

Improving Programming Education in Tanzania: Teachers' and Students' Perceptions

Mikko Apiola

University of Helsinki, Finland, firstname.lastname@ieee.org

Matti Tedre

University of Eastern Finland, and CPUT, South Africa, firstname.lastname@acm.org

Josephat O. Oroma

Tumaini University, Tanzania, firstname.lastname@tumaini.ac.tz

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. This paper reports an empirical study of students' and teachers' perceptions of challenges of programming education in Tanzania. The results support the anecdotal evidence from various developing countries that programming education is hindered by shallow learning strategies, unfamiliar pedagogical approaches, language problems, extrinsic motivations, free riding in group assignments, and cultural differences.

Index Terms – Developing countries, Contextualization, Programming Education, Students' perspectives.

INTRODUCTION

Tumaini University is the oldest private university in Tanzania. In the course of developing Tumaini's B.Sc. program in information technology (IT) since 2007, we have pinpointed a range of issues that need to be addressed. Those issues—which derive from the natural, cultural, and technical environment of a developing country—bring forth major questions, which concern, for example, infrastructure, ICT facilities, lack of resources, and lack of qualified staff. It has become clear to us that the ACM/IEEE IT curriculum [1] is not sufficient for a developing country [2], and thus, a contextualized curriculum was developed.

Our contextualized curriculum has been under constant development since the launch of the program in 2007, and it aims to address a wide range of educational, socioeconomic, and technical issues on multiple fronts. Whilst the socioeconomic and technical contexts include issues such as support models and procurement processes, the educational context—which this paper discusses—involves issues such as pedagogy, student motivation, the broader educational system, learning styles, and communication patterns. Since our contextualized IT program was launched in 2007, a significant amount of knowledge and experience has been

gained from research literature, local collaboration partners, and experiences of local and foreign staff: IT staff, administrators, students, researchers, and teachers. We are constantly working on improving our pedagogical approach, and the need for that improvement is most marked in programming courses, where students' learning techniques, motivation, and prerequisites for learning have, over several income cohorts of students, shown to be insufficient.

First, we have noted that students often resort to rote memorizing of code snippets instead of aiming at a deep understanding of program mechanisms [3]. And when rote learning fails, students often descend to plagiarism from the Internet and from each other. Students also employ group discussion techniques to help each other—sometimes to a fault. Second, we have argued that students' motivations are mostly extrinsic, which dissociates students from deep learning and leads them to attempt to pass the course through whatever means available [3]. Third, we have noted that a lack of adequate access to computers limits hands-on learning experiences. Also, in many cases, limited background education demands supplementary teaching.

A number of local IT educators, as well as foreign professional teachers from Spain, Denmark, USA, and Finland, have tested a variety of pedagogical approaches to address these issues in programming classes, yet the learning outcomes are still far from satisfactory. Therefore, in this study, we adopted a contextual approach, and investigated the challenges of programming education from a local, grassroots perspective. We aimed at exploring our above-mentioned assumptions about students' study habits and at deepening our understanding of the challenges of programming education from both teachers' and students' perspectives.

PEDAGOGICAL BACKGROUND

A.A. Zaky's views on engineering education in developing countries [4] give some directions for development of IT education in a developing country. Zaky [4] argued that curriculum development in a developing country should start from the social, cultural, and physical realities of the educational context, and that a clear understanding of these factors is essential for developing engineering education.

Common challenges include the lack of technical jobs and literacy, lack of quality science education in primary and secondary school, lack of practical training in industry as a part of engineering education, and lack of industry-academia interaction [4]. Other factors of note include, for example, the challenges that IT professionals face in Tanzania [5], educational infrastructure, facilities, funds, university organization [6], staff members' interests and competence areas [2,7], and, most importantly, differences in working pedagogical approaches.

We have earlier identified five important factors for development of contextualized pedagogy for programming courses [3]. Those factors are students' background knowledge, learning strategies and skills, motivational orientations, group work tradition, and local university tradition and culture—including language issues and students' social and cultural backgrounds. We argued that these five factors differ greatly between educational contexts—especially between industrialized countries and developing countries. These factors also complicate pedagogical design and design of learning environments for programming in a developing country context. As our foreign teachers are usually accustomed to a very different educational context and culture, these five factors make it challenging for them to adapt their teaching in their new, unfamiliar environment. Conversely, a local teacher might miss things that an outsider can see more clearly; hence, both insider's and outsider's perspectives are essential.

