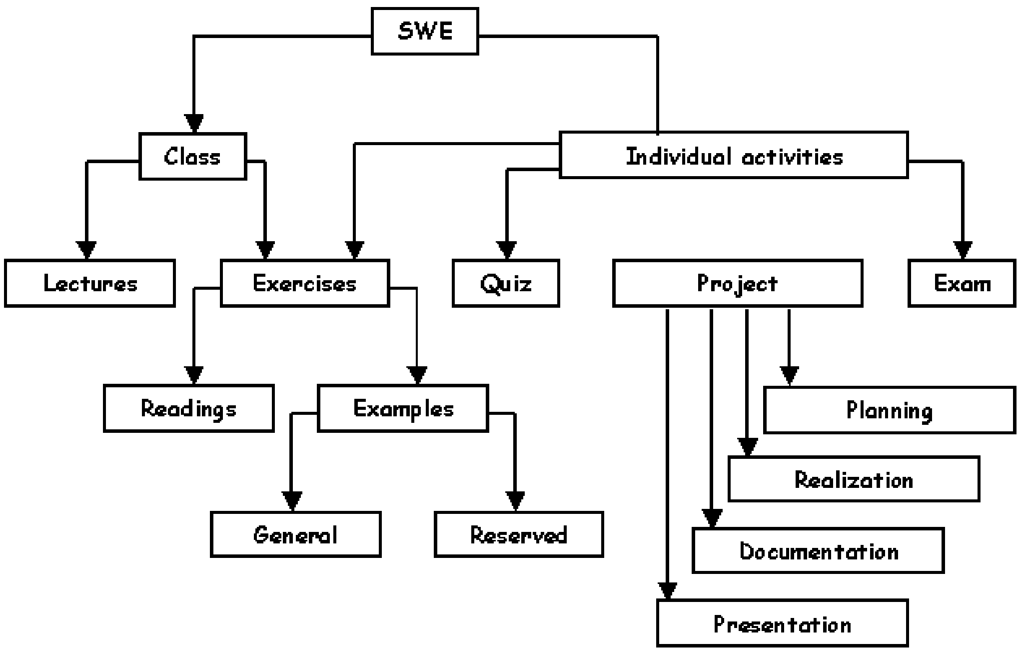

Figure 1: Structure of the presented Software Engineering course.

Table 2: Single parts of the course and related points profit including a case study of an average student.

Activity	Maximum points per unit	Amount [Times]	Maximal profit [Points]	Average student Effort[%]/Points
Quiz	10	14	140	0.5 / 70
Activity in exercise	10	13	130	0.5 / 65
Reading	50	2	100	0.5 / 50
Sample solution	50	2	100	0.5 / 50
Project deliverables	50	4	200	0.5 / 100
Extra project points	100	1	100	0 / 0
Exam result	400	1	400	0.8 / 320
TOTAL			**1170**	**655**

capable to present for the whole group. Moreover, each student had to have own solution on the paper along in class. Then TA selected randomly one person from the list and asked to demonstrate the solution. In case that everything was explained properly, the student obtained full amount of points for exercise; otherwise his/her total point gain for that class was zero. Correctness of each solution was preferably verified by the person who had booked the concrete example in advance. In such case a paper version of a sample solution had to be available in class and also sent to TA beforehand. Only then the extra points were acknowledged. If nobody was able to solve the particular problem, TA demonstrated the procedure. The main difference between the sample solution and a standard preparation for exercise was that the sample solution included detailed analysis and discussion of the problem and was connected with the creation of a structured report.

2.4 Project

From the technical point of view the project task was relatively simple. Its main idea was to create and regularly update a time accounting software in MS Excel. Students had to submit four deliverables (project outline, project plan, implementation plan and final project binder) till the given deadlines. The main emphasis was given to the proper understanding of the exact content and format of single documents as well as to the regular work with time accounting database.

3 Time management

Beyond the course content itself, we educated students to adhere the given timetable and optimize own workload. Converting 3 credits roughly into 120 hours, we demonstrated students' that a dummy person spending with the course an average time (121 hours) with an average effort (655 points) should also obtain average classification. There are seven basic points-producing activities listed in Table 2 altogether with recommended duration. At the beginning of the course it was explicitly mentioned that teachers are aware of such timetable and their demands are in accordance with this. That's why all the time spent on top of the upper estimate, which is 187 hours, is additional, essentially own time of those who are really interested in the subject.

The final classification was based on Table 3, which included two scales for separate groups. Group 1 denoted the students passing the presented structure of SWE course, group 2 was designed for students who had not been involved in the proposed teaching system and completed exercises or submitted projects earlier. Also these students must have a possibility to complete the course, but, naturally, based on different conditions.

Table 3: Grading scale for internal (group 1) and external (group 2) SWE students.

Grade	Group 1	Group 2
5	>800	>1100
4	746-800	1001-1100
3	651-745	901-1000
2	551-650	801-900
1	451-550	650-800
0	<455	<650

4 Results and discussion

At the end of the course, there were two main information resources available. We had the final project data, expressing students' total workload and also their exam results. The both parts are summarized in Table 4.

Table 4: Results for internal students (group 1).

Activity	Students
Registered for the course	89
Passed the project	73
Passed the course with the first exam	64
Passed the course within one school year	73
Study average for the whole study group and school year	4.2
Real average workload for the whole study group [hours]	127

From the resultant figures it is evident, that:

- More than 80% of enrolled students passed the course.
- Everybody who had completed the project also passed the course.
- More than 70% of enrolled students passed the course with the first exam.
- Average final grade for group 1 is high.
- Real average workload is almost in accordance with our original presumption although the classification is better than the predicted average.

5 Conclusions

The proposed active approach to SWE education was found as useful and motivating. Also international students appreciated and employed the obtained SWE knowledge after starting own professional careers. All good practices were reused and improved during the school year 2001/02 with comparable results. Virtual teaching techniques used in the both courses guaranteed a proper portion of interactivity and efficient bilateral communication. Instant feedback allowed students to monitor and manage their points' profit continuously and, consequently, optimize their total workload and final successfulness.

