Kolin Kolistelut — Koli Calling

Proceedings of the
Third Finnish/Baltic Sea Conference on
Computer Science Education

Oct. 3-5, 2003 in Koli, Finland

University of Helsinki
Department of Computer Science

Series of Publications B
Report B-2003-2
Helsinki 2003
Helsinki University Printing House
## Contents

What is needed to be considered in addition to learning objects in computer science education
B. Cavas (Izmir, Turkey), G. La Russa, S. H. Pitkänen, P. Silander, J. Suhonen (Joensuu, Finland), P. Silvasti (Helsinki, Finland) .......................................................... 1

Enhancing Groupwork with Social Navigation in Collaborative Learning Environment
J. Kuruila, M. Miettinen, P. Nokelainen, H. Tirri (Helsinki, Finland) ........................... 9

Producing Interactive Web Lectures with Authorware
T. Korola (Helsinki, Finland) ................................................................................. 18

Distance Learning Experiences from the Mathematical Modelling Course
K. Saastamoinen (Lappeenranta, Finland) ............................................................. 22

Virtual cross-border education: technical and teaching arrangements
S. Alaqulat, J. Voracek (Lappeenranta, Finland) ...................................................... 27

Virre - Virtual Reflecting Tool
P. J. Eroën, I. Jormanainen, M. Virnes (Joensuu, Finland) ................................. 35

TRAKLA2: a framework for automatically assessed visual algorithm simulation exercises
A. Korhonen, L. Malmi, P. Silvasti (Helsinki, Finland) ................................. 41

Problem-based learning of theoretical computer science
W. Hämäläinen (Joensuu, Finland) ................................................................. 50

A Narrative perspective to students’ experiences in problem-based learning in theory of computation – New deals with learning
M. Naumanen (Joensuu, Finland) ................................................................. 59

Introduction to Text Graphs
E. Nuutila, S. Törnä (Helsinki, Finland) ............................................................ 68

C-cards: using paper and scissors to understand computer science
A. Valente (Esbjerg, Denmark) ........................................................................ 77

Assessing Time-Efficiency in a Course on Data Structures and Algorithms
H. Hansen, M. Ruuska (Tampere, Finland) ......................................................... 86

Software Mathematics as a Course Topic
A. Valmari (Tampere, Finland) ....................................................................... 94

An Introductory Web-based Course to Enhance High School Students’ Knowledge of Computer Science
L. Grandell, J. von Wright (Turku, Finland) ....................................................... 103

Desmond - An Interactive Tool for Study Planning
A. Rytkönen (Helsinki, Finland) ............................................................... 106
What is needed to be considered in addition to learning objects in computer science education

Bulent Cavas
Dokuz Eylul University, Turkey

Gaetano La Russa, Sari H. Pitkänen, Pasi Silander, Jarkko Suhonen
University of Joensuu, Finland

Panu Silvasti
Helsinki University of Technology, Finland
pasi.silander@cs.joensuu.fi

Abstract

When using learning objects (LOs) in computer science education it is not enough just to consider the learning objects and their contents. Requirements from educational technology, virtual learning environments, implementation of LOs and pedagogical methods should also be considered in order to facilitate learning. These fundamentals are not described or defined by the contemporary Learning Object Metadata Standard (LOM IEEE, 2002). In addition, the LOM specifications do not support advanced learning applications such as cognitive tools. There are no metadata elements that support the learners function as designers using technologies as tools for analyzing the world, creating, interpreting and organizing their personal knowledge. We conclude that pedagogical metadata specifications are needed in order to facilitate the meaningful use of LOs to support students’ learning processes.

1 Introduction: Definition of the Learning Objects

Educational hypermedia systems and web-based learning environments enable the use of a new kind of learning material like Learning Objects (LOs). Currently, there are various definitions of LOs that differ from each other (17). In this context a LO is defined to be a small piece of learning material (e.g. short video clip, animation, interactive simulation, interactive exercises etc.) that is re-usable and compact as well as a unitary entity. The LOs are reusable in the sense that a particular LO may be used in various learning processes and different stages of a learning process. There may be also many pedagogical functions for a LO in a learning process (20). Therefore, learning objects should be compact enough to enable educators in constructing individual learning processes that consist of several learning objects stored in a database.

The definition or classification of learning objects may be based on the description of used media or on the description of the LOs” outlook” (e.g. presentations, drills, simulations). However, in order to utilize them meaningfully in educational settings, the broader context should be considered for the taxonomy and for the definition of the learning objects. The following figure (Fig. 1) visualizes the taxonomy of different types of learning objects.

When LOs are considered as cognitive tools (mind tools) (11), they refer to technologies that enhance the learner’s cognitive powers during problem solving and learning by functioning as a tool for distributed cognition. Learning objects as cognitive tools may activate complex cognitive strategies and critical thinking. They are essential components of a learning environment in which learners are required to mentally process deeper about the subject-matter domain and to generate mental models that would be difficult to create without these tools.

A TRAKLA2 (13) exercise applet is an example of a LO that is based on visualization and interactive simulation. TRAKLA2 is educational software for learning data structures and algorithms. In the algorithmic exercise applet (see Fig. 2) the learner modifies the visual
Learning object types

1) **Learning seeds**: LO that intentionally initiates a learning process or causes learning processes in a learner (guides the learning process)

2) **Content modules**: The content or object that should be learned (e.g. list of words, glossary) (do not guide the learning process)

3) **Learning tools**: Cognitive tools that guide the learning process and learners' activity
   - a) **Context bound (dependent) learning tools**: Tools that are related to a particular subject and content
   - b) **Context free (independent) learning tools**: Tools that can be used with any content and context (e.g. concept maps, etc.)

4) **Tools / Utilities**: Software for word processing, graphics etc. that do not guide the learning process

**Figure 1**: The different types of the learning objects described in relation with learning process (20).

representations of algorithm's data structures as the actual algorithm would do. The learner simulates the execution of a binary search tree insertion algorithm by dragging & dropping keys from the stream of keys structure into the tree structure. In the model solution window the learner can browse the discrete states of the binary tree structure as keys are inserted into the tree. This learning object may be used for several situations with different pedagogical functions, such as cognitive activations, hypothesis testing and reflections.

**Figure 2**: TRAKLA2 educational software for data structures and algorithms. The learner simulates the execution of the algorithm by dragging & dropping keys.

Also interactive analogies can serve as examples of learning objects. Real concepts have been described through metaphors and analogies. An example of the use of analogies is Elec-
tricity Analogy (12), where empty vans collect bread from a bakery (the bread is equivalent to load energy charges in the battery) and each of the vans delivers bread to the supermarket (see Fig. 3). In this context an abstract concept of current is described in a very concrete way. Similar interactive analogies can be used in computer science education for teaching highly abstract issues.

![Electricity Analogy](http://sycd.co.uk/electricity-analogy/)

**Figure 3:** Interactive analogy for electricity topics in science education. (http://sycd.co.uk/electricity-analogy/ and http://web.deu.edu.tr/buca/fenbil/BIDE 29.8.2003)

## 2  Educational technology for learning objects

### 2.1 Environments

Because LOs do not work alone in instruction they need an environment and a framework that define how to use them in teaching and learning processes. In order to use LOs in different situations they need to work on multiple platforms e.g. web, mobile devices like PDAs. LOs are more reusable if they are not heavily bound to particular learning theory or a pedagogical model. Still LO should guide sub learning processes and students cognitive processing. The pedagogical focus and framework is often provided by a teacher or by an environment.

In order to have access to LOs and build up learning processes there is a need for database and learning content management system or and virtual learning environment. LOs should be easily accessible via databases. Current standards support the retrieving of LOs, but rarely facilitate the implementation of LOs. In addition, ongoing standardization of learning contents, LOM standard (10), contains technical and content specifications and coarse pedagogical characteristics. However, this is not sufficient for the new type of digital learning material like LOs.

### 2.2 Adaptation in LOs

It is possible to support different kind of learners and learning processes by using adaptive methods (6; 7). There is an obvious need for adaptation because learning objects may be used in several contexts and for different learners on various levels. Content level adaptation, like adaptive presentation techniques and methods (6), can be done in some degree on a level of individual learning object, but when considering the adaptation to the student’s learning process and the adaptive navigation support (6) it is not possible to provide adaptation by the unique learning object. It may even be impossible to create a sufficient student model based on the users’ use of one LO. Adaptation to the learner’s learning process, adaptive navigation support and most of the adaptive content must therefore be provided by an education system, like a Learning Content Management System (LCMS) or a Virtual Learning Environment (VLE).
2.3 Automatic assessment

Automatic assessment and instant feedback in computer science education have been utilized, for example, in assessing programming exercises (1; 19), algorithmic exercises (4; 14) and diagrammatic exercises of object-oriented design (9). In the exercise presented in Fig. 2 the learner can ask the system to grade his answer, which is a simulation of the execution of the algorithm. The system assesses the answer and gives instant feedback, which reports the number of correct steps in learner’s answer.

Feedback provided by a learning environment provides the base for learner’s reflection and thus enhances learning. Automatic assessment of learner’s answers is needed for the creation of the feedback. Automatic assessment is also a valuable tool in saving teachers time and increasing cost/efficiency of the education. Feedback created using automatic assessment of learner’s actions may be provided by an individual LO or in broader context by LCMS or VLE.

2.4 Developing educational technology in the perspective of the software design

The following aspects need to be taken into consideration when looking at the learning objects from the perspective of software design:

- **Authoring tools can be used to create learning objects.** For example, there are authoring tools for creating algorithm visualizations and animations. With the user interface of Animal (18) one can create an animation that can be saved as a file and embedded into a web page. Jawaa (16) is a script language for generating algorithm animations to be embedded into a web page as applets. Jawaa requires some programming skills from the developer, but the programming is more about creating the content of the animations, rather than implementing, for example, user interface operations.

- **Software frameworks can help in the design of interactive learning objects.** For example, the interactive exercise in Fig. 2 was created using TRAKLA2 software framework (13). The framework hides most of the details of the implementation from the designer. One merely needs to define the data structures of the exercise, what are the rules for creating randomized initial values of the structures and the algorithm to produce the model solution.

- **The development process of learning objects is a continuos process of development and prototyping.** Formative evaluation is needed at the end of each prototyping cycle. The goal of the evaluation must be to analyze the effect of the LOs in students learning. For example, Mitrovic et al. (15) have gone through many development cycles and conducted several evaluation studies in the development of an intelligent tutoring system for SQL, SQL-Tutor. They have studied educational technologies applied in the SQL-Tutor from different perspectives; whether the constraint-based student modeling techniques used in the SQL-Tutor supports learning, the impact of different level of feedback in learning, and the effectiveness of the pedagogical agents.

3 Pedagogical Approach

In active learning students are involved in their learning process and they should engage in such higher-order thinking tasks as analysis, synthesis, and evaluation (2). In addition, during the interactive learning process software or educators should provide feedback to the student. Social interaction, conversation, shared work space and tools can be defined as key instructional elements for collaboration. The level of interactivity has a major impact on the quality of computer assisted learning (3). Interaction and learning do not simply occur but must be intentionally designed into the educational applications. LOs alone do
not necessarily support the learning process sufficiently because they usually support just situational cognitive processes but not the whole learning process. The learning environment or educators should provide support for the whole learning process.

3.1 Learning Theories

Learning theories with different perspectives describe how the learning occurs and what is the best way to support the learning (8). Learning theories, on which the instruction is based (implicitly or explicitly), define the purpose and pedagogical function of the learning object (see Table 1).

Table 1: The primary role of the learning objects in learning theories.

<table>
<thead>
<tr>
<th>Learning theory (learning concepts) and the role of learning objects</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Behavioristic theories:</strong></td>
</tr>
<tr>
<td>According behavioristic learning theories learning is basically based on the idea of conditioning and enforcement by providing feedback (8). The behaviorist pedagogy can be described as teacher-directed, text- and work-book-dominant curriculum (11).</td>
</tr>
<tr>
<td><em>The role of learning object:</em></td>
</tr>
<tr>
<td>The main role of the learning object could be a content module as well as drill &amp; practices that guide learning process by giving simple feedback.</td>
</tr>
<tr>
<td><strong>Cognitive information processing theories:</strong></td>
</tr>
<tr>
<td>Cognitive learning theories follow a general approach that views learning as an active mental process of acquiring, remembering, and using knowledge. New knowledge is built upon the existing knowledge structures (8).</td>
</tr>
<tr>
<td><em>The role of learning object:</em></td>
</tr>
<tr>
<td>The main role of the learning object could be a trigger for learning process or a cognitive tool that guides student’s observation and information processing.</td>
</tr>
<tr>
<td><strong>Constructivist learning theories:</strong></td>
</tr>
<tr>
<td>Learning is seen as an active process where learners construct and reflect their individual meanings from experience by interacting with the surrounding environment (21). According constructivist theories learning is described as a metaphor of knowledge building (construction) (2).</td>
</tr>
<tr>
<td><em>The role of learning object:</em></td>
</tr>
<tr>
<td>The main role of the learning object could be a content module as well as a cognitive tool for knowledge construction process. Learning objects may be used in exploring, hypothesis testing and reflection.</td>
</tr>
<tr>
<td><strong>Socio-cultural theories:</strong></td>
</tr>
<tr>
<td>The emphasis in Socio-cultural learning theories (22) is on collaborative construction of the knowledge. There are similarities with the constructive learning theories, but learning is seen more as a social action than an individual endeavor (8). The social context, like teachers and peers, has a significant role in the learning process. Hence, the learning situation should not be separated from its social, cultural and historical roots.</td>
</tr>
<tr>
<td><em>The role of learning object:</em></td>
</tr>
<tr>
<td>The main role of the learning object could be a cognitive tool (ideally collaborative tool) as well as content modules. Learning objects may be used e.g. in exploring, hypothesis testing and reflection.</td>
</tr>
</tbody>
</table>

3.2 Learning methods

Learning methods are derived from learning theories that more or less specify the nature of learning activity. In order to use LOs pedagogically in the most efficient way the learning methods (that facilitate students’ learning processes) should be somehow linked to the LOs.
Support for learning process may be provided by an individual LO or in broader context by LCMS or VLE.

The learning goals determine very often the use learning objects. If the learning goal is just recognizing information, the very basic learning object may be used. There is no need for interactivity or adaptivity in the sense of supporting knowledge building activity. The desired level of learning can be achieved e.g. by using LOs like behaviorist drills. If there are no needs to learn problem solving skills or cognitive strategies (repetition of the information is the goal of learning), LOs that describe the procedural information and are pedagogically structurized, may very often be sufficient. Naturally student’s observations and information acquisition should be guided.

In order to learn skills that are needed for applying learned information in various situations students should process the content of the LOs more deeply. Interactivity that enables reflection and “learning-by-doing” is required. There may also be triggers or scaffolds (23; 5) for student’s cognitive processes and elements that methodically support student’s information processing like problem solving.

More complex cognitive strategies and processes are needed (e.g. in design of algorithms or data structures) in computer science in addition to the factual knowledge and the capability to apply it. Students should achieve a capability to constantly create new information and solutions. Therefore more sophisticated mental models and higher levels of cognitive strategies are definitely needed. These kind of mental models and cognitive strategies can not be learned just by using a LO. Collaborative knowledge building and the use of abstract conceptual systems as tools for thinking are needed (22; 2).

4 Conclusions and Discussion

Learning Objects may be used in various learning processes and at various stages of a learning process. They are essential tools when learning abstract issues that require students’ mental development of models and cognitive strategies. However the learning context (educator, learning environment, situation, etc.) often creates the pedagogical framework and the pedagogical function for LOs. Therefore LOs may not be observed without the broader context of educational technology and pedagogy. Pedagogical metadata is needed in order to facilitate meaningful implementation of LOs as a part of learning process. Pedagogical metadata should also support cumulative descriptions of best practices for educators.

References


Enhancing Groupwork with Social Navigation in Collaborative Learning Environment

J. Kurhila
Department of Computer Science, PO Box 26, FIN-00014 University of Helsinki, Finland

M. Miettinen, P. Nokelainen and H. Tirri
Helsinki Institute for Information Technology, PO Box 2700, FIN-02015 Helsinki University of Technology, Finland

kurhila@cs.helsinki.fi

Abstract

EDUCO is a computer-supported collaborative learning environment that is designed to serve as a platform for self-evolving ad hoc study groups. Instead of trying to modify the environment to the individual, the underlying idea behind EDUCO is to equip the learners with information about the other learners so that meaningful collaboration is possible. This is achieved by revealing the actions of others with real-time social navigation and supporting interaction via synchronous and asynchronous discussions. EDUCO has been used in an advanced university-level course, where self-organizing and self-directing group work was required. This paper examines how the groups were formed, how the students viewed the group work, and what was the effect on group performance if the group consisted of students with varying motivational factors.

1 Introduction

Student-centred learning often requires working in groups. One popular approach is that the teacher or a tutor assigns students to groups, possibly according to various characteristics of the students. This paper examines how groups can be formed in self-organizing manner in computer-supported collaborative learning so that each individual can benefit from the group. The key issue is to provide the students with information about their study companions and make the learning process transparent to other students so that they can see and make use of the actions of other learners in the environment.

The tool used in this study is called EDUCO. EDUCO relies on aspects of social navigation to make the actions of others visible. When Dourish and Chalmers introduced the concept of social navigation, they stated it to be "navigation because other people have looked at something" (2). The concept has evolved since then and various categories of social navigation have emerged.

In the seminal book of social navigation, Munro et al. (8) use an on-line grocery store as an example: if people visiting the store are given recommendations what other people have bought, it is a form of indirect social navigation. If a shopper in the grocery store has a sense of other people moving about the store and can engage in seeking e.g. assistance, it is a case of direct social navigation.

Direct social navigation and awareness issues have been applied in various contexts (see e.g. (6; 5)), but experiments with social navigation in educational setting have mostly fallen into the category of indirect social navigation (see e.g. (1); of course, exceptions exist (4)). However, EDUCO is a system for collaborative learning to employ real-time direct social navigation in education. Real-time aspects of EDUCO create the feeling of live companions in the system (7). Other important facilities include tools for synchronous and asynchronous communication, and support for forming study groups and publishing their work.
2 EDUCO tool

2.1 Map view for social navigation

From the users' point-of-view, EDUCO tool consists of six different views for various activities, a document area and a discussion area (Fig. 1). The views are map, chat, search, alarm, preferences and help. They are presented in a tool resembling a handheld computer (upper-left corner in Figure 1, now in map view). The map shows documents anywhere in the Web that are linked to EDUCO that serve as a learning material for the course.

![Course Sequencing for Static Courses? Applying ITS Techniques in Large-Scale Web-based Education](http://educo.cs.helsinki.fi/educo/educo/educo)

**Figure 1:** Educo tool.

In addition to presenting documents currently available in the environment, map view provides a way to navigate to them directly. By double clicking a document a user can open it in the right-hand frame in the browser window, as seen in Fig. 1. A user is represented as a coloured dot around the document he or she is currently viewing. Other users are visible to every user in real-time, so that their navigation is visible to everyone present.

The colour of a dot indicates a group membership. This type of group membership is assigned by the teacher or course administrator and is independent of the groups the students form. An example of grouping could be teachers, tutors, observers, and students. Another suitable way to differentiate students is their motivational profiles or learning strategies, as described later in this paper. By representing different groups with different colours, the participants have one additional source of information they can use when selecting their study partners or teams for group work.

The documents change their brightness level and colour on the map depending on how much they have been viewed relative to the other documents. The total time all users have spent viewing each document is recorded every hour. The change in the brightness and in the colour of an individual document are determined by the distance of its moving average for the last 24 hours from the same average for all documents.

Map view provides the users of EDUCO with two essential features of social navigation. Colouring the documents according to how much they have been viewed is a form of asyn-
chronous social navigation. Presenting users as moving dots next to the documents they are currently viewing is a form of direct synchronous (i.e. real-time) social navigation. Both of these features can help the users to follow the footsteps of others.

2.2 Finding and forming groups

Finding team members to complete an assignment is often obligatory in student-centred learning. When searching for a companion for group work, regular search is useless unless a student already knows the person he or she is willing to team up with. However, another type of search can be used for the purpose. EDUCO’s alarm offers each user a possibility to set "triggers" into the documents, groups and the overall system. In other words, a user can set EDUCO to alert when certain conditions occur. This feature is useful in a case where a user searches for a companion (possibly from a certain group) showing interest for a certain document or topic, or wants to contact a particular person when he or she enters the system. The alarm function also enables making combinations of triggering events.

Even though the navigation of others is visible to the other participants, initiating the communication between the students also has to be made as simple as possible. EDUCO has a built-in chat integrated to the map view to enable ad hoc synchronous communication between peers and other users. The chatters can be picked up from the map view by clicking the dots representing users. The number of participants in the discussion is unlimited, but one person may use only one chat channel simultaneously. Thus the functionality of the chat is restricted when compared to many commonly used chat services. However, the chat in EDUCO is targeted to a small exchange of ideas when searching for a team member for group work.

Of course, there is also a search function in EDUCO. It can be used to find both persons and documents. The search is targeted to the titles of documents and names (and nicknames) of users, both online and offline. The search results are shown in the search view, but also in map view by highlighting the document with a blue rectangle, as can been seen in the map in Fig. 1.

2.3 Publishing group work

Working in groups is an essential part in student-centred learning. An important feature in EDUCO is the support for forming groups. Alarms, chat and navigational patterns can be used when screening for potential partners for group work. When the students have found and agreed upon their study companions, any one of them can start a group by clicking a button "Add a new group". Other people can join such a group, or they can start a new group and try to persuade people to join. People not yet in any group are shown in the column on the left, groups already formed in the middle column, and the work published by the group in the right column (Fig. 2). Publishing joint work, such as a report or the draft of a report, is done by submitting the URL of the Web page containing the work.

The way group forming is handled in EDUCO is particularly suitable for a course where the groups are not stable for the entire course, and more importantly, the groups are formed by the students themselves without teacher assistance. This approach requires metacognitive skills by forcing the students to think about what kind of expertise they will need for the task at hand. However, the features in EDUCO makes the actions of others visible so that the students can form better opinions about fellow students for selecting study companions.

Document icons in the map view of EDUCO can each represent a collection of student reports. Otherwise, the space for documents in the map view could be overwhelmed especially in large courses. These documents that are actually a collection of documents, are used for reports that students produce. In Fig. 2, the group of nine bright documents in the lower part of the map each represent a collection of weekly reports. The reports are accessible by clicking the corresponding link anchor (i.e. report name) on the right-hand column.
In addition to the synchronous discussions described above, EDUCO has two possibilities for asynchronous communications. A document in the EDUCO map can have one of these types of communication chains "attached" to it, i.e. the comments are always document-specific. A discussion chain for a report document published by a student group working together for a weekly assignment is a typical hierarchical discussion chain, where previous discussions can be quoted and a new discussion chain can be started. The discussion is meant for building knowledge together, by publishing an early version of a report for an assignment, for example, and soliciting comments from other course participants.

Another type of comments is shown in the lower left-hand corner in Fig. 1. Whereas the documents with student reports have newsgroup-type hierarchical discussions, other documents gathered into EDUCO have a space for general notes or comments. A typical use of these comments is to ask others about an unclear point in an article, or note an issue in the article. Course participants used this type of comments frequently to announce that they are seeking study companions to form a group. Both types of comments are open to everyone in the system.

3 Study setting
3.1 Course description

EDUCO was tested empirically in an advanced course in Computer Science entitled "Computer Uses in Education". There was no final exam but the students had nine weekly reports (assignments) to produce from nine different topics in varying teams. The reports were to be prepared in groups. Group size was not fixed, but groups of two or three were recommended. Moreover, the groups were not allowed to stay the same during the course. An important requirement was that a student had to be involved with at least three different groups.

Weekly assignments were presented as open-ended problems, projects or cases, such as consulting a higher education institute on what kind of course delivery system they should

![Figure 2: Students not yet in any group, groups and their published reports.](image-url)
choose or drafting an approach on how to evaluate web courses. There were no lectures and no pre-made learning material other than research reports and other appropriate resources. Although the deadlines were very strict, within the deadline the students were using self-paced collaborative learning.

Apart from the documents containing the reports of the student groups, the documents in the EDUCO map discussed the issues covered during the course. The documents were organized into eight different clusters under a common theme. The themes were close to the weekly topics but not completely the same. The document cluster sizes varied from two to ten, giving a total of ca. 40 documents. The exact amount of documents varied slightly during the course, since new resources were added or replaced occasionally.

The course included only two face-to-face meetings lasting 45 minutes. The first was an initial meeting where the structure and requirements for the course were explained. The second face-to-face meeting was organized the next day, and participants had an opportunity to get familiar with EDUCO. After that, every communication took place in EDUCO apart from some email announcements.

The course relied heavily on peer-commenting, since the teacher and the teaching assistant were not able to extensively guide or comment the reports of the student groups. The topics covered different aspects of Web-based learning, such as platforms, learning theories, interaction and adaptive educational systems. The reports were graded by two teachers together on a scale of 0 to 6.

3.2 Motivational profiling of the students

Before the course started, the students filled in a questionnaire to clarify their motivational profiles. The results were used in colouring the dots in the EDUCO map view, so that every participant knew the motivational profile for every student. The group descriptions with clear explanations were published for all the participants, so that the students were able to use the information when choosing a study companion or group.

Motivational profiling in this study is based on the Motivated Strategies for Learning questionnaire, MSLQ (3). MSLQ measures both motivational factors and learning strategies. The motivation section of MSLQ consists of 17 items that were used to assess students' value for a course, their beliefs about their skill to succeed in the course, and their anxiety about tests in the course. A 5-point Likert-scale ranging from 1 ("Not at all true of me") to 5 ("Very true of me") was used for all items.

The theoretical model of motivation (10) is constructed out of six factor solution: (1) Intrinsic goal orientation, (2) Extrinsic goal orientation, (3) Meaningfulness of studies, (4) Control beliefs, (5) Efficacy beliefs, and (6) Test anxiety. We expected to find a similar structure in the sample data and thus to be able to construct sensible motivational groups.

The analysis of the questionnaire was carried out with a Bayesian dependence-modeling tool named B-Course (9). The results indicated that the theoretical model of six factors (11) was a viable solution for this small number data set. Based on the motivational level scores on six dimensions, respondents were divided into three groups:

- Group 1 ("Red", N=20) characteristics: (2) Extrinsic goal orientation, (6) test anxiety and (3) meaningfulness of studies.
- Group 2 ("Blue", N=11) characteristics: (5) Efficacy beliefs, (1) intrinsic goal orientation and (3) meaningfulness of studies.
- Group 3 ("Green", N=11) characteristics: (4) Control beliefs and (1) intrinsic goal orientation.

The classification accuracy of the theoretical model was confirmed with both a linear and nonlinear discriminant analysis. There was no statistical significance between the group memberships of male and female respondents.
4 Results

This section examines self-organization of the groups, how the students viewed the group work, and the effect of different group formations. The quotes in this section are excerpts from the students' learning diaries with no specific questions given beforehand. The quotes have been translated from Finnish to English.