Students' Educational Background

Students' educational background in Tanzania varies greatly due to the multidimensional nature of the Tanzanian education system. The quality of schools varies a lot between locations and between private and public schools, and especially the quality of science education varies. Although the Ministry of Education has, for years, advocated constructivist pedagogy, primary and secondary schools still usually follow instructivist principles. Often, for instance, their teaching relies on memorizing important facts presented by teacher. In primary school the medium of instruction is Swahili language, but in the secondary school the medium of instruction is switched to English, and the students' language skills vary greatly depending on their teachers' skills. Like most developing countries, income differences in Tanzania are very large and students come from very different economic backgrounds. Other factors that vary between students are family background, tribe, age, work experience, and academic background [2]. From the teacher's perspective, this all leads to a situation where the students in a university classroom differ greatly from each other along multiple dimensions.

Learning Strategies and Skills

The Tanzanian students' typical cognitive and metacognitive learning strategies are not always directly transferable into the learning of programming. Due to their background education, the mainstream learning strategy that students are

familiar with and are good at is memorizing. However, in learning programming, it is essential to assume a deep-level learning process [8], which involves problem solving skills and the ability to learn by actively experimenting in a constructivist, student-centered manner. Not all students are used to experimenting, some of them possess simplistic views of knowledge [9], and their cognitive development [9], [10] as well as metacognitive skills suitable for studying programming may vary widely.

Motivational Orientations

Intrinsic motivation is defined as the motivation to engage in an activity primarily for its own sake, because the activity is perceived as interesting, involving, satisfying, or challenging [11], [12]. In contrast, extrinsic motivation is defined as the motivation to engage in an activity primarily in order to meet a goal extrinsic to the work itself. In our work [3] we have found students to easily generate extrinsic motivational orientations, which, according to research studies, are linked with surface-level learning and instructivist teaching models [13]. We have found students to be motivated to find and use whatever means they can for passing courses instead of performing actual study activities. We see several reasons for this: Students find it difficult to see a connection between programming and their future work. Some students consider programming as irrelevant, abstract, or extremely hard. Some students also possess erroneously negative views about their own ability to learn. Those students who perceive programming as useless and impossibly difficult are not motivated to even try.

Group work Tendencies

The communal, group work tradition is deeply rooted in many African cultures [14, p. 34]. This means that many students strongly prefer working in groups, which has both positive and negative effects for learning [14]. The dynamics of group work is a complex social issue with various side effects. It seems that bad motivation and lack of proper learning skills too often result in a situation where a group's main focus is not studying and learning together but completing an assignment by either utilizing the skills of the most competent group members or just plain plagiarizing. There is also wide variation among the independence levels of group members in terms of whether the opinions of the group or the opinions of the individual dominate the group's dialog.

Study Environment and Habits

One of the issues that we frequently face is plagiarism and copying of coursework. The habit is strongly discouraged by the university management and academic community, but it always takes new students some time to learn the rules of the academia [7], [15, p.58-59]. Working in English language causes problems for some students, too.

The study environment is not always conducive to best possible learning. Financial issues are often a problem to

students and the college. Internet connection does not always work reliably, teaching facilities are not always in order, and services and support are sometimes unavailable [16]. Students' socioeconomic statuses differ greatly, and students receive very different levels of support from their relatives. For example, scarcity of ICT facilities limits students' access to computers, and as not all students have money for laptops of their own, some students get more practice than others do.

CONTEXTUALIZING PEDAGOGY

In the course of building the B.Sc. program in IT since 2007, a great deal of experience, information, and ideas regarding programming education have been gained from local staff as well as foreign visiting teachers from Spain, Denmark, Finland, and the USA. All foreign teachers have based their teaching of programming on their previous experience on teaching programming in western universities. However, none of the teachers has been satisfied with the learning outcomes in the course.