SESSION 5

Virtual education: experience and visions

J. Voracek
Department of Information Technology, Lappeenranta University of Technology
P.O. Box 20, FIN-53851, Lappeenranta, Finland

Jan.Voracek@lut.fi

1 Introduction

There are several disadvantages connected with the traditional way of teaching. Primarily it is a weak contact with students during the lectures and also absence of a valuable feedback from classes. But also other aspects, like a lack of interactivity during the administrative part of course such as problematic reservation of project topics or missing informal communication platform were clear impulses for the searching for "something" improving the current status.

As a logical consequence of the virtualization process, also the formal structure of university courses must be innovated. The traditional in-class model, including regular lectures and exercises does not properly reflect the actual level of technology and more or less isolates students and teachers. Therefore the new teaching practices and techniques solving the above mentioned disadvantages must be included properly into an updated framework.

2 Tools and techniques

The following specific tools were developed for teaching of Computer Science courses in the Lappeenranta University of Technology and used regularly during school years 1998/99 and 1999/00 in several courses with about 300 students altogether. From the school year 2000/01 onwards they were completely replaced with the commercial product WebCT and its functionality was verified in 3 courses with about 400 students.

All tools were developed and applied or adopted in order to attract students, provide them an alternative way for passing the mandatory exercises and obtain a feedback on the lectured topics.

2.1 Realized activities

Considering all the mentioned weaknesses I decided to concentrate on the following areas:

- Regular **quizzes** were initially identified as activity with the widest scope. If properly designed, it is easy to evaluate them and well-asked questions could cover a large area of the presented topic. Therefore they can conditionally replace traditional exercises and partially even lectures, give a realistic picture about knowledge of students and express their overall involvement in the course. See Figures 1 and 2 for sample screen shots.

- Another automatized task was the **on-line reservation system**. Students could electronically choose or propose an atomic activity required for the course like readings, sample solutions or project topics and maintain this reservation in accordance with the course policy. Figure 3 demonstrates the combined reservation window used in the Software Engineering course in school year 1998/99.

- **Internal discussion forum** (Figure 4) developed as a simple web page with a public access facilitated well our internal communication and was used regularly in several courses with time-increasing popularity. It preserves anonymity, minimizes standard emails, archives all discussed items and opens new areas of interest.

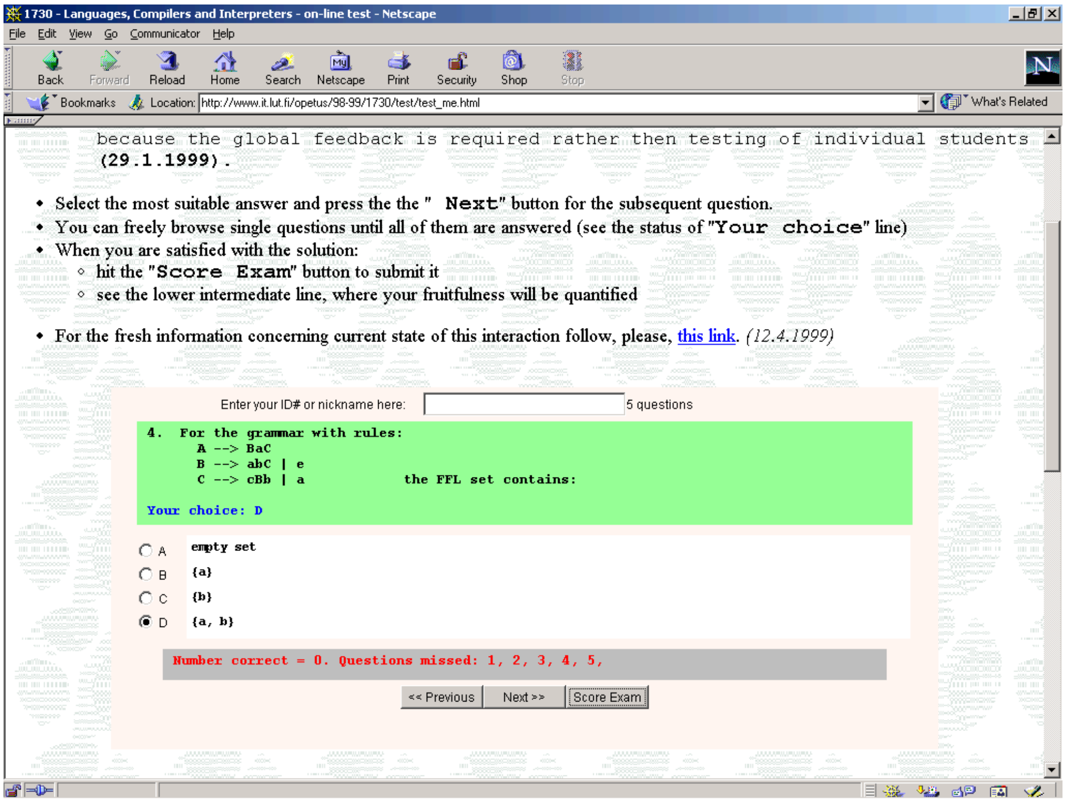

Figure 1: JAVA user interface for the first generation of own quizzes (1998).