4.1 About the study groups

Learning diaries of the students revealed different types of behaviour for group formation. "Active" participants started a group and tried to lure other members to join in. "Hopeful" participants started a group but did not actively try to find other members, just waited people to join in. "Scared" participants formed groups with people they knew outside the environment. "Rude" participants joined a group without asking permission. "Picky" participants evaluated carefully the quality of the previous work of others before asking to join a group. "Planners" made agreements into the future with a particular group to ensure that there is a safe backup group if no better comes along.

These types of behaviour evolved throughout the course so that every student was acting according to different behaviour in different stages of the course. It was typical that students stayed with their friend for the first assignment (or, in a case where the student did not know anyone from the course, asked the first person visible in the system to be a study companion). When the confidence grew, many of the students started to use different strategies for selecting study companions, as can be seen in this quote:

"Forming groups was stressful at first, but one got used to it quickly, and it didn't cause any problems."

It is important to note that the logged data revealed that there were no clear leaders or followers (always starting a group or always joining an existing group). However, some students saw this differently:

"I noticed such behaviour that some students wanted to start their own group even though there were groups of one already."

In many cases, where the study group was viewed to function well, group members agreed to come back together to complete at least one weekly report at some point of the course:

"I would like to be a part of this group also next week, but I think that these two members of my group are just on leave from their 'actual' groups, so that they fulfil the requirement of participating in different groups. [...] I also wonder if we, who do not know anyone or do not succeed in finding a good group right from the start, are we just drifting from one group to another every week?"

Only a few students consciously used the work produced by others as a basis to evaluate potential group members. However, when using EDUCO as a learning environment, the actions of everyone are highly visible to everyone, so the reports and comments made are likely to have an influence on the learners' behaviour, consciously or not.

The amount of study companions for each participant during the course varied heavily. The lowest possible amount (when acting according to the course requirements) was 3, and there were five students who chose not to have more study companions than 3. On the other end, one student had 11 study companions, while the average amount of study companions was 6.05. The amount of companions did not have any effect on the course grade, as can be seen in Tab. 1.

Overall, students valued the self-organizing and self-evolving groups:
Table 1: Amount of study companions, size of clusters, and average amount of points for each cluster type.

<table>
<thead>
<tr>
<th>Amount of companions</th>
<th>N</th>
<th>Points (average)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>30.2</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>30.9</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>28.3</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>29.2</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>29</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>32.2</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>30</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 2: Differences in motivational profiles within groups, amount of such groups and average amount of points.

<table>
<thead>
<tr>
<th>Entropy</th>
<th>N</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>36</td>
<td>4.3</td>
</tr>
<tr>
<td>0.81</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>0.92</td>
<td>41</td>
<td>4.6</td>
</tr>
<tr>
<td>1</td>
<td>26</td>
<td>4.5</td>
</tr>
<tr>
<td>1.58</td>
<td>8</td>
<td>4.6</td>
</tr>
</tbody>
</table>

"There were few groups, whose composition did not change much during the course. I wonder if they get as much out of the course? I think that forming groups [without the teacher] was educational. You had to learn teamwork skills, since my groups were very different in character."

4.2 About the motivational groups

It is clear from the student learning diaries that in the beginning many of the students did pay attention to the motivational groups of others, and used the colour of a dot as one factor for group formation. In the end, no one admitted to that dot colour was a factor. Motivational groups were still seen as one additional piece of information characterizing fellow students:

"Colour [of a dot on the map view] gave a student a personal character, which was nice compared to just a name."

Generally, different motivational profile of a study companion was not seen as problematic as differences in timetables and study habits. This view is also supported by the final results. Study groups with students with the same motivational profile (the entropy of profiles is 0; during the course there were 36 such groups) averaged 4.3 points per assignment, whereas groups with the greatest variety in motivational profiles (entropy 1.58) scored 4.6 points on the average (Tab. 2).

We classified the entropy profiles into five (0.00, 0.81, 0.92, 1.00, 1.58) and three (0.00, 0.81-1.00, 1.58) classes and used the Kruskal-Wallis test to examine if distribution functions would differ by average points per assignment. The results showed no statistically significant differences between groups classified into five ($\chi^2(4)=1,931$, $p=0.748$) or three groups ($\chi^2(2)=1,232$, $p=0.540$).

Students in the second motivational group ("Blue") were slightly more eager to seek different study companions; the average was 6.6 compared to 5.4 and 5.5 for the first ("Red") and third ("Green") motivational groups, respectively. Because of the small sample size, these differences are anecdotal.
5 Conclusions

Computer-supported collaborative learning consisting intentional active learning and knowledge-building resources necessitates a flexible and synchronous learning situation (12). EDUCO offers an environment that can be used in flexible manner to make the learning process more transparent so that the students have enough information of peers to organize meaningful study groups independently from the teacher. The study presented suggests that self-organizing groups are equal in performance, regardless of the group formation (i.e. groups consisting of people with the same or different motivational factors). Staying with the same study companions (i.e. only a few companions in total) or rotating study companions rapidly (i.e. a lot of different study companions) appears to have no impact on performance, either.

6 Acknowledgments

This research has been partly supported by the Academy of Finland.

References


Producing Interactive Web Lectures with Authorware

T. Kerola

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

Teemu.Kerola@cs.Helsinki.fi

1 Introduction

There exists a general demand to utilize the web (Internet and Intranet) more and more in education. New teaching/learning methods are now needed that were not applicable before. One can broadcast live lectures to students in their own homes or at least in remote classrooms, one can build large systems co-operatively with other students without ever meeting them face-to-face, or one can do independent studies using interactive material accessible in the web.

Such new learning methods are not necessarily simple to use, nor is the teaching/learning material associated with them easy to produce. This paper concentrates on one learning method, interactive web lectures, and how to produce it in university settings. We are not emphasizing (in this paper) how this material will be used in university course settings even though that is ultimately its planned use.

We had the opportunity to study the IBM process to produce just this type of learning material, and we found the process too cumbersome and expensive for our current needs. The IBM process is based on a specific Knowledge Factory Team that can independently produce the any material with just one subject matter expert (e.g., university instructor) advising the team on the material to be taught. The team can be quite large (15 different roles for 5-30 team members) and seems better suitable to industrial settings than for universities. The course contents must be specified in a detailed manner during the process, and it is not easy to modify the course later on without bringing in the whole team again. (3)

The IBM process is built around their own content development software, Knowledge Producer (1). Knowledge Producer generates educational material that is suitable for IBM Lotus LearningSpace (2), but not for generic web browsers. Before the production phase begins, all non-content details must be specified in a thick High Level Design Document and all content details in even thicker Detailed Design Document. In business environment both these contractual documents are important as they define in a precise manner all expectations and demands for the final product. So, the content development process is cumbersome and costly because of the large team involved, and the use of the material is expensive. We wanted to have a smaller process and the final product in a form that easily and freely accessible to all students.

We needed commercially available, economical software for our content development. It should produce learning material that would run in a standard web browser, or with a free plug-in. We needed a tool with which one teacher (in our Computer Science department) would do all the content development by herself. She could use help in specific instances, e.g., in graphics or animations, but in general she would develop the content alone in a similar manner that is used to create PowerPoint slides for lectures. It was also important, that it would be easy to update the contents annually.

After a short study we selected Macromedia Authorware (4) as our content development platform. Other platforms exist, but this one seemed to offer the functionality we needed and lots of associated software that may become handy (E.g., Fireworks for graphics editing and Flash for animations). Also, we wanted to have a quick start, and we felt that this was a solid platform to start our experimenting. Later on, we may try out other platforms.

The main goal is the production process, not the produced contents. However, it is obvious to us that the process cannot be very good unless the learning material produced with it is of high quality.
2 Resulting Learning Material

We wanted to produce stand-alone learning material that is somewhere between a lecture and a book. The idea is that the student will get pretty much similar learning experience to listening a good lecture, but having no possibility to interrupt the teacher and ask questions. However, one can go back and forth in the material, and listen (and read) difficult parts of the lesson as many times as needed. The lesson can be browsed anywhere, anytime, and at a pace determined by the student.

The material is also like a book, because it can be browsed here and there. It is better than a book, because it may include not only pictures worth of million words but also moving pictures and animations worth of billion words. It has sound and it may include built-in interactive problems to solve.

Terms interactive book or interactive web lecture would be suitable to describe this type of material. We prefer the latter term, interactive web lecture. So far, the interaction is mostly navigation, but in the actual use of the material there will also be interactive animations, problems and practice questions for students. Those components could be directly linked into this material, or they could be separately accessible from the course home page.

Students using this material need a modern multimedia PC with Internet connection. It is recommended to have broadband access to Internet. With a 56 kb/s connection the prototype material can be browsed, but it behaves in sluggish manner. However, we have not yet even tried to optimize the learning material for smaller bandwidth connections. If the material is to be used in public settings (e.g., in a computer lab), each student should have headphones. The department has a pool of (cheap) headphones that the students may check out in case they do not have their own headphones.

3 Starting Point for Development

We assume that the course material already exists in PowerPoint format. In addition to the PowerPoint slides, the instructor of course knows very well the learning objectives of each slide and how he would present the material in a standard lecture. The instructor himself will be the person doing all the content development, and so he must be also familiar with the production software, which in this case is Authorware.

We assume that most CS dept teachers can learn to use Authorware well enough for this purpose with relatively small training, either as independent study or using a departmental short course. If this process would be extended to other departments, where teachers are not involved with computer technology so much, we assume that one consultant would be needed to help the instructor at least for a while in web lecture production.

The instructor needs a modern multimedia PC, Internet connection, and an Authorware licence. She also needs a good quality microphone, preferable one connected to headphones. The recorded voice volume should be uniform and that is easier to achieve with a separate microphone as compared to using a built-in microphone in laptop or desktop PC.

4 Production Process

The content production process advances in two phases. The overall general design is produced in Design Phase and the actual learning material production in Content Development Phase. Once a good overall design has been developed, that can be used with all new modules.

4.1 Design Phase

Design Phase defines the layout, navigation, language, look and feel, colour scale, fonts etc for the whole material. This is implemented by setting up basic structures in Authorware with an already existing Knowledge Object (Authorware term), and modifying it to implement general design requirements.
Specifically, we do not produce any big documentation on these decisions as was done with the IBM process. With the IBM process it was important to lock in to the detailed design specification in written form as that specification was then used as a contractual document between IBM and the client. In departmental course material production such large documentation is not needed.

4.2 Content Development Phase

Actual production of the learning material happens in the Content Development Phase. This is where PowerPoint slides and instructor know-how are translated into web lectures. It should not be started before Design Phase is completed. If there are any major changes in Design Phase after the content development has already begun, one may need to implement those changes to each already produced page one page at a time.

Each PowerPoint slide set for a (1-2 hrs) lecture is implemented as one (Authorware) module, which may then be independently browsed at any time. The lecture is constructed of multiple sections (topics), and each section has many pages. Each page consists of one or more views. Each view has some visual modification to the page background and instructional text relating to that specific view. A view is smallest educational unit in this material, and it corresponds closely to what an instructor would teach between two mouse clicks during a lecture based on PowerPoint slides.

Content developing is executed by translating each PowerPoint slide (or a few tightly coupled slides) into one (Authorware) page that may have multiple views. Each page is started as a copy of the prototype page (produced in Design Phase) and it includes a background definition that applies to all views in that page. The drawing area for the background picture is marked, and it must be removed once the page is complete.

Each view in the page is built separately, again starting with a copy of a prototype view (which was also produced in the Design Phase). The view has its own visualization part that is overlaid on top of the page background. In this way, the view looks pretty much like one PowerPoint slide. In classroom setting, the teacher would have some explanation at this time to the students, and that explanation is now given both textually and vocally in the web lecture. The instructional text is first written down, then recorded separately, and finally tied in to this view. So, the information in PowerPoint slides is shown visually in each view, and the teachers know-how or instructional guidance is shown in text window and heard as voice with that view.

In the resulting web lecture, the students may elect whether they want to hear and read the instructional text, just hear it, or just read it. If the instructional text is neither heard nor read, then the material can be used by the instructor as lecture slides. However, the material is not as good as original PowerPoint slides for this purpose, because it has been developed for independent learning and not for lecture based teaching.

The material is easy to modify, which is important to university settings where courses are often updated. Once the modification is implemented in the content, the resulting learning module is updated with a one click of the mouse.

5 Summary and Future Work

We have produced one prototype module, corresponding to 37 PowerPoint slides, designed for a 2hr lecture on Compiling, Linking and Loading for a Computer Organization course, in Finnish. The web lecture has 24 pages in 6 sections, and altogether 58 views. The production was very slow to start with, but the last 80% of the material took some 7 working days to finish. The initial slow start was due to Authorware learning curve, and our need to first develop the process, and only then to do the content development. We would anticipate that the 11 remaining 2 hr lectures could be translated to similar web lectures in 1-2 weeks per lecture. The time of course depends very much on what type of tools one would use. For
example, any animation would be time consuming. Translating normal (text with simple graphics) PowerPoint slides to web lectures is straightforward and fast.

So far, we have not experimented with animation. Also, we will still fine tune the prototype before continuing with content development for new modules. We will use departmental know-how with both the user interface and web learning pedagogy.

We will try out this material with our normal lecture course this autumn. One normal lecture is replaced by this module for either all students or some section of them. Student feedback is collected with our normal course evaluation.

Once the complete lecture course is available as web lectures, one should take careful consideration on how to use that material. There should be ample room for student-teacher communication, either face-to-face or via web. There should be practice sessions, again either face-to-face with assistants or via web. However, running the complete course in web will introduce new problems that we have not yet really even considered.

Specifically, one should not simply replace usual lectures with web lectures and keep the rest of the course as is. We think that attending traditional live lectures is still the best mode for learning for most students in higher education. However, there are many situations where that is not possible, and then this type of web lecture could provide one good alternative.

Once we gain more experience in building and using web lectures, we can define useful criteria for evaluating tools to create them. That criterion must include, in addition to various components included with each tool, usability features that are more difficult to measure. Each tool should also be evaluated on how well it supports given production process, i.e., "how good is it for me?" Each tool should be tried out in practice and that will require lots of human resources.

Our emphasis will be in content production (i.e., how to produce web lectures, which learning methods to use in web lectures) and end-product use with web courses (i.e., how to use web lectures), and we will let many of the underlying technical issues (how to port to Linux, best way to store voice, best way to play videos, etc.) be solved elsewhere by the tool makers. We will not try to develop new tools, but instead find the best ways to use and evaluate them. One can always have better tools, but the current ones are quite suitable for useful work.

References

URL http://www.elearn.ihost.com/

URL http://www.lotus.com


URL http://www.macromedia.com/software/authorware/
Distance Learning Experiences from the Mathematical Modelling Course

K. Saastamoinen  
*Laboratory of Applied Mathematics*  
*Lappeenranta University of Technology*  
P.O. Box 20, FIN-53851 Lappeenranta, Finland  
kalle.saastamoinen@lut.fi

Abstract

Paper in hand presents a collaborative method for creating a distance learning studying environment. It will show how this method has been applied in practise in a course called mathematical modelling. We have used both collaborative and problem based learning methods in our teaching and this is also exposed in this paper. Paper will also explain what we have learned so far from distance education and discuss the future of this mathematical modelling experience. Mainly I will show a case where traditional classroom teaching is not an option for the reasons of long distances between students and teachers.

1 Introduction

The focus of this paper is to present a way of creating a functioning distance learning course. From the beginning, the main method for establishing the course has been collaborative work with all the main Finnish universities. Also, the learning method for the students has been collaborative. The paper will concentrate on demonstrating the benefits of the methods used and will also identify the main drawbacks of this project.

During 2001, a group of Finnish mathematicians representing different fields of mathematics decided to provide distance education in mathematical modelling for all the Finnish universities that were willing to join this experiment. The main reason was that, so far, there had not been any common studying entity for this field of mathematics at Finnish universities. As a result, we now have a functioning Web environment for the studying of mathematical modelling in the form of distance education.

In this paper, I discuss how this modelling experiment was implemented. At the end of this paper, I will also show what we have learnt so far from distance learning platforms and the future of our modelling course. In Finland, distance education is becoming an ever more vital part of the family of higher education. Just about every major Finnish university offers such courses. Distance education offers many benefits such as access to a broader student audience, a better medium for meeting the needs of students, cost savings and, sometimes and importantly, its use of the principles of modern learning pedagogy (1). In Finland, the geographical as well as historical distances between universities are relatively high, and during the last five years, significant political and public interest has been shown towards distance education. Similar trends can also be observed in other countries than Finland (2). The mathematical modelling course was offered in co-operation with other Finnish universities. The modelling platform also aims to provide a consistent system for the teaching of modelling subjects all around Finland.

2 Distance Mathematical Modelling Course in Finland

Some main baselines for the initiation of the virtual mathematical modelling course project are listed below:

1. Mathematical modelling and simulation are essential methods for research whenever mathematics is to be applied.
2. From this, it follows that it is of vital importance for a person, who is studying applied mathematics or working in an area of research and development in industry, to master the methods of mathematical modelling.

3. There has been no comprehensive study environment for mathematical modelling in Finnish Universities (3).

The funding for this project has been provided by the Finnish Ministry of Education. The funds are shared among the participating universities mostly the basis of how much material they produce for the Web pages of the course. The project is maintained by the seven major universities in Finland, University of Helsinki, University of Jyväskylä, Lappeenranta University of Technology (LUT), University of Oulu, Tampere University of Technology (TUT), University of Tampere and Helsinki University of Technology (HUT), as well as the Finnish IT centre for science (CSC). The main goals of this project are to firstly collect expert knowledge in the field of mathematical modelling into the Internet and secondly to create both basic and advanced courses in mathematical modelling for the use of distance education in Finland at the university level. All the participants have contributed staff to produce the material. The consistency, or rather the relevance, of this material is assured by monthly meetings and later with the use of questionnaires for students.

2.1 How Co-Operation has been Carried Out?

When many universities work in collaboration, a lot of time and effort has to be spent on planning and becoming acquainted with different people from all over Finland. This has mainly been done in monthly meetings at which all the participants have met and been able to voice their opinions on the direction in which the course is going and on whether or not the goals that have been set in the previous meetings have been met yet. Also, one university has to take a charge of the project, and in our case, the leading university has been Tampere University of Technology.

The purpose of the first meetings for both the basic and advanced mathematical modelling courses has been mainly to acquaint the team members with one another, to establish the contents of the courses and to discuss the budget. Then, once some material has been put onto the Internet, the discussions have concentrated mainly on the practical matters of the course, which means discussing the suitability and technical quality of and student feedback on the material and possible delays in the delivery time of the material in addition to other matters. The final discussions have been concerned with the following year's budget, final student feedback, how to improve the material for following year as well as with timetables. During the first year, there were also a lot of discussions on copyrights (to whom they belong etc.).

2.2 Course Arrangements

At the beginning of the course, the students formed working groups of 1 to 3 persons. Once the groups were formed, they started working on their weekly exercises, giving comments to other groups and, finally, on doing their final assignments. It is important that the throughout the course the group works as a team. All the participating universities have an assistant whose responsibility it is to help the students if they are experience any kinds of problems with the use of platform or their exercises. The mathematical modelling courses have been implemented using a course delivery platform known as AO Fig. 1 with the help of a web site created by an instructor. The delivery platform has been used for discussions and as the base where students submit their weekly exercises and final assignments.

The lectures have been videotaped and have been delivered using a technique called SMIL. Finally, we give a presentation in which a lecturer talks, while on their right slides are projected in synchrony with their talk Fig. 2.
Problem solving on BB

Figure 1: View of the AO.

Moniarvologiikan sovellukset
Ilmion mallintamisesta ajattelun mallintamiseen
Esdo Turunen
Tampereen Teknillinen Korkeakoulu

- Matemaattisen mallinnuksen tavanomainen lähtökohta on ensin havainnoida illmitöitä, abstrahoida sen keskeiset osat ja kirjoittaa
  illitöistä sitten matemaattinen kurva, vaikka Newtonin jälkein mitali
  missa T(t) on kuuman kappaleen ja sen ympäristön
  välinen lämpötilaero ja k vaki.
- Esittää mukavien yksinkertaisetkin ilmiöt ovat sellaisia, että asiainvointi
  ilmiöin - hallitsee ne yksinkertaisilla säätöillä, mutta matemaattisen
  kannan kirjoittaminen on työlästä, esimerkiksi
- Liikennepoliisin toiminta perustuu yksinkertaisiin säätöihin
  Jos jalankulkijoita on kaksi ja he ovat odottaneet kahdeksan
  aina jalankulkijoiheippeen ylälaitteen
  Urheilulääkäri osaa määrittää huippu-urheilujen aerobisen ja
  anaerobisen kymmenen muutamalta yksinkertaisilla säätöillä

Figure 2: Example picture of a lecture.
All the weekly lectures are provided with exercises which students are expected complete and submit to the AO platform within a specific time period. Students are also required to comment the answers of other student groups every week. Finally, the answers and comments of students are graded by the weekly lecturer. At the end of the course, the student groups have to present their final assignment in a video conference that is arranged simultaneously between all the universities, and here again, students are expected to comment the work of others students. At the end, all the student assessments are combined with weights, and students are awarded their final grades.

3 Benefits and Problems or What has been Learned from the Project

The main benefits that we have been obtained with this distance education platform have been

1. student satisfaction; more than 50% of the students, who have taken our modelling course, would rather choose this kind of course again rather than a traditional course.

2. We now have a working teaching entity, which can be used with little effort in the teaching of mathematical modelling.

The main problems that we have encountered have been the use of our course delivery platform. Most of our students and teachers have found the delivery platform to be time-consuming, unstable and useless. However, many instructors are now using course delivery platforms, and many surveys indicate students are satisfied with using platforms such as WebCT (4) and (5). Also, as in the previously cited research reports, also all our students responded that they found the course materials on the course Web site helpful.

We have learned that the following things are of vital importance for a successful distance education course:

1. All the material should be available for the course use as soon as possible, because there will be problems with technology and the absence of lecturers.

2. It is better to make all the agreements concerning the copyrights of the material at as early a stage of the project as possible.

3. The course delivery platforms should be tested properly before they are used in practise.

4 Future and Conclusions Concerning of the Modelling Platform

The material, which was produced, can easily be used as compensatory or associating material for modelling. If properly used, it will make it easy to design both good and diverse courses on mathematical modelling. We are now building courses that can be used as associative materials for mathematical modelling and that can also be used as stand-alone courses. The materials that we are preparing at the moment are, for example, stochastic processes, the mathematics of visual motion and discrete mathematics.

Many research studies have shown that cognitive factors, such as learning, performance and achievement in distance education classes, are comparable to those observed in traditional classes (6), (7), (8), (9) and (10). It is also my humble belief that students will certainly learn as much in distance education courses as they would in normal classroom instruction.

References


Virtual cross-border education: technical and teaching arrangements

S. Alaoutinen, J. Voracek
Lappeenranta University of Technology, Department of Information Technology, P.O. Box 20, FIN-53851 Lappeenranta, Finland.
satu.alanpunen@lut.fi

Abstract

In years 2000-2002 we realized a set of experiments searching for a convenient structure of virtual classes combined with teacher’s personal involvement as well as their proper inclusion into the international degree education in information technology. This paper presents and justifies the way we achieved the final technical solution and teaching arrangements related especially to the lecturing process. Due to our cumulated teaching and researching roles we present our findings mainly from the practical point of view.

1 Introduction

Virtual education has many facets and all of them must be properly balanced for final success. The state of the art there is given, for example, in the latest studies conducted by the International Institute for Planning Education D’Antoni (1) or The Commonwealth of Learning Mishra and Bartram (5). To keep this report concise, we are limiting our interest only to the technical and teaching aspects of virtual education, although the proper placement of resultant courses into the institutional strategy is, at least, equally important. For example, the work Farrel (3) concentrates on such system level aspects of virtual education as new venues for learning, necessary organizational changes or accompanying quality assurance processes. Although the literature widely uses the term virtual education, there are few successful examples of purely virtual programs until now, i.e. programs where face-to-face contact with the teacher is completely excluded. Such fully non-personal type teaching is interesting, especially for earnings or numbers driven organizations, where the learner is considered primarily as a profit generating object. On the other hand, those authors and institutions, who are concerned with the quality of education, are increasingly recommending a balanced combination of virtual features with classes where the teacher is also actively involved Smith et al. (6); Koen (4); Dexter and Gurwitz (2). Although we belong to the latter group and, consequently, our requirements for the teaching framework, applicable both locally and internationally, were the following:

a. A single lecturer must be able to deliver the same course, during the same period, to students in several locations,

b. The quality of such education, subjectively evaluated by the course participants, must be comparable with that delivered by traditional teaching methods,

c. For course development and study, a standalone, standard PC must be enough.

For our cross-border project this meant that we were searching for a technology, satisfying the following needs:

- To run common virtual courses with Russian universities as a part of a future joint degree program, where not students, but teachers are travelling. Such a solution is cheaper than the current system of students’ scholarships and minimizes initial cultural tensions.

- To subcontract the content production as a service from the Russian academic sector, which solves the shortage of available experts in Finland.
• Does not require the direct employment of foreign staff and, again, is less expensive.

As an indirect consequence of such an approach to cross-border education we were expecting a package of high quality virtual courses also applicable for other institutional purposes (internal courses, adult education, transfer education, vocational training etc.). Financial resources, previously invested in mostly volatile and sensitive scholarships would, thus, contribute to the establishment of a more persistent entity - repository of virtual courses. We presented our very first ideas concerning this topic in Voracek (7), but at that time a deeper context for this model was missing. In this paper we want to disseminate our developers' and teachers' experience with several courses lectured as a part of IMPIT program Voracek et al. (8) at LUT during the years 2000 - 2003.

2 Applicable particular teaching techniques and arrangements

2.1 Real-time videolecturing

We have concentrated on the formal face of lectures since the year 2000. The initial and the most straightforward solution seemed to be to transfer all in-class lectures directly to the remote audience. A complete AI course, including 42 hours of lecturing, taught Finnish and foreign students in Lappeenranta and Joensuu using this technique in school year 2000/01. Although the majority of registered students passed this course, there were also multiple negative findings from this experiment:

• On-line videolecturing is technically expensive and a personally demanding solution, for which special equipment, fast and reliable connection or extra supportive staff are needed,

• Lecturer cannot concentrate just on the taught subject and local audience, but, on top of this, has to manipulate the complicated equipment and follow the quality and content of transmitted information. His/her contact with the distant learners is close to zero (Was everything I wanted to show taken by camera and broadcasted? Was it transmitted in a fair quality? And how does my lecture look on the remote side? Is it possible to establish a fruitful discussion with the outside students during such a videolecture? Being a regular student, would I personally profit from taking this kind of course?).

We concluded that the interactivity of such a class is limited, since not everything can be explained in the same way for local and distant students and so one group is suffering from the existence of the other. Also, the quality of lecturing, expressed by the level of personal satisfaction of students and teacher was not appropriate. That is why we decided to exclude real-time videolecturing from the "main development stream" totally. According to our opinion this technique is suitable only for prepared meetings among professionally close and rather small groups of participants.