Over the years, we have utilized the teachers' views, opinions, approaches, explanations, and ideas for improving programming education. For addressing issues about students' learning strategies we adopted the *coding-while-lecturing* technique [3]. In coding-while-lecturing, all programs are built from scratch during the lecture, while at the same time reflecting on the process. The aim is to emphasize that memorizing code snippets will not help one to learn programming. Instead, we encourage students to think about what happens when a program gets executed, and we market a specific mental model on program construction. By coding-while-lecturing, we also aim at preventing the students from falling back to rote memorizing and to shallow learning strategies.

Another approach that we adopted for supporting the development of students at varying stages of their learning was that we designed a large set of practical exercises aimed at providing intellectual challenge for students at varying skill levels [3]. We created an adaptive model for giving students exercises, and by that model we aimed, firstly, at ensuring that every student would experience feelings of success, and, secondly, that as each student would advance, his or her tasks would become more challenging. The tasks started from extremely easy. This way we aimed at supporting the feelings of achievement and competence by providing a safe base for building programming knowledge, and by providing everyone the chance to succeed.

We also opened the learning environment by reorganizing classroom activities and by getting rid of the traditional lectures-homework-exams arrangement. The plan was that the teacher would act in multiple roles in different classroom sessions. During coding-while-lecturing the role of the teacher would be an instructor and a role model. During exercises the role of the teacher would be a coach, who would be available to provide guidance and affective support. The time for instructivist lecturing was reduced, and the amount of supervised exercises was increased. Since the students in Tanzania are used to environments where the

teacher is in full control, and where students are used to obeying the teacher, it was quite challenging to execute this shift to an open learning environment, and to ask for students to assume more control over their learning.

Emotional support is rarely a recognized issue in western higher education. Due to this issue, we assume that the western university tradition may face a challenge in Africa, where there is an urgent need for closer contact and emotional support between the teacher and the student. Our solutions regarding the issue of affective support arises from the viewpoint of intrinsic motivation research [11], [12], and from our aim to support the students' feelings of autonomy, relatedness, and competence. We aimed at creating an atmosphere where students would be free to work without restraints, and where the teaching staff is not there to give orders but to support and facilitate learning. We aimed at supporting feelings of competence by promoting positive attitudes towards the difficulty of programming and promoting positive views about the students' capacity for learning programming.

These pedagogical changes, which were aimed at tackling the pedagogical challenges in our programming education, show promising results. However, although these changes in pedagogy have gained us promising results and a lot of experience [3], we still lack a coherent pedagogical model for teaching programming. Furthermore, even after these changes, the results were not yet satisfying by any measure, course grades, or the teacher's opinions. What still happens is that a small minority of students actually starts to learn programming skills, while the majority continues to struggle. To bring more input for our pedagogy development mission, in this research study we wanted to go deeper into a qualitative analysis of challenges with programming education. Especially we wanted to explore the students' viewpoints and we wanted to systematically collect information from the teachers.

METHODOLOGY

This study aimed at exploring students' and teachers' opinions and ideas about programming education at our college. The research question of this study was: "What are the students' and teachers' perceptions about improving programming education?"

We investigated the teachers' perspectives by interviewing five previous teachers either face-to-face, via Skype, or via email. One of the previous teachers was not reached. We investigated students' perspectives through conducting two group interviews. The first group consisted of the best five students (4 male, 1 female) in our Programming II course, as measured by midterm grades. The second group consisted of five other randomly picked 2nd year students (3 male, 2 female). Interviews were set up at one of the campus offices, and each interview took roughly 2 hours. The interviews were thematic interviews, built around two themes: "What are the biggest obstacles for students in learning programming?" and "How could we improve the teaching of programming?"

After the interviews were recorded, each of the researchers independently analyzed the interviews, identifying central issues that the students raised during the interviews. Based on the analysis, we generated a questionnaire, and administered that questionnaire to a comprehensive sample ($N=50$) of our 2nd year students, getting a response rate of 60% ($n=30$). The questions in the student interviews and survey were generic to all programming courses, but the students' answers were especially relevant to Programming II course, which the second year students had just completed.

The questionnaires were analyzed for descriptive statistics and even though the sample size is quite small ($n=30$), for basic measures of association (Spearman's rank correlation coefficient, ρ).