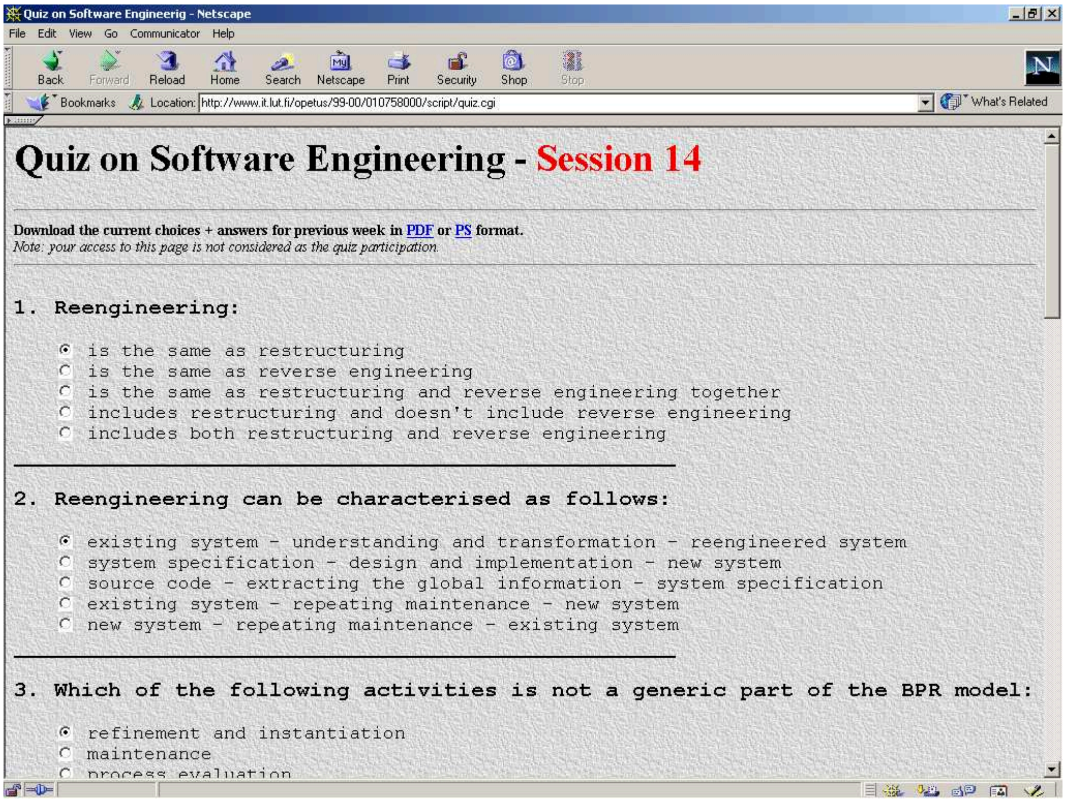

Figure 2: HTML form as user interface for the second generation of own quizzes (1999).

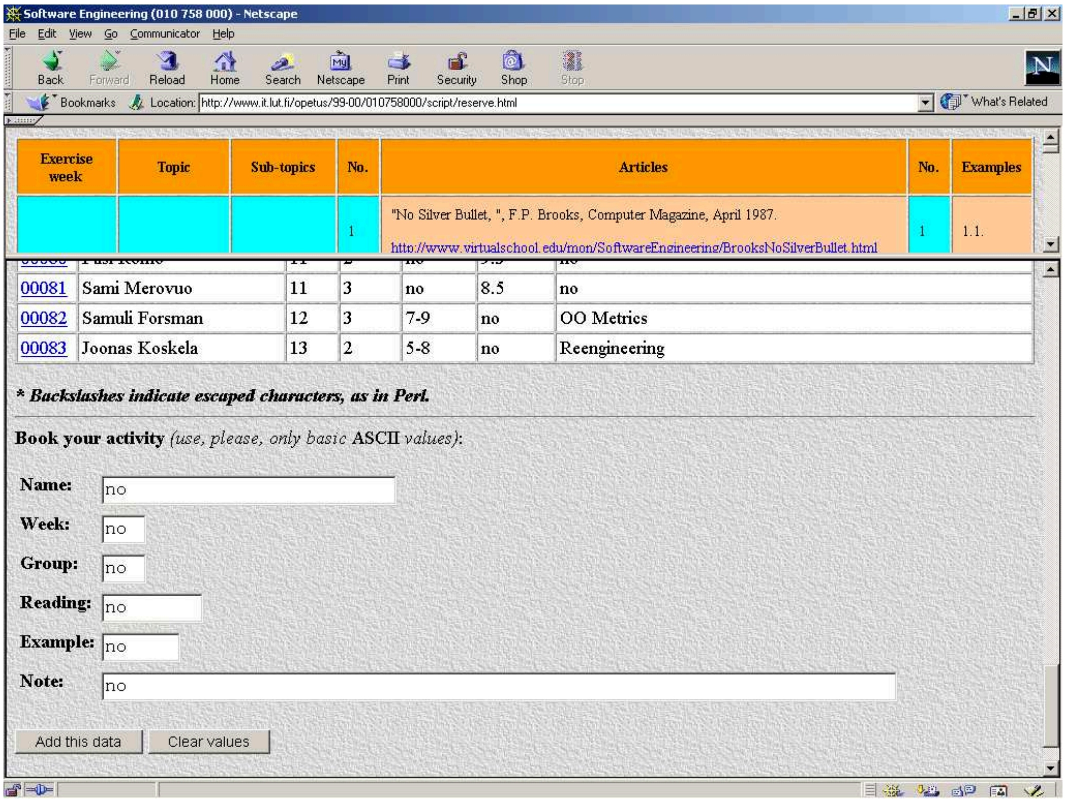

Figure 3: Sample screen of reservation system. Topics for booking are in the upper part of the window, interactive part at the bottom.

- Simple prototype of a **system for automated evaluation of submitted exercises** included a pattern matching mechanism based on regular expressions and allowed exact or approximate classification both for lexical and numerical answers. For programming tasks also publicly available generators of random test cases were included and used by students during the task development as well as by computer classifying their later submissions.

  A simple plagiarism preventing system, evaluating selected structural and lexical elements of sent programs was included to minimize evident duplicities. Although such automatic checkers are useful and time-saving mechanisms, their proper utilization and wider distribution are connected with many essential, say, theoretical problems.

- On top of the directly teaching-oriented tools, a set of **auxiliary service programs** realizing the passwords, users and security management as well as data conversion and post-processing was realized and used as the natural consequence of a regular interaction with tens of students.

2.2 Platforms

The first version of quizzing system was realized in JAVA, which provided a non-uniform interface and some advantageous performance features. On the other hand JAVA interpreter permanently running on the the server originated some technical difficulties. Because of this limitation, PERL was used as the main language for the second generation of own tools.

