Towards a Contextualized Pedagogy for Programming Education in Tanzania

Mikko Apiola
Tumaini University, Tanzania
and
University of Helsinki, Finland
Department of Computer Science
Email: mikko.apiola@ieee.org

Matti Tedre
CPUT, Cape Town, South Africa
and
University of Eastern Finland
School of Computing
Email: firstname.lastname@acm.org

***Abstract*—Contextualization of curriculum and course contents has been central to development of IT education at Tumaini University in rural Tanzania. However, as the development of the IT program has progressed, pedagogical challenges have become increasingly evident. The pedagogical difficulties materialize most markedly in programming courses. We identified a number of sources of pedagogical difficulties: reliance on rote learning, group dynamics that support free riding, extrinsic motivations, and greatly varying educational backgrounds. In this paper we describe a pedagogical approach that utilizes the theories and principles of creative problem solving to overcome those difficulties. Firstly, we provided optimal challenges to cognitive development by switching to an open learning environment, and we adapted course arrangements accordingly. Secondly, we encouraged intrinsic motivation and sense of autonomy by increasing the amount of affective support. Thirdly, we encouraged reflective learning processes by introducing a new concept of coding-while-lecturing. Fourthly, we stimulated the positive aspects of group work and promoted individual learning by providing adaptable multi-level exercises.**

I. Introduction

Since the founding of the B.Sc. program in information technology (IT) at Tumaini University, Tanzania, in 2007, it has become clear to us that the standard ACM/IEEE IT curriculum [1] is not adequate for a poor developing country [2]. The environment—natural, cultural, and technical—brings forth a number of challenges that are not challenges at all in industrialized countries. Those challenges include major issues with basic infrastructure, ICT infrastructure, lack of qualified staff, and lack of resources. The needs of the local environment, that is, the jobs that a college graduate should be able to do, also differ from industrialized countries.

The sociocultural, environmental, and technological reality of Tanzania necessitates a contextualized curriculum—a curriculum that takes the local context into account [2]. Since the launch of this contextualized curriculum in 2007, a great deal of experience, information, and ideas have been gained from both local and foreign visiting teachers. Especially the need for appropriate pedagogical solutions has become evident. That need is the most marked in programming courses, where—despite of a number of attempts by local teachers as well as foreign teachers from Finland, Spain, Denmark, and the USA—teachers have failed to adequately prepare students to meet the learning goals of the prospectus.

We have identified a number of issues, related to pedagogy and learning strategies, that negatively impact students' learning outcomes in programming courses. The first issue arises from the surface-level learning strategies of students, which, in turn, derive from the tradition of rote learning (e.g., [2], [3]). Students are not used to explore a topic, solve problems, and synthesize and apply knowledge, but they are mainly used to memorizing facts. The second issue arises from nearly non-existent computer literacy. The fact that most students have not used a computer before their university studies, combined with the abstract nature of programming, leads some students to generate negative attitudes and poor self-esteem, which easily leads students to adopt extrinsic motivational orientations. Students who do not feel competent are only motivated towards passing the courses by any available means. The third issue arises from the communal, groupwork tradition, which is deeply rooted in some African cultures [4, p.31]. The students usually work in groups, which has both negative and positive effects on learning [5].

Zaky [6] argued that successful engineering education in developing countries has to be based on a clear understanding of the social, cultural, and physical reality of the educational context. He wrote that four main factors need to be taken into consideration in curriculum development for developing countries: 1) social background and technological awareness, 2) the degree of preparation in secondary school, 3) the requirement for professional qualification, and 4) the wide university–industry gap. In addition to Zaky's [6] concerns, a number of other elements differ between industrialized and developing countries. First, workable pedagogical approaches in IT education are somewhat different between industrialized and developing countries [2], [7]. Second, many of the challenges that IT professionals face in industrialized countries are different from the challenges that IT professionals face in developing countries [8], [9]. Third, educational infrastructure, facilities, available funds, and university organization differ between universities in industrialized countries and universities in developing countries [10]. Fourth, generally speaking, staff members' interests and competence areas differ [10], [11]. These differences further undermine the attempts to import

curricula, pedagogical approaches, or other educational ideas from outside.

In addition to the external challenges, our students themselves consider programming to be difficult. In two earlier studies on students' perceptions of the challenges of IT education, programming was ranked as the most challenging subject in IT [12], [13]. On a scale from 1 (very easy) to 5 (very hard), programming scored mode 4 (hard) in both studies, whereas all other subjects scored mode 2 (easy) or 3 (average).

