Evaluating student motivation in constructivistic, problem-based introductory computer science courses

Lukas Faessler, Hans Hinterberger, Markus Dahinden and Marco Wyss
Institute of Computational Science, ETH Zurich
CH-8092 Zurich, Switzerland
E-mail: {faessler, hinterbe, markus.dahinden, marco.wyss}@inf.ethz.ch

Abstract: Keeping students motivated is a particularly challenging goal in undergraduate service courses such as introductory computer science for the natural sciences. Our experience shows that to jump-start motivation, students must experience an increase in their problem-solving competence, a capability that is built upon a combined mastery of concepts and skills. To achieve this, we integrate and support problem-solving from the beginning, in conjunction with a self-directed construction of knowledge structures for deep and sustainable learning. Student motivation was monitored during two introductory computer science courses involving a total of 500 students. A process analysis was used to investigate the relationship between motivation and task-specific aspects of problem-based learning. Our findings indicate that solving a sequence of selected, small problems representing fundamental concepts leads to an increase in motivation, provided these are followed-up with tasks that are ambitious rather than trivial.

Keywords: Student motivation, problem-based learning, constructivism, blended learning, didactic design, concepts, process analysis, application oriented learning, problem-solving skills, ICT instruction

1. Realistic problems: the essential food for motivation

All natural science students at ETH must complete introductory computer science courses. Ideally, in such a course students not only hear or read about computers but acquire *skills* in using computers to solve problems. Computers are complicated tools that demand from their users not only skills but also a good measure of *self-confidence*. Developing self-confidence takes time and depends on a strong *motivation* "to keep at it" throughout the semester. We have observed that motivation vanishes rapidly if the learning objectives are reduced to memorizing facts or going through routine drills with application software. This is understandable because it is difficult to detect sense in material that is presented as a collection of loose fragments.

Concepts as teaching objectives
To provide our courses with a framework, we make *concepts* rather than a compilation of facts the center of our teaching objectives. Concepts can provide structure and thus help students to see the course content in a meaningful context; but to be useful during the learning process concepts must be “connected” to useful skills. Combining concepts with skills requires a didactical and pedagogical preparation that leads to a course in which students learn more than the sum of the concepts taught and acquire new *capabilities*. By this we mean that a representative set of concepts must be embedded in a process that guides students through increasing levels of “computer competency”.

Instruction that relies on problem-based learning (PBL) supports this process best, because learners come into contact with the concepts through their own activities and thus can better differentiate between them (Flammer 1996). This differentiation lays the ground for a perception of the underlying ideas that enables students to *construct* the concepts by themselves, to successfully *apply* them, and do this while they are in *control* of their own learning process. The crucial point, however, is that the problems which guide through this process must be *interesting*, *relevant*, *realistic* and, if at all possible, also *entertaining*. We have learned that these are the primary ingredients for instruction that motivates.

Dimensions of cognitive competencies
Concepts, skills and capabilities form a triad useful when categorizing the factors that contribute to "Information Technology Literacy" (Snyder 1999). To illustrate our application of this triad we locate the level of these three

competencies along three axes, representing the *declarative*, the *procedural* and the *conditional* dimensions of a "cognitive space" (Mietzel 2003). The declarative dimension describes a concept (the *what*), the procedural dimension represents the skills required to solve a problem (the *how*), the conditional dimension measures the *capability* to apply the concept in a problem setting (the *where* and *when*). The procedural dimension presupposes the practical application of a concept, contrary to the declarative dimension where concepts are reduced to terms and definitions.

Development in this three-dimensional learning space is driven by personal experiences which are constantly reinforced by continually applying previously learned skills.

Choosing the right problem for each competence level
During the past 5 years we developed and refined instructional units to teach the natural science students skills to apply information and communication technologies (ICT) and to introduce them to computer programming. The first ICT course covers the topics of the internet, information visualization, modeling, data management, and macro programming. The second course teaches Java programming, covering subjects like programming tools, variables, control structures, developing algorithms, objects and graphics, applets and events. Each year, 500 to 600 students spend a total of 60 to 80 hours with each of these two courses (including lectures, exercises, review and exam).