2.2 Full content videos recorded in studio

Excluding the direct transfer of information, there remained only off-line techniques. Analyzing their properties, we found the following groups of limiting features:

Environmental features:

• Need for a special room, with good sound and lighting,

• Teacher must feel comfortable in a studio environment and have enough space, as well as, the right presentation equipment (blackboard, overhead projector, computer-based projection),
• Teacher should also pay special attention to his/her own visual appearance (color and pattern of clothes, way and speed of movements or gestures, regular eye contact with camera),

• Supportive staff are needed for recording the lectures and editing the video material.

**Technical features:**

• Properties related to target files (size, storage and transport media, bandwidth needed, streaming arrangements, software and hardware for manipulation with multimedia resources).

**Content features:**

• Readability of the lecturer’s written message (overall quality of handwriting, size of text taken by camera, clearness of information written on the previously cleaned part of blackboard, image saturations, shadows and contrasts given by the overhead projector),

• Quality of sound (background noise, way of sound recording if the speaker moves).

![Image](image.png)

**Figure 1:** Videolectures with the full content prerecorded in different environment. Overhead projector/Blackboard.

Many of the interfering features mentioned above do not meet our requirements for the teaching framework described in the introduction. Fig. 1 also show that the relevant information occupies only a small part of the image (typically not more than 50% of active visual field) and is relatively static. Moreover, discussions with involved students highlighted that the educational value of videolectures recorded in studio does not justify their expense.

This, in practice, meant that we had to search for another technique, minimizing the remaining negatives and making the content production more effective. One might argue that there is an alternative solution - to record a normal lecture - but this approach preserves the majority of outlined weaknesses and even combines them with the problems of real-time videolecturing.

### 2.3 Limited content videos recorded anywhere

The next step we made was towards off-line videolecturing with a limited content, as illustrated in Fig. 2. The only dynamic element there is teacher’s hand and the useful information covers a significantly larger space than previously. Thus, the environmental features were removed completely and the content features minimized considerably; the technical features, however, were left almost untouched. We used this technique in one regular course for about 100 students during school year 2001/02 and the message from students’ feedback was that there still existed some interfering factors (expenses vs. educational value ratio).
2.4 Electronic documents including audio files

Finally we decided to concentrate in particular on the technical features of the problem. At that time it was already evident that the next virtual course would be the international one. We were also negotiating possibilities of design of courses in Russia. The only feasible solution, the total replacement of videos with audio files, was adopted. This removed the technical features almost completely and still preserved the basic principles of our vision of distant education. As the first software development environment we used a freely available version of Real Presenter, which can add voice comments into the existing set of PowerPoint slides - as it is shown in Fig. 3 - and save the enhanced presentation in the streaming format for Real Player (RM, RMA).

After the students’ feedback from school year 2001/02 we changed the format and delivery of target multimedia from streaming Real Player sources to off-line available MP3 files. We also found that the exclusive use of Power Point and Real Presenter was limiting and decided to replace it with Acrobat Reader, so in the latest version we can add voice into all PDF documents. Such a solution also fulfills all the originally formulated requirements.

2.5 Auxiliary tools - feedback mechanisms

Experimental distant education is very sensitive to various misunderstandings, irregularities or just symptoms of normal human behavior like improper planning, unevenly distributed

Figure 2: Videolectures with limited content (context-free). Original format used in 2001/02. One example was spread on several pages, that is why the overall context was unclear and clarity of explanation degraded in case of larger examples. Next version with printed headings and larger screen size used in 2002/03 (the original picture is reduced - notice the font size in Real Player menu). Here we strictly followed the rule 1 example per 1 page.

Figure 3: Audiovisual lectures using different types of source media. PowerPoint and Real Presenter/PDF and MP3.
effort, problems with prioritization of tasks or changing professional focus. Moreover, distant students cannot approach their teacher with problems immediately and have either to record and postpone them or quit the course. They also have a natural tendency to solve the local study matters prior to those distant ones.

That is why some kind of tailored communication and feedback mechanism must be established, keeping the distributed community together and encouraging the distant learners to progress. Due to large amount of external students, the exchange of individual e-mails is impossible and the selected technique(s) must become inherent standard part of the virtual educational process. We finally designed the following three "bridges":

2.5.1 Personal records

Each student had a duty to mark every completed activity from the actual teaching block (listened lectures and exercises, done and submitted homework or projects) onto their own database-like record, accessible only for him/her and the teacher and the teacher can store all his/her own comments and grades onto this record as well. All entered items have deadlines, so the records provided enough quantitative information about a student’s current status.

2.5.2 Surveying

In order to collect general opinions from the whole group, a surveying tool with a statistical summary was implemented. It incorporates a web questionnaire and constantly updated resultant statistical information as publicly available web page. Such an arrangement allows the teacher to monitor the atmosphere, study moral, attitudes and trends among the students and also establishes a real "team feeling" across the course community. An interesting consequence of this survey was a regular comparison and a kind of competitiveness among students from different places. Sample survey collecting students’ final opinions is shown in Fig. 4 and 5.

![Survey Questions]

**Figure 4:** Final questionnaire and its statistical evaluation for AI course 2002/03. Input to the surveying tool.
2.5.3 Individually directed mail merging

In virtual international education a teacher must frequently pass numerous messages to distributed students. The content of these messages differs not only from group to group but even within single groups (classification results, optional study arrangements, personal consultation periods, project negotiations etc.). In such cases it is confusing and impolite to use mass posting and spread a similar, impersonal and, thus, confusing text among all participants.

Moreover, we believe that it is particularly important to approach individual students, who are studying by distance learning, to compensate for the lack of personal contact. That is why we were, for example, always using the addressee's first name in the message heading or distinguished gender inside the text. Also a prompt response to any kind of learner's request was a part of this strategy.

The whole task of exchanging messages is not as straightforward as it might look at a glance and its implementation required writing several data processing scripts as well as a special parametrization of standard software packages.

With the outlined techniques we succeeded to reduce "the other" meaning nonstandard communication (individual emails or personal meetings) significantly. From students' responses it was evident that they appreciated our attitude and considered it one of the main driving forces.

3 Overall structure of virtual course for cross-border education

Teaching in the final version of our semi-virtual model was designed as block-based with all teaching arrangements, deadlines and deliverables known in advance. This means that students can plan all activities from the very beginning and work on the required tasks continuously. Teacher is personally available regularly (although not continuously) to specify all unclear matters from lecturing materials as well as discuss the content of exercises, home assignments or quizzes. These face-to-face discussions, which typically took place at the end of each teaching block, were unlimited by time and topic. For example in the artificial intelligence (AI) course taught in spring 2002/03 to about 300 students from 4 universities in Finland and Russia, we divided the whole teaching into 5 blocks, the discussion meetings were every third week and each took from 3 to 6 hours depending on the group sizes, which varied from 5 to 50 students.

Concerning its internal structure, our final model of a semi-virtual course applicable to Russian conditions includes the following entities:

a) Audiovisual lectures: several per block, individual listening,

<table>
<thead>
<tr>
<th>Variable</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Deviation</th>
<th>Average</th>
<th>Median</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>students_involvement</td>
<td>1</td>
<td>3</td>
<td>0.53</td>
<td>1.45</td>
<td>1</td>
<td>67</td>
</tr>
<tr>
<td>English_level</td>
<td>1</td>
<td>4</td>
<td>0.98</td>
<td>2.09</td>
<td>2</td>
<td>67</td>
</tr>
<tr>
<td>English_before</td>
<td>1</td>
<td>2</td>
<td>0.37</td>
<td>1.16</td>
<td>1</td>
<td>67</td>
</tr>
<tr>
<td>English_accept</td>
<td>1</td>
<td>2</td>
<td>0.17</td>
<td>1.03</td>
<td>1</td>
<td>67</td>
</tr>
<tr>
<td>web_learning_before</td>
<td>1</td>
<td>2</td>
<td>0.45</td>
<td>1.28</td>
<td>1</td>
<td>67</td>
</tr>
<tr>
<td>adequate_knowledge</td>
<td>1</td>
<td>2</td>
<td>0.12</td>
<td>1.01</td>
<td>1</td>
<td>67</td>
</tr>
<tr>
<td>interested_in</td>
<td>1</td>
<td>2</td>
<td>0.24</td>
<td>1.06</td>
<td>1</td>
<td>67</td>
</tr>
<tr>
<td>reading_or_listening</td>
<td>1</td>
<td>2</td>
<td>0.49</td>
<td>1.39</td>
<td>1</td>
<td>67</td>
</tr>
<tr>
<td>web_or_norns</td>
<td>1</td>
<td>3</td>
<td>0.69</td>
<td>1.63</td>
<td>2</td>
<td>67</td>
</tr>
<tr>
<td>homeworks_useful</td>
<td>1</td>
<td>3</td>
<td>0.57</td>
<td>1.36</td>
<td>1</td>
<td>67</td>
</tr>
<tr>
<td>need_to_hear</td>
<td>1</td>
<td>3</td>
<td>0.62</td>
<td>1.99</td>
<td>2</td>
<td>67</td>
</tr>
<tr>
<td>blocks_and_deadlines</td>
<td>1</td>
<td>3</td>
<td>0.66</td>
<td>1.34</td>
<td>1</td>
<td>67</td>
</tr>
<tr>
<td>enough_materials</td>
<td>1</td>
<td>2</td>
<td>0.42</td>
<td>1.22</td>
<td>1</td>
<td>67</td>
</tr>
</tbody>
</table>

Figure 5: Statistical summary for the whole group.
b) Audiovisual (numerical) exercises: several per block, individual listening and solution,
c) Quizzes: one per block, individual solution,
d) Homework: one per block, individual solution,
e) Classes with teacher: one per block, intended for overall communication,
f) Project: one for the course, individual or team work,
g) Projects presentation and discussion session: one for the course, presentation of projects by single authors or teams, overall communication and dissemination of experience and good practices,
h) Exam: one for the course.

The final classification incorporated both the overall working morale of students and the level of their professional skills. The significance of the exam itself was limited up to 30% from the full scale where the lowest limit for passing the course was 50%. This means, that even a student with an excellent exam result, but having completed nothing else from the above, was finally unsuccessful.

4 Conclusions

The presented framework could serve as an example of how to design and run international virtual courses in computer science. Our solution is, however, very specific due to its close orientation towards Russia and, consequently, its direct comparison with already existing techniques is difficult. We did not find any examples of regular courses, established between western and Russian universities, which are standard curricular items for all participating institutions.

On the other hand, the majority of information, published about various shorter courses, fully virtual international classes without personal interaction, on-line videolecturing to international students, multicultural project-based learning or initiatives established between well-developed universities (e.g. EU or US) represent fundamentally different approaches so we could not use them for a direct comparison as well. This lack of comparative platforms does not concern only the teaching itself, but holds also for the process of building of the course materials.

We are naturally aware of many questionable points, which we are proposing here - for example the role and technology of lecturing is different than usually - but we were trying to compensate these with intensive - although only periodic - personal interaction and continuous exchange of information among stakeholders.

Overall, a positive feedback was provided by many of participants during a three-year long research period. A total absence of negative statements and the study results of several hundred participants in our single semi-virtual course indicate that this direction is promising. On top of this subjective, qualitative information we collected also encouraging quantitative results concerning, for example, a comparison of students' final classification in similar traditionally and semi-virtually lectured courses, differences between local and distant students or some statistically significant nationality dependent features. Their detailed presentation and analysis is out of the scope of this paper and we are planning to publish them later.

Purposely we are also not dealing with the related expenses or workload matters in this paper, because our main goal was just to investigate if there exists a solution satisfying the requirements and, possibly, prove its functionality. If a promising prototype is finally available, there are always reasons for its further analysis and optimization.
References


Virre - Virtual Reflecting Tool

P.J. Eronen, I. Jormanainen and M. Virnes
Department of Computer Science
University of Joensuu
P.O. Box 111
FIN-80101 Joensuu
FINLAND

{peronen, ijorma, mvirnes}@cs.joensuu.fi

Abstract

Virtual Reflecting Tool (Virre) was developed to support children’s reflection process and tutors’ data gathering in Kids’ Club technology club. Returning back to learning process by reflecting strengthens one’s self-guidance and learning. This is realized in Kids’ Club and with school students by using Virre. According to Kids’ Club tutors’ experiences, Virre is the tool that gives motivation to children to answer to questions, which benefits children’s learning and collection of research data. From research point of view Virre has made the contents of children’s answers richer and their thinking processes more explicit. Virre is at the moment in the development phase and new versions with improved functionality will appear later.

1 Introduction

Virre, which stands for Virtual Reflecting Tool, is a tool to support children’s learning in its reflection phase. This is achieved by allowing children to use their natural means for communication, that is oral verbalization. The idea about a virtual reflecting tool aroused when tutors and researchers participating in the Kids’ Club project noticed difficulties in getting useful information from the children through web form questionnaires. Neither were Kids’ Club people satisfied with the depth of the reflection process among the children in the club.

Kids’ Club, where Virre was developed, is a bi-directional research laboratory where children of age 10 to 15 work in collaboration with university students and researchers, apply and create novel information and communication technologies (ICT) for learning. To children Kids’ Club appears as a technology club where they can learn technology skills and work playfully and collaboratively with tasks that they are interested in (Eronen et al. (2), Eronen et al. (1)). Kids’ Club can also be seen as an idea generator, which creates new research questions and solutions based on requirements found in Kids’ Club.

At first, Kids’ Club tutors and researchers realized data gathering about children’s experiences and learning in the club activities by using the web form. The web form was introduced on spring 2002 to offer both tutors and researchers with easy access to children's answers to reflection form. It was also considered that storing the answers with help of the web form would facilitate their future use. After each Kids’ Club meeting, children had about twenty minutes time to answer to the questions on the web form concerning for example what they had done and learned during the club sessions, what kind of problems they had found, how they had solved those problems and how would they like to develop their robot next. Besides data collection, the questions were meant to children for reflecting their learning aiming to increase their self-direction skills and improve their learning results.

However, children’s reflection with the web form did not give expected information to researchers. Children’s different skills and interest to produce text in written format could be seen in the answers and in given feedback during writing. Most of the answers were very short and the content of answers was insufficient both for research and reflecting. Therefore, the reflection process with the web form was proven to be unsuitable for the children and providing little or no information for the research use. Thus, Virre was developed for needs of
Kids' Club by Kids' Club tutors and researchers on the fall 2002. Implementation was realized by using Visual Basic for Virre application, web-camera, microphone and video program.

2 Usage of Virre

In this chapter we will describe how Virre works. There are two different kinds of users in Virre. User is the one, for example a pupil, who reflect his or her learning with Virre and administrator, who prepares question sets. The administrator can be for example a teacher or a researcher. Fig. 1 presents interaction between user, administrator and application.

![Interaction between user, administrator and Virre.](image)

**Figure 1**: Interaction between user, administrator and Virre.

2.1 User

The user begins her reflecting by pressing the button in the user interface of the application. Virre plays a question and shows it at the same time as text to the user. The user answers to the question and continues to the next question by pressing the button again. User sees always his or her own picture on the display located in the center of the application. Fig. 2 presents the user interface.

![User interface for reflecting.](image)

**Figure 2**: User interface for reflecting.
2.2 Administrator

The administrator prepares the application for the use by making question sets for the users. Administrator menu can be obtained by pressing the right button of the mouse. The menu contains items for controlling the camera and preparing the question set. There are also items for setting dialog of the application. Administrator prepares a new question set by launching the dialog presented in Fig. 3. The administrator selects questions for the question set from the list, which shows all the questions available. Selecting another item from a menu above the list can change category of the questions. From this dialog it is also possible to set a description and a path where answer files will be saved.

![Figure 3: Dialog for creating a new question set.](image)

3 Implementation

In this chapter we will describe technical features of Virre. The basic structure of the system will be shown. Also hardware and software needed by Virre are described.

The whole system of Virre contains four pieces. Virre application is used to control the system and it takes care of video compression. A web camera is used for video capture. A microphone is used to capture sound and it can be either an external one or built-in microphone for example when the laptop is used. The fourth piece is an external video program used for playing the captured video. Basically, any video player is suitable if it can handle Indeo 5-compressed video.

Virre uses web camera for recording the video file. At the moment, only some models of Logitech’s cameras are supported. Virre uses at the moment vportal2.dll component which is an easy-to-use COM-object for Logitech’s camera (4). For future releases, there will be possibility to use basically any camera which supports Video for Windows API or DirectX (5).

4 Virre in Research and Education

Virre has two principal purposes that are a tool for research and a tool for education. In this chapter we describe usage of Virre from the research and education points of views in Kids’ Club context.
4.1 Tool for Research

As a tool for research, Virre is found to be suitable especially for data gathering purposes. According to experiences in Kids' Club, data that is gathered by Virre is richer in content than data collected from web form and gives possibility to gather also non-verbal information that is user's mimics.

Current version of Virre saves each child's reflecting session to one file. From research point of view, it would be meaningful give to a researcher possibility to save an answer of each question to separate files, especially if answers are wanted to be compared to other answers. Also possibility to add comments to a record would be an useful feature. From a standpoint of data handling Virre takes more time because of the need of the transcription of answers for data analyzing purposes. However, data that is collected by Virre is richer and give contextually more information than data from the web form.

Kids' Club tutors do not need to be along in the reflecting process, which make usage of Virre flexible during Kids' Club meeting. According to children's feedback, Virre is an interesting tool increasing children's interest to use it and through this adds motivation to answer the questions. It also seems to be suitable tool for most of children despite of different verbal skills.

Children can speak to an anonymous object, which can make them feel more free to answer the question when compared to a situation where interview is done with tutor. Naturally, children are aware that their answers are recorded and can be used for research, which is important especially from the ethics point of view. The user interface of Virre was created to appear simple as possible for the users, which means that they do not need to pay attention to usage of the usage itself but only to answers to the presented questions.

Both personal reflecting and reflecting with the pair are possible by Virre, which gives to researchers possibility to observe also non-verbal interaction and roles in a group work. For instance, when a child has more active role or role of a leader in a group, the role can be seen also in reflection. In an opposite situation more passive child consents his or her pair in reflecting. In addition to verbal answer, children's reactions to questions and expressions are shown in recording. Thinking processes and reactions were difficult to observe from web forms' answers since they presented mostly final solutions. Answers recorded by Virre make child's thinking process explicit when she thinks aloud when forming an answer to the question.

4.2 Tool for Education

From the standpoint of education Virre is a tool for learning and teaching. Pedagogical standpoints of Kids' Club emphasize constructionism, problem-based learning and group processes, where self-direction and reflection are essential issues (2). For reaching self-directed learning children need to understand is being main part in his or her learning process. Understanding goes through observing own cognitive processes and self-assessment (3). In this reflection process Virre works as a supportive tool.

Virre makes children to think about their learning and plan their work after each Kids' Club meeting, which probably would not otherwise actively happen. This can be seen as the first step of reflecting, but children should be able to return back to their thoughts and plans and get feedback from tutors as the second step. Kids' Club tutors task is not teach, but instruct in working and help in problem situations. Tutors' instruction could be realized for instance as discussions between tutor and children or with Kids' Club group. Some tips could be added also to Virre and also availability of tutor's comments on children's reflections could be helpful in this.

The usage of Virre is not tied to time, thus children can use it independently in different phases of the working process. Virre can serve as a data-collecting tool for various purposes to administrators, that is tutors and teachers, according to recorded questions. It can be used for instance for assessment and following children's learning processes, when assessment do
not reach only the final product but whole process from several dimensions that is difficulties and group dynamics.

4.3 Creativity

One dimension of Virre is related to creativity that has been realized as unforeseen questions during reflecting session. Solution was found when Kids’ Club tutors realized that children stick easily to same solution models as in their earlier projects. Tutors wanted to bring new ideas to children’s working processes the way that children find ideas themselves. As an alternative method for brainstorming or discussions in a group, the brainstorming was realized in Virre by using distant thinking models (6). Tutors also wanted to disconnect children from their routines to answer to the questions and help them to detach from typical ways of thinking.

There are some questions in Virre’s basic question set that are meant to shake children’s thoughts and through that to find novel ideas and approaches to working processes. Question may be for example ‘Are cake receipt and computer program related to each other? Give also reasons.’ or ‘What happens if a robot builds and programs another robot?’ One purpose of questions is to arouse interest to find for example connection between cake receipt and algorithm. In the best case, new solutions or ideas can be found from unforeseen questions and also transfer the skill to other situations, for example Kids’ Club project.

5 Future Plans

The first version of Virre is a prototype with limited set of features. However, many ideas for future releases have been raised and the following features will be added:

- Any web camera could be used with Virre. With this feature Virre would be more universal and more easily adapted to any system available using different camera than Logitech’s.

- Each answer could be saved to its own file. This feature is especially important for the latter use of the answers. By saving answers to separate files, it is possible to take for example any two answers for the same question by different users and compare them to each other.

- Administrator can record questions with Virre. At the moment the administrator needs to have a separate program for saving the .wav format questions. It is too complicated for non-professional user.

- Support for non-linear question sets. Here the idea is to present the idea of flowchart to question sets, which would help the administrator to create more adaptive question sets responding better to different kinds of users.

- Support for adding comments to answer files. In this feature idea is to support qualitative research work. A researcher can make her comments on user’s answers and save them with the answer itself. It also helps collaboration between several researchers working on same material.

- XML-based project file. With this feature all the files, which are belonging to the same project are collected into one project file. It would include metainformation (that is for example comments) over all the files. It serves also the purpose of sharing research material among several researchers.

- No need to the external video program. This feature will be one of the first, which will be realized in future versions of Virre. In this feature the video player is an embedded
component, which will diminish the need for several different programs to be ran at the same time with Virre session.

6 Conclusions

Virre has proved to be a useful tool for research in Kids’ Club environment. Changing the question set Virre can be tailored for various purposes both in research and education. During the fall 2003 Virre will be used as a reflecting and data gathering tool among school and Kids’ Club children visiting our department’s premises and features especially for administration use will be further developed. After development work Virre will become a helpful tool for a teacher in her work with children or a researcher gathering qualitative research material from children.

References


URL http://developer.logitech.com

URL http://msdn.microsoft.com/

TRAKLA2: a framework for automatically assessed visual algorithm simulation exercises

A. Korhonen, L. Malmi and P. Silvasti
Helsinki University of Technology
Department of Computer Science and Engineering
PO Box 5400, 02015 HUT
{archie,lma,psilvast}@cs.hut.fi

Abstract

Interaction and feedback are key factors supporting the learning process. Therefore many automatic assessment and feedback systems have been developed for computer science courses during the past decade. In this paper we present a new framework, TRAKLA2, for building interactive algorithm simulation exercise applets. In these exercises students directly manipulate conceptual visualizations of data structures in order to simulate the working of given algorithms. The framework supports randomized input values for the assignments, and automatic feedback and grading of students’ simulation sequences. Moreover, it supports straightforward generation of model solutions as algorithm animations and logging statistical interaction data about the process of solving exercises. The system has been used in Helsinki University of Technology on a course with some 600 students, and the student response was very positive.

1 Introduction

Interaction is one of the key issues in computer based learning environments. It is used for many different purposes, among which some of the most important are the following. First, the environment can support various forms of collaboration among learners and teachers, such as chatting, email or discussion groups. Second, the environment can include active learning modules in the form of interactive applets that visualize or animate important concepts. A good example of this is the Snapshots project by (11). Third, the user can solve exercises and receive constant feedback from the system in terms of simulation and automatic assessment. In this paper we concentrate on this last form of interaction, and we emphasize that in the exercises the user actually manipulates data structures instead of viewing ready made visualizations possibly accompanied with questions. Our application is a web-based learning environment for teaching data structures and algorithms.

Automatic assessment systems have been developed over the last decade to aid grading of exercises on mass courses, especially in programming (1; 6; 12; 14). Other applications include grading algorithmic exercises (2; 5; 7) and analyzing object-oriented designs and flowcharts (4). The main purpose of the systems has been to reduce the workload of human teachers. However, there are other very important issues involved. First, the learners have the freedom to submit the exercises at any place and time, a feature of great value in distance learning, but also for ordinary learners. Second, most systems allow resubmission of exercises after the feedback. This option can promote learning considerably, since it supports independent self study and allows the students to learn from their errors without waiting for the feedback from the teacher (10). Third, automatic assessment allows personalization of the exercises (5; 7) that would be impractical on large courses if the teacher marks the exercises manually.

At Helsinki University of Technology (HUT) we have developed software for supporting the data structures and algorithms course since early 1990’s, when the first version of the TRAKLA system (5) was built. The system checked answers to algorithmic exercises and communicated with learners by email. The fundamental idea of the exercises was to present the changes in the states of the given data structures, and thus simulate the execution of an algorithm (a method called algorithm simulation). Instead of concentrating on implementation issues, this method concentrated on promoting conceptual understanding about the algorithms
involved. Each exercise was individualized for each learner by randomizing the initial values of the algorithm. The answer consisted of a sequence of data structure states that had to be coded into a predefined format. Initially, the email based submissions required textual formats, which were impractical and error-prone to use. Therefore a graphical user interface for constructing the answers was developed. This user interface was also integrated to course web pages, and thus formed our first web-based learning environment, the WWW-TRAKLA (7).

WWW-TRAKLA was our first system that brought algorithm visualization technology from theory into practice in teaching data structures and algorithms at HUT. Graphical editing of the algorithm exercises was a huge improvement compared to text-based editing and the feedback from the students was encouraging. However, the feedback of the exercises was still sent to the learners by email in textual form that was not very informative. Moreover, implementing new exercises was time consuming and required a lot of programming and testing. Furthermore, even though the learners created the answers graphically in terms of algorithm simulation, and were allowed to run the exercises step by step backwards and forwards, the model answer generated by the system was solely in textual form. Actually, it typically included only the final state of the data structure in question.

Due to these limitations, WWW-TRAKLA could not support the full power of visual algorithm simulation, and therefore a new framework, called Matrix was developed (8). Matrix is a general purpose framework for building applications in the field of algorithm animation and simulation. The framework allows building applications, in which the user has full control of the data structures through GUI interaction in terms of direct manipulation. Operations in an application program are carried out by invoking context sensitive drag-and-drop operations, triggering menu commands, and pressing push buttons. Thus, the user can modify, for example, the contents and relations of nodes, activate operations on abstract data types, and use different visual representations for a single data structure.

In this paper, we present the TRAKLA2 framework for straightforward creation and publishing of interactive exercises for data structures and algorithms. TRAKLA2 is based on Matrix and allows its full power of building different types of visual algorithm simulation exercises that employs direct manipulation. The new framework supports automatic assessment, and provides a well defined schema for programming new exercises. Our experience shows that the new exercises can be build within the same time and effort than with the old system. However, the new exercises are fully visual including visual algorithm simulation functionality and visual model solutions. This is new compared to the old text based system. Moreover, TRAKLA2 provides automatic logging of the data generated by the user interaction and submissions (13). Thus, gathering data from the exercise sessions for research purposes is easy.

Some other systems also apply visual algorithm simulation in exercises such as PILOT (2) and SALA (3). However, neither of them include the automatic assessment in terms of grading and storing submitted answers. Thus, these systems are more suitable for formative than summative assessment. For example, PILOT allows stepwise execution of graph algorithms, but provides feedback only on a single step at a time, and does not log nor grade learner’s work. TRAKLA2, on the other hand, supports both summative and formative assessment.