This paper describes our attempts to address the pedagogical issues concerning programming education at Tumaini University by applying theoretical viewpoints adopted from psychology of creative problem solving to pedagogical approaches in programming courses. We will describe the theoretical underpinnings of our pedagogical choices, and analyze the preliminary results. This paper is aimed at educators who work in similar, challenging environments as well as for theoreticians who wish to gain an alternative view on the challenges of the pedagogy of programming.

II. Creative Problem Solving

Creative problem solving is a special type of problem solving, which embraces the ill-defined nature of some problems (e.g., [14]), in contrast to well-defined problems. In math education, for example, solving ill-defined problems is called "problem posing" or "problems without a question" [15, p.124]. Creative problem solving entails a persistent process of idea generation, and the ability to transfer those ideas into action [16]. The required persistence is connected with *deep-level learning strategies* [17], where a topic is approached more intensively than in surface-level learning strategies.

Deep-level learning strategies are also connected with higher–order cognitive skills in Bloom's taxonomy of the cognitive domain [18]. Those skills involve evaluating, analyzing, and creating: a process of self-reflection, or the "active, persistent, and careful consideration of a belief or knowledge" [19], [20]. In addition to the higher-order cognitive functions, the creative process requires *intrinsic motivation* [21], which refers to a state of motivation where the activity per se is motivating, and which stands in contrast to extrinsic motivation, which refers to motivation by external rewards. A third necessary component is *domain-relevant skills*; every domain of knowledge requires some basic skills as a prerequisite to creativity. Thus, creative problem solving is best suited for a situation where intrinsic motivation, deep-level learning strategies, and domain-relevant skills overlap the most [21].

A. Intrinsic motivation

An intrinsically motivated person engages in an activity because activity is perceived as interesting, involving, satisfying and challenging [21], [22], [23]. An extrinsically motivated person acts primarily in order to meet an extrinsic goal, such as attaining a reward, winning a competition, or getting recognition. The self-determination theory (SDT, [22], [23]) argues that the intrinsically motivated state requires three factors: autonomy, competence, and relatedness, which Ryan and Deci [22], [23] also see as innate to human needs. A person perceives his or her own *competence* as a feeling of being in control of challenging tasks. *Autonomy* refers to the freedom to choose actions based on one's own interests. *Relatedness* refers to the basic need to be connected and close to other human beings. In contrast, a person who feels incompetent, not in control of his own actions, and isolated from others, is typically not intrinsically motivated and may experience a lower level of general well-being [22], [23].

According to Ryan and Deci [22], [23], in a university context intrinsic motivation can be supported by social interaction that enhances the feelings of competence, by optimal challenges, by supporting internal perceived loci of causality, and by providing optimal choices [22], [23]. Research evidence also shows that student-centered approaches to learning might be linked with deep-level learning, while teacher-driven methods are connected with surface-level strategies [24].

B. Cognitive Development

According to Perry [25], students' cognitive development involves three stages. At the *dualistic stage* students believe that for every question there exists exactly one correct solution. At the *multiplicity stage*, students believe that there are multiple answers, which all are worthy and can be equally good. At the stage of *contextual realism* students believe that the correctness of answers depends on how well those answers are reasoned in their context [25]. According to Schon [20], students' cognitive development can be boosted by supporting reflection; the "active, persistent, and careful consideration of any belief or knowledge" [19].

In the university context, students' cognitive development can be supported by pedagogical practices such as experiential learning, discovery learning, and problem based learning (PBL). For example, in experiential learning, meanings are made from direct, concrete experiences, and reflection is an essential part of the process. Experiential learning leans on ideas of John Dewey and Jean Piaget, but was made popular by Kolb [26], who developed ideas from earlier models of experiential learning [27]. In the "Kolb-learning-cycle" learners get fully and freely involved in new experiences, after which they have time and space to reflect on their experiences from different perspectives. Then, learners must form, re-form, and process their ideas, and integrate them into new ideas and understanding and into sound, logical theories [27].