2.3 Experience with teaching tools

The overall experience with the set of homemade scripts is positive. Immediately after the deployment students were appreciating this teaching innovation, accommodated all the related user interfaces smoothly, participated in quizzes from very beginning although this activity was not mandatory and fully accepted automated evaluation of submitted tasks.

There were no technical problems and the main disadvantage of presented programs was the mutual isolation of single services and data incompatibility. Also the maintenance and solution of possible technical problems were completely up to the author.

Starting from the school year 2000/01 a short study concerning SW market at that time was done and considering also non-technical aspects of the searched product like overall support or distribution in Finland, finally WebCT was chosen as an optimal alternative. From the technical point of view WebCT provides a solid central data structure and acceptable level of security. On the other side, its problematic user friendliness, low processing speed and some missing useful functions make the routine utilization still difficult.

3 Future techniques for virtual education

University courses are currently given more or less according to the standard "in-class" model, which includes regular lectures and exercises. Although the impact of information technologies is huge nowadays, the main effect of this process to the educational sector in comparison with the situation ten years ago is, that teachers are currently more publishing their materials on web and using on-line references, textbooks or tutorials instead of the paper resources. However, also in this approach teachers and students are isolated and mutual information flow is unsatisfactory.

Current level of communication, rapidly growing information society and consequently changing life style are appealing to incorporate all new aspects also to the educational process.

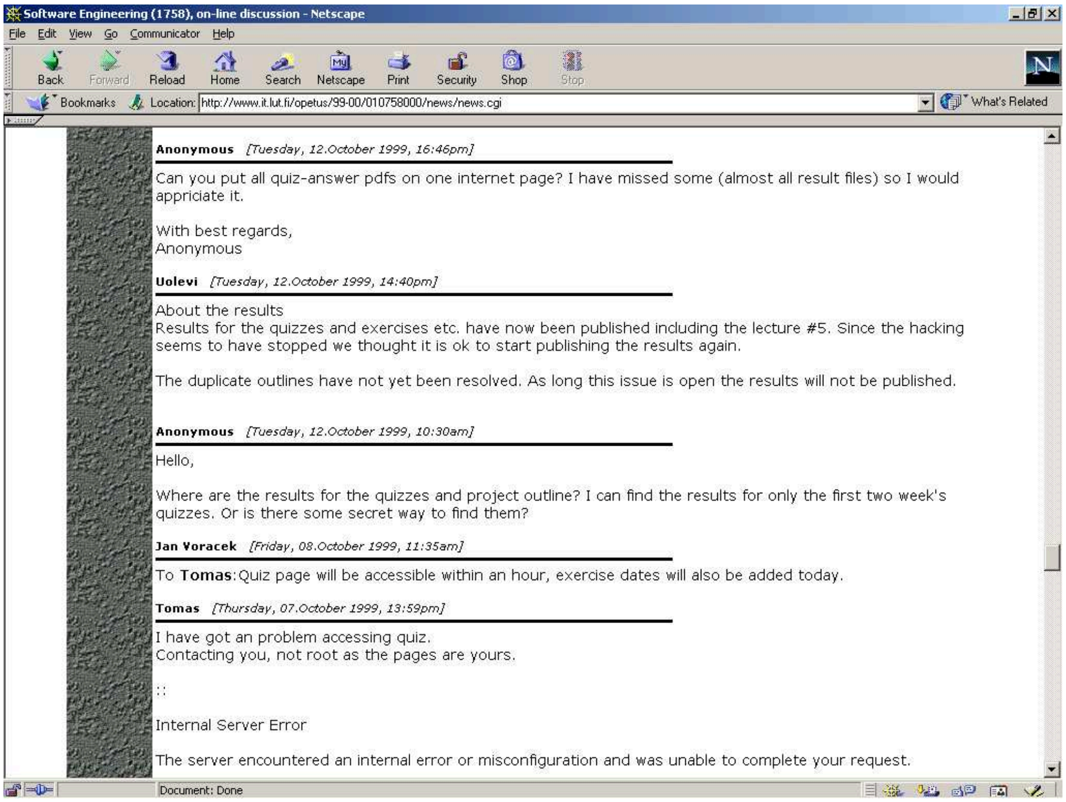

Figure 4: Example of the format how our discussions were realized.

It is evident that standard realization of university teaching process must be updated into a qualitatively higher platform. Based on the personal experience, the following parts should be included there:

- **Off-line videolectures**: the lecturer records the course or its selected parts on video. The complete program should consist of a shorter, say, 15 minutes long blocks between which both the surrounding environment and presentation techniques are changing. Possible formal candidates for the video-presentation are paper handwriting, blackboard-based explanation, slides given through the overhead projector, streaming electronic slides or parts of the textbook taken directly by the camera.

- **Regular common sessions with the lecturer**: instead of a traditional lecture, where students are introduced to the new facts and cannot typically ask detailed questions at the same time, a separate session with the lecturer is arranged after a certain virtual teaching period. In such special class those students who are interested in the topic could ask more qualified questions and refer directly to a possible unclear parts of videolecture.

- **Optional in-class and virtual exercises**: students could decide whether to participate in the real exercise session and communicate with the teaching assistant or realize the given tasks individually and submit solutions in electronic form for automated assessment.

- **Electronically submitted projects** with subsequent discussion or a common experience summarizing seminars: students could have a possibility to present results of own project work to the rest of the group, if they wish. If there are more volunteers, a separate seminar session can be established for this purpose. In case they just want to pass the project with a minimal personal involvement, fully electronic way of submission and evaluation is possible. Anyway, people using such way of delivery must be aware of all the related risks.

4 Conclusion

The routine, systematic utilization of any kind of virtual features as well as the structural changes of educational process will finally increase the quality of teaching, minimize workload of teachers during the course preparation stage and enable to share the virtual courses between universities.

Also students will profit from this arrangement with a wider study freedom and better course materials. Development of a global service unit coordinating all the related activities as well as a suitable motivation for teachers to take part in the proposed process are two essential prerequisites for the final success.