The contents of these learning units are problem-based. PBL, however, is easier said than done because one quickly realizes that for it to work it is imperative that the chosen problem's difficulty is adapted to the student's level of competency. This matching of difficulty to competency can be challenging and it has been our experience that sometimes several attempts are necessary until the right problem has been found. To make sure that we found the right problem, we closely observed how the competencies of our students developed as they worked their way through several different instructional units; six for ICT, seven for programming. These units, based on blended learning (Hinterberger 2004), had not only increased our students' motivation, they also improved the quality of instruction (Faessler 2005). The constantly positive feedback from our students had led us to analyze the processes that go on during these courses so that we could generalize our experience.

The next section discusses how PBL can fill the space spanned by the three cognitive dimensions mentioned above; in section 3 we show how we built a scaffolding that can support our students while they navigate through this space. After we started to erect this scaffolding the motivation of our students increased continuously. A possible explanation for this is the topic of section 4. Our findings are discussed in section 5; section 6 completes the paper with conclusions and an outlook to further work.

2. Transferring skills from exemplary to real world problems

PBL is attractive because it captures the learner's attention by appealing to his or her natural curiosity and ambition. To challenge this ambition, however, a problem must be reasonably demanding; a condition that carries with it a high risk for overtaxing a student's comprehension. If this happens, the student is not willing to spend the time necessary to solve the problem.

Bootstrapping a student's capabilities
The success of PBL therefore stands and falls with the method chosen to increase the level of difficulty of the problems during different phases of the course. During our courses we start with small problems that together embody a minimal set of selected concepts and continue with a more complex problem in which these concepts come to bear in the context of a more demanding real world problem. Figure 1 illustrates these two levels graphically by embedding the problem-solving tasks in the space spanned by the dimensions of cognitive competencies.

At the application level A of Fig. 1 the focus is on competence that enables students to connect the *what* with the *how* for a single concept. Consequently, students solve small problems that are exemplary in character and chosen more to stimulate the learner's curiosity than to provoke his or her ambition. The concepts learned while completing tasks T1 to T3 prepare students to solve a more ambitious problem with task T4 at application level B. At this level students become skilled enough to apply several newly learned concepts in combination. This ability to combine concepts successfully does not come automatically, however; it is only possible if the students have learned to apply their knowledge *flexibly*.

An example from our introductory programming course

An ambitious task T4 can be: program a Galton Board – a triangular arrangement of pins placed above a row of small bins into which balls fall after they bounce from one pin to the next, starting at the apex of the triangle. Galton boards are typically used to demonstrate a process that leads to normally distributed data. The corresponding real world problem is to illustrate a stochastic system, a concept fundamental to the understanding of many simulations.

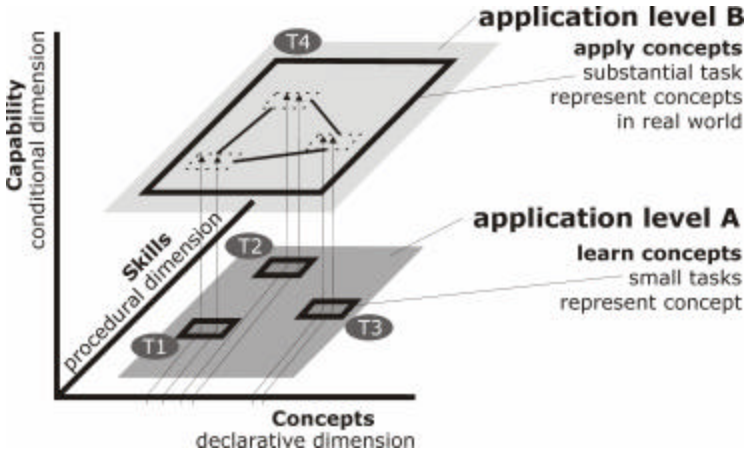


Figure 1: Training for increasingly demanding application levels (task T4) with a sequence of small tasks during which a few selected concepts are first practised (tasks T1 to T3). Tasks T1 to T3 provide a bootstrap to help a student pulling herself up to level B. The diagram shows an idealized abstraction. In reality, individual concepts at level A will reappear at higher levels in different combinations, also repeatedly, for reinforcement.