III. Pedagogical guidelines

The learning outcomes in Tumaini's programming courses are cognitive as well as technical; that is, students are expected to learn new knowledge as well as gain a number of new skills. In the 3-credit *Programming I* course the main objectives are to introduce students to algorithmic thinking, to familiarize them with the fundamental concepts of programming, and to enable them to write, compile, and run small programs. In the 3-credit *Programming II* course cognitive skills are emphasized, as the course goes deeper into Java language and the object-oriented paradigm. The 3-credit *Programming*

TABLE I
FROM CREATIVE PROBLEM SOLVING TO CONTEXTUAL GUIDELINES

Theory Base	Method of Support	Contextual Solution
Intrinsic motivation - Autonomy - Competence - Relatedness	Provide: social interaction, optimal challenges, freedom from demeaning evaluations, support for internal loci of causality, and choice for self-direction.	Opening the learning environment (section III-A1) Affective support (see section III-A2)
Cognitive processes - Deep-level learning - Reflection, self-reflection - Cognitive development	Provide tasks to support all cognitive levels. Use pedagogical theories such as experimental learning and PBL.	Coding-while-lecturing (see section III-A3) Provide excercises for all cognitive levels (see section III-A4)

Project course aims at getting students comfortable with writing larger programs, and at introducing them to the basic software development process. Finally, the 3-credit *Procedural Programming* course introduces students to a different programming paradigm, and the course, again, aims at cognitive as well as technical learning outcomes.

Students' tendency to treat programming as a theoretical topic instead of a practical skill has been one of the greatest challenges in programming education at Tumaini University. However, adopting a theoretical approach to programming makes learning unnecessarily difficult, which sometimes leads students to fall back to the learning strategy they have relied on during their secondary school studies: rote memorizing.

To eliminate unsuitable learning strategies, we drew content from the theories of cognition and education, and adapted those theories into our curriculum design as well as design of classroom interaction. Our outcome—that is, contextual guidelines for programming education—is summarized in Table I. The first column in the table lists the cognitive and educational theories we relied on, the second column introduces the literature-derived support mechanisms that we decided to utilize, and the third column summarizes our implementation of those mechanisms in our Tanzanian context. The approaches and how they worked are further explained in the following section.

A. Pedagogical choices

We contextualized our pedagogical approach by building it upon a number of central theoretical notions of intrinsic motivation and cognitive processes. First, with regard to the central ideas of intrinsic motivation—autonomy, competence, and relatedness—we designed our pedagogical approach to provide social interaction, introduce optimal challenges, cut down on evaluation, support internal loci of causality (move students away from beliefs that their good or bad performance is caused by their own internal properties), and provide opportunities for self-directed learning.

Second, with regard to the central notions related to cognitive processes—deep-level learning, experimental learning, reflection, and contextual relativist views of knowledge—we designed our pedagogical approach to employ constructivist models and experiments, to support reflection, and to demonstrate persistent solving of problems. Yet, we tried to keep an open mind for not being single minded but for finding a fruitful mix of different approaches. Sfard [28] argued that "*because no two students have the same needs and no two teachers arrive at their best performance in the same way, theoretical exclusivity and didactic single-mindedness can be trusted to make even the best of educational ideas fail*." In this section we describe the pedagogical solutions that we discovered to work well. The students' comments in the following sections are based on two group interview sessions, involving 5 students each, which we conducted after the course to gain insight on students' views on the course.

1) Opening the learning environment: The pedagogical choice of opening the learning environment meant breaking the traditional lectures-and-homework model by organizing the classroom activity in a way that the role of the teacher became that of a facilitator, who is available to provide not instructions but guidance and affective support (explained further in section III-A2). The number of lecturing hours was reduced to 1/3 of the original, and the rest of the time in class was dedicated to practice. This meant that a lot of practical exercises had to be provided, utilizing optimal challenges for cognitive development (see section III-A4). This approach utilized the central ideas of an open learning environment [29], where the learner is in control over his or her learning process, as compared to a closed learning environment, where the learner has zero degree of freedom and the environment controls the learning situation.

In practice, we planned the learning situation so that students had a lot of exercises, on which they worked on their own. Students solved the exercises, and then brought the problems they faced back to the class sessions. During class sessions, the course facilitator did not provide students with answers, but helped and coached students to find answers by themselves and to help each other with the challenges. By this arrangement, we dramatically changed the learning environment from what the students are used to. Our students were used to an approach where a lecturer talks about programming, while students quietly listen. Our new approach is an open environment that supports active search of an actual working strategy that is required to learn programming and to write programs. The open environment approach was utilized in roughly two thirds of the class hours. However, at the beginning, some students reported confusion, and requested direct, step-by-step instructions on what to do.