RESULTS: GROUP INTERVIEWS

The first broad theme that arose in the interviews of the first group—the group of high achievers—was a lack of prerequisite skills for programming. Firstly, students reported a lack of learning skills: *"I would say they could have learned programming, but, the method they are using to learn [in their previous education], it is something they need to change. Because, programming, as I know ... is more of understanding each code, each line, not that all bunch of codes does this, exactly one line, what does it do."*

Secondly, lack of deep-level learning processes was perceived as a systemic problem: *"This is not only programming, it is almost like a country problem. They [teachers] don't want us to think much deeper on something, using, thinking, preparing. We don't! The students only do what the teacher is saying, and that's it..."*

Thirdly, students considered the new pedagogical ideas of foreign teachers, such as class interaction, to be alien to them: *"I think the teaching methodology, you people [foreigners] are using, is different from what most of us are used to. In the sense that most of the people from outside Africa, they did much more of interaction, they shoot the questions for students, you pose a question, they give their views, and all that kind of thing."*

Fourthly, language was also mentioned as a major issue: *"This [language] is the problem. With maybe schools you came from. Some people went to very nice schools, very nice teachers, who speak well English ... but some of us have to take additional courses to be able to write English sentences."*

The second important theme that arose from the first group interview was emotions and motivation. Firstly, many students do not see a connection between their studies and their future job, which leads to extrinsic motivation: *"I would not expect someone to like the process, because ... most of us, we don't see the future of programming. And then if we don't see something to do with it... then just pass the test, and that's it".* Motivation is also brought down by negative emotions from lack of success: *"Now for some who is not able to do that, then it is impossible! It's like, very stressed out: How will I do this?"* In some cases it is clear

that when programming is perceived as "impossible", students do not even try—even if teaching starts from the zero level. There again, there were emotions of joy when starting to learn: *"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!"*

One student described losing track of time and having flow-experiences when working with code: *"It reminds me of something funny. [...] If I am tired, if I read something [...] then I start falling asleep. Kind of tired. But if I write Java, then I work more. It is kind of maybe interesting, I take my time... time just goes, I can't go to sleep!"*

The group work tradition has been a controversial issue from the beginning of the IT program [15]. When students work in groups, there is a tendency that only few people in the group do the work. In addition, there is a lot of copying between groups. Students recognize this issue, too, and acknowledge it: *"Strictness needs to be there [...] I remember, on one of our assignments, some of the groups, switched from other groups, and the printed them as their own and they submitted the assignments."* There is some group pressure between students, too. There seems to be an interesting aspect in the study culture that when everyone works together, no one is allowed to outperform his companions: *"But the rest three members did not do nothing. They wanted me to print their names, but they didn't even come when we were sitting for this exercise. Now why do it? It was very bad because I was taken as a bad man who fails the group [when refusing to put extra names on the assignment]."* Students considered group selection methods to be problematic. Instead of the class representative doing the grouping, students suggested free forming of groups.

One of the improvement ideas that the students presented was programming clubs: *"It is not easy to program alone. If you are three or four, it becomes easier. To be able to interact with others and share this. A learning club."* Another thing that was asked was career counseling, as students clearly lacked knowledge about the opportunities that programming skills would give them. Students had many good ideas for improving group work: there should be support for spirit that every group should be held responsible for making every member of the group learn. Also, it was suggested that there should be more group presentations—though we are not assured of the usefulness of group presentations. One thing that students constantly mentioned was that things should start from the bare basics—although that is exactly what we have attempted to do from the beginning. Students also mentioned that there is a shortage of computers and computing facilities at the university.

The first theme that arose from the second group interviews was classroom pedagogy. Again in this group, students raised the issue of unsuitable pedagogical practices, such as the students' lack of confidence with classroom interaction. Another cultural difference that was mentioned considered interaction: *"When the teacher asks the class,*

have you understood? Oh then they can say, yes! But really they don't know." - "I can say it is the ... the ... tendency of Africans, many Africans." One reason given was politeness: "We don't want to hurt the teacher's feelings." Students also mentioned that the teacher may have too high assumptions about their previous knowledge: "Maybe he [the teacher] can start with the simplest programs. And assume that they [students] don't know [anything about programming]. But he started as if we actually knew something [about programming]". However, the same student also reported that "we had very good introduction with former teacher [of programming]."