In the following sections we describe the TRAKLA2 framework, its architecture and the process of creating new exercises. We also present some observations on TRAKLA2 exercises that were used in our data structures and algorithms course with approximately 600 enrolled students in 2003.

2 Overview of TRAKLA2

TRAKLA2 is a framework for automatically assessed visual algorithm simulation exercises (9). Not only does the system provide a Java applet that portrays a variety of algorithms and data structures, but it also distributes individually tailored tracing exercises and then automatically
evaluates the learner’s answer to the exercise. The learner simulates the algorithm by directly manipulating the visual representations of the data structures the algorithm employs. The purpose of a tracing exercise is that the learner changes the data structures as the algorithm would do. Thus, the answer is a sequence of discrete states of the structures and the task is to determine the correct operations that cause these changes.

Let us consider the exercise in Figure 1. The learner has manipulated the visual representation of the data structure by invoking context-sensitive drag-and-drop operations. In the next step, for example, he or she can drag the key C from Stream of keys into the left subtree of R in the binary heap. After that, the new key is sifted up via a swap with its parent until the parental dominance requirement is satisfied (the key at each node is smaller than or equal to the keys at its children). The swap operation is performed by dragging and dropping a key in the heap on top of another key. In addition, the exercise applet includes a push button for activating the Delete operation. The Delete button is applied in the second phase of the exercise to simulate the deleteMin operation. Of course, there are several correct ways to heapify the tree while inserting or deleting a key, thus the drag-and-drop operations can also be targeted to “empty keys”, and the deleteMin operation does not have to start with removing the root node (but swapping it with the last node, in which case the delete should be targeted to the key at the last node). Thus, the delete operation is performed by selecting the target node before pressing the corresponding push button.

The exercise applet is initialized with proper randomized input data. The binary heap exercise, for example, is initialized with 15 alphabet keys (Stream of keys), that do not contain duplicates. This means that the exercise can be initialized in more than $10^{19}$ different ways. The learner can reset the exercise by pressing the Reset button at any time. As a result the exercise is reinitialized with new random keys.

After solving the exercise, the learner can review the answer step by step using the Backward and Forward buttons. Moreover, the learner can ask feedback on his or her solution by pressing the Grade button, in which case the learner’s answer is checked and the immediate feedback is delivered. The feedback reports the number of correct steps out of the required steps in the exercise. Figure 2 shows an example of the feedback report. After the exercise is solved, it is possible to submit the answer to the course database using the Submit button. Note that the exercise can be solved unlimited times. However, if grading has been requested, but the answer was not submitted, the answer cannot be graded anymore. Instead, the learner has to reset the exercise and start over with new randomized input data. Another option is to set up exercises with a limited number of submissions. After the learner requests grading, he or she has to continue with the same data.

A highly important feature is that the learner can examine the model solution of an exercise. It is represented as an algorithm animation so that the execution of the algorithm is visualized step by step. In Figure 1, the model solution window is opened. The states of the model solution can be browsed using the Backward and Forward buttons. Obviously, after opening the model solution, the exercise cannot be submitted until it has been reset. Moreover, if the number of submissions has been limited and the user continues operating on the same data, the model solution should be disabled before submission deadline is over.

TRAKLA2 was brought into production use in spring 2003. The learning material is organized around exercises, and each exercise is described in one page. Each TRAKLA2 exercise page (e.g., Fig. 1) consists of a description of the exercise, an interactive Java applet, and links to other pages, which introduce the theory and examples of the algorithm in question. Table 1 lists the current set of exercises in TRAKLA2.

3 Frameworks

TRAKLA2 is based on the Matrix algorithm visualization, animation, and simulation framework (8). Matrix provides the following reusable fundamental data types (FDT) to be used
5 Heap operations (4 points)

First, insert the Stream of Keys one by one into the Heap below. Second, after the insertions, perform three deleteMin operations.

Insert a key by dragging and dropping it into an empty node of the heap. Delete a key by clicking it and by pressing the "delete" button. After each operation make sure that the heap-order property "the parent is smaller than its child" is preserved.

It should be noted, however, that drag and drop will swap the source and destination keys with each other.

Figure 1: TRAKLA2 applet page and the model solution window.

Figure 2: TRAKLA2 feedback window. Here, the exercise can be submitted or one can cancel and try to solve the exercise again with new input data.
Table 1: The visual algorithm simulation exercises in TRAKLA2 system. The column name describes the topic and the description characterizes the exercise. The roman numbers (i-iv) indicate the separate exercises and the number of sub-topics.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insertion into (i) Binary search tree, (ii) Digital search tree, and (iii) Radix search trie</td>
<td>The learner is to insert random keys into an initially empty search tree by dragging and dropping them into the correct positions.</td>
</tr>
<tr>
<td>Binary search tree deletion</td>
<td>The learner is to remove 4 keys from a binary search tree of 15 to 20 keys. Pointer operations are simulated by directly manipulating the edges that connect the nodes of the tree in the visualization.</td>
</tr>
<tr>
<td>Faulty Binary Search Tree</td>
<td>The learner is to show how to bring the following binary search tree in an inconsistent state: duplicates are allowed and inserted into the left branch of an equal key, but the deletion of a non-leaf node relabels the node as its successor.</td>
</tr>
<tr>
<td>AVL tree (i) insertion, (ii) single rotation, and (iii) double rotation</td>
<td>The learner is to (i) insert 13 random keys into an initially empty AVL-tree. The tree (i-iii) has to be balanced by rotations. The rotation exercises (ii-iii) require pointer manipulation, while the insertion exercise (i) provides push buttons to perform the proper rotation at the selected node.</td>
</tr>
<tr>
<td>Red-black-tree insertion</td>
<td>The learner is to insert 10 random keys into an initially empty Red-Black-tree. The color of the nodes must be updated and the tree must be balanced by rotations.</td>
</tr>
<tr>
<td>BuildHeap algorithm</td>
<td>The learner is to simulate the linear time buildheap algorithm on 15 random keys.</td>
</tr>
<tr>
<td>Binary heap insertion and delete min</td>
<td>The learner is to (a) insert 15 random keys into a binary heap and (b) perform three deleteMin operations while preserving the heap order property (see Fig. 1).</td>
</tr>
<tr>
<td>Sequential search: (i) Binary search, and (ii) Interpolation search</td>
<td>The task is to show which indices the algorithm visits in the given array of 30 keys by indicating the corresponding keys.</td>
</tr>
<tr>
<td>Tree traversal algorithms: (i) preorder, (ii) inorder, (iii) postorder, and (iv) level order</td>
<td>The learner is to show which keys in a tree the algorithm visits by indicating the visited keys in the required order.</td>
</tr>
<tr>
<td>Preorder tree traversal with stack</td>
<td>The learner is to simulate how the stack grows and shrinks during the execution of the preorder algorithm on a given binary tree.</td>
</tr>
<tr>
<td>Graph algorithms: (i) Depth First Search, and (ii) Breath First Search</td>
<td>The learner is to visit the nodes in the given graph in DFS, and BFS order.</td>
</tr>
<tr>
<td>Open addressing methods for hash tables: (i) linear probing, (ii) quadratic probing, and (iii) double hashing</td>
<td>The learner is to hash a set of keys (10-17) into the hash table of size 19.</td>
</tr>
</tbody>
</table>
in the algorithm simulation exercises: arrays, linked lists, common trees, binary trees, and graphs. However, the Matrix framework also provides an extensive library that includes ready-made implementations for many basic data types, search trees, priority queues, and so on. Such conceptual data types (CDT) can be applied to provide more functionality to the simulation process. The environment is not limited to simple operations (e.g., changing keys or manipulating pointers), but allows also ways to implement more abstract operations, such as insertion into an AVL tree or deletion from a stack with single drag-and-drop operation. However, all CDTs have been implemented by reusing the FDTs. Thus, raising the level of abstraction requires no additional work what comes to the visualization, animation, or simulation. Moreover, for all reusable FDTs above (and thus also for the CDTs), there exist visual counterparts that enable the visualization and animation of the structure, as well as the simulation functionality. We refer to these visualizations as representations.

TRAKLA2 heavily employs the different kinds of data structures and their visualizations implemented in Matrix. While creating a new exercise, one can choose to (re)use any combination of the Matrix structures. For example, the exercise in Figure 1 applies an array to hold the keys to be inserted into the heap, and a binary tree to implement the binary heap. Moreover, the layout for the representation can be adjusted to better suit the exercise. For example, one can choose the layout for the binary heap between an array representation and the binary tree representation depicted in the figure. Furthermore, both representations can be visible together, if required.

3.1 Creating new exercises

The process of creating a new exercise starts by writing a textual description, a manuscript, of the exercise. Based on the manuscript, the programmer is able to implement the exercise in Java using the features and components of the TRAKLA2 framework. Each exercise has a corresponding Java class, which defines the elements of the exercise by implementing several Java interfaces.

In general, TRAKLA2 exercises are defined by determining the following elements: 1) data structure types employed in the exercise (e.g., array, binary tree), 2) names for the visual representations (e.g., “Stream of keys”, “Heap”), 3) the layout for each visual representation, 4) the layout for the representations in the user interface, 5) methods to create randomized initial values for data structures, 6) an algorithm to create the model solution for the problem instance with the specified initial values, 7) push buttons for the exercise, if required, and 8) allowed/disabled user interface operations for a representation (e.g., is it possible to drag and drop keys into the structure). Each exercise is implemented as a Java class that minimally implements methods for the items 1, 2, 4, 5, and 6.

Each exercise has randomized input values. Although our example exercise can be initialized with any set of keys, an exercise may require, in general, that the input values must be tailored by a particular initialization algorithm. We call this process validation. For example, the validation of the AVL tree insertion exercise (described in Table 1) should allow no duplicates, and at least one simple rotation and one double rotation should take place in the solution.

An implementation for the algorithm in question, however, is essential for two reasons: to check the learner’s answer, and to produce the model solution as an algorithm animation. Contrary to the old TRAKLA system, there is no need to implement a checking algorithm that evaluates the exercise. This is because the current TRAKLA2 evaluation is based on running the implemented algorithm and comparing the resulting data structures step by step with the simulation sequence generated by the learner.

Drag-and-drop operations, however, may not be suitable for all kinds of data structure manipulations. It is difficult, for example, to find an intuitive way to change the color of a node in a red-black tree using only drag-and-drop operation. Thus, such operations are
performed by using push buttons. In a red-black tree, for instance, the color of a node is changed by selecting the node by a mouse click and then pushing the Toggle color button.

The Matrix framework allows various user interface operations on the representations. Therefore, it is important to define which operations are enabled and which are disabled for each structure in a TRAKLA2 exercise. For example, in Figure 1, the task of the learner is to drag and drop keys from the array representing the stream of keys into the binary heap. Therefore, dragging keys from the table must be enabled, while dropping a key into the table must be disabled. The operations that can be enabled for each visual representation includes enable/disable dragging a component, enable/disable dropping a component, and enable/disable highlighting a component while the cursor is moved over it.

Finally, the TRAKLA2 exercise framework provides several functions that the exercise programmer does not have to know about. For example, if the environment is applied to deliver compulsory exercises that are also marked, there should be an infrastructure for delivering and submitting an answer as well as collecting data on learner’s performance. Moreover, the exercise can be taken into production use by embedding it into a web page using a Java applet provided with the framework. These issues are discussed further in Section 3.2.

3.2 Integration to Learning Environment

All TRAKLA2 exercises are presented within an applet that is merely a user interface for an algorithmic exercise. In practice, the applet must be embedded into a web page, which should contain the actual description of the assignment. The surrounding web-based environment also takes care of authenticating learners and selecting exercises. The relevant information about the user and the selected exercise is passed on to the applet by setting up the applet parameters. The parameters define the name of the exercise (Java) class, the course name and the exercise number, the learner’s user id, and the IP address of the TRAKLA2 server.

The TRAKLA2 applet connects to the server when it is initialized and the random seed to create the randomized input data for the exercise is created at the server side. The server also takes care of the bookkeeping of learners’ grades and submitted answers. The communication between the applet and the server is carried out over Java Remote Method Invocation (RMI) protocol.

The server also logs data of the user interface operations the learner performs in the applet. A log entry is saved each time the applet is initialized, the model answer window is opened or closed, and the exercise is graded or reset. Each log entry contains a timestamp, course id, exercise id, learner’s userid, and the name of the operation performed. The information about how the model solution is used (which states were visited and how many times) is also logged. At each grade operation, a snapshot dump of learner’s answer, which is a sequence of data structure states, is saved. Log entries are saved into a text file and learners’ answers are saved as serialized Java objects.

4 Experiences and Discussion

TRAKLA2 exercises were used for the first time in the basic data structures and algorithms courses at HUT in spring 2003. There were two versions of the course, one for the CS majors and one for students of other engineering curricula. The TRAKLA2 exercises were a gradable and compulsory part of both courses. In total, some 600 students participated the courses and there were 14 different TRAKLA2 exercises, 8 for the CS majors course and 6 for the other course. The system worked well, except that two technical problems arose.

First, the TRAKLA2 applet requires a Java 2 compatible browser, and it appeared that many popular browsers did not support Java 2 well enough. However, even though all students were not able to get TRAKLA2 working in their home computers, they were able to use the system in the computer classrooms at HUT, in which TRAKLA2 was tested to work well. Second, the communication between the TRAKLA2 applet and the server did not work if
there was a firewall blocking traffic of certain TCP ports at the client side. This problem appeared only in case the student was trying to use TRAKLA2 at his or her work. The security policy of a company firewall is usually stricter than, for example, at universities, libraries, and homes.

Students' opinions of the system were asked by using a web-based survey at the end of the course. 364 students answered. The attitude to the system was very positive: 94% of the students thought that the system was very easy to use. In addition, they held the system as a good learning aid: only 1% thought that the feedback provided by TRAKLA2 was not useful, and only 3% did not get any help from the model solutions. Moreover, 96% of the students thought that the visual representations of the data structures were fairly logical or very logical. Finally, 80% gave the overall grade 4 or 5 to the system in scale 0–5.

As a whole, we argue that the new concept worked well. Students' response was positive. Moreover, the implementation work of the exercises using the new framework was quite straightforward. Although the exercises include visual manipulation of complex data structures, allow automatic assessment, and provide model solutions as interactive algorithm animations, the work required to implement a new exercise was typically 2–3 workdays for a competent Java programmer. Furthermore, the programmer needs to have no deep knowledge about the Matrix framework. Knowing the basic concepts and interfaces is enough. After the test course, we have implemented over ten new exercises on various topics such as basic graph algorithms and hashing methods. It seems to be that the most challenging part is to design a good manuscripts for an exercise and make the user interface intuitive for the learner, and not the actual implementation of the exercise.

The data logging feature that is invisible for the exercise programmer, seems promising. We can now gather detailed information about the use of the applet and the model solutions, together how much time a student spends on an exercise. This can be used to test or verify hypothesis about the difficultiness of exercises, and to find out which phases in the algorithms students find the most difficult. This allows us to direct the education better to the actual needs.

5 Acknowledgements

We thank Ville Karavirta, Pekka Mård, Jussi Nikander, Harri Salonen and Petri Tenhunen for designing and implementing major parts of the TRAKLA2 framework and the exercises, which were used on HUT courses in spring 2003.

References


Problem-based learning of theoretical computer science

W. Hämäläinen
Department of Computer Science, P.O. Box 111, FIN-80101 Joensuu, Finland
whamalai@cs.joensuu.fi

Abstract

Problem-based learning has become successful in the education of medical science and it has been applied also in higher education of computer science. The problem-based learning has been considered especially suitable for learning programming and software engineering, but as far as we know, it has not been tried in the more theoretical courses of computer science. This paper reports a successful experiment in learning theory of computability in problem-based way in Department of Computer Science, University of Joensuu.

1 Introduction

The theory of computer science, especially the theory of computability and computational complexity are important but difficult part of computer science studies. The courses are quite theoretical and require mathematical skills. Students usually find these courses very difficult and interrupting the courses is a serious problem. However these courses are central for overall understanding of computer science and offer several important skills for further studies and work. In fact the whole computer science deals with computation and an educated computerscientist needs the abilities to decide, whether a given problem is solvable or not and how difficult or demanding it is.

Traditionally, the courses of computer science in Finnish universities consist of lectures, exercise sessions, and one or two exams. The exercises have been found to be the most effective part of the traditional course setting, but lectures suffer from passive role of learners and are not fully utilized by the students. So the focus is how to develop the lectures or utilize their time more efficiently. The problem solving skills are practised in the exercises, but tasks are mainly mechanical practising of the new methods and constructive processing of the topics and understanding the subject as a whole are not supported. The exercises are solved individually, but in the exercise sessions the group works can be used in checking the correct answers and the students have found it very useful.

According to constructivist view of education (3) learning is an active process, in which students organize the new ideas and concepts based on their previous knowledge and experience. Also the ability to work in groups supports active learning, as we have noticed in our exercise sessions. Problem-based learning fulfils these needs of active and collaborative learning (15; 19), but in the same time it also meets more existential challenges by supporting students in their process into intellectual and emotional maturity(16, 8–9).

In the next chapters we shall first consider, what the problem-based learning is about. Then we shall describe our experiment in a course on theoretical computer science and evaluate the results. As a complement Naumanen (14) has studied the same experiment from the students’ point of view.

2 Problem-based method

2.1 What is problem-based learning?

The main idea of the problem-based learning (PBL) is to use problems, queries or puzzles as starting point for learning. In fact problem-based learning is not just a single method or technique, but a variety of problem-based approaches, from lecture-based teaching to pure problem-based learning without any teaching or assessment by the teachers (4; 2). Ellis et al. (7) divide the problem-based learning methods into three categories. In the modest forms,
which Ellis et al. call problem-based approach the material is presented in normal lectures, but problems are used to motivate students and to demonstrate theory. In the hybrid models or guided problem-based learning the problems are solved in the groups, but also lectures are used to present the fundamental concepts and conceptually most difficult topics. In the full problem-based learning the problems guide and drive the entire learning experience and no formal exposition of knowledge from the "expert" is given. However there are some general characteristics which are typical for problem-based courses (4):

- Acknowledgement of learners' experience.
- Emphasis on students taking responsibility of their own learning.
- Crossing of boundaries between disciplines.
- Focus on the processes of knowledge acquisition rather than the products of such processes.
- Change in staff role from instructor to facilitator.
- Student self- and peer assessment of learning.
- Focus on communication and interpersonal skills.

In our experiment we have adopted the seven step model, which is quite often used in the education of medical science (5; 9). The learning succeeds through the following steps:

<table>
<thead>
<tr>
<th>Phase</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Defining unclear concept</td>
<td>The students look for concepts, which are unclear and try to define them.</td>
</tr>
<tr>
<td>2. Defining problem</td>
<td>The students discuss about the problem.</td>
</tr>
<tr>
<td>3. Brain storming</td>
<td>Students try to construct, test and compare different hypothesis and explanations.</td>
</tr>
<tr>
<td>4. Constructing hypotheses</td>
<td>The problem is analyzed carefully by comparing different hypothesis. The ideas are argumented and organized into an integrated whole.</td>
</tr>
<tr>
<td>5. Defining learning goals</td>
<td>The students write down their learning goals for self-studying.</td>
</tr>
<tr>
<td>6. Self-studying</td>
<td>The students acquaint themselves independently with the topic. In this phase also lectures can be offered to support the self-studying.</td>
</tr>
<tr>
<td>7. Sharing the results</td>
<td>The students compare their solutions and try to help each other to understand the topic. Learning goals are checked and final conclusions are drawn.</td>
</tr>
</tbody>
</table>

In this model the steps 1-5 constitute an opening session and step 7 works as a closing session. At least some tutors or facilitators should be available during the group sessions. For the self-studying phase students can be given some reference material, but they are encouraged to use different sources. Also other group members or teacher consultation can work as information source, if the process gets stuck.

2.2 Evaluation of the method

The problem-based learning has produced excellent experiences in medical science, which have encouraged other faculties to adopt it in higher education. In computer science, the problem-based methods have been experimented mainly in the areas of programming and software engineering, but in Linköping University the entire curriculum of Information Technology is problem-based(11). A lot of good effects have been reported: The students have deeper understanding of the issues and they are better motivated. PBL improves also communication and cooperation skills as well as metacognitive skills like problem solving and ways of thinking.
Individual learning goals support different learners and also poor students manage well in PBL. In addition, PBL gives practice in research and information retrieval (11; 10; 8).

Ellis et al. (7) have argued that problem-based learning suits especially well for computer science. A computing discipline matches the characteristics of problem-based learning: computing itself is mostly problem-driven; life-long learning is necessity in the area and the professionals must constantly update their skills; the project works in groups is dominating working mode in industry; and finally, computing crosses discipline boundaries.

The evaluation studies agree that the problem-based learning produces better skills than the traditional education. However it has been speculated, whether the students acquire as good theoretical knowledge as in traditional education with lectures and exams. Dochy et al. (6), Albanese and Mitchell (1) and Vernon and Blake (18) have made independent meta-analyses of the reports, which compare problem-based method to traditional education. The conclusions are uniform: immediately after the course the problem-based learners have better skills but slightly poorer knowledge than traditional learners. However the difference in knowledge level is statistically quite unimportant and even better results have been reported. The most interesting observation is that after some time the theoretical knowledge of the PBL learners is at least as good as the traditional learners – i.e. the PBL learners remember better what they have learnt.

The results of problem-based learning match well with constructivist view of learning (3): during the problem-solving process the students actively construct their knowledge on the basis of their own experience and reflections. It is quite obvious that this kind of processing leads to deep learning instead of memorizing facts for the exam.

However, a good method can also fail and it is good to be conscious of possible problems. McCracken and Waters (13) have reported about some problems in their experiment in software engineering: Real world problems are essential for PBL, because the students do not like artificial “toy problems”. However such problems are often too worksome and difficult even for the facilitators. In the PBL the students may not develop deep learning issues, because they cannot set efficient learning goals or they concentrate in particular solution, not the general method. The students may also abandon the PBL process as they move further into problem. The students can also have difficulties in developing the desired metacognitive skills. McCracken and Waters noticed that partly the problems due to staff recourses: enough tutoring by experienced facilitators is essential, especially when the method is new for the students.

As a solution for the first problem Ellis et al. (7) suggest well-structured problems for novice students. For advanced students PBL course should emphasize open-ended and ill-structured problems, which are typical for real life. For other problems the learning diary can be especially valuable complement (7). In the learning diary the students are asked to process further what they have learnt, what was unclear and to construct overall pictures. In addition they analyze their own learning process and develop the ”meta-skills” of learning. Learning diary offers also an effective forum for feedback between the facilitator and the student (12; 7).

3 The course description

3.1 Resources

The problem-based method was tested in teaching theory of computability in the Department of Computer Science, University of Joensuu on spring term 2003. In Joensuu there is only one course about this subject, ”Theoretical Foundations of Computer Science” ("Tietojenkäsittelytehteen teoreetiset perusteet"), which should cover the whole area, both the theory of computability and computational complexity. The status of the course in the curriculum is very problematic, because it is reserved only 3 credit units (about 120 hours working time), which means 40 hours lectures and 20 hours exercise sessions. In other universities of Finland the topic is usually reserved 4-5 credit units in one or two courses.
In addition to the lecturer only one or two exercise session assistants were available. Compared with the course contents these resources were too scarce and it is widely known that problem-based method requires even more resources. For these reasons we managed to separate the theory of computational complexity as a separate self-study course and the course covered only theory of computability from the finite automata and regular expressions to context-free grammars, pushdown automata, Turing machines and solvability. In addition the students got one extra credit unit for problem-based learning methods, i.e. learning general problem solving and studying skills.

The course material consisted of Finnish lecture notes and several recommended books in the area. In addition students could buy a Finnish booklet, which covered the whole course. The material is widely thought to be very theoretical and hard for self-study by most of students. The lectures tried to complement this by offering more practical applications and demonstrating the techniques.

3.2 Arrangements

The course was given during ten weeks, with one week break, which was reserved for an art exhibition. In the art exhibition the students represented the pieces of art they had made with the aid of formal languages: animations, pictures and music by LL-systems and poems generated by finite automata.

Each week the students had two hours traditional exercise session, but the lecture time was scheduled in a new way. Half of the lecture time was used in opening and processing the problems in small groups according to the seven step model, and the other half was reserved for lecturing. A couple of times this normal program was replaced by playing problem solving games. During the group works the lecturer wandered in the groups and tutored them. In a couple of the most difficult problems the assistants were also voluntarily available.

In the first half of the lecture the students first processed the last problem for about half an hour. Then they were given a new problem to be opened. They were also encouraged to meet the group members in the free time, and at least some groups had very active communication through chat. When the problem reports were returned we still had some general discussion with the whole group and final conclusions were drawn.

The students were given freedom to participate either problem-based learning or perform the course by traditional way with exercises and exams. However, the amount of lectures was only half of normal and they were given in Finnish. For foreign students the course was more demanding, because they could not participate the lectures. However they had an English exercise group and in the problem-solving part they had an English speaking group.

3.3 Problems

During the course the students solved total 14 problems. In addition the first problem was processed along the whole course and especially in the last lecture, when the students played problem solving game. The problem cases and their learning goals are represented in the table 2 in the Appendix. It is considerable that the learning goals were not told to the students, but everybody should determine their own individual learning goals. In the end of the course the students could compare, what they have learnt against the teacher’s learning goals, and thus perform self- and course evaluation.

For example the first problem, which was quite central for the whole course, was following:

You are working in a problem solving company, the motto of which is "What we cannot solve, cannot be solved". The company has all kind of computer science experts, for example top programmers of all existing programming languages, and super computers with all possible utilities. Your task is to receive the problems of clients and decide, if they are solvable. If the problem is solvable, you give it to a suitable software engineering group and tell, how "difficult" problem it is, so that they can allocate the required recourses.
The full descriptions of the other problems can be found in

In the table 1 the problems are classified according to whether the initial problem setting,
the goal and the methods are open or closed, although sometimes it is not obvious, into which
category they belong. In the open cases the starting point and the problem are vaguely defined
and no solution method is suggested. As a contrast in the closed cases the initial setting and
stopping point are clearly defined and also the solution method can be hinted.(17)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Setting</th>
<th>Goal</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 0</td>
<td>closed</td>
<td>closed</td>
<td>open</td>
</tr>
<tr>
<td>Problem 1</td>
<td>open</td>
<td>open</td>
<td>open</td>
</tr>
<tr>
<td>Problem 2</td>
<td>open/closed</td>
<td>closed</td>
<td>open</td>
</tr>
<tr>
<td>Problem 3</td>
<td>closed</td>
<td>closed</td>
<td>closed</td>
</tr>
<tr>
<td>Problem 4</td>
<td>open/closed</td>
<td>open/closed</td>
<td>open</td>
</tr>
<tr>
<td>Problem 5</td>
<td>open/closed</td>
<td>closed</td>
<td>open</td>
</tr>
<tr>
<td>Problem 6</td>
<td>open</td>
<td>open/closed</td>
<td>open</td>
</tr>
<tr>
<td>Problem 7</td>
<td>open</td>
<td>open</td>
<td>closed</td>
</tr>
<tr>
<td>Problem 8</td>
<td>closed</td>
<td>closed</td>
<td>open/closed</td>
</tr>
<tr>
<td>Problem 9</td>
<td>closed</td>
<td>closed</td>
<td>open/closed</td>
</tr>
<tr>
<td>Problem 10</td>
<td>closed</td>
<td>closed</td>
<td>open/closed</td>
</tr>
<tr>
<td>Problem 11</td>
<td>closed</td>
<td>closed</td>
<td>closed</td>
</tr>
<tr>
<td>Problem 12</td>
<td>open</td>
<td>closed/closed</td>
<td>closed</td>
</tr>
<tr>
<td>Problem 13</td>
<td>closed</td>
<td>open</td>
<td>open</td>
</tr>
<tr>
<td>Problem 14</td>
<td>open</td>
<td>open</td>
<td>open</td>
</tr>
</tbody>
</table>

Table 1: Classification of the problem setting, goals and methods as open or closed.