Table 1 lists the concepts, the skills and the problems chosen for three small tasks (T1 – T3) which successively increase the students competencies required to solve a more substantial task (T4) as part of our programming course.

Task	Concepts	Skills	Problem
T1	Variable Conditional statement	Managing a programming environment, storing & processing data, controlling program execution	Check if 3 digits typed into the keyboard correspond to the key of a lock
T2	Iteration	Applying repeated program execution	Calculating standardized paper formats, e.g. deriving the size of DIN A4 given the size of DIN A0
T3	Random number Array	Using program libraries Storing and processing lists of data	Illustrating the birthday paradox
T4	Simulation	Dissect a problem into parts that can each be solved by applying different concepts	Program a Galton Board

Table 1: The concepts practiced at application level A (tasks T1 to T3) and the application of these concepts at level B (task T4). The problems listed are representative, not exhaustive.

When we started with our problem-based programming course (Java), we made the error of choosing the programming of a Galton Board as the first task for our students and in doing so robbed them of the chance to bootstrap their capabilities. Not surprisingly they became demotivated after this first task and never really recuperated during the entire course. Motivation improved dramatically after we changed the syllabus and introduced three small tasks before assigning the Galton Board, as listed in Table 1. Our experience shows that not all tasks need to be spell-binding, but students must be given a genuine challenge to apply their newly acquired skills after a reasonably short time (following about three small tasks).

The bootstrapping process, schematically illustrated in Fig. 1, poses three constraints when applied in practice:

1) Developing skills with small tasks at level A happens in a time frame of a few *hours*,
2) Capabilities develop during an entire course, typically over a number of *weeks*,
3) The time and support required by each student for different tasks is highly *individual*.

Recognizing that capabilities develop in a 3-dimensional learning space is a necessary but not a sufficient condition for successful instruction because students can easily get lost in this learning space unless it is embedded in a structure that supports the learners. The structure we chose is based on the *four-step-model* described in the next section.

3. Four steps to flexible knowledge and skills

We have found that instruction is most effective and efficient if it makes the learning process "brain friendly" by breaking it up into four discrete steps (see Fig. 2):

See: students must be given the opportunity to ***see*** the concepts
Try: students should have the chance to ***try*** to apply concepts actively with appropriate guidance
Do: then they ***do*** apply them independently
Explain: to verify their understanding, they ***explain*** their solution to an instructor