In the interviews students explained their learning paths in ways that we found to be similar to what we expected. Many of the interviewed students emphasized practice as a key aspect to learning, and many outlined a basic problem-solving process: "*For example when I was starting, if I write a code line in Netbeans, if I write it wrong, there is an error. So right then I started thinking, what went wrong. So I made a program, but I just forgot to make a curve bracket* [sic]. *After doing so*

many practices, I begin to know, if I see red light, I know this is something I will need to fix."

Another student wanted to tell us about the motivating effect of the open exercises, which gradually increased in difficulty: "*When I started succeeding, I got motivation, and put more effort, that's when I started to see, now I am learning something.* [...] *and I saw how I can even write my own codes in methods, then I can call those methods, and then I realized, this is something*." More students' opinions regarding the increase in amount of exercises can be found in section III-A4.

2) Affective support: The need for affective support is not a recognized issue in western higher education pedagogy or practice. This is one aspect where western university tradition fails in Africa, where there is an urgent need for developing closer contact between teacher and student, as well as for affective support. We aimed at improving affective support from the viewpoint of intrinsic motivation studies, and through the three components of intrinsic motivation: autonomy, competence, and relatedness [22], [23]. We supported a sense of autonomy by giving students freedom to work with a number of different exercises. We tried to create an atmosphere where students are free to work as they wish and their work will not be controlled: the teaching staff is not there to give orders or instructions, but to help students to learn.

Support for competence is extremely important. Students often perceive programming as something very difficult, and they easily generate a negative internal loci of causality that inhibits their learning. Support for relatedness differs also from the western world, since group work is inherent to many African traditions of learning [4, p.31]. We consider the most important aspect in that respect to be some control over group dynamics. It is necessary to emphasize the importance of getting everyone in the group to learn individually, too. In order to emphasize the individual learning outcomes, we also provided a large number of individual learning tasks.

In practice we supported positive feeling of competence and empowerment by telling the students, in many ways and in many occasions, that learning of programming does not depend on some rare mental property, but that everyone can learn programming. We frequently marketed the idea that in the students' future career, programming skills will have plenty of applications and will widen their career opportunities. We also drew connections with the real IT work, emphasizing those careers that many students find interesting, such as web site development.

In general, we attempted to create a close and warm relationship through, for example, discussing the students' personal interests and their future plans. Considering the group work aspect and relatedness, we told students many times that in a healthy and responsible group, students will help each other learn, but in an unhealthy group students will help each other to pass the exercise by copying from each other. In the interviews we explicitly asked the students' opinions about the issues with group work, and all interviewees acknowledged the problems with group work.

Some students proposed solutions to problems with group work; solutions, such as introducing public group work evaluation sessions, where potential loss of face—a very negative thing in Tanzanian culture—would work as a deterrent from copying exercises without understanding them: "*So, during the time of presentation, somebody is standing in front of the class, then presenting what he knows. While he or she is presenting, the others are there to ask him questions. So if he or she know nothing, he can not respond to the questions*." In general, it is difficult to measure the impact of affective support, but our experiences are positive and suggest that affective support plays an important role.

3) Coding while lecturing: This approach was developed especially to support students' cognitive development by helping the students to step beyond their previous learning styles, and by underlining the fact that rote memorizing does not work with programming. We developed a pedagogical tool, which we call *coding while lecturing*, where the process of writing programs is constantly demonstrated by programming on-the-fly, while at the same time reflecting on the process with the students. This approach stresses the fact that every programming task can be solved in various ways. Programs are also debugged immediately at the class. While desk-testing the programs, students are repeatedly queried about the state of the program and about the values of variables. This practice is aimed at encouraging students to think about what happens when a program gets executed. This approach strongly discourages the students from falling back to rote memorizing learning styles. An additional benefit of coding while lecturing was that the teacher was able to keep aware of students' cognitive development.

In most of the coding-while-lecturing sessions, we started by opening an empty Java programming environment on the data projector. Then, a programming task was presented. After considering, with the students, what kinds of programming constructs and variables the program requires, we started coding. We constructed the program on the screen while explaining each and every step in program construction.