The Effective Virtual Strategy - An Example

K. Saastamoinen
Laboratory of Applied Mathematics
Lappeenranta University of Technology
P.O. Box 20, FIN-53851 Lappeenranta. Finland

kalle.saastamoinen@lut.fi

1 Introduction

The mathematical knowledge and ability of the new students at a universities - in Finland, but probably also in any other country - is not so firm as one should be able to expect. Some revision of the school level mathematics is needed before the actual university studies can start. Here, computer-based study materials can give one solution. It has been claimed [1] that development of software and network technology would give enormous possibilities to teaching and learning. However in practice, it does not seem so clear how these possibilities should be used or even what these so called new possibilities actually are.

First there were some questions to be taken into account before starting the project, like: 'What is the use of distributing material into the internet?' Some answer for this could be for example [2]:

- It is easy to reuse, transform and combine with other materials.
- It is fast at least with fast connections.
- It is accessible 24 hours a day.
- It is reachable for the great amount of people.

One more difficult question is this: 'What will be the usage of this virtual pages and are they usable?' The good usage means that pages are for users effective and nice to use. Pages are usable if you will find exactly what you have came to look for and you also will find out straight that if this pages does not contain the information that you are looking for. You can check out more information about usability by looking from pages [3]. Anyway in my opinion these pages fulfils these both criteria.

In this paper I will represent a material [4] that was done for the Lappeenranta University of Technology or shortly LUT. I will also tell about philosophy, planning principles, realization and future of this experiment. The core of the experiment is a study packages based on hypertext, web browsers, computer algebra systems and visualizations based on java applets. It is my humble believe that this kind of projects can be carried out in the similar way also in other areas than just in the field of mathematics. Except at universities, the material delivered and categorized here can also be used as additional reference at mathematically oriented schools. Situations I told above has been the starting points of this material developing project at the Lappeenranta University of Technology.

2 Starting of the project

In the beginning of the year 2001 Lappeenranta University of Technology virtual university part decided to support project called 'Johdatus korkeakoulumatematiikkaan' freely translated 'Introduction to the university level mathematics' supporting money was as high as 10 000 Finnish marks which is about 1667 euros. Which actually is not much. Money or more or less lack of it was at first one of the main reasons why it seemed so hard at first to carry out this project in a successful way. This was of course attached with imperfect skills of programming and also lack of viewpoint what actually is the usage of the final material. So there were many reasons which were saying that this interactive project could not be fruitful and perhaps not worth of doing at all. But since the object was kind of interesting one for a mathematician who is a bit worried about level of mathematics in Finland nowadays the temptation was just too much. So the project was then finally accepted to be done.

This project was at first intended only to build material which would help mathematically not very skillful students to get a grip about mathematics studies and for which they could use by themselves without any supervision. But since money and time was short we did not wanted to do our material from the beginning cause this would have for one thing been expensive, secondly time consuming and finally there would have been no guarantee that we would at the end have in our hands any better material than what already has been done and which is available for use also.

So I started to look over what kinds of materials have been done in other universities. Soon I was in a situation that I had more than enough good materials what could be used for previously mentioned needs. Because material was plenty it would have been a nonsense to give a material 'just' for our unskillful students so I decided to do a pages which would help students of different levels.

3 Goals of the project

The objectives of the project are:

- To collect computer-based mathematical study materials of various kinds.
- To offer computer-based mathematical study materials of various kinds in the suitable way.
- To study what kind of learning processes should be supported.
- To study how the material should be offered.

For one thing at the end we are at least able to offer a study package for revising and slightly enlarging the school level mathematics. At the moment, the main components are already completed. A test of the one package is planned for Mars 2002, when some students of the Lappeenranta University of Technology will use the package of differential equations during the second period of the second basic course of mathematics. The use at upper secondary schools and in distance education systems has also been planned.

4 Basic philosophy of the project

The student will be given an open study environment: This consists of information about mathematics, problems to be solved, computational tools, guidance, general help (both technical and mathematical) and visualizations. The student may study what they feels the most interesting. Students hopefully begin to do their own research and to create a view on mathematics. Thus, the view on the learning process is constructivist. In principle, the criterion of the learning is the ability to solve the given problems. On the other hand, the students are not graded. The purpose is only to give them a study environment that will be interesting enough to make them study mathematics at their own pace.

5 Planning principles of the project

The planning principles for the usage are set as follows:

- To create mostly a hypertext based mathematical forum with visualizations, animations etc.
- In the chosen material the mathematical formulae and expressions must be written as beautifully as possible.
- System has to work fluently.

- Students should get feedback and give some too.
- Material has to be well organized.
- Material has to be relevant for students.
- The distribution of the most important parts of the material should be free.
- Use material what others have done as much as possible.
- Ask permissions for usage of material in all cases.

If the material should be free, the project must have the funding from public sources.

6 Some aspects into the procedure

Supervision is one of the key aspects when we want to produce good material into the web cheap and efficient way. Firstly the supervisor has to decide the purpose in for which he or she wants the new material to be used and think a bit also about how is this material to be organized. Secondly it has to be decided what does this new pages include e.g. applets, interactive games, theory etc. After this it is suitable to do a fundamental search from the WWW and look out materials which would be best for previously decided purposes.

Layout of this new pages is of course also an important question. In my opinion it is recommendable to use some kind of outside help in design of this new pages cause for learning purposes it is important what does this new pages look like. If we think for example usage of this pages. For outside help we chose to use a student who was hired to work during the summertime in to the our laboratory. The work what he was given in this project was simply to design layout of this new pages. I think it is good to use students cause they are cheap for one thing and it is good for their studies for second and what is good for the project is that you will get a fresh view for this layout. Students of course needs a supervision cause for example course material has to be linked in an appropriate manner and visualization has to be done in a meaningful way. Finally it is important:

- To check that everything onto new interactive pages is ordered in a pedagogically meaningful sense.
- All material is proper in a sense that everyone who has done materials used has been informed and licenses to their material is asked.
- It has been checked carefully that everything is working as wanted.
- One should also ask other peoples (co-workers, students) opinions and improvements proposals concerning new pages.
- Possibly do some changes and ask again.