The second theme that arose in the second interview contained some very important improvement suggestions. The first suggestion concerned the choice of programming language (Java): "maybe if there are some simple programming languages, directing to Java maybe, they could have been taught before". This is an excellent point. Java might not be the best language to start with, because the import-clauses, class- and main-method definitions, and other object-oriented concepts and Java-related syntax bring in a lot of unnecessary cognitive overhead [17, p.77-81]. Better languages in that sense would be, for example, Python or Scala. The second suggestion concerned the well-known issue about access to computers "But this semester, all the labs are closed. So we have no access to computers in here." The third suggestion concerned group work and prevention of copying answers from each other; the students suggested that the amount of presentations should be increased to put social pressure on those copying: "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."

THE SURVEYS

A structured survey was generated from the central themes of the interview. That survey—a questionnaire with 59 items on a 7-point Likert-scale (1=not at all true, 2=very little true, 3=slightly true, 4=moderately true, 5=quite true, 6=very true, 7=completely true)—was administered to a sample of 30 second year IT-students ($n=30$). The sample consisted of 19 males (63.3%) and 11 females (36.7%), whose average age was 29.9 (min 20, max 45). Nineteen students (63.3%) had received only secondary school education, two students (6.7%) had advanced secondary school education, seven (23.3%) held a diploma in education, and two students (6.7%) had received a diploma in some other field. Due to the page limitation in this paper, we present only an overview of the most important results in the survey.

The survey results support our earlier suggestion that students' learning skills and strategies may not be well suited for learning programming [3]. To the statement "Learning programming requires a different approach to studies than what I am used to", the mean response was 4.97, with a mode of 5 ("quite true"). The statement "When I

study, I do experiments, try different program codes and try to figure out how programs work" resulted in a mean of 5.13 with a mode of 6 ("very true"). The statement "When I study programming I look at program codes for hours and try to memorize them" resulted in a mean of 4.14 and a mode of 4 ("moderately true"). It seems that both learning strategies are in use (experimenting and memorizing). Interestingly, those variables were not associated: those students who reported the most experimenting are almost uniformly divided among those who do a little and a lot of memorizing, and vice versa. We interpret that this implies that many students are struggling between their old and new learning strategies.

The survey results rejected the hypothesis that students do not see a connection between programming skills and their future. The results were quite the opposite: the mean response to the statement "There are such jobs available where programming skills are very useful" was 5.48 with mode 7. Also to the statement "I see a connection with programming skills and my future", the mean of responses was 5.07 with mode 7 ("completely true"). The survey also rejected the hypothesis that students find programming extremely hard: the mean response to statement "I feel I understand programming very well" was 4.38 with mode 6 ("very true"), and the statement "Often I do not know how to start solving the programming exercises" resulted in a mean 3.34 with mode 1 ("not at all true"). Hence, lack of trust on one's own skills might not be a reason behind learning challenges in programming courses.

Those statements that measured attitudes towards group work reveal that students recognize some of the issues regarding group work. To the statement "There is a lot of copying between groups", the mean response was 4.34 with mode 6. There again, students felt that group work was well distributed within their groups. The statement "In the group I worked, one or two persons did mostly everything", resulted in a mean of 2.97 with mode 1 ("not at all true"). Respectively, when it came to the individual level, students strongly emphasized their personal participation in group work: "In my group I was among those who put the most effort in the exercises", resulted in mean 5.63 with mode 7. Of the three statements that measured reasons for copying (options: exercises too hard, wrong teaching methods, not enough time), the statement "If people copy, it is because the exercises are too hard" was clearly agreed the most: mean 4.45 and mode 7.

The survey results confirmed that majority of students considered class interaction and language to be an issue, yet "limited access to computers" was the survey item that was most commonly agreed to be a problem. The most widely supported improvement ideas were the group work presentation and changing the programming language. Individual assignments, however, received very little support. Students also strongly agreed on the statement that programming teaching should start from a more basic level: The statement "Programming II should start from the very basics", resulted in mean 5.30 and mode 7. The course, however, is already the students' second full-semester

course on programming, and the course cannot repeat the basics again. The results also implied that students enjoy challenges, many felt joy, and some experienced losing track of time when succeeding with challenging tasks.