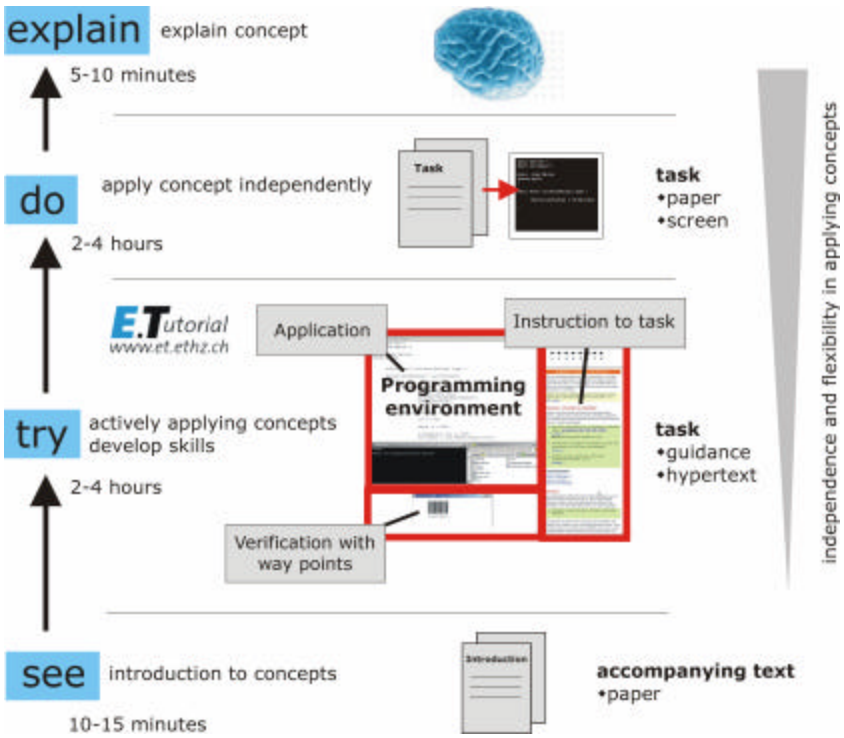


Figure 2: The *four-step-model* used to structure the learning environment illustrated in Fig. 3 (for details see text).

All tasks of our introductory courses are organized based on these four steps. During the *Try* step, we combine PBL with constructivistic methods in an e-learning environment (Hinterberger 2004) to allow individualized guidance.

E.Tutorials: an effective learning tool to reach high competence at the application level
In order to support our students while they are applying a given concept actively for the first time, we have developed, evaluated and refined hypertext-based course material we call E.Tutorials (Faessler 2004). Figure 2 illustrates how E.Tutorials are integrated into a blended learning environment in which students complete the four steps mentioned above. For each step we have chosen a medium that best suits its purpose and content.

In the first step, the concepts involved are briefly introduced on paper (***See***) to lay the foundation for the second step in which the concepts will be applied when students work with the E.Tutorial (***Try***). The E.Tutorial consists of an *application window* (e.g. Excel or Eclipse), in which learners are led step-by-step through small problems with instructions that are displayed in an *instruction window*. In a separate verification window students can check whether they are on right path in the problem-solving process. The guidance in the instruction window of the E.Tutorial must be structured in such a way that learners are neither overtaxed nor under-challenged. It is during the second step that knowledge becomes active and learners become increasingly independent as they progress through the learning material. We count on this independence during the third step, when they have to solve a new problem on their own (***Do***). The fourth step closes a learning unit with a short oral presentation (***Explain***), in which students are given the chance to show what they have learned and an instructor can evaluate the learning outcome.

The caveat: "Choose the right problem for each competence level" must be heeded during the design of steps two (Try) and three (Do). For both steps we first define the level of competence required when working with a given concept before we construct a problem that represents the concept in question. Next we divide this problem into a set of smaller tasks that allow step-by-step instructions for the Try phase and that facilitate the process of continuous verification to provide the learners with way points on their journey through the E.Tutorial. The development of these steps typically requires several iterations until a satisfactory solution has been found.

This four-step-approach with its explanations, way points and discussions provides a scaffolding for our students that helps them feel they are in control of a learning process during which they acquire skills that they confidently apply. Both this sense of control and their self-confidence, in turn motivate them to learn more about computer science. This motivation shows in pre-/post-evaluations administered during the past four years (details can be found on: http://www.et.ethz.ch/). Pre-/post-evaluations, however, give no insight into what happened in between, i.e. which parts are most likely responsible for a given result. We wanted to observe how the motivation of our students *fluctuated* during the two courses and therefore we adopted the process-oriented approach described next.

4. A pragmatic evaluation of student motivation

When we talk about motivation in the context of the process analysis described in this section we refer to *changes in motivation* over time. To measure this, we ask a student if she or he preferred one E.Tutorial to the previous one. An affirmative answer is interpreted to reflect an increase in motivation.