7 Final work

All the pages have a feedback form, which is met for students that they could be able to send us any information what they want about pages. Final work consists of the following main parts:

7.1 Main page

- Finnish main pages [4].
- English main pages [5].

This both pages have many links, main mathematical features are:

7.1.1 Introduction to the university level mathematics

By clicking this link [6] student gets links to:

1. Basics of mathematics [7].
2. Basics of high school level mathematics [8].
3. Great pages done by Franz Embacher [9].

All of these pages includes a lot of theory, exercises, visualizations and even quizzes and games about mathematics for students.

7.1.2 Ordinary differential equations

Which consists of material [10] done by Simo Kivelä and his work group and it includes theory, visualizations, exercises and examples about ordinary differential equations.

7.1.3 Virtual mathematics

By clicking this link [11] student gets links to:

1. Examples [12] which includes a lot of useful examples about java-coding and mathematics.
2. Sorted applets [13] which will help to visualize mathematics.
3. Ready Materials [14] which includes all kinds useful mathematical materials for example previously mentioned pages done by Franz Embacher [9].

It should be mentioned that this is just a main part of materials what we have putted into the our pages.

8 Future

The development of the packages will continue. The following new features are planned:

- Start active advertising of this pages.
- Make categorizing better.
- To create mostly a hypertext based mathematical forum with visualizations, animations etc. There must be more animations and visualizations. They may be in different forms: for example series of gif pictures, Java based etc. It is rather easy to animate many mathematical phenomena, but to find animations really useful for learning is not so simple.
- When the package is used through the web, the students could send their answers of the problems to the server. Here, the answers are checked, some feedback is given and a statistics is collected. Thus, it would be possible to give the student a survey of the work done during several sessions (which of the problems are solved, how many correct and wrong answer are given, etc). It must be emphasized that the idea is not to grade the student, but to give some feedback for self-evaluation. On the other hand, the data collected in this way can be used for developing the system further.
- The aim is to develop a system for collecting the data about the students works and analyze the data.
- Constant maintenance to this pages.
- Link and use materials into the specific courses.

9 Acknowledgements

Acknowledgements to the Lappeenranta University of Technology virtual university that have funded this project.

References

[1] S. K. Kivelä, "An Example of Computer-Based Study Material in Mathematics: Planning Principles and Realization," *IMS '99 THIRD INTERNATIONAL MATHEMATICA SYMPOSIUM 23 - 25 August, 1999* .

[2] A. Nevgi, *Opettajan ja ohjaaajana verkkopedagogikkaa käytännössä* (Presentation TieVieseminar at Dipoli, Espoo 3.10.2001).

[3] Käytettävyyslaboratorio, "http://usability.hut.fi/laboratory/," (Helsinki University of Technology, Espoo)

[4] Collected by Kalle Saastamoinen, "Home page of the LUT laboratory of applied mathematics: in Finnish http://www.it.lut.fi/fac/mat/newfin/index.htm," (Lappeenranta University of Technology, Lappeenranta, 2001).

[5] Collected by Kalle Saastamoinen, "Home page of the LUT laboratory of applied mathematics: in English http://www.it.lut.fi/fac/mat/new/index.htm," (Lappeenranta University of Technology, Lappeenranta, 2001).

[6] Collected by Kalle Saastamoinen, "http://www.it.lut.fi/fac/mat/newfin/Materiaalit.html," (Lappeenranta University of Technology, Lappeenranta, 2001).

[7] Collected by Kalle Saastamoinen from Internetix databases, "http://www.it.lut.fi/fac/mat/newfin/Linkit/internetix.html," (Internetix Campus, Mikkeli).

[8] S. K. Kivelä, "http://www.it.lut.fi/fac/mat/newfin/isom.html,"(Helsinki University of Technology, Espoo).

[9] F. Embacher, "http://www.univie.ac.at/future.media/moe/," (Institute for Theoretical Physics at the University of Vienna).

[10] S. K. Kivelä, "http://www.it.lut.fi/fac/mat/newfin/delta.html,"(Helsinki University of Technology, Espoo).

[11] Collected by Kalle Saastamoinen, "http://www.it.lut.fi/fac/mat/newfin/applets.htm," (Lappeenranta University of Technology, Lappeenranta, 2001).

[12] Collected by Kalle Saastamoinen, "http://www.it.lut.fi/fac/mat/newfin/examples.htm," (Lappeenranta University of Technology, Lappeenranta, 2001).

[13] Collected by Kalle Saastamoinen, "http://www.it.lut.fi/fac/mat/newfin/appletsjava.htm," (Lappeenranta University of Technology, Lappeenranta, 2001).

[14] Collected by Kalle Saastamoinen, "http://www.it.lut.fi/fac/mat/newfin/linksjava.htm," (Lappeenranta University of Technology, Lappeenranta, 2001).

Contextual Computing Studies in Tanzania

E. Sutinen, M. Vesisenaho, and M. Virnes
Department of Computer Science
University of Joensuu, Finland
P.O. Box 111
FIN-80101, Joensuu, Finland

erkki.sutinen@cs.joensuu.fi
mikko.vesisenaho@joensuu.fi
marjo.virnes@cs.joensuu.fi

1 Introduction

There is a connection between spread of information networks and national levels of income [1]. In general, countries with a higher level of income have more information networks than countries with a lower level of income. Information networks and technology can give various possibilities to narrow the gap between developed and developing countries and create e.g. cultural and educational connections between countries.

Special knowledge and skills in the field of information technology are factors that are increasing the gap between developed and developing countries [8]. It can be asked whether culturally sensitive digital material, e.g. in the field of Computer Science (CS), would be a cost-effective benefit for developing countries. The challenge and goal for our project are to narrow that gap by creating an educational program that includes virtual studies of information technology for a developing country from its own cultural perspective and needs.

2 Contextual Computing - Teaching Technology in Tanzania

The program of Contextual Computing began at the University of Joensuu in summer 2000. At that time the studies of Computer Science in Tanzania began from the Tanzanian context. After two months of introduction courses, studies of two Tanzanian students continued at the University of Joensuu in Finland. The program is still continuing at the University of Joensuu, and the aim is for two Tanzanian students to take their master's degrees during this time.