Although the sample size is quite small for inferential statistics, a few associations can be noted. The statement *"When I study programming, I look at program codes for hours and try to memorize them"* correlated negatively with *"I feel I understand programming very well"* ($\rho = -.485$, $df = 28$, $p < 0.05$). The first statement also correlated positively with statement *"If the task is too hard, I think it is the right thing to copy"* ($\rho = .508$, $df = 28$, $p < 0.01$).

The first correlation indicates that a shallow learning strategy correlates with poor image of one's programming skills. The second correlation indicates that a shallow learning strategy correlates with positive attitudes towards plagiarizing.

TEACHERS' PERSPECTIVE

In addition to students, we also interviewed the previous programming teachers concerning their views about issues related to programming education. There were five out of six participants attending the interviews, which were conducted electronically via Skype and email, or face-to-face. One of our previous teachers could not be reached.

The teachers' interviews reinforced some of the ideas that we had gathered from the other informants. The teachers noted problems with hardware and software, lack of study material, antiquated learning strategies, erroneous assumptions and fear towards programming, time management issues, counterproductive group work dynamics, and issues with English language. In addition, two teachers noted that the imported virtual courses were not sufficiently adapted to the Tanzanian context.

Concerning what worked well, the teachers named easy interaction in the classroom. The Moodle server—which is not in use anymore due to technical issues—had also contributed positively to the courses. In addition, one of the teachers had successfully used a visualization tool Jeliot for algorithms and program concepts. The teachers argued that, roughly, one third of students learned to program quite well, another third learned to construct simple programs, and the bottom third struggled with programming a lot.

Concerning reasons for difficulties in programming education, the teachers considered the students' educational background to be the main reason. The teachers argued that Tanzanian students have a relatively poor mathematical background. One of the interviewed teachers felt that many students find mathematics and science hard and unrewarding. The acute shortage of good mathematics and science teachers deepens the problem for some time for the future. The teachers' proposed several solutions: Firstly, they suggested that students should get individualized exercises; secondly, they proposed that students should be given more time to work both individually and in groups; and thirdly, they recommended inclusion of tasks where the students have to complete documented but unfinished

example programs. Fourthly, the teachers noted the importance of easing the students' fear of programming—many students give up too easily since they perceive programming to be too difficult.

CONCLUSIONS AND DISCUSSION

This research study aimed at gaining a better understanding of the factors that influence the learning of programming in a developing country. We looked at the issue from both teachers' and students' perspectives. We have had many teachers attempting various pedagogical solutions, mainly imported from the western world, but unfortunately with only moderate success. Based on those experiences, we have previously proposed five factors that influence learning of programming: students' educational background, learning strategies and skills, motivational orientations, group work tendencies, and study environment and habits.

Our results corroborated but also elaborated and added to our previous observations about the teaching and learning programming. It seems that students and teachers share some agreement about the underlying issues. For example, the students are very aware about the need to acquire new learning strategies, which many of the teachers have also stressed. Some issues affecting motivation, especially the fear of programming and the idea that programming is irrelevant to one's future work, were brought up in the interviews, but were not confirmed by the survey results. In our future research we will further investigate that issue.

The results also confirmed the students' awareness of influential issues such as group work dynamics, yet the results also revealed some students' protective attitude towards their own group and especially to themselves. Other groups plagiarize, one's own group does not. The research results also revealed important cultural issues about classroom pedagogy, which especially the western teachers need to take seriously. Pedagogy cannot be based on western practices and ideas alone, but it is important to first understand the local practices of teaching and learning. In general, the students were extremely outspoken, open-minded, and positive, and they suggested a wide range of improvement ideas, which we take seriously.

In the future, we plan to improve the programming education at our IT program in two ways. Firstly, our plans include conducting a deeper qualitative study about the trajectory of a student's path throughout a programming course, systematically collecting and identifying issues which affect the student's learning. Secondly, it is crucial to experiment with new pedagogical approaches, and test them in controlled, experimental setups.