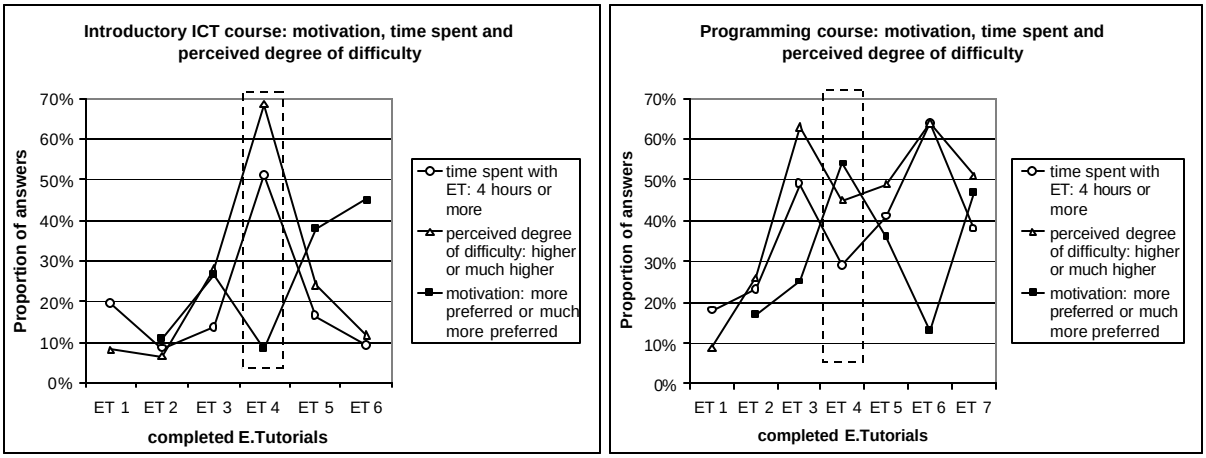
The process analysis as evaluation procedure

The data for this analysis came from polling the students with an online survey every time they completed an E.Tutorial. Since this happened every two weeks, we limited the questionnaire to five questions (Table 2) that allowed comparisons of key indicators over time. From these indicators we expected information representing the progress of the learning process.

Question	Possible answers	Interpretation
1. How much time did you spend with this E.Tutorial?	<2h, 2-4h, 4-6h, 6-8h, >8h	Individual effort invested by the students
2. How difficult was the E.Tutorial?	too easy, easy, just right, somewhat difficult, difficult	Subjectively perceived degree of difficulty is an important feedback that allows adjustments to the level of difficulty
3. How difficult was this E.Tutorial when compared to the previous E.Tutorial?	much lower, somewhat lower, approximately same, higher, much higher	Subjectively perceived change in the degree of difficulty
4. Did you prefer this E.Tutorial to the previous one?	preferred much less, preferred less, no preference, preferred, preferred much more	Reflects a change in motivation
5. What did you like? What can be improved?	free text	General feedback

Table 2: Questions asked for each E.Tutorial as part of the process analysis.

The online survey was placed at the end of each E.Tutorial using a HTML-form integrated directly into the instruction window (see Fig. 2). The answers are continuously collected and saved in a log-file. We have been careful to choose a selection of indicator variables and a wording of the questions that capture the subjective experience of the students as completely as possible. Students were asked to identify their answers with a mnemonic to allow us to anonymously trace the answers of each individual.



(A) The introductory ICT-course (246 students, avg. return rate: 57%).
ET 1: Internet, WWW
ET 2: Simulation with Excel
ET 3: Multivariate data visualization
ET 4: Data mgt. with lists and tables
ET 5: Data mgt. with relational databases
ET 6: Macroprogramming in Excel

(B) The programming course (292 students, avg. return rate: 75%).
ET 1: Variables, conditional statements
ET 2: Iteration
ET 3: Random numbers, arrays
ET 4: Simulation (Galton board)
ET 5: Objects, line graphics
ET 6: GUI-components
ET 7: Modelling, software development

Figure 3: Results of the process analysis of the introductory ICT and the programming courses.

Results
Figure 3 shows data for the following three variables:

- the proportion of students spending more time than expected with a given E.Tutorial
- the perceived degree of difficulty
- the comparative preference (change in motivation)

The graphs in Fig. 3 allow comparisons between the different variables as the students progress through all E.Tutorials of a course, allowing us to make the following observations:

Motivation compared with task difficulty and time spent
A numerical analysis of our process data over all E.Tutorials shows a highly significant inverse correlation between motivation and time spent on an E.Tutorial ($r = -.17$; $p < 0.01$). There is also a highly significant inverse relationship between the variable motivation and perceived degree of difficulty ($r = -.30$; $p < 0.01$).