4 Evaluation

4.1 Results

In total, 79 students registered for the course, from which 75 participated at least the first
exercise session. 63 students begun in the problem-based way, but two of them changed to
traditional method in the beginning. 12 students dropped the course, 7 traditional learners
and 5 problem-based learners. 56 students completed the course in problem-based way, but
one of them was failed because of missing sections. It is remarkable that those problem-based
learners who dropped the course did it in the first weeks, when they have seen what the course
was dealing. The problematic droppers of traditional courses, who waste their own and staff’s
resources until the first exam were missing. Only two problem-based students dropped the
course in the middle of it. In the traditional courses of theoretical computer science usually
much less than 50 % of students pass the course. Thus, we can conclude that the students
commit themselves better to a problem-based course than a traditional one.

The grade distribution of the problem-based learners and the traditional learnes of the
course are represented in table 4.1. (The grade 1- is the lowest accepted performance and 3 is
the best grade.) There is no significant difference in the grades, but the amount of traditional
learners is too small for any conclusions. However it is considerable that the population, which
passed the course was about three times more than in the traditional mode. Thus, also the
weak students managed well in problem-based learning, even if the evaluation was quite strict
and reaching the fundamental learning goals was checked. Another interesting phenomenon
is the amount of the best grade in the distribution. Many of those students did not only learn
Figure 1: The grade distribution of problem-based and traditional learners in the TFCS course.

everything required for the best grade, but became real experts in the area and can even work as tutors in the future courses.

4.2 Summary of student feedback

In the middle of and after the course the students were asked to fill the department’s course evaluation forms anonymously. In the query the students were asked about overall satisfaction with the course, its usefulness, the lecture material, lecture’s and exercise teacher’s expertise and pedagogical skills and free comments.

Only 36 students answered the final query and the answers of problem-based and traditional learners are mixed. The students were also asked to give some free comments about the problem-based method. 26 students considered this question. Most of them were either very satisfied or quite satisfied with PBL and only 3 were unsatisfied (from whom one did not even participate the PBL). 4 of the students found the problem-based method heavy and total 14 students mentioned that the course would need more time. 6 of those missed more traditional lectures. Only two students were unsatisfied with the learning diary. The conclusions about satisfaction with the course and PBL are represented in figure 2

Figure 2: The students’ overall satisfaction with the course and satisfaction with the PBL method (vs=very satisfied, qs=quite satisfied, n=normal, qu=quite unsatisfied).

Much more feedback was given in students’ learning diaries and most of it was very positive: the students found the method very effective and they believed they will also remember the things longer. Some students even found the problem-based method as the only possible way to pass the course. It is also noticeable that participating the group works was considered crucial for solving problems. Naumanen (14) has made a further study on students’ experiences in
learning diaries.

4.3 Teachers' experiences

The course was organized by a lecturer and two assistants. The assistants led exercise sessions and checked the problem reports weekly. They also participated in problem sessions a couple of times. The goal was that they were not required more work than in the traditional courses, e.g. they did not have to check any exams. However, the lecturer's work was enormous, because she was lecturing, tutoring in problem solving sessions, leading one exercise group, checking her part of problem reports and reading the learning diaries weekly. She also had to check the exams of the traditional learners and prepare English material for the foreign students. Reading the learning diaries and checking problem reports took both about 10-15 minutes per each student weekly, but deviation was large.

It was clear that one tutor was not enough for assisting 50-60 students in the problem sessions. It was much easier, when the assistants were available, and we can conclude that one tutor can take care of 20-25 students. Anyway this experiment proved that problem-based method can be applied for mass teaching, if the department is eager to invest more staff resources for better results, both in aspects of quantity and quality.

All the teachers agreed that obviously the students learnt better and were well motivated. Only the group of foreign students seemed to be more divergent, but the reasons for less motivation and poorer results are not yet studied. Also the teachers got new learning experiences, both in the field of the subject and education in general.

5 Conclusions

The problem-based learning was applied to theory of computability in Department of Computer Science, University of Joensuu. The application of problem-based method was a hybrid model with short lectures and the main focus was in problem solving process according to the seven step method.

This experiment proved that problem-based learning suits very well for hard theoretical subjects. The students were very satisfied with the course and the number of students, who passed the course was significant, even if the evaluation was strict. This experiment suggests that the problem-based learning could also be used in other theoretical courses of computer science like data structures and design and analysis of algorithms.

References


## Appendix A

<table>
<thead>
<tr>
<th>Problem</th>
<th>Topic</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 0</td>
<td>Morse alphabet Basic concepts</td>
<td>To learn the basic concepts like alphabet, strings, languages, consideration that all symbolic languages can be coded by binary alphabet. Practising the problem-based method.</td>
</tr>
<tr>
<td>Case 1</td>
<td>Problem solving company Modelling problems, their difficulty and solvability</td>
<td>Overview of modelling problems, their difficulty and solvability. The need for formal languages (descriptions). Chomsky hierarchy of languages, existence of unsolvable problems. (In a more profound way in the end of course.)</td>
</tr>
<tr>
<td>Case 2</td>
<td>Search for mail addresses and precompiler for programming language Regular expressions</td>
<td>To understand the idea of regular expressions and construct a describing expression for a given problem,</td>
</tr>
<tr>
<td>Case 3</td>
<td>Coffee automaton Deterministic finite automata</td>
<td>To learn how to construct a deterministic finite automaton and represent it as a program,</td>
</tr>
<tr>
<td>Case 4</td>
<td>Nondeterministic editor - Nondeterministic finite automata and determinization</td>
<td>To understand the idea of nondeterministic automata and how to transform them deterministic,</td>
</tr>
<tr>
<td>Case 5</td>
<td>Roman checking exams - Finite automata vs. regular expressions, ε-automata</td>
<td>To learn how to construct a finite automaton corresponding a regular expression and vice versa. Also ε-automata and determinization and minimization are needed,</td>
</tr>
<tr>
<td>Case 6</td>
<td>Grandma’s rhyme - Pumping Lemma</td>
<td>To understand the limits of regular languages and how to recognize irregular languages, Pumping Lemma,</td>
</tr>
<tr>
<td>Case 7</td>
<td>L-systems - grammars</td>
<td>To learn how to construct pushdown automata and how they can be programmed,</td>
</tr>
<tr>
<td>Case 8</td>
<td>Parentheses parsing Pushdown automaton</td>
<td>To understand the idea of grammars, especially context-free grammars, and how to program simple grammar based systems. Artistic pleasure,</td>
</tr>
<tr>
<td>Case 9</td>
<td>Arithmetic calculator - LL(1)-grammars, recursive parser</td>
<td>To understand the idea of LL(1)-grammars, how to transform nearly LL(1)-grammars into LL(1) form and how to implement a recursive parser for it. Also the idea of “hidden” stack in the recursive programs,</td>
</tr>
<tr>
<td>Case 10</td>
<td>General parser the CYK-algorithm and Chomsky normal form</td>
<td>To understand how CYK-algorithm works and why Chomsky Normal Form is needed. How to modify an arbitrary context-free grammar into CNF,</td>
</tr>
<tr>
<td>Extra</td>
<td>Attribute grammar and parser tools</td>
<td>To study the idea of attribute grammars or test some parser tool,</td>
</tr>
<tr>
<td>Case 11</td>
<td>Summing machine - basics of Turing machines</td>
<td>To understand the basic idea of Turing machines and how to construct them, Understanding that multiple tape machines are equivalent with standard TM’s,</td>
</tr>
<tr>
<td>Case 12</td>
<td>Library functions for TMs deeper practice on Turing machines</td>
<td>To study more carefully, how to construct TM’s for different purposes,</td>
</tr>
<tr>
<td>Case 13</td>
<td>Programming competition - Universal machines and universal languages, solvability proofs</td>
<td>Also consideration of equivalence between programs/TM’s, functions/submachines,</td>
</tr>
<tr>
<td>Case 14</td>
<td>Philosophical considerations</td>
<td>To understand the idea of universal machines and universal languages,</td>
</tr>
<tr>
<td>Case 1 rev.</td>
<td>Problem solving company again Overview of the principles learnt</td>
<td>How to study properties of Turing machines? Introduction to the unsolvability of semantic properties,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>To consider the meaning of main results concerning solvability and Church-Turing theses, Limits of computation,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>To create an overview, how we can use the techniques and principles learnt in this course to analyze the computability and difficulty of any given problem,</td>
</tr>
</tbody>
</table>

**Table 2**: Problem cases and their learning goals.
A Narrative perspective to students’ experiences in problem-based learning in theory of computation – New deals with learning

M. Naumanen

Department of Computer Science, P.O. Box 111, FIN-80900 Joensuu, Finland
mnauma@cs.jouneesu.fi

Abstract

In this paper we shall see the problem-based learning in the context of theoretical computer science course, from the students’ point of view. We shall consider how the students experienced the new way of learning and reflected the things learnt in their learning diaries and how they adjusted to the process in general. How to make learning more meaningful for everyone. Aim is also to raise discussion about the methods in interpretation and analyzing.

1 Introduction

In this paper we shall shortly present the experiences and feelings of the students in a problem-based course on theory of computation (TFCS) in Department of Computer Science, University of Joensuu. The detailed description of the course can be found in Hämäläinen (4). In this paper the emphasis is on how the students evaluated their own learning, experienced things and adjusted to this new system. We shall partly reflect the extracts taken from students’ learning diaries against "Dimensions of Learner Experience"-framework by Savin-Baden (7). As Savin-Baden (7, 6) says: "The consideration of personal experience in learning is something that is noticeably lacking in general, and especially in PBL, yet for many, personal experience is that which makes learning both possible and meaningful". That is also what we are hunting for.

We shall define the problem-based learning and the background of the study in chapter 2. Next, in chapter 3, we shall get to know the framework of Dimensions of Learner Experience: its structure and the means to make transitions within it. We shall get acquainted with the disjunction and the ways to encounter it. The importance of enabling disjunctions as well as the contribution of group-work to the learning will also be discussed. Finally, in chapter 4, we shall have a look at the selected diary-clips to create an overview of the students’ learning experience.

2 Methods and background

2.1 Problem-based learning

Problem-based learning (PBL) is a learning method first introduced by McMaster University in 1960’s, as a demand for applying the learnt knowledge to real life situations – to meet the world and its challenges (6). In PBL students are provided with practical problems that induce the learning. It consists of learning in small groups, emphasizes how to learn now and in the future, how to end up with solutions rather than getting them ‘ready-made’, how to develop critical thinking and how to learn to learn. It is a process of continuous concept defining – problem presentation – discussion – self studying – reviewing – evaluation, a play of actions, goals and results with individual learners guided by external facilitator (i.e. teacher) and supporting team (group-work, discussions, brainstorming). (1; 2)

Torp and Sage (8) point out the importance of the concept of "Learning to learn and understand", as well as retrieving information (and thus to get to know how to cope with the "information overload" of today’s society). It is also notable that by PBL students can
develop new abilities to be self-directed learners and learn the methods of scientific research, i.e. finding information, proposing and testing hypotheses and presenting results. These higher-order thinking skills (analysis, synthesis and evaluation) are not learnt through direct instruction. They emerge from experience of doing things by yourself and PBL can offer one way to develop these learner abilities (8, 104).

2.2 Learning diaries analysis and research methods

During the course students were asked to write learning diaries. They were part of the process and one target of evaluation. In the diaries students wrote about their experiences, thoughts and feelings and analyzed their own learning process. Students could freely express themselves in their diaries and they received comments and feedback weekly.

The selected parts of learning diaries have been further analyzed. No specific qualitative method has been used and the focus is to provide interesting and general descriptions and thoughts emerged during the course, especially at the beginning and at the end of TFCS. The author's aim was to find out the general readjustment to this new way of learning.

3 The framework

The Dimensions of Learner Experience (DLE) is a framework created by Savin-Baden, who constructed it through the stories of the students undergone the PBL-method (7, 54–67). Essential is that students should learn how to deal with their conflicting real experiences and the imagined pictures they possess about learning, i.e. to bring themselves into the learning. Literally speaking students are not learning for the school but for the life, the life which they construct by themselves, creating their own culture of knowledge based on their dynamic epistemological view.

![Diagram of Dimensions of Learner Experience (DLE)]

Figure 1: Dimensions of Learner Experience

DLE contains three different, but intertwined, levels/points of view, which Savin-Baden calls for stances. A stance is a kind of an extended version of attitude, which covers the sense
of one's attitudes and beliefs towards a particular thing (e.g. context, person, experience) as well as our unconscious prejudices, prior learning experiences etc. A Stance can be further divided into domains, which are the particular overlapping areas between which the learner travels, gradually constructing her/his stance by means of transitions. Transitions are the shifts in one's learner experience and are strongly connected with the concept of disjunction (7, 100-109). The term disjunction means the sense of fragmentation that a student feels in her/his world (of ideas and cognition). It usually involves frustration, confusion and the loss of sense of self and can be either enabling or disabling by nature (7, 89). The structure of the DLE-framework is presented in figure 1.

Next we shall have a closer look at the stances, the hierarchy of the experience world of the student and the issues concerning about the change-points and the sub-structure of different stances. We shall also define the disjunction and how to approach it. We shall reflect the extracts taken from students' diaries in TFCS-course against the framework as examples.

3.1 Stances

The stances are strongly intertwined to each other, i.e. one's personal attitudes and cultural background affect the position she/he may take in an interactional stance, e.g. in a group-work.

The term personal stance is used to describe how students and teachers see, define and place themselves in PBL environment (7, 58). It is a playground where experiences are created and one can search one's place in it – it is a particular view of the world at certain point of their "learning career". The personal stance is very transient and induced e.g. by the self perception of students and learning context (which both are changing objects).

E.g. "Defining my future self" can be read on the pages of some diaries:

"I loved the logic of those regular expressions in Problem 3. They are very useful in developing you logical reasoning. It was good for me to learn those logical expressions considering my future profession as a teacher. I will definitely teach these skills also to my students."

"...I abuse the course so that the theoretical foundations do not interest me in general, but it provides me better understanding to things and thus I can make my work [as a professional] better."

A pedagogical stance is the dimension how students see themselves as learners. This is affected by prior learning experiences, some taken-for-granted notions of learning and teaching and the instruction they have received. The domains of this stance represent approaches to learning, the kinds of strategies students have. These are: reproductive pedagogy ("traditional", learning as safe and predictable happening), strategic pedagogy ("maximise the profit" and choose the methods according to it), pedagogic autonomy ("what and how", choosing the methods that satisfy one's own, personally defined needs) and reflective pedagogy ("flexible", critical evaluating and deep engaging of the things). (7, 61-63)

An interactional stance brings the relationships into the picture. Its domains are specified according to the nuances of relationships between teacher-student, student-student and student-group (7, 63-66).

3.2 Disjunction

The meaning and purpose of disjunction is gradually to bring towards deeper understanding of the world and the self. Effective "disjunction-management" can provide transitions into students' lives and thus make re-valuing of knowledge and values to emerge – you can call it deeper learning. According to Savin-Baden (7, 89) there are two types of disjunction, enabling and disabling, and both of them can produce a shift in one's life. The main point is to learn to deal with these disjunctions. Savin-Baden has introduced four different approaches to face the disjunctions (7, 96-98). She reminds that they are just generalized ways to handle a situation,
since conflicts, ambiguity and quarrelling facts experienced by each individual student vary a lot, so do the methods used to overcome them.

![Diagram showing the centrality of disjunction in learning framework]

Figure 2: The centrality of disjunction in learning framework (7)

The importance of the disjunction in a learning experience and the ways of coping with it are presented in figure 2. The PBL itself acts as an initiator for disjunction by its novel methods, partly stimulating, partly irritating the learner. Later these same values serve as feedback enabling the life-long learning to occur. Next we shall discuss the ways to handle these irritating and confusing moments.

3.3 Four ways to meet the disjunction

Savin-Baden (7, 96–98) describes four means to attack the disjunction. We shall give an overview of these ways of “lockworkshop”.

The student may retreat the situation e.g. by means of excuses, or building a safe-wall against the threats it seems to pose for their learner-identity. This may worse the situation and thus often causes a disabling experience and heavy tendency e.g. for blaming others or the system used for her/his incapabilities and/or demand for right answers.

This could be well noticed during the TFCS-course, too. Especially for some foreign students it was very difficult to adapt into this new independent method of learning, partly because of the lectures, which were given only in Finnish. Here are some good examples of retreating the situation by demanding answers and blaming others:

“"We are going somewhere deeper and deeper to the theory of computer science with nothing provided to read, with nothing provided from where to get the really required information. And more and more time is spent to find it, somewhere outside the course page, somewhere outside the lecture notes … I think it would be useful if the lecturer mentioned every time the exact place where to get the required material for completing the problems. Especially for foreign students, most of which have to participate in this course.”

"…Please, tell me, maybe am I so clear understand the main goal of our Problems, and maybe must I read somewhere the instructions how to do them and do not going to consider my own ideas?"

This is where the facilitators/tutors enter the picture and try to unlock the situation, e.g. by appreciating the knowledge the student possess and trying to offer assistance in searching for the material, and this is the point where discussions are needed.

The student may temporize the disjunction situation. Temporizing means postponing the situation. Students recognize that something needs to be done in order to effective results to take place, but they are not ready to take the transition. The tight schedule and ”sticking into one’s habits” could also be counted into this category. They inhibit successful transition and thus new ways of seeing the situation and the most effective learning to take place.

Using the avoidance the student both temporize and adopt some mechanism to circle the disjunction. She/he finds a way to bypass the situation but this may still be more effort-taking than engaging the situation, because the student cannot escape the situation and it confronts her/him gradually and can make learning an unpleasant phenomenon for her/him.
In order to proceed, the difficulties need to be identified and engagement has to enter the picture. It requires to become aware of the existence and roots (internal and external) of the difficulties through reflection. This enables transition to take place (intra or inter domain) and the puzzle to get a bit clearer — until the next disjunction fight emerges. The key point is taking responsibility of one’s own learning, which is also one of the fundamental issues in problem-based learning.

Traditional teaching method and its collision with PBL naturally causes a disjunction and starts a cycle. Students begin to view things differently and some type of questioning occurs, like:

"I totally couldn’t understand what was the reason to begin the course with solving so called problems, just some spontaneous situations not connected to anything in common …”

"At the beginning my goal was just to pass the course. When I noticed this new teaching method I jumped to the roof. And when teaching went on, I realized that I do have a capability to learn something, and so it turned out to be at the end, that those previously totally unknown and ‘all Greek to me’ sounded things like Turing machines, automata, regular languages, context-free grammars, etc. have been cleared up to me. Also getting to know them and using them is not as scary thing than earlier.”

3.4 Enabling disjunctions

By providing meaningful examples and new ways of doing and seeing things these disjunction situations can be made easier and more flexible to cause the action of learning – both the subject and yourself. E.g. Ellis et al. (3) have focused the importance of the meaningful problems as motivating actors in PBL. Realizing the fact that her/his prior knowledge and experiences are valued usually acts as a trigger of engagement and starts a journey in the world of different stances of Learner Experience, where the student finds new doors opening and part-time harmony within her/himself. The journey from disjunction to disjunction has just began, taking a form of challenging formula of recursion with ever increasing variables and lots of possibilities.

We shall also discuss the group-work as an enabling factor for creating transitions and how this was experienced by the students of TFCS –course.

"I have once again noticed that this theoretical subject can be approached interestingly, this week with forms of pictures, animations and a game."

"I enjoyed the Art Exhibition. I saw a lot of amusing things. …I think that an Art Exhibition is a very nice way of teaching, as students do their best at creating pieces of art and share their ideas at the exhibition …”

And some personal interest to the things:

"Woh!!! I like automata! It’s so interesting to me! And I even know why – when my mammy wrote her diploma her theme was ‘Finite non-deterministic automata …’, of course there was more mathematics and I think more theoretical things, but the interest is in our blood!”

More creative styles of handling the theoretical issues, e.g. in the form of the poems (generated by the simple automata made by students) can be seen on the Internet.¹

3.5 Group-work

A group work can function both as an enabling or a disabling factor in the machinery of DLE. Group-dynamics is a difficult issue and e.g. McCracken and Waters (5) have presented that these skills must be taught explicitly to the students. Just working in a group does not facilitate that team-work is learnt.

¹http://www.cs.joensuu.fi/~akautone/poems.html
The use of a group work and the possibilities it offers in PBL were seen mainly as a positive thing in TFCS. It offered many students a new sense of themselves both as giver and receiver of information. It gave them a surface to reflect to and from and get a deeper picture in the problems solved and increased the sense of integrity. Group work as its best is sharing yourself with others and discovering the new waters of the ocean of the knowledge together. Mutual respect is valuable in learning and for learners.

This kind of co-operation and interactional stance is considered fruitful also in the "Real talk" domain by Savin-Baden (7, 65). The following extracts from the learning diaries describe the situation:

"We have solved the problems, as well as demos via Messanger between the members of our team. My friends (and the members of our group) deserve the biggest thanks for the fact that I have managed the course so far, we have supported one another along the journey and helped each other."

"At this phase the benefits in group work started to get concrete. When you have to explain the half you understand to your friends, you usually have to find out the rest of it, too, when someone is asking the classical 'why'-question ... Because lectures are given in Finnish, the things in this course may not be that hard after all, and it's amazing what you can find in the secret closets of your memory ..."

Of course the student takes her/his own position in the field of interactional stance and relates its importance and usage for her/his point of view: what may be gained or lost and in what sense. And it is nice to observe things, too:

"Observing others in different situations is interesting. If you have strengths to listen and observe, you may in some point learn to even understand something."

The concept of group-dynamics play here an essential role, too. It is a challenge. There were also some shifts to be seen during the course from a grouper to an individual and also the other way around.

4 Some more insights into learning experience

Clear and easy classification according to the clips taken from students' learning diaries is quite complex, but it is easy to see the change in their values and reflections on what they know and how they see themselves as learners. Next we shall have a look at some selected diary-extracts.

Finding the appropriate strategy

"At first it took some time to get used to the working method. I think it's also the reason why the first reports didn't cover all. I simply didn't know how thoroughly you should write in them."

"It was nice to notice that I realized how these things have to be studied, because my points from the problems were great after couple of first problems."

"It seems that this course is ok after all. Getting the things clear comes about two weeks late, but it comes after all - actually this is nice way to learn, even if it takes nerves when you have used to lecture based mode. ... In fact this course is getting better towards the end, works are getting along smoothly when you have got the idea what to do and how."

Evaluation of own learning (at the end)

"Science is not my strength, so I want to study it. ... Afterwards the most rewarding courses have been just these hard courses, after which you are no longer the same person than before them. ... I have learned, but there seems to be longer and longer way to Knowing. The course gave me answers, but now I have once again new questions ..."
"I have learned about the content of the course more than I expected. And, if some thing was a bit unclear at this point, I can at least start searching for information in right directions. Finding relevant information may be the most important area of learning in the university, when you can’t possibly remember everything by heart. ... The problems forced you to study and learn without planning how to do it separately."

"After all I was surprised by the fact, that I could get the ideas, and get them well. When I compare the situation to the last year, when I couldn’t learn these things in any way."

And creating a "big picture" was important for quite many students:

"How all I have studied so far and in the future really connects with anything? Engaging with separate problems has taken all the attention away from the 'big picture' ... If you could see the 'big picture' it would boost the studying and help finding the essential things."

The same student continues later:

"The problem-game played on last lectures was a good idea. It was good for recap and even cleared the things. I am always in a habit of loosing the 'big picture' and now it was there – on the table in front of me."

"The things covered in the course appeared scattered at first, you couldn’t get a grip of the things at hand, but towards the end all started to get clearer: ... it’s very pleasant to learn, even if it tooks place so late stage of the course."

"When our course was only started I often asked myself why we study this or that, and why not something else. Now I can think that all we have learnt can be included into this course, because I can find a transition between our themes and computer science."

"Liking and disliking"

Almost everybody liked the PBL-method and some mentioned it to be suitable for themselves, e.g.

"PBL suited me excellently. This was my first trial using it, and I just noticed it to be 'my case'. Things that I tried desperately learn last year on this same course using 'ordinary' method, finally opened to me thanks to the problems. I’d love to take other courses in problem based way, too." ... but some had put it this way:

"The course was very unusual in terms of both teaching and requirements ... I'm not sure if I’ll ever take a risk to participate in the similarly based course ..."

This was the conflict where traditional way of learning (getting both information and the instructions how to work from the teacher) was hard to abandon by the student and the ways to treat the situation did not provide the best possible solutions. Also the lack of group-work was a drawback, but the course was passed. The most remarkable point of disjunction seemed to be at the beginning of the course (as mentioned earlier), when traditional method met the PBL and confusion about the new ways of studying, searching the information by yourself and creating a strategy to cope with the course.

Even those who first took very protective attitude against the new way of learning and who could not deal with the arising disjunctions in an enabling way, were adjusting to the way towards the end of the course or finding the meaning.

"At first I couldn’t make any specific goals, because the purpose of the things had not cleared, yet. E.g. in programming the goal is clear, but the plain word 'theory' makes the hair rise up."

"I begin, to my horror, gradually to get warmer to this kind of course, I mean problem based learning. Now when there’s no exam in the course, you can better concentrate on what you find interesting and what is of more use and the things that don’t interest me can be left with lesser observation, because you don’t have to be scared of what the examiner might come up to ask."
Hard but fun under the caring eye of the cognitive coach

All agreed that PBL -method was harder than the traditional, but regardless of that the results achieved usually were considered worth that.

"During the spring I have been obliged to say, that this course has been the hardest one among the courses thus far in the university. Usually it's enough that you just read a little bit the night before the exam and visit few exercise classes. Now when you have decided to do these problem-reports, you have something real to do for every week. On the other hand, learning seems to be a bit more efficient in this way ...”

"The course has been extremely giving, but very hard! I have also found a learning-diary writer in me! This counts for learning, too, I suppose. Previously I used to hate learning diaries, as mentioned in week 1.”

And the teacher does make a difference:

"It's very rarely when lecturers are so interested and worrying about studying process. I think W and R are devoting the huge amount of time to the course. It is laudable. And of course it is the reason of our problems, but it's real life.”