For example, on one lecture we worked with arrays, and we wished to develop accessor methods for a Java class that belonged to a larger example project that was developed over several class sessions. We started by thinking aloud about the problem and planning the solution. We then proceeded to slowly write and talk about a method for finding out if a value, given as a parameter to the method, exists in an array that was also given as a parameter. While gradually constructing the method, it was debugged, and before each test run the students were kept active by asking what they think will the outcomes and the state of the method be after each step. After that, we developed an alternative solution, and invited a student to step in front of the class and explain how the code worked. Our aim was to support students to develop a mental model about how programs are *actually* written, and how does the process look and feel like.

Students reacted to the coding-while-lecturing approach in various ways. On the one hand, students did enjoy the

approach and were interested, often active, and intensively observant of the class. On the other hand, this kind of an approach was new to students, which led to some confusion. Some of the students attempted to write down all the code and code changes using a pen and paper, even though all the programs constructed during lectures were made available in the course home page. Sometimes students considered activating questions to be too hard. Some students also considered the approach good but the difficulty too high: "*So maybe if he could start with the most simplest programs, what I am saying, then I think I could understand. But he was just taking many big programs, and he can extend a simple program to many outcomes, so that it becomes difficult*."

4) Excercises and Practice: As the cognitive development levels of students in a group may vary greatly, it is important to support students' development on varying levels. Therefore, we designed a large set of practical exercises, which were aimed at providing intellectually challenging tasks for students at varying stages of development. The exercises ranged from extremely easy to slightly more difficult to very challenging.

In practice, six sets of exercises were presented for students, each consisting of about 20 tasks. Some smaller tasks were combined to larger projects in later tasks. At the beginning many students reported that they had to work and struggle a lot doing the exercises, which we did not consider a negative thing at all. For example, one student noted: "*... maybe I feel that ... we had a lot of exercises in Java ... It kept us really busy! Some other courses, maybe they don't give so many assignments*." Some students also reported that while working with the exercises, there was a specific point when they started to understand programming: "*Maybe it was because of these exercises. I looked at it, I tried it ... to understand ... what is behind some method, for example. My programs had more and more mistakes, and when I tried to run them ...* [laughter] *sometimes the computer can go really crazy ... But in fact, it was this time I really started to understand*." Another student commented on the utility of the exercises: "*I think you are doing better in by teaching, by giving a lot of exercises ... So if you* [a student] *do a lot of exercises, then you understand more and more*."

IV. Discussion

Similar to curriculum design, also pedagogical design must be *strength-based* in the sense that it utilizes the strengths of the students, teachers, and each specific context. In our contextualization effort of Tumaini University's B.Sc. curriculum in IT, we have identified a number of pedagogical characteristics that derive from a variety of reasons external and internal to Tanzanian reality. In order to respond well to those characteristics, we adjusted our pedagogical approach in a number of ways. Below we outline the changes that we made and how those changes were received by our students.

First, our choice to shift to an *open learning environment* was welcomed by the students. Students especially liked the hands-on practices. But since students are used to closed learning environments where the teacher is in full control, the changes in mindset did not happen overnight. And the changes are still somewhat incomplete. Traditionally, students are used to obeying the teacher, and the shift to assuming more control takes time. One important observation is that the students were extremely well behaving when the learning environment was opened. That is, when students were given freedom and control, they did not become active instantly, but they did not start to lose their focus or disturb the learning situation either. This makes a great foundation for further opening the environment. For future development, Dron [30, pp.47–48] makes wise recommendations when he differentiates between choice and control in open learning environments: A learner can make choices, but assuming control can happen only if the learner understands the consequences of his or her choices [30].

Second, changing the *group work dynamics* without destroying the well-functioning group work spirit was not an easy task. It is clear that the cultural group dynamics offer great benefits, but those dynamics did also hinder the learning of some group members. In our future research we will explore how encouraging more individual work affects the students' work, and we will test how group dynamics games and tasks (such as the 3+ method [31]) change the dynamics. Such methods could demonstrate the students that the dynamics of a group can be changed, and that those changes may have a significant impact on the overall group results and on the learning outcomes of every individual group member.

Third, *coding while lecturing* was developed as a tool for eliminating the tendency of students to consider classroom examples in programming classes to be ready-made, final answers. Each lecture started from definition of a programming task, partly designed by the students, in order to ensure the uniqueness and spontaneity of each task. The teacher used a sort of a thinking-aloud method (self-reflection [20]), with help from the students, for writing and testing a program that solves the task. In our experience, this method worked well in forcing the students out of rote learning styles. We found that students need time and space to become more active and reflective in the classroom, since they are not used to this kind of learning.