Motivation to practice fundamental concepts
The decrease of motivation during the introductory ICT-course after ET 3 (Fig. 3 (A)) is significant (two sided t-test, $t(108) = 5.70$; $p < 0.01$). In Fig. 3 (B) one can observe that during our programming course the motivation also decreased significantly (two sided t-test, $t(106) = 2.37$; $p < 0.05$) after students were led back to practice fundamentals again (ET 5) after programming the Galton Board in E.Tutorial 4.

Motivation to learn fundamentals that are embedded in increasingly ambitious tasks
Motivation in E.Tutorial 4 of the programming course (Fig. 3 (B)) was significantly higher than in E.Tutorial 3 (two sided t-test $t(100) = -3.06$; $p < 0.01$). There is also a significant increase in motivation from E.Tutorial 6 to E.Tutorial 7 (two sided t-test, $t(47) = -2.81$; $p < 0.01$).

5. Discussion

Three points summarize our experiences:

- Motivation can depend on perceived task difficulty and time spent on task
- Rote practice of fundamentals lowers motivation
- Motivation can be recovered with an ambitious task

Motivation can depend on perceived task difficulty and time spent on task
Diagrams (A) and (B) of Fig. 3 show a marked drop in motivation after students completed time-consuming tasks which they also found difficult to master (ET 4, marked with dotted rectangle in (A), ET 5 and ET 6 in (B)). Upon closer examination we realized that in both courses we underestimated the effort required to complete these tasks. When students feel that the effort demanded from them is reasonable, however, they become motivated again (ET 5 and ET 6 in (A), ET 4 in (B)). This observation coincides with Atkinsons "risk-taking-behavior model" which predicts maximum motivation when the task to be solved is perceived to be of medium difficulty (Atkinson 1957).

Taking into consideration that the *perceived* and not the objective difficulty influences motivation can turn a course from a failure into a success as we have learned with our introductory programming course. Figure 4 illustrates that the actual task did not become less difficult by moving it six weeks into the course but that the students' perception of the tasks difficulty changed remarkably. The assessment of student motivation in 2003 is based on personal observations as we did not carry out a process analysis at that time.

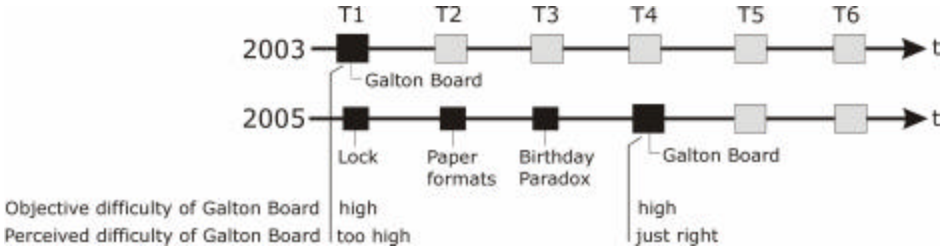


Figure 4: The motivational problems caused by the Galton Board in 2003 were solved in 2005 by placing this challenging assignment after three bootstrapping tasks.

Rote practice of fundamentals lowers motivation
Low motivation can also be observed when students are asked to practice only isolated concepts, even though they are essential to the understanding of the material that will follow. We see the reason for this low motivation in the lack of an opportunity to practice in a meaningful way the concepts just learned. Learners typically want to immediately see a return on investment and if deprived of it suffer from motivational "growing pains". Artificial constructs that do not convincingly illustrate the benefit of the skills these fundamentals will subsequently support are no remedy.

Motivation can be recovered with an ambitious task
Applying newly learned concepts with an ambitious task is more motivating than the introduction of new, fundamental concepts. On the other hand, an ambitious task is only motivating if the skills required to master it have been learned before. The data for ET 4 of the programming course (dotted rectangle in (B) of Fig. 3) show that the task to program the Galton Board is now at the right level of difficulty. Combining the *Try*-step with the *Do*-step (more independence during *Try*, a little guidance during *Do*) for this task might have helped in finding the right level.