5 Conclusions

The students found the PBL-course mainly a very positive experience. It was hard but fun, and many achieved a result they never could have imagined to gain. The students also grew to adjust to the new situation and some new deals with learning were committed by them. The study itself serves as a stepping-stone towards further and deeper research in the area and encourages the educators to try and to adopt the PBL in their teaching – in some form.

References


2as described by Torp and Sage (8, 15)
Introduction to Text Graphs

Esko Nuutila and Seppo Törmä
Helsinki University of Technology, P.O.Box 5400, FIN-02015 HUT, Finland
esko.nuutila@hut.fi, seppo.torma@hut.fi

Abstract

We introduce text graphs, a simple and easy-to-learn notation to represent complex conceptual content. Text graphs have been designed to replace essays and concept maps as constructivist assignments in our computer science courses. The advantages of text graphs over previous similar notations is accuracy and well-defined meaning, which make them especially suitable for representing technical content.

1 Introduction

Text graph notation provides a way to write text divided up into the nodes of a graph structure. The graph offers an explosion view to the contents of the text. When used properly, it makes explicit the meaningful parts of the text and the relations between these parts.

The nodes of a text graph contain fragments of text. These fragments may include so-called anchors that are placeholders for other text fragments. The edges of a text graph connect nodes and anchors to each other.

Figure 1 below contains an example of a text graph that describes concepts related to the file system of a computer. There are seven nodes and eight edges, and some of the nodes contain anchors shown as small rectangles.

![Text Graph Example](image)

**Figure 1:** A text graph of file system concepts

It is possible to define a meaning of a text graph with respect to natural language, namely, the language used in the nodes of the graph. This enables us to specify if the text graph is valid (has any meaning at all) and argue whether it is true or false.

The natural language meaning of a text graph can be derived as follows. Each node that has no edges connected to it (but may have edges connected to its anchors), has a text expansion that corresponds to a sentence in natural language. The text expansion of a node can be created by replacing the anchors in the node with the text expansion of the node connected to the anchor. The text graph in Figure 1 corresponds to the following set of sentences:

| Document is a File. |
| Folder is a File. |
| File belongs to a Folder. |
| Folder contains many Files. |

Despite their simplicity, text graphs turn out to have many different applications. The nodes may contain different kinds of text fragments denoting concepts, individuals, propositions, or questions. This allows the text graphs to be used for various purposes such as concept mapping, argumentation diagrams, or text annotation.
In addition, text graphs suit to different modes of use. The correspondence of a text graph with a set of natural language sentences can be used in both directions. On the one hand, an existing text can be analyzed to produce a text graph that reveals its conceptual or propositional structures. On the other hand, a text graph can be produced first and then be used to derive the text that describes the essential structures of the domain.

2 Background and motivation

We have developed the text graph notation for the students of our computer science courses. The goal is to enable the students to externalize the content of their cognition in an easy-to-learn but accurate manner. During the past few years, we have attempted to create new kinds of constructive assignments for introductory programming courses to improve the students’ understanding of the central programming concepts (see, for example, Nuutila et al. (11)). The idea of text graphs arose in autumn 2002 as a result of our earlier experimentations with mind maps (4) and concept maps (10; 8).

Before these experiments we used to give the students assignments to write essays about conceptually complex topics faced in the courses (e.g., “Exception handling in Java”). Essays, however, offer only an indirect way to develop conceptual understanding. Students end up spending time in producing fluent and readable text instead of focusing on the refinement of the concepts and their relations. In addition, the teachers waste time in separating the level of the conceptual understanding from verbose rhetoric and surface quality of the text.

The mind maps and concept maps provide a more direct way to work with concepts. However, mind maps — that consist of concepts and unspecified associations between them — soon turned out to be far too vague a notation for working with systems of concepts, especially in a technical domain such as computer science. It seems that mind maps are primarily a technique for taking notes and sometimes a useful memory aid.

Our experiences with concept maps were encouraging but not completely satisfactory. Concept maps are, however, the closest precursors of text graphs: the overall nature and intended uses of these notations are similar. Like text graphs, concept maps are easy-to-learn and are intended for the analysis and refinement of complex conceptual material. They seldom are finished with the first attempt but are revised as the understanding of the domain improves.

Concept maps were developed by Joseph Novak (10; 8). A concept map is composed of concept nodes connected to each other with named relation nodes. Each relation \( R \) between concepts \( A \) and \( B \) could roughly be read as “\( A \) is in \( R \) with \( B \)”. Figure 2 gives an example of a concept map.

![Concept Map Example](image)

**Figure 2:** A concept map describing the file system concepts

When we instructed our students in the use of concept maps and studied the maps they produced, we repeatedly encountered the following kinds of problems in this notation:

- **Semantic vagueness.** The relations seldom specify uniquely the intended reading. The readers must use their background knowledge and imagination to read a relationship. As a result, different people — and also the author of the graph at different points of time — may produce different readings of the same relationship.

- **Expressive inadequacy.** Fundamentally, concept maps are meant for representation of
concepts and binary relations between them. In our experience, the attempt to represent other kinds of relations leads to cumbersome or unnatural maps.

- **Concept/relation separation.** The relations can be used to make propositions about concepts, as in “File is a Folder”. However, it is not possible to specify relations that make propositions about other relations in a concept map. This separation between concepts and relations probably makes the use of concept maps easier in simple cases but reduces their applicability to other more advanced uses.

Åhberg (15) has presented some improvements to concept mapping. He stresses non-hierarchical nature of knowledge, the importance of accuracy in the labels of concepts and relations, and that each concept should appear only once in a map. One concrete technique is to draw concept nodes inside others that represent more complex concepts. Unfortunately, this is only a partial solution, since a node cannot be inside more than one other node.

Text graphs do not have these problems, as will be shown later on in this paper. Before that, we briefly contrast text graphs with other techniques that have similar outlook or purposes.

**Knowledge representation** research is an area where different kinds of graph notations for conceptual information has been developed. The goal in knowledge representation is to encode knowledge in a form whose meaning can be processed by a machine. In practice, when a graph notation is used, it should be possible to interpret it as a set of sentences in some logic language to enable the use of automatic inference procedures. Many different graph notations have been developed, with the name of *semantic networks* (1; 3; 5) or *conceptual graphs* as specified by John Sowa (13; 14). All of these notations are extensive and cannot be characterized as easy-to-learn.

It should be stressed that text graphs have not been intended as a notation for knowledge representation. While knowledge representation aims to transmit to a machine the kind of domain information that it can independently act on, text graphs are meant to aid the human cognitive processing. By constructing a text graph, a person can externalize the content of his or her cognition and in the process develop better understanding of that content. To serve this goal, the simplicity of the notation and a clearly defined meaning to the user are crucial.

Text graphs may also appear to have similarities with *hypertext graphs* that represent associations between (hypertext) documents. However, text graphs do not deal with inter-document relations but with the *internal structure* of documents. The most widely used hypertext system, *The World-Wide Web*, uses a variety of *markup languages* (HTML or XML) for the internal structuring of the documents. Markup languages are based on an implicit model that a text is an *ordered hierarchy of content objects (OHCO)*, i.e., the structure of a document can be represented as a *tree* whose nodes are *content objects* (for example, paragraphs, figures, or headings) and the children of each node always come in a specific *order*.

Renear et al. (12) argue that the OHCO model, even if refined and extended to cover multiple simultaneous hierarchies, is insufficient to model many important structures in text. Based on our experience it seems evident to us that the *relationships between concepts* or the *chain of reasoning* presented in a text are difficult if not impossible to represent in a hierarchical manner; instead, more general graph structures are required. Indeed, markup languages offer ways to define identifiers for the nodes in the tree and to refer to these identifiers as a way to break out of the hierarchical strait jacket. In text graphs this is possible in a simple and natural way without the overhead of notational tricks required in markup languages.

Recent developments of the *Semantic Web*, that combines knowledge representation techniques with hypertext systems, have some common traits with text graphs. For example, the basic model of the *Resource Description Framework* (7) is a graph structure of nodes (called *resources*) connected with edges. Each edge can be interpreted as a proposition composing of the *relation name* and the *subject* and the *object* (or value) of the proposition. Thus, in prin-
ciple it is possible to produce a natural language sentence that documents the meaning of the proposition. Moreover, the relations can be *reified*, making it possible to make propositions about propositions.

However, the emphasis in the Semantic Web research is in knowledge representation techniques and automatic reasoning. Its goals as well as tools differ from those of text graphs.

3 Structure and Meaning of Text Graphs

Since we have claimed that text graphs are easy to learn, and can be used to accurately present conceptual and propositional content, we decided to use text graphs below to describe the structure, properties, and meaning of text graphs themselves. The additional advantage is that we can present multiple examples of text graphs.

3.1 Basic structures

The basic structures of text graphs are described in Figure 3. The nodes that have different roles are shown in different graphic styles.

![Figure 3: The basic structures of text graphs and the resulting text expansion](image)

Note that the target of an edge may be another text graph. This allows the building of nested text graph structures. The nodes in a text graph can be in different roles. Figure 4 identifies two important categories of nodes: *root concepts* and *root sentences*. Root concepts are shown in Figures 3 and 4 inside rounded rectangles and root sentences without a border.

There can be nodes that are neither root concepts nor root sentences, as the “set of” nodes in Figure 3. The non-root nodes can be structural concepts, non-independent sentences, or parts of other sentences. If text graph only contains text nodes without any anchors, they are all both root sentences and root concepts.
3.2 Meaning through text expansion

A simple text graph is one where each edge connects a node and an anchor, each anchor is the target of exactly one edge, and the transitive inclusion relation contains no cycles, as specified in Figures 5 and 6. Note that the text graph in Figure 5 contains a subgraph. An edge connects the first anchor in the If-node to the whole subgraph; there are also edges that connect anchors outside the subgraph to nodes inside the subgraph. This is quite legal by the definition of Node in Figure 3.

Figure 5: Acyclic text graphs

The meaning of a simple text graph is specified with respect to the (natural) language used in the nodes. We define that the meaning of a text graph is the meaning of its text

Simple Text Graph is an Acyclic Text Graph where each Anchor is Target of exactly one Edge and one Target of each Edge is an Anchor and the other Target is a Node

Figure 6: Simple text graphs
The text expansion is a string containing expressions in a natural language and its meaning possibly depends on the interpretation of the people who read it. However, the following can be said: (i) Two different text graphs mean the same for an individual reader, if their text expansions mean the same for that reader. (ii) One text graph means the same for two different readers, if its text expansion means the same for them.

The text expansion of a simple text graph can be derived as specified in Figure 7. These rules assume that there is exactly one edge connected to each anchor and the other end of that edge is connected to a node. The text expansion of a text graph is unique, if the order of the sentences produced is not considered relevant.

Note that the conjunction of text expansions is handled differently in a top-level graph and in the subgraphs. In top-level, the text expansions of the root sentences are ended by full stops and concatenated, whereas in the subgraphs they are connected with the word 'and' (see Figure 5). There are some other linguistic issues in the forming of the text expansion, which are outside the scope of this paper.

**Figure 7:** A text graph describing the text expansion

A text graph is valid, if it has a text expansion and the size of the text expansion is finite. Every simple text graph has a finite text expansion (Figure 7) and is therefore valid.

### 3.3 Multiply connected anchors

As a shorthand it is possible to draw text graphs where anchors are connected to multiple edges. This kind of graphs can always be automatically converted to simple graphs as shown in 8. Therefore the text expansion rules shown above are sufficient to specifying also the meaning of these non-simple cases, as long as the graphs are acyclic.

Figure 8 shows how a node containing two multiply connected anchors (in the left) is split into a set of nodes (in the right) that is the cross product of the connected nodes.

When one node is transformed into multiple singly connected nodes, it is possible that other nodes will get multiply connected anchors as the result. The transformation must thus be propagated in the text graph but only toward the root sentences. The root sentences may
be multiplied but that does not result in any additional multiply connected anchors. Therefore the propagation always terminates.

As a consequence, every acyclic text graph is valid.

3.4 An example

Figure 9 shows a text graph that contains the Peano’s five postulates. It should be noted that the four central concepts are clearly shown. The use of the concepts in the postulates is clear and it is easy to see that natural number is the central concept. An attempt to represent this content with a concept map would lead to a very unnatural map.

4 Text graph tool

Creating a graphical presentation like a text graph by paper and pencil is a tedious task. It is difficult to determine in advance the space requirements and the relative positions of the different parts of the presentation. This problem is greatly magnified when the understanding of the presented structure evolves over the time. The manual labor and the problems with the layout may take most of the capacity of the author.

To enable the authors to pay most of their attention on the conceptual content of the subject matter, a suitable drawing program is a necessity. There is a freely available drawing program called CMapTool (9) which in our experience is indispensable when working with concept maps. However, it is not completely adequate for text graph editing. Text graph editing requires means to flexibly modify the relative positions of the elements so that the edges remain connected to the nodes and anchors. In addition, it requires support for nested graphs and fluent editing of mixed content of character strings and anchors. CMapTool does not satisfy the latter requirement. Common drawing tools lack both of these capabilities or provide them in a cumbersome manner.

To satisfy these requirements, we have implemented a text graph editor called TGE. The figures in this paper have been drawn with TGE. The basic capabilities of TGE include node creation and deletion, editing of mixed content of character strings and anchors in an Emacs-like manner, creation and deletion of edges, and moving the graph elements maintaining the topological relations.

More advanced functionality of TGE includes:

- Direction constraints for the edge segments and automatic constraint propagation. This allows the creation and maintenance of edges that consist of orthogonal segments.

- Node and edge styles. In the figures in this paper, the core concepts and the core statements have different styles to highlight their different roles.

- Unlimited undo and redo functionality, and complete logging of the editing operations. In addition of helping the editing task, these capabilities make it easier to have different versions of the same text graph.

![Diagram](image)

**Figure 8:** An example of a multiply connected anchors and resulting simple text graph
- Multiple simultaneous graphical layouts of the text graph. With this functionality, the user can experiment with different layouts for the same graph, while maintaining the topological structure and the textual content of the graph intact.

TGE is implemented in Java. It stores the text graphs and editing logs into Scheme files, and uses Kawa, a free Java-based Scheme system (2) for processing the Scheme expressions. Currently we have only a test version of TGE, but a more reliable product version will be freely available later.

5 Summary

We have introduced text graphs, a simple notation to represent complex conceptual content. Their goal is to aid human cognitive processing, for instance, in educational settings. The main advantages of text graphs over previous notations with similar goals are accuracy and well defined meaning. These properties make text graphs especially suitable for technical domains like computer science. We have also implemented a text graph editor called TGE.

We intend to use text graph notation and TGE tool in our future computer science courses. We anticipate that this tool will also be used in other domains such as argumentation analysis (6), knowledge management, and initial stages of the development of ontologies.

References


C-cards: using paper and scissors to understand computer science

A. Valente

Aalborg Universitet Esbjerg, Niels Bohrs Vej 6, DK-6700 Esbjerg, Danmark
av@cs.aue.auc.dk

Abstract

We define a simple card game, where cards are computational elements; computing machines can be defined, built and animated in a concrete way by disposing cards and moving pegs around them, following formal rules. We discuss how to use this card game as an educational tool, to introduce children 8 to 10 years old to the concept of computation, seen as manipulation of symbols. Students will not need any mathematical knowledge to explore information theoretic concepts by means of our tool; moreover the students can easily expand the tool with new components for exploring new concepts. Graph-based syntax and graph-rewriting semantics are given to formalize cards and their behaviour.

1 Introduction

In this paper we aim at defining an educational tool, suitable to teach basic computer science concepts to classes of 8 to 10 years old, or high-school students without a strong mathematical background. Such a tool has a number of desirable features:

• no mathematical knowledge needed to understand and use it. In CS courses, computing machines are classically introduced using mathematical definitions and exercises are usually based on algorithms taken from number theory (e.g. factorial, Fibonacci). This approach does not scale to children.

• the tool should offer a concrete, mechanical metaphor for reasoning about computation: it should enforce the idea that computation is a property shared by many different systems, some of which made of a few simple pieces. We think formal semantics should be hidden inside real world objects, that can be animated in game-like simulations.

• direct manipulation and learning-by-doing. The tool should enable students to define their own computational elements; it should also encourage the exploration of problem-solving and design methodologies (top-down and bottom-up project development, for example).

• cheap and flexible support (paper), wide availability (web-based simulator) and individual learning. The tool should be both web and paper-based: in this way even when it is not possible to have a PC for each student, the tool could be used on individual basis. An optional advantage of a tool made of paper, is that in the era of Internet, distribution will be inexpensive.

Taking all these requirements into account, we conceived a card game, with both a paper and a javascript implementation (both available at the author’s home page). The game has a precise, formal set of rules and it is suitable to build and study various computational devices: we call this game c-cards, that stands for computational cards.

The cards are introduced informally in section 2, via some examples; in section 3 c-cards are provided with graph-rewriting semantics. Then we discuss how to turn c-cards into an educational tool, i.e. how to express exercises for young students; we give some insight about what teaching areas can be covered by our tool. We present the web implementation of our tool in section 5, and finally we discuss some related projects and some problems that need further investigation.
2 The deck

A card is a square piece of paper (see Fig. 1, on the left); directions, and card sides will be addressed as north, east, south and west. Sometimes cards will be labelled, i.e. named, to address them more easily in the discussion. Each card has two ports on each of its four sides: an input port and an output one, so that when two cards are laid aside, their ports connect in a complementary way to form a complete port (Fig. 1, on the right). The area inside a card can contain arrows, going from an input port to an output one, defining the behaviour of the card.

The basic c-cards set is called the deck (Fig. 2). It is composed of seven card types: the peg-source card and the peg-pit card (first row); the cross card, the east and west turn cards and the confluence card (second row). All the cards in the second row of Fig. 2 are basically pipes, useful to build stational circuits on which pegs will move. The switch card (last row of Fig. 2) is the only one double-faced: after cutting it, we have to fold it along the thick vertical line. Together with these seven card types, there is a peg (the small rectangle at the far right on the first row).

![Figure 1: A generic card.](image1)

![Figure 2: A basic c-cards set: the deck.](image2)

To play a game at c-cards, we have to compose a card circuit, by connecting cards together; every circuit must have at least one source and one pit card. To execute a computation, a number of (identical) pegs will be placed on the source cards of a circuit; then we will move them, one at the time, to represent the flow of information traversing the circuit. Pegs will run through the cards following arrows, from port to port, until they reach a pit card, where they must stop.

By now we leave parallelism aside, for the sake of simplicity, and we assume that pegs are released sequentially by source cards, each after an arbitrarily long delay: in this way we can reason as if a single peg at the time is ever present in a circuit. Under these assumptions about execution, our card circuits will have a deterministic behaviour.

Source cards contain (possibly empty) queues of pegs, being scheduled out southwards; every source shall then be described by a string, or a regular expression, defining the (finite) sequence of pegs it will release. E.g. a source defined by (white.black)* produces a finite sequence of alternating white and black peg; to specify the precise number of pegs produces, a string can be used instead, e.g. white.black.white.
The last card type we have to detail is the *switch* card (third row of Fig. 2), the only one capable of changing its state dynamically. A switch card has two faces, each describing a specific state and behaviour. When a card of this type is placed in a circuit it is in its *initial state*, i.e. the first peg arriving from north will exit westwards; when leaving the card the peg will force a *flip*: the switch card will be flipped in its other state (face). A second peg arriving from north will then exit eastwards. The state of the card will flip back to the initial one, and so forth.

The switch card represents a simple form of memory (similar to a flip-flop boolean circuit) and is the only active computational element in the deck.

### 2.1 A card circuit

To demonstrate the behaviour of c-cards, we compose a circuit (Fig. 3) with two pegs placed on the *In1*-labeled source card, and one on the other, labeled *In2*. After a number of steps, the two pegs will arrive at the *Out2* pit, while the other one will stop at the *Out1* pit. In the figure all unreachable ports are light-gray, to simplify spotting the two main paths pegs can follow in the circuit.

The action performed by this circuit is to *swap* its inputs: in fact we shall say that pegs starting from the *In1* source card are in state TRUE, while pegs starting from *In2* are FALSE pegs. For the same reason, a peg stopped at *Out1* pit will be in TRUE state, and FALSE if it ends up on the other pit.

![Card Circuit Diagram](image)

**Figure 3**: The not card circuit.

The whole circuit should then be seen as a boolean *not gate*, because it turns TRUE pegs into FALSE ones, and vice versa. This suggests that boolean operations are realizable with c-cards, although we know that c-cards do not reach Turing-completeness. However we feel that boolean operators are still too *mathematically oriented* for young students, so we limit the discussion of c-cards in education to regular languages and simple pattern-recognition exercises.

We also want to draw attention to an interesting concept related to our card circuits: *time-reversibility*, closely connected with the idea of representing information by means of physical objects. Standard boolean gates create and destroy boolean values as they compute; a gate in which no bits are to be lost or freshly created during computation, is called *time-reversible*, meaning that we could totally recover what the input was using the output. All the circuits that we can define with c-cards have this property (for a presentation of time-reversible gates and their expressive power, refer to Fredkin and Toffoli (2)).

Finally we would like to remark that although all pegs are treated as identical by our cards, pegs can came in different colors, with labels on them and possibly in different size, but
only for helping the user spotting them and following them in their way through the circuit. Only their physical location, or better their arrival direction, can be used as information by cards, and this is why a circuit changing peg positioning, is executing a computation. Pegs in c-cards have a status similar to the one of tokens in Petri Nets (see Ajmone Marsan et al. (1)), but more persistent. Moreover these circuits bear similarities with flow graphs (see Milner (5) and Gardner (3)), because all these graphical notations express movements of information flowing into and out of ports, in a topology of communicating neighbours.

2.2 Shapes, macros and custom cards

The shape of a circuit is defined as the structure obtained by replacing every card, different from source and pit, with a blank square of paper. Arrows can be drawn freely on these blank parts to show the pegs’ paths in the circuit.

![Figure 4: A macro-card and a card.](image)

The shape of a circuit is its functional specification, and reasoning about shapes (i.e. incomplete circuits) is useful to encourage top-down analysis and modelling in students.

A standard feature of programming languages is the possibility of definingacro-instruction, and use them to write more readable and compact programs; macros, automatically expanded by translators and compilers, encourage a simple form of reuse and abstraction.

The corresponding concept in c-cards is the one of macro-card. Once a circuit is defined and we like to use it repetitively to build more complex circuits, we can map the whole circuit it into a single large card, preserving the circuit’s behaviour. The idea here is simple: source cards became input ports, while pits are turned into output ports; arrows (and some informal annotations about actions to be performed) will express the high-level behaviour of the large card. The macro card in Fig. 4 on the left has the same behaviour of the not-circuit: its ports are labelled after sources and pits of the original circuit, and two crossing arrows represent the swapping of incoming pegs.

It is also possible to define new custom cards behaving as a whole circuit, although in these cases the geometry of the original circuit is hard to preserve. Since cards are realized in the real world as paper squares, students can easily define their own cards. An example of custom card is visible in figure 4 on the right: the new card will behave roughly like the not-circuit.

3 Graph-rewriting semantics

Cards and the way they can be connected to build circuits form the syntax of the c-cards game. Here we give a graph-rewriting semantics to our tool. As a card is mainly a set of ports (of two types) plus some arrows connecting them, we can consider a card circuit as a graph of ports, each one possibly containing pegs.

In Fig. 5 we show the rewriting rules for source and pit cards (first row), cross and switch cards (second and third row respectively). All rules are intended to work up-to card rotations, this explains why there is only one rule for pits, while otherwise four should be needed. Card rules are deterministic, triggered by the presence of pegs, and depend upon the current state of cards (in fact the switch card has two rules). We omit rules for turn and confluence cards because they behave essentially as the cross card: pegs movements are directed by arrows.

As c-cards have a sequential execution model, no more than one peg can be present on a single port at any given instant; to simplify design we assumed from the beginning that pegs
flow in the circuit one at the time. We think that graph-rewriting rules are rather simple to explain to young students and fit naturally with our visual and physical idea of computation.

Moreover we believe it is evident that children can easily manage games with formal rules (e.g. the goose game or RPG games) therefore hide formal aspects behind a game-like look seems a good, general trick for designing educational tools.

4 Educational applications

Wanting to organize a basic course in CS, and adopting c-cards as our main tool, we can follow ideas from the pedagogical pattern project (7): at the beginning of the course some simple statical circuits (such as the not-circuit) can be shown and animated with pegs, so to explain the behaviour of all seven types of cards through direct manipulation. This strategy conforms to the spiral and early bird pedagogical patterns.

After this first, hands-on introduction to the c-cards tool, some top-down modelling examples should be shown: a shape, together with the specification of its desired behaviour, can be proposed to the class. The solution of the exercise is an implementation of a card-circuit performing the required task.

As an example of such an exercise, let us consider the shape in Fig. 6 (on the left), where the In source card is defined by (white,black)*, and the desired circuit have to separates whites from blacks (we recall that colors are meaningful only for users, not for the cards). If we count pegs as they exit from their source card, here white pegs will be odds, and the black ones even; a circuit implementing these specifications correctly must make white (odd) pegs stop at the Out1 pit, and black (even) ones at the Out2 pit card. Students are asked to give an implementation of the shape-specification; a possible answer is depicted in figure 6 (on the right).

A mathematical description of this exercise could be to implement a circuit counting modulo 2, but we don’t need to use such terminology with children, since speaking of patterns of pegs is expressive enough to specify the correct behaviour of the circuit.

![Figure 5: Rewriting rules for c-cards.](image)

![Figure 6: A shape with specification, and a possible implementation.](image)
This is one of the main reasons to propose c-cards for teaching to young students: no mathematical knowledge is needed for reasoning about card behaviours, nor to build them. Patterns and physical manipulation of symbols, i.e. pegs, seem to be general enough concepts to introduce the study, construction and use of computational machines.

Suppose now we want to make a more complex exercise, working with a source defined by \textit{(white.slashed blacklist)*}. The resulting circuit will have three pits, in which pegs of the same color will accumulate: the first (western) pit will receive all white pegs, the second all the slashed ones and the black pegs will stop at the third. The circuit in Fig. 7 implements a solution for this exercise.

There is a general technique (that can be discussed with students), for building a \textit{modulo k} peg splitter, for any given \(k\). When \(k\) is a power of 2, we simply have to put a cascade of switch cards, forming a tree with \(k\) levels and placing the pits in the correct positions; note that by using some copies of the not-circuit, we can always re-arrange the final (southern) part of a circuit so that pits are in any desired east-to-west order. With this method will have a pit card collecting each class of integers \textit{modulo k}; in fact for all other values of \(k\), we can start building a circuit for the first power of 2 larger than \(k\), then short-cutting all pits in excess.

As these \textit{modulo k} circuits should be widely used to implement bigger circuits easily and in a modular way, it seems reasonable to convert them into macro-cards, and consider them as off-the-shelf components.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{circuit_splitting_modulo_3.png}
\caption{A circuit splitting pegs modulo 3.}
\end{figure}

We can decide to name \textit{k-color splitters} these macros, in this way the card in Fig. 6 will be called \textit{2-color splitter}.