Fourth, an *adaptable model of exercises* ensured that students experienced feelings of success: we gave each student tasks that they were able to solve. As each student advanced in the course, his or her tasks got more challenging. The design of exercises that suit different cognitive development phases [25] was a clear success. Each student was able to go hands-on and get the important sense of competence and grounds for cognitive development. Students did struggle when moving to higher cognitive levels, but by using this adaptive model we were able to provide them with a safe environment to try their best.

Another factor that definitely affected the students' learning experience was their access to computers. The students' access to computer labs was limited. That aspect of learning, however, we did not study this time. But the slowness and general unavailability of IT support did not help things, either. Although

the learning outcomes this year are good, unfortunately the numerical results cannot be compared with previous years' results. That is due to the incommensurability of the whole learning situation each year. Each year a different teacher has given the programming course; the student selection has been, due to various external reasons, done differently each year; and even the course description has changed during the past few years. It is our subjective view, though, that the changes had a positive impact on learning outcomes. Nevertheless, in this paper we are limited to presenting our pedagogical choices and leaving their broader and deeper empirical analysis for further research.

In our future research, we plan to extend our focus of interest into a very important part of the learning environment; the work that students perform on their own, outside the class sessions. We plan to extend our learning environment to allow students to do practice in more guided environments. As opening the learning environment and utilizing problem based practices have given promising results in the classroom, the working strategies that students utilize outside the class sessions are equally important. Our future research plans include conducting qualitative study on one student's pathway through our programming courses, and comparing different aspects of the learning environment in controlled experimental setups.

ACKNOWLEDGMENT

The authors wish to thank Dr. Dan McIntyre and Mr. Andrés Moreno for sharing their insights on the topic. This research was partly funded by the Academy of Finland grant #132572.

REFERENCES

[1] ACM Information Technology Curriculum Committee. (2005) Computing curricula: Information technology volume. [Online]. Available: http://www.acm.org/education/education/curric_vols/IT_October_2005.pdf

[2] M. Tedre, N. Bangu, and S. I. Nyagava, "Contextualized IT education in Tanzania: Beyond standard IT curricula," *Journal of Information Technology Education*, vol. 8, no. 1, pp. 101–124, 2009.

[3] The Task Force on Higher Education in Developing Countries, "Higher education in developing countries: Peril and promise," World Bank, Washington, D.C., USA, 2000.

[4] J. S. Farrant, *Principles and Practice of Education*, 2nd ed. UK: Longman, 1981.

[5] S. Loft Rasmussen and E. Larsen, "Social empowerment through ICT education: An empirical analysis of an ICT-educational program in Tanzania," Master's thesis, IT University of Copenhagen, Copenhagen, Denmark, March 3 2008.

[6] A. A. Zaky, "Developing engineers: Some reflections on university education in developing countries," *IEE Review*, vol. 35, no. 6, pp. 229–232, June 1989.

[7] M. Tedre, F. D. Ngumbuke, N. Bangu, and E. Sutinen, "Implementing a contextualized IT curriculum: Ambitions and ambiguities," in *Proceedings of the 8th Koli Calling International Conference on Computing Education Research*, A. Pears and L. Malmi, Eds., Lieksa, Finland, November 13th-16th 2008 2009, pp. 51–61.

[8] E. Brewer, M. Demmer, M. Ho, R. J. Honicky, J. Pal, M. Plauché, and S. Surana, "The challenges of technology research for developing regions," *IEEE Pervasive Computing*, vol. 5, no. 2, pp. 15–23, 2006.

[9] J. Kemppainen, "Building ICT facilities for education in a developing country. Analysis of an ICT project at Tumaini University/Iringa University College 2000–2004," Master's thesis, University of Joensuu, Department of Computer Science and Statistics, Joensuu, Finland, December 11 2006.

[10] M. Tedre and N. Bangu, "Implementing a contextualized IT curriculum: Changes through challenges," in *Proceedings of the 9th Koli Calling International Conference on Computing Education Research*, Koli, Lieksa, Finland, October 29–November 1, 2009 2010.