Using the four-step-model in the described way, an instructor could possibly influence motivation in a positive way by telling the students that the next tasks will be less exciting but that they can look forward to a more challenging assignment in, say, two weeks time.

At the risk of being repetitive we mention again that an ambitious task by itself will not necessarily raise student motivation. It is the coordination of this task with the bootstrapping of the students' capabilities based on the concepts relevant to the task that is central for successful transfer.

6. Conclusions and Further Work

Motivating natural science students to actively participate in introductory courses that teach the fundamentals of computer science is possible. For us, e-learning materials that support PBL have been the answer because they allowed us to design introductory courses structured so that students can build their own conceptual resources and become competent in applying their skills in new situations.

With the four-step-model each student has the opportunity to construct these conceptual resources in a self-directed way. Always starting out by combining the *what* with the *how* ensures that students can bootstrap their capabilities. The result is a gradual acquisition of increasing self-sufficiency, reinforced by a sequence of increasingly difficult tasks that lead to higher levels on the conditional axis of the cognitive space (*where* and *when*). Once a student is at a higher level he or she has the competence to solve real problems. If, furthermore, students are given an opportunity to realize this, a positive effect on motivation is certain.

The data from our survey illustrate some of the experiences our students made during the course and provided some task-specific information. We have learned that the didactic design described in this paper can be highly effective and motivating but that it can only work if the degree of difficulty and the time required to complete a learning unit are chosen carefully. Because our learning units are problem-based we can embed the concepts to be taught in a "story" which can unfold during an excercise. It also gives learners a chance to generalize concepts on their own which in turn provides for deep and *sustainable* knowledge. The students leave the course with newly acquired *capabilities*.

The type of process analysis that we applied is an inexpensive but powerful measuring instrument which, when used in combination with a pre-/post-evaluation and with results from exams, can provide invaluable information to assess and improve the quality of complex educational methods.

Outlook

Our experiences with the E.Tutorials are restricted to first year courses at the University level. How our integrated learning method can be adopted at the high school level or even with younger students has not yet been explored. We plan to do this and also investigate the limits of the method, in particular the E.Tutorials, i.e. when are its possibilities exhausted? Is there a clear cut border or is it perhaps useful for certain topics to begin with E.Tutorials and halfway through the course switch to another didactic scenario?

With the two courses described in this paper we have implemented the paradigm "from sage on the stage to guide on the side". When applied to a traditional, lecture-based course, what are the consequences for the lecture? How does this novel didactic design affect the role of the "sage"? We are working to find answers to this question.

References

Atkinson, J. W. (1957). *Motivational determinants of risktaking behaviour*. Psychological Review, 64, 359-372.

Faessler L., Hinterberger H., Bosia L., Dahinden M. (2005). *Assessment as an Instrument to Evaluate Quality of Instruction*. ED-MEDIA 2005, World Conference on Educational Multimedia, Hypermedia and Telecommunications 2005, 3555-3562.

Flammer, A. F. (1996). *Entwicklungstheorien. Psychologische Theorien der Menschlichen Entwicklung.* Bern: Hans Huber Verlag.

Hinterberger, H., Faessler L., Bauer-Messmer B. *From Hybrid Courses to Blended Learning: A Case Study.* International Conference on New Educational Environments (ICNEE), 2004, University of Neuchatel, Switzerland.

Mietzel, G. (2003). *Pädagogische Psychologie des Lernens und Lehrens*. Göttingen: Hogrefe Verlag.

Snyder, L., ed. (1999). *Being Fluent with Information Technology*. Computer Science and Telecommunications Board, National Research Council. National Academy Press, Washington, D.C.

Acknowledgements

This work is based on research supported by Fonds Filep of ETH Zurich. For more information about the project, see http://www.et.ethz.ch (in German). We are also indebted to all our students at ETH who diligently answered our survey questionnaires and to Sarah Shephard from the Didactic Center of ETH for her critical and helpful comments.