The possibility to use macro-cards and forgetting about their real implementation (i.e. the circuit they are based on), together with the idea that different circuits can solve the same problem, enables the teacher to discuss \textit{optimization}. A circuit implemented using macro-cards, or better its \textit{expansion} (i.e. the whole circuit obtained replacing every macro with the actual circuit it represents), can in the general case be simplified, without changing its functional behaviour. Investigating in this direction, students can be guided to understand, in a practical way, the meaning of \textit{complexity} and \textit{performance} in computing machines.

4.1 Frequencies, probability and transmission

In order to design exercises about probability, information and transmission theory, we propose a simple extension to the basic deck: a \textit{random card}, that is the non-deterministic version of the switch card. The random card is shown in Fig. 8.

A peg entering a card of this type will be sent out towards east or west, with \{1/2, 1/2\}
probability distribution. Again the students do not need to know probability, they can simulate the random choice needed by this type of cards, by tossing a coin.

After a small number of trials, they should be convinced that half of the pegs will leave the random card from the output port and half from west: in this way we can use frequency instead of probability for discussing about these cards and the circuits that students will build with them. Using two or more of these cards in cascade, we can implement almost every probability distribution needed; macros and custom cards can be devised to create specific probability distributions.

A possible exercise for the class about the use of random cards can ask to build a circuit for dividing white pegs from black ones, generated by the source (white.black)*, with 50% probability of making an error (note that this is the random version of the modulo 2 counter discussed in the previous section).

When the students have realized that a single random card is sufficient to solve this exercise, we can propose them the exercise in Fig. 9 part A, called the pegs’ telegraph.

We suppose two friends want to communicate sending pegs to each other. Pegs come in two colors: white and black. The circuit in Fig. 8 is incomplete and we want the students to design the grayed part so that there is exactly 50% of errors in sending the pegs. The solution is shown informally in Fig. 9 part B, at the bottom (the top part shows the design to be used to have a perfect transmission). Here an error is defined as sending a white peg and receiving a black one and vice versa, i.e a peg starting from the source card labelled Sender’s White arriving at the pit card labelled Receiver’s Black and vice versa.

This last exercise demonstrates how, with the simple extension of the random card, we can discuss of data transmission and errors within c-cards. The concepts of redundancy, message coding and error detection can also be introduced using similar exercises. We can propose the class to modify the circuit in Fig. 7; the modified circuit should work sending a couple of pegs, like blue.blue, instead of the Sender’s White, and red.red instead of the Sender’s Black.

The receiver will then get blue and red pegs, and will have to consider couples to understand the original message sent from the sender.

In this modified circuit there are illegal couples: couples of pegs that can be received but that could never be sent, like red.blue, which is obviously caused by a transmission error. We

![Figure 8: The random card type.](image)

![Figure 9: (A) The pegs' telegraph example. (B) Solution.](image)
can show the students that illegal couples are the result of our decision of using a code with two pegs for each one of the original message, and they give us the ability to detect errors.

We are currently investigating possible ways to give a visual meaning (using c-cards circuits) to Shannon’s information measure: we would then be able to reason about most of Shannon’s results within the c-cards tool, and most important without mathematical notations (e.g. Shannon’s entropy for a source of symbols is computed using logarithms and probability distributions, which are very complex for children to manipulate and understand).

5 Web implementation

As we write in the introduction, we believe that an educational tool for teaching young students should both be physical and computer/internet based. This is why we developed c-cards to have both a paper and an html-javascript implementation.

Fig. 10 shows E-Si (pronounced easy), the c-cards’ editor and simulator. The program (available at the author’s home page) runs entirely on the client machine, so it can be used on-line or downloaded and deployed in a school’s lab.

![Figure 10: E-Si’s interface.](image)

The current version of E-Si is a simple experimental realization and implements only the basic card set; it enables the user to design, save/reload and test circuits.

In the next version we plan to add the definition of custom cards and the possibility of executing circuits with multiple pegs.

6 Related work

It is kind of difficult to locate the subject of this study: toy-making, educational and computer science issues have been closely related in developing c-cards. However there are other works that go along the same interdisciplinary path. The requirement for our educational tool to be a physical toy, can also be found in the concept of tangible programming (see McNerney (4)), defined as the activity of arranging the blocks to build (as opposed to "write") computer programs. As we are discussing about toys as didactic tools, we want to cite Papert’s constructivism (6), that in his words, attaches special importance to the role of constructions in the world as a support for those in the head; he also maintains that children should learn better by self-directed activities, such as bricolage and thinking, both closely related to c-cards.

For the methodology to adopt when using toys in education, we found that our ideas match with a pedagogical pattern called toy box, presented in Sharp et al. (7) by the pedagogical pattern project: it aims at giving students broad historical and technological knowledge of the field by letting them "play" with illustrative pedagogical tools.
7 Conclusions and future works

In this article we presented c-cards, a card game with formal rules, and discussed its application as educational tool suitable for teaching computer science concepts to young students.

With c-cards students can create and manipulate their own computing machines, reason about design and implementation of card circuits. Also concepts like probability and information (intended as Shannon's entropy) can be discussed on a concrete basis using a simple extension of the standard card deck; likewise we showed how to set up experiments about redundancy and transmission errors.

However there are still many problems (and spin-offs) that we would like to investigate:

- Measure/prove the real computational power of c-cards.

- Give a parallel semantics to our cards; will they become similar to cellular automata? Shall we then be able to model and study classical concurrent systems (such as the dining philosophers) with our tool? In fact parallelism and concurrency are considered a fairly complex subject for bachelor students, so scaling them to children is an intriguing goal.

- An implementation the cards in a mainstream educational language such as Logo/StarLogo to see whether it is possible for the students to pass smoothly from cards to a real, universal programming language.

- Design a three-dimensional, real-time implementation of c-cards, based on connectable cubes, that could compute automatically, by gravity; pegs should be replaced by balls running through cubes. Can such a tool be made of paper? Could we use plastic parts to create the equivalent of the switch card?

- Enrich the number of exercises and examples for c-cards, to build a teaching methodology on top of it; we feel that reverse-engineering exercises should play a major role in the methodology. We wish to be able to draw inspirations from automata and language theories (deterministic finite automata for example), Petri nets and queue networks.

References


Assessing Time-Efficiency in a Course on Data Structures and Algorithms

H. Hansen and M. Ruuska

*Tampere University of Technology, Institute of Software Systems, PO Box 553, FIN-33101 Tampere, FINLAND*

hansen@cs.tut.fi

Abstract

Time-efficiency is one of the key points in the design and analysis of data structures and algorithms. We study the use of two methods for assessing time-efficiency of student programs. The benefits and drawbacks of asymptotic analysis and time measurements are studied. The effects of changing the test data used in assessment and of differences in asymptotic requirements are studied by using student programs from two years. The design of test data is found out to be critical in cases where asymptotic requirements are absent.

1 Introduction

In a course on data structures and algorithms one is expected to learn theoretical and practical aspects related to the subject. The practical part is the proper use and implementation of data structures and algorithms. Students should implement programs of their own at some point in order to fully appreciate the practical aspects of the things they have learnt. These programs then have to be assessed from the point of view of correctness and efficiency in order to give feedback to the students. Some requirements on the efficiency of programs are necessary, as in most cases a problem may be solved with a naive algorithm, from which there is nothing to be learnt. Efficiency aspects may be considered essential also when learning basic programming, as stated by (2).

An assessment method for efficiency should meet several criteria (6). It should be fair and focus on issues relevant to the purpose of the course. Its use should not require too much time from the staff so that they may focus more on instructing the students. The feedback given to the students should be instructive and support the learning process and it should be provided with as little delay as possible.

Some of these requirements for the assessment method are, in fact, conflicting. In our research we have studied the use of two kinds of approaches to the assessment of efficiency on a course on data structures and algorithms.

The first kind is common in theoretical context, e.g., algorithm textbooks. An upper limit for time consumption is given in $O$-notation ("Big-Oh") as a function of the size of the input (3). This can currently only be evaluated reliably by actually having someone read the code and analyse it by hand. The other possibility is to measure the time taken by the program when it is given some suitable test case and then compare it to some benchmark value. This can be done automatically if proper test data is available.

In our research we took student programs from two previous years of our course on data structures and algorithms and analysed how the measurements and asymptotic requirements relate and how changes in test data or asymptotic bounds affect the quality of submitted programs. We found out that improving test data when asymptotic requirements are present does not have a dramatic effect on the distribution of programs. On the other hand, when there are no asymptotic requirements, the quality of the test data is absolutely critical. Also we found out that a tightening of asymptotic requirements has a similar effect as improving the test data.
2 Practices of the course

There are two alternative courses on data structures and algorithms in our institution. The
course we refer to in this research is the more advanced of the two. There are three mandatory
programming assignments on the course, two preparatory ones and a third one which is the
real challenge. In the final assignment, each student has to implement one randomly selected
problem from a set of ten.

Currently all assignments are assessed as either fail or pass, with possible comments and
tips for further development from the teacher. A system for computer-assisted learning, pre-
sented in (1), is used in handling the programming assignments. The system tests correctness
and, partly, efficiency of programs submitted by students.

The problems fall roughly in two categories: those with restrictions on asymptotic behav-
ior and those without such restrictions. The efficiency of programs is first checked by
time measurements in which the time consumption of the student’s program is compared to
that of a model program. When there are asymptotic requirements, the assistant responsible
for grading also has to analyse the asymptotic behaviour of the program and check whether
or not it meets the requirements.

Some problems have a fixed maximum size for the data structure. In such cases the student
programs are not required to be able to handle problems larger than the given limit.

Currently the time efficiency is measured when the student submits a program on a ma-
chine of his or her choice. The program is compiled with a compiler using code optimisation,
runtimes and its running time measured. To compensate for differences in the running envi-
ronments, the model program is also run on the same machine and its running time is measured.
These two times are then compared and a feedback is decided on. If the time consumption of
the student program is more than ten-fold compared to the model then the program fails and
the student is asked to make it more efficient. If the time is more than three-fold, the student
is warned that the program is possibly slightly inefficient, but is within accepted limits. If
the time is three-fold or less, the student is congratulated on an efficient program. It is also
possible that the program does not finish in a reasonable time and the execution is aborted,
in which case the program fails and the student is told that the test was not finished in due
time.

During the spring of 2003 we adjusted the system to give more detailed feedback. The
measurements made in the submission phase are recorded and students are allowed to view
the time consumption of their own programs and compare it with that of the model and
results with other students’ programs. This type of arrangement has been welcomed by the
students as inspiring and informative.

3 Selected problems and measurements

We measured the time consumption of the student programs from springs of 2002 and 2003
using the test data from 2003. We singled out four problem assignments we consider to be the
most representative of the issues in our research. One of the problems (C1) only has programs
from spring of 2003, as it is a new assignment.

To make our measurements more readable we divided the programs into five groups accord-
ing to how much time they took in the measurements. For the programs of year 2002 (drawn
in white), the middle group (group three) consists of those programs that are within ten per-
cent of the median time. Group one contains the fastest programs whose time consumption
is less than half of the median. Group two has the programs whose time consumption ranges
from 50 to 90 percent of the median. In group four the time consumption ranges from 110 to
150 percent and group five has the programs that take more than 150 percent of the median
time. In the figures, the height of the bars denotes the number of programs belonging to the
corresponding group.
The programs from 2003 (drawn in black) are grouped using the scale obtained from the 2002 programs and the results are drawn in the same diagram to show how the time consumptions of the programs differ.

3.1 Problem C1

In problem C1, the student program has to handle large amounts of text input. The assignment contains a mapping from numbers to sets of letters. The program is given a set of words and a set of numbers. Each number corresponds to a subset of words from the set of words according to the mapping given in the assignment and the program must output this set of words. The program is not given any asymptotic bounds and any program that passes the efficiency measurements is considered efficient enough.

We took problem C1 under investigation with the idea that the data structures and algorithms used by students might not be the most important factor in time efficiency. The problem is such that a simple hash table would be quite sufficient to produce an acceptably efficient program. As the inputs consist of sequences of strings it seems plausible that I/O might have a significant effect on time consumption in this problem.

The distribution of the programs is relatively wide, as shown in Figure 1. We found out that in groups one and two both the data structure (either a hash-table or character tree) and I/O were quite efficient. In particular, the fastest program uses a very efficient I/O method. In group three, the data structure was efficient in most instances and the I/O was neither particularly efficient nor inefficient. The solutions in groups one to three were mostly $O(n)$.

The majority of the programs from groups four and five were implemented using some sort of binary search-tree structure and one (the slowest) used an extremely inefficient solution (a list of words). In addition, in most of these programs the I/O was implemented using library functions that are known to be somewhat inefficient, except the most time consuming program in group five which would not have passed the test were it not for an extremely fast use of I/O. All programs in groups four and five use at least one inefficient solution, either for I/O or storing data. The latter were mostly $O(n \log n)$ solutions (the word list is $O(n^2)$).

So, even if the I/O is probably the most important factor, the “bad” solutions tend to have both slow I/O and slow data structures.
3.2 Problem C6

In problem C6 the program must calculate the sliding median of a sequence of numbers. The students are given some hints about possible data structures and there is a strict requirement of $O(n \log k)$ on the asymptotic behaviour of the program, where $k$ is the window size and $n$ is the total length of the sequence. There is also a limit of size 1000 for the window.

We chose this problem, because the test data for the time measurements was changed in between the two years. We had found out that in 2002 the test data was inferior and did not contain any instances with $k > 3$. Therefore it did not test the data structures but mostly just I/O. In 2003 the tests were redesigned to test the case for 1000 numbers.

As we can see from Figure 2, the change of test data has not had a big impact on the distribution of time consumption of the programs that have passed the test. This is, we believe, due to the strict asymptotic bound given in the assignment. When we ranked the 2002 programs in terms of time taken with both test data as in Figure 2, there was a marked difference between the test data of 2002 and 2003. The program that ranked first with the 2002 test data ranked 10th with the 2003 test data and so on. So, even if the tests in 2002 did not measure efficiency correctly, this had no impact on the actual efficiency distribution of the programs. One might go so far as to say the measurements were redundant, as the asymptotic requirements made sure that no inefficient solutions could pass.

3.3 Problem C9

C9 is a graph search problem. The input contains id-numbers of courses and for each course a set of predecessors is given along with the information of whether the course is given in the spring or fall semester. The program should calculate the earliest possible time in which a requested course may be taken. There are no limits to the size of the input and no asymptotic requirements.

We chose this problem for similar reasons as we did the C6-problem. Although the test cases were big enough, they did not contain cases that form a dense graph. The tests for 2003 were redesigned to cover cases that are quite tricky and, if the algorithm is not chosen carefully, result in exponential time consumption. The size of the data increased by approximately 50 percent. However, for many student programs the increase in time consumption was much more. A majority of programs from 2002 would not have passed the test in 2003. Moreover,
Figure 3: Distribution for problem C9 and rankings of 2002 programs tested with both 2002 and 2003 data

A closer inspection of the programs revealed that in 2003 all the programs were of almost the same asymptotic order, with mostly logarithmic differences.

As we can see in Figure 3, the time consumption of the programs from 2002 is in general much worse than that of the programs from 2003 and the distribution is also very wide. The difference between the fastest and the slowest program is about 50-fold. All of the programs from 2003 go to group one. It can be seen from Figure 3 that the number of programs that eventually passed in 2003 was smaller than in 2002.

When compared to the result obtained for problem C6, we can see a marked difference. In the case of C9 the better test data resulted in a dramatically higher efficiency standard for passing, which in turn caused some negative attitude among students. No such effect occurred with problem C6. The only explaining factor seems to be the lack of strict asymptotic requirements. The programs that seem exceptionally slow with the new test data are almost invariably exponential in their worst case behaviour.

Perhaps surprisingly, though, the rankings did not change as dramatically as it was the case for C6; The ones measured as fastest with the old data are more or less the same ones as measured with the new. Even when the old test data did infact measure efficiency to a reasonable extend and the cases were big, it was not carefully designed to detect the really troublesome dense graphs.

3.4 Problem C10

C10 is an optimisation problem, where the program should find a maximum profit for renting a ballroom. The assignment restricts the size of the input to 1000 events at a time. As a consequence, testing can not differentiate well between programs of different asymptotic order.

In this problem, the test cases were the same in both years, but the asymptotic requirements were changed from $O(n^2)$ to $O(n \log n)$. We can see from Figure 4 this had a similar but less dramatic effect on the programs as the redesign of test data in problem C9. The measured times of 2003 are clustered around the fastest programs of 2002.

4 Comments on asymptotic analysis

The traditional method of assessing time efficiency in computer science is asymptotic analysis. The order of time consumption of a program is analyzed by reading the source-code and
analysing the behaviour in detail. It requires a lot of detailed work by the evaluator and it may take a lot of time before the evaluator will be able to provide feedback to the students.

Asymptotic analysis does not take into account differences in the smaller details of implementation, but it narrows down the possible choices for design solutions and may guide the students too forcefully towards a particular type of data structure or algorithm. This is not exclusively beneficial to the learning process. On the other hand, understanding the theoretical considerations of efficiency is one of the main objectives of the course and asymptotic requirements encourage the students to analyse their solutions before actually implementing them.

During the assessment process valuable information about the quality of the student’s program is gained, as the evaluator has to do a detailed analysis. This information can be used to give complex feedback, the effects of which were studied in (4).

5 Comments on measuring clock-time

A simple method for estimating time efficiency is to measure the time consumed by a program while performing a given task. Our approach is to compare the measured time to that of a benchmark program. These empirical measurements of time consumption are quite easy to use, can be fully automated and provide immediate and concrete feedback to the student.

Both the efficiency of the reference program and the quality and size of the test data are critical to the success of the assessment process. With the test data it is crucial that the program is given a large enough instance of the problem to bring out the order of time consumed by the program. The problem remains, however, what it means that the instance is “large enough”. As we saw with problem C9 (Figure 3), a large test data alone does not guarantee anything. The design of test data requires an understanding of the problem and possible solutions.

The load on the machine at the time of submission has a tremendous effect on the measurements. Uncompatible measurements result when students use different computers for submission. These problems appear when immediate feedback is required, although we were of course able to eliminate them in our measurements.

There is however a more serious problem in the process of measuring time. As the computing power of computers has increased dramatically over the years, ever bigger test sets are needed to make the effect of the chosen algorithms and data structures measurable. Large test data and powerful processing easily produce a situation where most of the time, the program is actually performing I/O. In a language like C++ (which is used in the course), there
are various methods for reading the input and there are differences in the efficiency of those methods. This may lead to a situation, where most differences in time consumption between programs submitted by students are due to factors almost totally unrelated to algorithms and data structures. This seems especially to be the case in assignments that have to deal with strings of text and the problem itself can be solved in linear or almost linear time as we saw in Section 3.1. This is a serious issue, because elegant and generic programs typically use high level I/O-solutions and are therefore slower than clumsier low-level implementations.

Competition has a tendency of making people try a little harder, and this has been applied to learning as well (5). It is possible to use the measurements to encourage competition among students by making the measurements visible to all students. There is, however, a risk that this type of numeric feedback from the measurements encourages students to resort to implementations that are of poor overall quality. Even more generally, it seems highly questionable to encourage students to resort to methods totally unrelated to the course material. We plan to investigate this in the future.

6 Conclusions and further research

It would appear that in assignments with a strict asymptotic time bound like our problem C6, the measurements are not of a great importance. Typically any program that has the correct asymptotic behaviour is efficient enough in any case, and the differences in execution times are in the order of tens of percent. Strict asymptotic requirements seem to guarantee a relatively even quality in the programs.

On the other hand, if the model solution and test data are of good quality and the problem assignment is complicated, like the 2003 version of problem C9, the less laborious measurements provide reasonably reliable results. Even though the measurements cannot provide absolute certainty of the asymptotic bounds of programs, under suitable conditions only programs of the same or almost the same order as the model should be able to pass the measurements.

One of the goals in our research was to study the possibility to use the measured time consumption as a factor in grading and thus encourage students to pay more attention to efficiency aspects. In the light of our results it seems apparent that this approach has serious problems and cannot be used as such. There are too many factors unrelated to data structures and algorithms having too great an impact on the measured time. There is hope, however, because most of these unwanted differences are due to I/O solutions. It is possible to provide students with a module that handles the I/O and only leave the implementation of the algorithms and data structures to the student. After this, the differences in time consumption should be due to solutions relevant to the course material.

In order to guarantee some standard of time efficiency of students' programs, it would be possible to use either one of the assessment methods we have studied here. From the point of view of the learning process, however, it might be justified to use a suitable combination of the two, as they result in different type of feedback. Asymptotic analysis provides complex and theory-oriented feedback with a delay, whereas measurements provide concrete and immediate quantitative feedback. Asymptotic analysis of the programs can also be used as a method of checking that the time measurements differentiate between efficient and inefficient solutions. This was actually the way we discovered that the test data for some of the assignments was inferior.

References


Software Mathematics as a Course Topic

A. Valmari
Tampere University of Technology, Institute of Software Systems,
PO Box 553, FIN-33101 Tampere, FINLAND
ava@cs.tut.fi

Abstract

Although it is widely believed that mathematical background is important for producing software, programmers and software engineers do not seem to appreciate mathematical education. Indeed, it seems that traditional continuous mathematics is of little help in software production. Discrete mathematics has proven more useful, but even its success has been moderate, despite of good textbooks that are intended for computer science students. In this article it is argued that the best service that mathematics education can do to software professionals to-be is to give them tools and skills for handling abstractions with rigour. Then a new software mathematics course that was designed with this goal in mind is introduced. Although the course partly contains common topics with traditional discrete mathematics and theoretical computer science courses, it commits itself to the point of view of software professionals, and therefore presents the material in a different way. First experience with the course is discussed.

1 Introduction

Opinions regarding the importance of mathematics in the software profession vary. It is clear from recent public discussion in Finland that top management in industry, public administrators responsible for science and technology, many educators, and perhaps also the public believe that programming and software engineering require mathematical talent and skills. The situation is most likely the same everywhere in the industrialised part of the world.

On the other hand, the many discussions I have had with practicing software engineers and their immediate superiors have drawn a totally different picture. A small minority mentioned the importance of mathematics as a means to develop abstract and precise thinking. Very few gave any concrete example of how they or their subordinates have benefited from mathematics in programming or software design. Furthermore, usually the need for mathematics had risen from the application area, not from the act of developing software. The vast majority saw mathematics as unimportant, and some even revealed an actively aggressive attitude. This phenomenon also occurs elsewhere, although perhaps in milder form than in Finland. This is suggested by the small but famous survey by Lethbridge (7, 8) and other evidence.

It is thus clear that the way mathematics has traditionally been taught to software engineering students is not a great success. This is not a new finding: “If we are to continue to teach the amount and type of mathematics, we must justify it by other means than by saying it is important to a software developer’s work: It is normally not.” (7, Sect. 10) “Unfortunately, most mathematics courses in universities — responding to the needs of the physical sciences and classical engineering fields — focus on calculus and other aspects of continuous mathematics rather than on the discrete mathematics required for most computing disciplines. As a result, the necessary discrete mathematics is often taught by faculty in computer science or related departments.” (6, Strawman Report, Sect. 6.1). (The more recent Steelman Report, Sect. 9.1.1 puts it more mildly: “Computer science programs must take responsibility for ensuring that students get the mathematics they need, especially in terms of discrete mathematics.”)

In this contribution this problem is analysed, and first experiences of a course that was designed to alleviate it are reported. A large part of what is presented consists of my opinions that are based on about twenty years of experience in first building commercially used software
systems, and then teaching computer science and doing software theory research while cooperating with the industry. To avoid creating the image of objective science where that is not justified, I chose to frequently use the words “I” and “my” instead of the passive voice.

2 State of the Art

The above citations from (7) and (6) tell that software engineers do not benefit much from studies in calculus, linear algebra, etc. I have analysed this view and some common arguments against it in (10).

IEEE and ACM specify a discrete mathematics core consisting of functions, relations, and sets; basic logic; proof techniques; basics of counting; graphs and trees; and discrete probability. Indeed, some textbooks are built around these; let me just mention (9; 5). Both contain additional topics, such as algebras, and finite automata. The book (2) lacks probability, but in return has entire chapters on logic programming with Prolog, formal specification in Z, program correctness proofs, and relational database systems.

Some Finnish universities have this kind of a course, but it is too early to say how well it works. The results of Lethbridge suggest that software engineers do not see probability, logic and set theory as particularly important, although they rank them much higher than calculus and combinatorics.

Unfortunately, there is a somewhat common tendency to put too much emphasis on combinatorics. An extreme example is (1). It has grown from a course where Knuth taught “what he wished somebody had taught him” (p. v), namely the mathematics he needed when writing his famous “The Art of Computer Programming”. The book is excellent, if the goal is to analyse the performance of algorithms extremely accurately (beyond the usual “O” or “Θ” analysis). However, squeezing the very last drop of performance is not necessary with today’s computers. The major problems today are development time and correctness.

The book (3) is a remarkable attempt to teach logic and the construction of proofs to computer science students. It has been written by two leading researchers in program correctness. It develops a special semi-formal style of proofs, in which fragments of proofs can be carried through by syntactic manipulation, pretty much in the same way as we apply formulae such as \( a(b + c) = ab + ac \) or \( \sin^2 x + \cos^2 x = 1 \) when dealing with real numbers. The book develops a new notation for quantified entities (understood in a general sense), and justifies that it is more systematic and flexible than the common mathematical notation.

For instance, if the ambiguous expression \( \{ n^2 - i^2 \mid 1 < i < n \} \) denotes a family of sets indexed by \( i \) while \( n \) is a constant, then it becomes \( \{ i \mid 1 < i < n : n^2 - i^2 \} \), but if both \( i \) and \( n \) are indices, then the result is \( \{ i, n \mid 1 < i < n : n^2 - i^2 \} \). The formula \( \forall i, j : (i \leq i < j \leq n \Rightarrow A[i] < A[j]) \) is represented as \( (\forall i, j \mid i \leq i < j \leq n : A[i] < A[j]) \) or \( (\forall i, j \mid i \leq i < j \leq n : A[i] < A[j]) \). While the new notation helps in understanding and writing proofs, it also makes the book less attractive: who wants to teach students to write \( (+i \mid 1 \leq i \leq n : i^2) \) instead of \( \sum_{i=1}^{n} i^2 \)? Yet one would get useful laws such as “if the sets described by the predicates \( R \) and \( S \) are finite, then \( (+i \mid R \lor S : a_i) + (+i \mid R \land S : a_i) = (+i \mid R : a_i) + (+i \mid S : a_i) \)”.

3 Where Should the Emphasis Shift

I believe that the most valuable service that mathematical education can do to the software profession is to teach the students to recognise, formulate and reason with abstractions. The challenge is to get important details correctly, even when they are subtle and the end users or customers do not see any problem with them.

To illustrate this, consider the stupid example “the M.Eng. degree consists of 180 credit units.” No human being would reject a course collection containing 181 credit units on the basis of this rule. However, if the rule were programmed literally, a computer would. Cor-
rectness of a university administration system thus depends on the fact that the programmer uses his or her common sense, and implements the rule as if it said "... at least 180."