As can be seen in Figure 1 below, there are three stages in the program. First the students begin with orientation in Tanzania and after that continue their studies for 2 years in Finland. In the third stage students go back to Tanzania and begin to teach IT courses at the local university. The students' educational background in CS will in the future be based on an IT-Diploma. After the IT Diploma level, students will continue more demanding studies of CS in Finland and receive a bachelor's degree. Consequently, we can develop a higher beginning level of knowledge of CS in Tanzania and the student can reach more demanding goals in education.

2.1 Beginning in Tanzania

Studies began with a six-week long contact course in Tanzania in July-August 2000. The aim was to learn basic skills in CS and educational technology. The teacher was an IT-teacher from Finland and the students were two business-bachelors from Tumaini University, Tanzania. The course program in Tanzania included the following courses:

- Basic skills (1.5 weeks)
- Multimedia project related to local culture (2 weeks)
- Internet and information (1 week) and introduction to programming (1 week)
- Literature and learning diary for supporting the learning process

Studying of basic skills included introduction to computers, software (MS Office), operating system (Windows), network (Intranet) and installing of programs. By using e.g. Lehrer's Learning by Design -method [7] and Koschmann's Problem Based Learning- method [5] in the contextual learning project the multimedia project introduced different learning methods and different ways to use computers in education. The third part concentrated on teaching basic knowledge about the Internet and its structure and limitations. It also gave introductions about HTML for web-page design and installation to the server. Programming began by studying MSW Logo programming language and continued with studies of Pascal.

2.2 The Goals of Contextual Computing -program

After introductory courses, the same students have continued their studies in Finland aiming at a master's degree. They are key members in the project that is producing a virtual learning environment of CS for Tanzania. The goal for the whole Contextual Computing program is to study the prerequisites and opportunities for a new IT-program that does not yet exist but which is needed - an program from the Tanzanian point of view. The aim is to create professionals of information technology for developing counties, at the moment Tanzania. Tanzanian students do not leave behind the knowledge that they get during the program to Finland but the knowledge goes with them to Tanzania and serves there as educational "capital", being a long-term asset for their country's own development. However, this program should not be considered to be only one-way novel form of aiding a developing country. Instead, our program should be seen as a starting point for a bi-directional relationship in developing culturally enlightened education, especially education in technology as a scientific and cultural dialogue between developing and developed countries [9].

3 Virtual IT Education

The foundation for creating Virtual IT-education courses for Tanzania is in the Virtual Certificate of Computer Science that has been created for high school students by the Department of Computer Science at the University of Joensuu. The aim of the Finnish Virtual certificate program is to create tools and materials for on-line learning and to give high school students a possibility to study the first year studies of computer science during their time in high school. Virtual Certificate studies take place mostly via information networks, but some contact teaching is also arranged [4].

Currently, we plan that IT education for Tanzania will consist of two parts, the Virtual Certificate and the IT Diploma. Virtual IT-studies for Tanzania is based on the Finnish Virtual Certificate. Studies are similar in terms of the contents of courses. Differences come along when the target group and its needs are noticed - the Tanzanian perspective receives an important role when the Virtual Certificate is made for Tanzanians. As a teaching method,

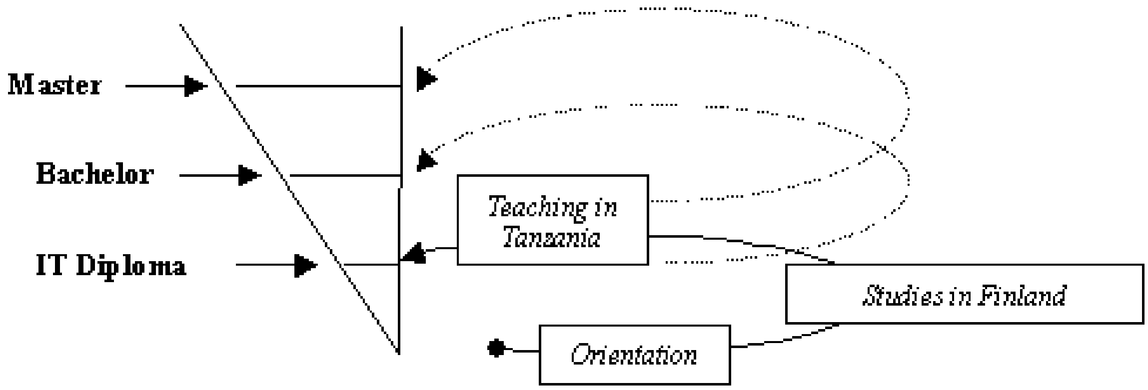

Figure 1: Program of Contextual Computing.

distance learning that is based on digital learning material (Virtual Certificate) will be used via network, tutoring, lectures given by visiting lecturers and their own teaching with their own traditions at the local Tanzanian university.

3.1 Virtual Certificate for Tanzania

The project of processing the Virtual learning material for Tanzania began in May 2001. At the moment the project is in the phase of producing on-line material, and the first prototype version of the virtual learning environment will be ready by summer 2002. In credit units the overall number is 15 cu credit (1 credit unit = 40 hours), and they consist of the following courses:

- Introduction to Computer Science 2 cu
- Introduction to Algorithmics 2 cu
- Programming I (Procedural) 2 cu
- Programming I (Object-oriented) 2 cu
- Computer Architecture and Operating Systems 2 cu
- An Overview of the Research Fields in CS 2 cu
- Ethics of Computer Science 1 cu

3.1.1 Courses

Courses are based on virtual learning material, except for the programming project that will be done after the programming courses and the Introduction to Computer Science, which will be passed in a book exam at the beginning of the CS studies. The aim of the material is to emphasize the Tanzanian point of view as much as possible. One of the student's ideas was to exemplify the abstract data type, stack, with a Tanzanian food skewer. The function of the food skewer is similar to that of a stack. The elements, which are onions, green pepper and mushroom, can be added to the stick or taken off like in a stack structure - last-in-first-out. Another cultural example can be recursion that can be described as the architecture of a village [2].