We plan to implement new kinds of exercise sets, and to dramatically increase supervised practice sessions. We are also currently working on changing from Java to Python. In addition, we plan to undertake research for gaining deeper understanding on the background characteristics of students, and the possible connections between students' backgrounds, learning strategies, and their performance in programming courses.

ACKNOWLEDGMENTS

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. Computing curricula: Information technology volume. 2005. [Online]. http://www.acm.org/education/education/curric_vols/IT_October_2005.pdf. Accessed: 5 April 2011.
- [2] Tedre, M., Bangu, N., and Nyagava, S.I. 2009. "Contextualized IT Education in Tanzania: Beyond Standard IT Curricula". *Journal of Information Technology Education*, vol. 8, no. 1, pp.101–124.
- [3] Apiola, M., Tedre, M. 2011. "Towards a Contextualized Pedagogy for Programming Education in Tanzania". *To be presented in the 10th IEEE Africon Conference*, Livingstone, Zambia, September 13–15.
- [4] Zaky, A. A. June 1989. "Developing engineers: Some reflections on university education in developing countries", *IEE Review*, vol. 35, no. 6, pp. 229–232.
- [5] Kempainen, J. 2006. "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, Finland.
- [6] Tedre, M. and Bangu, N. 2010. "Implementing a contextualized IT curriculum: Changes through challenges." *In Proceedings of the 9th ACM International Conference on Computing Education Research*, Koli '09 Finland, pp: 67–76.
- [7] Tedre, M. Ngumbuke, F., Bangu N., and Sutinen, E. 2009. "Implementing a contextualized it curriculum: ambitions and ambiguities." *In Proceedings of the 8th ACM International Conference on Computing Education Research*, Koli '08 Finland, pp: 51–61.
- [8] Marton, F and Säljö, R. 1976. "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.
- [9] Perry, W.G., Jr., 1970. *Forms of Intellectual and Ethical Development in the College Years: A Scheme*. New York, NY, USA: Holt, Rinehart and Winston.
- [10] Bloom, B.S., Englehart, M., Furst, E., Hill, E., and Krathwohl, D 1956. *Taxonomy of Educational Objectives: The Classification of Educational Goals. Handbook I: Cognitive Domain*. New York, NY, USA: Longmans.
- [11] Ryan, R.M and Deci, E.L. 2000. "Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being," *American Psychologist*, vol. 55, no. 1, pp. 68–78.
- [12] Ryan, R.M and Deci, E.L. 2001. "On happiness and human potentials: A review of research on hedonic and eudaimonic well-being," *Annual Review of Psychology*, vol. 52, pp. 141–166.
- [13] Hoskins, S., Newstead, S. Encouraging student motivation. 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.
- [14] Farrant, J.S. *Principles and Practice of Education*, 2nd ed. UK: Longman, 1981.
- [15] Rasmussen, L.S. and Larsen, E. 2008. "Social empowerment through ICT education: An empirical analysis of an ICT-educational program in Tanzania," *Master's thesis*, IT University of Copenhagen, Copenhagen, Denmark.
- [16] Tedre, M., Apiola, M., Oroma, J. 2011. "Developing IT Education in Tanzania: Empowering Students." *To be presented in the 41st ASEE/IEEE Frontiers in Education Conference*, Rapid City, SD, October 12–15.
- [17] Sajaniemi, J and Kuittinen, M. 2008. "From procedures to objects: A research agenda for the psychology of object-oriented programming education." *Human Technology: An Interdisciplinary Journal on Humans in ICT Environments*, vol. 4, no. 1, pp. 75–91.

AUTHOR INFORMATION

Mikko Apiola, Lecturer of IT, IT Department, Tumaini University, Iringa University College, Tanzania; PhD Student, Department of Computer Science and Department of Teacher Education, University of Helsinki, Finland, firstname.lastname@ieee.org

Matti Tedre, Ph.D., Project leader, Adjunct professor, University of Eastern Finland, School of Computing; Adjunct Professor, Cape Peninsula University of Technology; Professor, Tumaini University, Tanzania, firstname.lastname@acm.org

Josephat O. Oroma, Head of IT Department, Tumaini University, Iringa University College, Tanzania, firstname.lastname@tumaini.ac.tz