It is most likely that a programmer implements the "180" rule in its intended meaning. But what about "back-ups of electronic mail messages shall be kept for three months"? The rule cannot be interpreted literally, because old messages are destroyed in batches, so they are not all of the same age. Most people would interpret the rule to mean "at least three months". So this is what the programmer would probably implement. But, if the rule were stated by the Finnish data protection authority, it would mean "at most three months".

The computer is a hair-splitter, but (software) systems are built for humans. Therefore, a software designer must act as an arbitrator between the computer and the users. To do this, the designer must look at requirement and design documents at the same time both from the imprecise human and the nitpicking computer's viewpoints. It is this skill which I feel extremely important, and whose teaching seems to belong to nobody. I believe that software mathematics would be an excellent place to rehearse it.

In addition to the computer, a big organisation consisting of human beings may also be a hair-splitter. When frustrated with this, we often use the word "bureaucracy". I illustrate this with a stupid example. (I chose not to use an important example, because organisations seldom like their sore spots being dealt with in the public.) The rules according to which my work travel costs are compensated encourage me to go to the airport by taxi, although my wife taking me there by our family car would be cheaper for the university and (sometimes) more convenient for us. However, kilometre compensation is not paid for driving the car back to home, because then I am not in it — not even when the length of the travel exceeds the maximum for which parking fees at the airport are reimbursed. I know and the clerk knows that this is irrational, but the clerk is not empowered to break the rule in favour of common sense, and the rule has been formulated so high up and far away that there is no hope of getting it changed.

I am not claiming that software developers would knowingly build errors into systems because stupidly formulated rules force them to do so. But they may make errors, if they do not have enough knowledge of the purpose of a requirement to see that its formulation is against its intention. Although the requirements analyst may well know that "three months" means "at most three months", (s)he does not necessarily communicate that clearly enough to the system designer — perhaps because of having never realised that it is ambiguous, that someone could interpret it otherwise. At the end of the chain, the programmer implements "three months" either literally, which would be hair-splitting, or according to his or her guess of what it was really intended to mean. With some bad luck, the latter results in an error which is detected only when the system has been used for some time.

Of course, whenever a responsible programmer finds a specification ambiguous, (s)he goes and asks for more information. Indeed, I would like the skill of recognising potential ambiguity being practised during education — and I believe software mathematics to be an (although not the) right place for that. Even more importantly, the requirements analyst and system designer must have that skill. Otherwise the programmer will far too often have to stop working and start asking questions.

The problem has yet another crucial facet. In my experience, it is surprisingly difficult to get the right answer from the customer to a question about a potentially dangerous ambiguity. In many cases it has been almost impossible for the customer to understand a subtle issue which may have important implications for the use of the system. Therefore, software professionals must not only distinguish potential problems, but also express the alternatives to the users as use cases or by other layman-understandable means.

For instance, when the curriculum administration system for the Tampere University of Technology was being designed, the users pointed out that there may be several different versions of a course: one for undergraduate students, one for postgraduates, one for the open university, and one in English. The software company responded to this by reserving
four entries for each course, one of each kind. There were no software-enforced connections between the four entries. To my question how consistency between the various versions will be maintained the answer was “just cut text from one entry and paste to the other”.

It is unlikely that the secretaries who enter the data remember to update each version of the course, and even if they do, having to enter the same information into several places is unproductive and annoying. The interesting part of the story is that the designers visited the Institute of Software Systems at a late stage of the development, and before that nobody — neither from among the users to-be, nor from within the software company — had realised that a huge information consistency maintenance problem was being built into the system. I easily forgive the users to-be for that, but it is sad that the software company did not notice the problem.

The problem was solved (or this is at least what Software Institute representatives suggested) by dividing the information of a course into two parts: long-term and implementation-specific. The name, code number, number of credit units, name of the person responsible, total number of teaching hours, description of goals, list of prerequisites, etc. of the course are long-term. These are decided by the Council of the Department in question. There will be two copies of them: one in Finnish and another in English. The lecturing hall, weekdays and times of lectures, name of lecturer, language of this implementation, etc. are implementation-specific. There is one copy of this information for each implementation, and it is in the language in question. The copy also contains fields for specifying whether the implementation is intended for undergraduates and/or postgraduates and/or open university students.

Thinking about this kind of a structure was clearly painful for the person who was responsible for editing the curriculum books. She wanted to look at it all as nothing other than four different books. This is probably the reason why the software company designed the system in that way. Of course, a good system can print these four books from a differently organised database, so perhaps organisation of the data was implementation technology that she should not be mixed up with. However, someone has to decide what information is long-term and what is implementation-specific (do you agree with the classification of the total number of teaching hours?). That someone must come from within the university, and she was a primary contact for the project.

With all this, I am not saying that developing skills for dealing with ambiguities should be the only goal of software mathematics education. What I am saying is that it is an issue which today gets far too little attention. It should be given so much priority that it gets a strong position in the curriculum, even if that means that a lot of calculus and combinatorics must be thrown away. That some topic is potentially useful and educating should not result in there being no room in the curriculum for a vital topic.

4 Design of a New Course

As a penalty for all my nagging on the teaching of mathematics to future software engineers, the head of the Department of Information Technology told me that starting from spring 2002, the curriculum will contain a new three credit unit course with the title “Mathematical Methods in Software Engineering” (from now on MMSE), for which I will be responsible.

The course is intended to be taken during the second year of studies. As prerequisites it has six credit units on programming and a three credit unit “Mathematics for Algorithms” course on logic and set theory which, in turn, has no prerequisites. The MMSE course belongs to the common part of the Information Technology curriculum. It is not fully obligatory, as it may be replaced by the course “Basic Electronics 2”. However, that course is less attractive than MMSE to those information technology students who plan to specialise in software, and MMSE is in the (transitive closure of the) obligatory prerequisites of all courses on software theory. As a consequence, we expected roughly 100 students annually.

I (foolishly) ambitiously stated as the goal of the course
To learn to recognise, formulate, delimitate in an appropriate way, represent on an appropriate level of precision and detail, analyse and manage abstractions that result from, for instance, the analysis of user requirements, the specification of information content, or the design and implementation of software. To learn to exploit discrete mathematics to reduce software engineering costs and to improve the quality of software.

In the Institute of Software Systems, three credit units usually correspond to 42 hours of lectures and 26 hours of problem sessions. I saw no reason to deviate from that. No student project was included, because I had no idea of how it could be organised for the expected number of students in the absence of teaching assistants already trained in the subject.

The table of the contents of the autumn 2003 version of the course is shown in Fig. 1. Originally I planned to also include material on the general theory of state machines (which is not the same as the theory of finite automata), but after the first implementation of the course it was clear to me that there is no room for it. In the first two implementations Section 3 contained a large example on specification with set-theoretical machinery, but I dropped it with regret, because of lack of time and because of getting tangled with its details. The example was the “bubble diagram” notation of the SA system analysis method as presented by Haikala and Märijärvi (4, pp. 147–151). I wish to put it back one day!

The course discusses many topics that are commonly taught in discrete mathematics or theoretical computer science courses. However, the point of view of this course is systematically that of a programmer or system designer, not that of a mathematician, logician or theoretical computer scientist. For instance, context-free grammars are given only little emphasis, and they are derived from BNF. On the other hand, students are familiar with graphical notations, so the graphical “railway yard” chart notation is presented and shown equivalent to BNF.

Where possible, things are taught by appealing to ideas that are familiar to programmers. For instance, mathematicians’ definition of the notion of directed graph as a pair \( (V, E) \) where
$V \subseteq E \times E$ is compared to classes and instances in object-oriented programming. Special attention is paid to issues which, in my experience, are difficult for programmers to understand.

There are many discussions on the relationship between the reality and its formalisation. This explains why the brush-up sections are so long, although the students have had a course on logic and set theory. Furthermore, their lengthiness is partly an illusion, because they contain long lists of laws that are not discussed during lectures. They are there to liberate the students from the obligation to have a separate book on logic. There are also discussions on where and to what extent formalisation is useful — the course is far from claiming that everything should be formalised!

Section 4.1 is on the notion of expression. It is built around the idea that expressions are linear representations for entities that are trees by their "true nature". This view demotes precedence and associativity rules from mysterious laws to humble servants that ensure that the right tree structure is obtained. Regular expressions come in Section 4.2.

The last nine pages of Section 4.1 are devoted to an example, whose goal is to make students appreciate the power of abstract ideas and careful analytical thinking. It is a relatively simple computer program that uses the previous ideas to do impressive things. In it, expressions in a simple language for real functions (such as $(x + y + 2) \cdot \sin x^2$) are represented as trees, from which they can be printed as expressions with only the necessary parentheses, simplified according to a pre-programmed set of rules, and differentiated. For instance, differentiation converts $x \cdot y \cdot 2 + \sin x^2$ to $(1 \cdot y + x \cdot 0) \cdot 2 + x \cdot y \cdot 0 + (\cos x^2) \cdot (2 \cdot x^1 \cdot 1)$, from which simplification yields $2 \cdot y + 2 \cdot (\cos x^2) \cdot x$. In Section 5.3 the example is expanded with a recursive-descent parser that generates tree representations from expressions in the language.

The pretty long introduction contains, among other things, some analysis of the results of Lethbridge (7), and a lengthy discussion on the motivation of the course.

5 First Experience with the Course

At the time of writing this contribution, the course has been taught twice: during spring 2002 and autumn 2002. That it was taught in two successive seasons was due to some reorganisation of the curriculum.

During the first run of the course, a serious problem slowly surfaced. I had taught major parts of the material of the course (although not precisely in the same way) in two other courses: Introduction to Theoretical Computer Science (from now on ITCS) and Program Verification (PV). I had had to include elementary material in those courses, because the students' background and mathematical maturity did not allow for going directly to the topics proper. The idea was that in the future, that material would be removed from ITCS and PV, and something more advanced would be put into its place. Therefore, for students who had already passed ITCS and PV, the MMSE course did not contain enough new material to justify the credit units.

I and my superiors did not expect this to cause problems, because students that have passed ITCS and PV are advanced, and MMSE was clearly marked as a course for new students. Unfortunately, this did not prevent tens of old students from starting MMSE. Before its introduction, the "Basic Electronics 2" course was obligatory to all software students. Many students had done with it what many students tend to do with first and second year courses that are outside their main interest area: postpone. Now they saw MMSE as a chance to get rid of the electronics course altogether and do something more motivating instead. (Later on, discussions with many of the older students who had failed in the course confirmed that they had thought that with their experience, this course would be a piece of cake that would liberate them from the electronics course. Some were really angry and claimed that by failing them, I had unjustly destroyed their plan of graduating in the next month.)

To sort this problem out, in the beginning of autumn 2002 the Department of Information Technology forbade the combination MMSE + old version of ITCS + old version of PV.
1. \(j := 1; \text{limit} := A[n] \)
   
   for \( k := 1 \) to \( n \) do
   
   
   endfor

2. (a) What values can \( j \) have at the end of the program?

   ...  

   (f) If initially \( A[i_1] > \text{limit} \wedge A[i_2] > \text{limit} \), then can the relative ordering of the elements \( A[i_1] \) and \( A[i_2] \) change because of the program? Justify.

3. The alphabet is \( \{a,b\} \). Draw as small NFAs as possible that match the description.

   ...

(c) It accepts precisely three elements.

(d) It accepts precisely those strings, whose length is even.

(e) It accepts precisely those strings, which contain precisely one \( a \). (The number of \( b \)s may be just anything.)

Figure 2: Translation of sample questions from the most recent examination.

Students who have the latter two need not, however, take the electronics course, provided that they meet some additional not particularly restrictive conditions. (The decision had to be slightly reformulated later because, despite of all of our attempts to inform students about it, two of them took an MMSE examination against the ban. One of them fiercely argued that the original formulation allows him to take the examination and then pick any two of the three for his degree. These two students got very high scores in the examination.)

Before the ban became effective, seven students had completed the triple of MMSE, old ITCS and old PV. The following table shows the marks they got from them. “5” is the best possible, and “1” is the smallest that passes. Not surprisingly, they, too, did generally very well in the MMSE examination.

| mark from MMSE | 5 5 5 4 4 3 1 |
| mark from ITCS | 5 4 3 5 2 1 1 |
| mark from PV   | 5 3 3 2 1 2 2 |

To motivate students to participate in the weekly problem sessions, they were given points for attendance and for showing their solutions on the blackboard. The maximum that could be earned in this way was 7, of which at most 3 from participation. These points were added to at most 30 points from the examination. The threshold for passing was usually 17 points, and 20, 23, 26 and 29 points earned better marks from “2” to “5”. Therefore, if a student had been active in the weekly problem sessions, one third of the maximum examination points sufficed for passing the course. Fig. 2 shows some sample examination questions.

By summer 2003, altogether 175 students had tried the examination or got points from weekly problem sessions or both, and 128 of them had passed the course. The following table summarises their performance.

<table>
<thead>
<tr>
<th>mark</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td># students with that mark but not better of them tried the examination ( \geq ) twice</td>
<td>47</td>
<td>30</td>
<td>21</td>
<td>19</td>
<td>33</td>
<td>25</td>
</tr>
<tr>
<td>average problem session points</td>
<td>1.51</td>
<td>2.97</td>
<td>4.48</td>
<td>4.74</td>
<td>5.88</td>
<td>6.60</td>
</tr>
</tbody>
</table>

The distribution has clearly two peaks. 58 students have got one of the best two marks, and only five of them needed more than one examination to achieve that. In the opposite end, 19 have failed although they tried at least twice. They have 1.47 problem session points on
the average. Eight of them tried three times. Of the 47 students who have not passed, only seven have more than 3 problem session points. High problem session points are rare also among the students who have passed with 1 after trying more than once. Of course, problem session points correlate with passing because they contribute to the total points. But with +3 points ≥ +1 in the mark, this explains only a part. These statistics suggest that difficulties with passing can be attributed to not really putting effort to the course.

This is not the full story, however. I had discussions with many of the students who had failed, and I offered some of them the possibility to do something extra to pass the course or to reclaim the right to participate in the examination. (The right is forfeited after registering for the examination three times.) These discussions and experience of giving extra work revealed that many of them have only a vague understanding of even the easiet topics of the course. So it may be that the weekly problems were too difficult for them.

By the time of the most recent examination I had hypothesized that many students answer badly not only because they cannot use formal notations, but also because they do not understand programming-related issues well enough. The most recent examination was a good place to test this, because most of the participants had failed one or two earlier examinations. So, instead of my usual question asking state predicates to be added here and there into a piece of program, I posed a semi-verbal question, part of which is shown in Fig. 2.

Each item there is worth one point. On the average, the students got 0.39 points from 2(a). This is a disappointingly small figure, because the question only tests if the student understands a for-loop. Five got the “yes/no”-part of 2(f) wrong. Of the 16 who answered that part correctly, seven had big trouble in justifying the answer, although that could have been done by giving a three-element counterexample.

Question 3(c) provided another disappointment. The average for 3(e) was 0.71 points, which is not bad. For 3(c), which I had expected to be easy, all but two students got zero points, yielding the average 0.04. Many had drawn an automaton that accepts precisely the strings whose length is three, despite of clear difference in wording between 3(c) and 3(d).

In any event, because the system of voluntary weekly problem sessions yielding additional points resulted in many students not doing the weekly problems, the system will be different in autumn 2003. Problem session points do not any more contribute to the mark, but a student is not entitled to participate examinations unless (s)he has earned three points. A point can be earned only by acceptably presenting a solution on the blackboard. Priority to present a solution is given to the students who have collected fewer points than others who are present. When more than one student with highest priority want to present a solution, a lottery is held. This system ensures that a student has to prepare solutions to many more than three problems unless (s)he has exceptional luck in the lotteries; students who have fallen behind have good chance of catching others; and no one can earn all three points without doing exercises also at the late part of the course.

266 students have registered for the autumn 2003 edition of the course.

6 Conclusions

It is not unusual in Tampere University of Technology that students come to an examination without preparing for it. Therefore, if we want to concentrate on students who have seriously tried to pass the course, it is better to ignore those who have failed in one examination and not tried again. Of the remaining students, 36% had no difficulty in passing the course with one of the best two marks. On the other hand, about 13% have not passed, and 20% failed before passing (or after, when trying to improve their mark). The students are thus polarised.

Many of the students who have failed have probably not taken the course seriously enough, because they have not collected problem session points. However, among them are also many who seem to already have big problems in understanding the operation of a nontrivial four-line program. This is really worrying. The autumn 2003 implementation of the course will reveal
new information about this, because then collecting problem session points is obligatory.

Another unexpected finding was that many students who were near graduation saw the course as a newly available easy alternative to an unpleasant course that they had postponed, thought that with their knowledge they would easily pass, took the examination without seriously studying the material, and failed. This demonstrates that the course material is far from the common knowledge of many software professionals — which it of course should be, if it were to bring new kind of help to the software quality problem.

Formal notations are not important as such. However, I nowadays believe that inability to learn them is an indication of inability to think as clearly and precisely as is needed to design and implement software systems that are reasonably free of errors. The software professional must master logical thinking up to the level of nitpicking, because the computer is a nitpicker. The great challenge of the software professional that this course tries to help meet, is protecting the user from the nitpicking of the computer.

References


An Introductory Web-based Course to Enhance High School Students’ Knowledge of Computer Science

L. Grandell and J. von Wright
Department of Computer Science, Åbo Akademi University
Lemminkäisenkatu 14 A, FIN-20520 Turku, Finland

linda.grandell@abo.fi

Abstract

In order for high school students to be able to make a correct decision about potential future studies in computer science, it is essential for them to know what the field involves. At the Department of Computer Science at Åbo Akademi University, this was one of the reasons for developing and giving a web-based version of the introductory course to high school students. Other objectives were recruiting more students to the department and increasing the amount of girls among these students. In this paper we present the course, our experiences and some current issues. The results from the course were positive and indicated several advantages of this kind of educational programs.

1 Introduction

An introductory course for high school students in computer science (CS) was developed at the Department of Computer Science at Åbo Akademi University during year 2002. The main idea was to develop a course model as equivalent as possible to the corresponding university course. This was essential, since students who complete the introductory course while still in high school, do not have to attend the course again later at the university. When developing the model, the main goals and considerations were as follows:

- Introducing CS to high school students to improve their comprehension of the field.
- Diminishing the imbalance in the gender distribution among CS students.
- Attracting more students to studies in CS.

2 Background

CS has traditionally been intended mainly for students at higher education levels. On other levels the lack of tuition has often resulted in students having an incomplete and even inadequate comprehension of CS. It would, however, be essential for high school students in particular, to know the true nature of CS in order to be able to decide whether they want to engage themselves in university studies in the field.

The real pioneers in the field of introducing CS on lower educational levels can be found in USA (3) and Israel (1), where CS has been included in the high school curriculum since the early 1990s. Starting in year 2000, Finnish universities in the Eastern and Northern parts of Finland decided to make an effort in the same direction. They made CS tuition available to Finnish-speaking high school students in their regions using the Web for delivery of instruction. Swedish-speaking students, on the other hand, still lacked the possibility to study CS.

The Swedish-speaking population in Finland is distributed over a large area, reaching from Kokkola in the North to Helsinki in the South. This fact had so far complicated matters, since teaching CS to the students implies teaching groups of students as far as 500 kilometers apart. Since Åbo Akademi University is the only Swedish university in Finland, the university should be able to provide education to all interested Swedish-speaking students. When developing the model, the web-based programs used at the universities in Joensuu (5), Kuopio (4) and
Oulu (2) were of great importance. The course was marketed using a brochure containing information about different aspects of CS and studies in the field. It also brought up some of the myths in CS, e.g. that the field is not suited for girls and that one has to know a lot about computers in order to be able to study CS. It was important to sort out these claims, since one of the goals was to inspire and recruit as many girls as possible. For the same purpose, the contact persons in the schools were asked to encourage girls in particular to at least give the course a try.

3 Organization and material

The course organization involves one main tutor and several local tutors. The main tutor is situated at the Department of Computer Science in Turku, whereas the local tutors are found in the high schools. The main tutor has a Master’s degree in CS and is responsible for the development and updates of all material. In addition, the main tutor constructs and corrects the students’ assignments and is thus responsible for grading the students. One important part of the main tutor’s work tasks is giving support to both students and local tutors.

The responsibilities of the local tutors are mainly to support the students locally, making computer laboratories available to the students and arranging a joint gathering for all course participants once a week. In other respects, being web-based, the course assumes that the students mainly study on their own. This sets high demands on the students’ self-discipline and motivational level. The local tutors play important roles in encouraging the students to fresh and active efforts. The final grade is based not only on one single exam, but is derived from the results from assignments, quizzes and an exam.

The course model relies heavily on the use of information and communication technology as a tool for enhancing the students’ learning. The web material can be divided into two parts: informational and instructional material. The informational part of the material contains time schedules and information on other practical issues concerning the course. The instructional material includes reading material, assignments, discussions and computer based tools. The latter are for instance animated visualizations of algorithms, base converters and different kinds of games. The course also contains written material in form of a course handout, written at the department for this specific purpose.

4 Experiences from the first course

The course was given for the first time in spring 2003 and it was successful. At the start the course had 104 students, which was very satisfying, considering that the Swedish-speaking population is a minority in Finland. Over 42% of the participants was female, which must be seen as quite a good a percentage; the corresponding shares in Joensuu, Kuopio and Oulu were lower (13,5% , 11,5% and 22,4% respectively). It is possible that the measures taken to inspire girls mentioned above, did have a positive effect on the amount of girls. Another possibility is that the main tutor being female made it easier for girls to choose the course.

In March 2003, 65 students had completed the first part of the course. Of these 36,9% was female. The proportion of girls is quite high also in this respect in comparison to the courses given in Joensuu, Kuopio and Oulu. The dropout rate turned out to be very low, since only seven students interrupted the studies. There are, however, about 30 students who still have not completed the course, but have achievements from the course. It is likely that at least some of these students will finish the course during fall 2003.

By analyzing the results from a survey containing e.g. questions about the students’ comprehension of CS before and after the course, we found that most students, irrespective of gender, had a clearer understanding of CS upon completing the course. The students’ answers to open questions also indicated that instruction of this kind could attract more students, male as well as female, to the CS field. In addition, the evaluation showed that computer-based aids, which describe and explain different phenomena, can assist in developing an understanding
of CS. Especially multimedia presentations made with Macromedia Flash appealed to the students, facilitating their learning.

The main difficulties we experienced during the course were time scheduling and finding an appropriate work load for the students. The course should, on one hand, correspond to the 3 cu course at the university. On the other hand, we had to take into consideration that the students were still in high school, thus having many other subjects and activities. The time must be scheduled in very high detail according to how much time the students actually have. One must also remember that too high a work load contributes to students dropping out of courses.

5 Current issues

The course is being given for the second time as we speak. The work load has been adapted to the students' reality. The largest difference is, however, the replacement of the course web pages by the use of an open source course management system (CMS). The CMS moodle\(^1\), is, to our knowledge, not used by any other educational unit in Finland. It is thus a great opportunity for us at Åbo Akademi University to evaluate the system and its use in CS education in particular. Our experiences so far have been mainly positive. Since moodle is open source, we are able to make adjustments and changes to the CMS in order for it to be exactly what we want. moodle is a vast and flexible platform with many features, including for instance discussionboards, chat, quizzes, grading and logging. One further advantage, in our opinion, is that using this system everything is centralized. Another plus are the discussionboards on the moodle homepage, where teachers and developers from all over the world discuss problems, share ideas and give advice to each other on using the CMS.

References


\(^1\)Further information on moodle can be found on http://www.moodle.org
Desmond - An Interactive Tool for Study Planning

A. Rytkönen

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

Anni.Rytkonen@cs.Helsinki.FI

Abstract

Desmond is a tool to use in study planning at the Department of Computer Science at the University of Helsinki. This paper presents demonstrated parts of the system, with some future ideas. The design and implementation of the tool is still in progress and some additional features will be included later.

1 Introduction

The study planning process is one of the most important parts of successful studying. The planning can be done traditionally with pen and paper, using the study guide, the syllabus and the curriculum for the coming academic year. Alternatively, an electronic tool, like Desmond, can be used.

Desmond is designed for helping the student with her study planning process. It can be used throughout the studying years. In addition to guidance in course selection, it offers help to the student in automatic verification of the study module contents.

The idea of Desmond is presented in Koli Calling 2002 (see (1)).

2 Tool functionalities

The recommended study path in Computer Science, with most of the courses depending on each other, builds a tree of prerequisites. Desmond presents the recommended path as a graph of courses and dependencies. The first year student can easily see what courses to take first. The senior student gets an overview of courses she has passed and courses still missing.

A part of the first year student study path is presented in Figure 1 with visualisation as follows. The passed courses are presented with date and grade, while the planned courses are shown with the season they are going to be taken. The non-planned courses are presented with dashed-lined boxes, while the boxes for passed and planned courses are with normal lines. The next recommended, but not yet planned courses are marked with an exclamation mark (!) and the courses that cannot be yet taken with an X. All the future courses are presented with additional colour codes: green for next recommended courses and red for courses that should not be taken. Finally, the lines that combine the course boxes in the graph present the prerequisites between the courses. The normal lines are for fulfilled prerequisites and the dashed lines for non-fulfilled prerequisites.

Desmond presents possible courses that the student may use in the design of her timetable as a list. The list is generated according to the student’s study situation (i.e., what courses she has passed and what are missing) and the curriculum for the corresponding academic year. The data is received from a number of databases: the teaching database, the course prerequisites and correspondences, and the study register, which is used after the student has passed even the first one of her courses.

Part of the course list presented to a senior student is also shown in Figure 1. The plan is made one term at a time; the figure shows the plan for the Autumn term 2003. The student may include courses to her plan by selecting them from the list. The courses are presented with colour codes (this view is meant to be used mostly on screen): green for the next possible courses, red for courses that still lack prerequisite courses and cannot be taken yet. The list is updated for every term to come, according to the plans made for the previous terms.

When the student has passed all the courses that are needed for a study module, Desmond provides the student with a suggestion for contents of the module. The suggestion can of course
Figure 1: Part of the study path (left). The course list for a senior student (right).

be modified, within the limits of the study module regulations. The final course combination can be printed in official form for the professor to sign. Figure 2 presents the selection part of the study module design screen. Courses can be included in any of the study modules, within the regulations given for the module.

Figure 2: Part of the selection list for the "Cum laude approbatur" study module.
3 Future ideas

The study path could be developed towards the idea of concept maps (like presented in (2)). It could also be provided with functionalities connected to the courses (boxes on the study path), like planning the courses on a specific term or course enrolment.

The proposed timetable for the incoming term could include all the practise groups that are offered for the proposed courses (and that have free space). The student could then pick off the inappropriate practise groups and leave the best ones as her plan. Furthermore, the plan could be taken as course enrolment. This would save the student from the extra work of making the enrolments with a separate system. The idea of combining the planning process with the course enrolment is one future goal at the Department of Computer Science.

The plans for incoming terms (the practise times excluded) could be used in the process for planning the department teaching. The lecturer could see, what courses her students in average have taken before and how many students plan to participate in the course. This could help in planning different level practise groups for students with different prerequisites. The planning staff could see, what course combinations are the most popular and place the lecture times close to each other or at least not overlapping.

4 Further information


References