3.1.2 Ethics

Teaching computer science ethics is an important issue, because it is part of all that is done using computers and using networks. Consequently, we want to include the teaching of ethics in computer science as an integral part of the Virtual Certificate and IT-Diploma [6]. One part in the teaching of ethics will be based on case studies and discussions, e.g. on a "chat-channel". At the moment the topics deal with hacking, software piracy and copyright laws that are important to know when creating learning material now and in future.

3.1.3 Learning Environment

The Virtual Certificate consists of a learning environment that is divided into parts according to courses. As a tool for creating the learning environment, HTML was used. The basic structure of the courses is a text as the main source of information, plus examples and assignments that will both emphasize the Tanzanian point of view as much as possible. At this point of the process, the importance of Tanzanian students increases markebly. The learning environment will also include a "chat-channel" for students, a list of the essential concepts of each course, help for problem situations and a databank that includes book and link references. As the learning environment is built on web pages, it does not set complex requirements for used hardware. Not even a network connection is needed all the time if the material is loaded onto computers, for example, outside the actual studying hours.

3.1.4 Special Characteristics

The special characteristics of the digital material that supports the learning process will be Flash and gif animations, visualizations produced by the Jeliot environment and Woven stories software [3]. Jeliot, an environment for visualization of algorithms, will be used in programming courses and in the Introduction to Algorithmic course to produce example visualizations for the learning process and help students to understand the algorithms. While using Jeliot, students can both run their own java-programs and see how an algorithm works, but they can also run already existing examples from the material. The animations and visualizations will be part of each course and the mission is to exemplify and create a concrete grasp of theory. Woven stories software is based on learning by writing. In general, after studying a certain subject, the student writes a story in shared story space. Finally, there is a network of learners' stories in one space. In the planning of the material we also want to pay attention to different ways of learning and give different learning material to different learners. For example, to Bachelors of Economics there may be a problem to understand or apply to new concepts of abstract data structures on an abstract level. For these situations we want to present the abstract concept also on a concrete level like in the stack example or using animations.

3.1.5 Studying the Learning Material

Most of the learning material will be learned independent by the students. In problem situations students can discuss with each other on the "chat-channel" and share their knowledge. Another possibility is to contact a tutor, who will guide them in learning process. For each chapter or subject studied there are assignments that should be done and returned via the Internet to a teacher/tutor who will check the answers and give feedback according to the results.

3.2 IT Diploma

According to preliminary plans, the IT Diploma will consist of the courses in the Virtual Certificate with the additional courses listed below. In addition, there will be courses offered by the local university. Together these are 36 credit (1 credit unit = 40 hours) units and comprise one wider entity called the IT-Diploma, which will be realized in Tanzania.

- Computer Science Mathematics 3 cu
- Introduction to Office Applications 3 cu
- Basics of Visual Basic 2 cu
- Data Communication 3 cu
- Operating Systems 3 cu
- Computer Science English 3 cu
- Computer Science Ethics 1 cu
- Computer for Development (problem-based) 3 cu

4 Future Plans

The first version of the learning material will be ready by June-July 2003. This material will be then evaluated at local University, and necessary changes will be made before the material is given out for teaching. According to this schedule, studies using the digital learning material can begin during autumn 2003. In the very beginning it will be possible to load material into the computers of the local university to ensure its functionality and avoid possible breaks in network connections.

After or at the same time with the Virtual Certificate studies, there will be a possibility to study courses of the IT Diploma for supporting Certificate studies. The plan for the future is to give students a possibility to study for a degree with these studies. IT-diploma studies are not enough to graduate from university; but they will serve as a minor, computer-familiarizing subject, helping students from various backgrounds to apply information technology in their fields. The students showing academical excellence may also be offered a chance to come to study computer science in Finland for short period of time, thus deepening their knowledge and skills in the field. After studying in Finland, they will return to Tanzania to apply their new skills for the good of their nation.

The final goal is to support the development of the country with co-operation and a cultural approach. This means that the whole project would be realized in Tanzania, at a Tanzanian university by Tanzanians themselves. We also hope to give Tanzanian high school students the chance to start studying for the Tanzanian Virtual Certificate while attending high school, the same as their counterparts in Finland are doing.

References

[1] M. Castells, *Information Age 1: Economy, Society and Culture: The Rise of The Network Society* (Blackwell Publishers, Great Britain, 1996)

[2] R. Eglash, *African Fractals* (Rutgers University Press, New Brunswick, 1999).

[3] P. Gerdt, P. Kommers, C. Looi and E. Sutinen, "Woven Stories as a Cognitive Tool," *In Proceedings of The Fourth International Conference on Cognitive Technology, CT2001* (August 2001) University of Warwick, 2001.

[4] A. Haataja, S. Kontkanen, J. Suhonen and E. Sutinen,"High School Students Learning Computer Science over the Web," *In Interactive Multimedia Electronic Journal of Computer-Enhanced Learning* (2/2001).

[5] T. D. Koschmann, A. C. Myers, P. J. Feltowitch and H. S. Barrows, "Using Technology to Assist in Realizing Effective Learning and Innstruction: A Principled Approach to the Use of computers in collaborative Learning," *The Journal of The Learning Sciences* **3(3)**, 227-264, (1994).

[6] E. Kähkönen and E. Sutinen, " Context and Ethics as Factors in the Design of Educational Technology," *The Second Conference of Open Learning and Distance Education, Teheran, Islamic Republic of Iran* (2001).

[7] R. Lehrer. *Authors of Knowledge: Patterns of Hypermedia Design In S. Lajorie & S. Derry (Eds.), Computers as cognitive tools* (Hillsdale: Lawrence Erlbaum Associates, 1993)

[8] V. Naidoo and C. Schutte, "The Development of Virtual Education: A Global perspective, Virtual Institutions on the African Continent" (1999), Internet WWW-site, URL: http://www.col.org/virtualed/index.htm.

[9] E. Sutinen and M. Vesisenaho, "Teaching Technology in the context of the third world - case study Tanzania," (in Finnish) ITK Conference, Hämeenlinna, Finland (2001).

Author index