[11] M. Tedre, F. Ngumbuke, and J. Kemppainen, "Infrastructure, human capacity, and high hopes: A decade of development of e-learning in a Tanzanian HEI," *Revista de Universidad y Sociedad del Conocimiento*, vol. 7, no. 1, pp. 7–20, 2010. [Online]. Available: http://rusc.uoc.edu/ojs/index.php/rusc/article/view/v7n1_tedre_et-al/v7n1_tedre_et-al

[12] M. Tedre and M. Kamppuri, "Students' perspectives on challenges of IT education in rural Tanzania," in *Proceedings of IST-Africa 2009 Conference*, P. Cunningham and M. Cunningham, Eds., Kampala, Uganda, May 6th–8th 2009.

[13] M. Tedre, and M. Apiola, and J. Oroma "Developing IT Education in Tanzania: Empowering Students" *To be presented in IEEE Frontiers in Education (FIE) 2011 Conference*, Rapid City, South Dakota, USA. October 12–15 2011.

[14] M. Csíkszentmihályi, *Creativity: Flow and the Psychology of Discovery and Invention*. New York, NY, USA: Harper Perennial, 1996.

[15] E. Pehkonen, M. Hannula, and O. Björkqvist, "Problem solving as a teaching method in mathematics education," in *How Finns Learn Mathematics and Science*, E. Pehkonen, M. Ahtee, and J. Lavonen, Eds. Rotterdam, The Netherlands: Sense, 2007, pp. 119–130.

[16] N. Jackson and M. Shaw, "Developing subject perspectives on creativity in higher education," in *Developing Creativity in Higher Education: An Imaginative Curriculum*, N. Jackson, M. Oliver, M. Shaw, and J. Wisdom, Eds. London, UK: Routledge, 2006, pp. 89–108.

[17] F. Marton and R. Säljö, "On qualitative differences in learning – 2: Outcome as a function of the learner's conception of the task," *British Journal of Educational Psychology*, vol. 46, no. 2, pp. 115–127, 1976.

[18] B. S. Bloom, M. Englehart, E. Furst, W. Hill, and D. Krathwohl, *Taxonomy of Educational Objectives: The Classification of Educational Goals. Handbook I: Cognitive Domain*. New York, NY, USA: Longmans, 1956.

[19] J. Dewey, *How We Think*. New York, NY, USA: D.C. Heath & Co., 1933.

[20] D. Schön, *Educating the Reflective Practitioner: Toward a New Design for Teaching and Learning in the Professions*. San Francisco, CA, USA: Jossey-Bass, 1987.

[21] T. M. Amabile, "Social psychology of creativity: A componential conceptualization," *Journal of Personality and Social Psychology*, vol. 45, no. 2, pp. 357–377, 1983.

[22] R. M. Ryan and E. L. Deci, "Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being," *American Psychologist*, vol. 55, no. 1, pp. 68–78, 2000.

[23] ——, "On happiness and human potentials: A review of research on hedonic and eudaimonic well-being," *Annual Review of Psychology*, vol. 52, pp. 141–166, 2001.

[24] L. Postareff, S. Lindblom-Ylänne, and A. Nevgi, "The effect of pedagogical training on teaching in higher education," *Teaching and Teacher Education*, vol. 23, pp. 557–571, 2007.

[25] W. G. Perry, Jr., *Forms of Intellectual and Ethical Development in the College Years: A Scheme*. New York, NY, USA: Holt, Rinehart and Winston, 1970.

[26] D. A. Kolb, *Experiential Learning*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1984.

[27] H. Fry, S. Ketteridge, and S. Marshall, "Understanding student learning," in *A Handbook for Teaching and Learning in Higher Education*, 3rd ed., H. Fry, S. Ketteridge, and S. Marshall, Eds. New York, NY, USA: Routledge, 2009, pp. 8–26.

[28] A. Sfard, "On two metaphors for learning and the dangers of choosing just one," *Educational Researcher*, vol. 27, no. 2, pp. 4–13, 1998.

[29] V. Meisalo and J. Lavonen, "Bits and processes on markets and webs: An analysis of virtuality, reality and metaphors in a modern learning environment," *Tietoa ja toimintaa: Journal of Teacher Researcher*, vol. 6, no. 2, pp. 10–27, 2000.

[30] J. Dron, *Control and Constraint in E-Learning: Choosing When to Choose*. Hershey, PA, USA: Idea Group, 2007.

[31] J. Lavonen and V. Meisalo. (2009) Creative problem solving (luovan ongelmanratkaisun työtavat). [Online]. Available: http://www.edu.helsinki.fi/malu/kirjasto/lor/