Blended learning of programming in large classes: a reflection of students' experience from an Ethiopian University

Tesfaye Bayu Bati
Hawassa University, Ethiopia

Prof. Helene Gelderblom
University of Pretoria, South Africa

Prof. Judy van Biljion
UNISA, South Africa

ABSTRACT

The challenge of teaching programming in higher education is complicated by problems associated with large class teaching, a prevalent situation in many developing countries. This paper presents students' reflection about the use of a blended learning approach to teaching and learning of programming in a class of more than 200 students. A course and learning environment were designed with pedagogical underpinning and integration of large class and programming teaching best practices. Students' reflection on the course design, implementation trajectory and their satisfaction was gathered through a focus group discussion. The result of the study shows that students can positively accept large class teaching of programming with a provision that the course and learning environment design facilitates their learning. Fulfilling such provisions requires substantial time and effort; this study can contribute to the discourse on how to manage that process effectively. The contribution of the paper is the presentation of students' perspective reflection on course design and implementation for blended learning of programming in large class.

Keywords: *blended learning; large class teaching and learning, computer programming, novice programmer.*

1 INTRODUCTION

This paper presents an empirical study conducted to address the challenges of large class teaching and learning (T&L) of introductory computer programming courses. Beginner programming students (novice programmers) tend to make limited progress in learning to program due to numerous technical and psychological factors (Robins, Rountree & Rountree, 2003). Despite forty years of research-based intervention initiatives, the challenges continued to hamper students' learning of programming often leading to high failure and dropout rates in the courses (Sarpong, Arthur & Owusu, 2013).

Fast rate of population growth and resource constraints make large class teaching a compelling reality to higher education institutions in sub Saharan Africa (Altbach, Reisberg & Rumbley, 2009; Teferra & Altbach, 2004). It is widely believed amongst educators and students that large class T&L is disliked and difficult (Iipinge, 2013). Some of the problematic issues of large classes for students are decreased student-teacher interaction, anonymity, passivity and reduced accountability (Prosser & Trigwell, 2014). Teachers encounter increased workload and difficulty to give individualised attention to their students.

The study introduced a blended learning approach to enhance student learning of introductory programming within a large class teaching context. The blended learning design integrated accessible open-source and free technologies with innovative pedagogical approaches, which are described later in Section 3. The study was designed as an action design research (ADR), which is a variant of design research in organisational context. ADR targets a dual goal of continuous improvement of a design and evaluation of the design's impact (Sein, Henfridsson, Purao, Rossi & Lindgren, 2011).

The guiding research question of the study was "*how can a blended learning approach be used to improve large class teaching of programming?*" Two consecutive programming courses were used for the ADR study. The study was conducted with a large class of over 200 students at Hawassa University, Ethiopia.

This paper focused on the second cycle of the ADR, which focused on the sub-question "*how do the different components of the intervention contribute to support large class teaching of programming* ". The paper is organised into seven sections. The next section (2) presents a review of related literature followed by a description of the blended course and learning environment design in Section 3. The research design is presented in Section 4. The two succeeding sections (5 and 6) present and discuss the results of the study. The paper concludes in Section 7 by suggesting a need for a more student-centred instructional approach for improved programming education in large classes, but also for smaller classes.

2 BACKGROUND OF THE RESEARCH PROBLEM

2.1 The issue of large class teaching

Large class teaching is often considered as counterproductive to "meaningful T&L" (Iipinge, 2013, p. 105). Table 1 summarises the challenges of large class teaching widely cited in the literature (Prosser & Trigwell, 2014).

Table 1: The effect of large university classes from student and teacher perspectives

Student perspectives	**Teacher perspectives**
- Decreased student-teacher interaction causing student anonymity and passivity; - Poor student engagement in learning activities and lower level of motivation; - Student dissatisfaction with large classes; - Unpreparedness of first-year students for large class teaching; - Undesired behaviours such as late coming, absenteeism, and off-task engagement; - Less individual accountability, noise and destruction.	- Stress to function effectively in large class; - Difficulties to relate as individual with students; - problems in two-way communication and feedback; - Resorting to traditional teaching methods to allow time for other responsibilities; - Resource problems such as insufficient copies of textbooks.

The global trend in massification of higher education (Altbach et al., 2009) has emboldened the wider adoption of large class teaching strategies (Hornsby & Osman, 2014). There are two more factors that make large class teaching compelling to Sub-Saharan Africa. First, most of the countries in the region have economies that are underdeveloped but their population is growing fast (Teferra & Altbach, 2004). Secondly, though the region is still characterised by a lowest higher education enrolment, it leads the world in the rate of new enrolments (UNESCO, 2010). Thus, as Hornsby and Osman (2014) argue in favour, countries in the region need to search for and apply alternative pedagogies and technologies for large class education. In doing so, the countries can achieve both the access and quality issues of higher education.

2.2 The challenges of introductory programming

Novice programmers tend to make limited progress in learning to program, resulting in high failure and dropout rates in programming courses (Sarpong et al., 2013). Robins et al. (2003) show that the difficulties are both technical and psychological.

The technical challenges are diverse. Among them are the challenge to learn programming concepts taught through implementation in a particular programming language (Pears et al., 2007) and the need for acquiring many inter-related skills(code designing, writing, debugging, reading, tracing, etc.)

all together at the same time (Gomes & Mendes, 2007). There is also intensive debate amongst computer science educators on what and how to teach introductory programming courses, which is a further source of challenges and confusion for both students and teachers (ACM & IEEE, 2001).

Robins et al. (2003) explain that, psychologically, novice programmers are challenged to develop different kinds of mental models necessary for the task of programming. Learning to program requires an understanding of programming language constructs at the level of syntax, semantics, and machine implementation. Writing a fairly short program further needs a clear mental picture about the problem to solve, algorithmic and code representation of the solution, and about the behaviour of the code during execution. This increases the psychological pressure on novice programmers.

There are relatively a few studies on large class teaching of programming. Kay's (1998) observation on the difficulties associated with large class teaching of programming has a close resemblance to those summarised in Table 1. Chamillard and Merkle (2002) and Brown and Baatard (2008) outlined challenges related to learning resources, environment and assessment overload as student numbers increase.

The difficulties outlined above are broadly consistent with what we anecdotally experienced in large class teaching of introductory programming. The positive outcomes reported as a result of the proposed solutions suggest that the challenges and difficulties are solvable. A limitation of the existing literature is the lack of a holistic study examining the large-class, technical and psychological issues of programming in an integrated manner. The vast majority of programming education studies are experimental with intent of testing a prototype as suggested solution. With the exception of a few multi-national studies, there is a clear gap of continuity and linking within the existing body of literature. Moreover, both the educational and computer science education literature predominantly report the experience from the developed world with a sparse coverage of issues from the developing world, sub-Saharan Africa included. This study addresses the shortcoming in the literature to develop an integrated learning environment for improved programming learning within large class context in sub-Saharan Africa.

3 BLENDED LEARNING DESIGN AT HAWASSA UNIVERSITY

There are attempts to explore practical mechanisms to promote quality education within large class teaching, or at least to minimise some of the negatively perceived effects (Kay, 1998). This paper reports on experience at Hawassa University, Ethiopia, in introducing blended learning of programming in large class. A holistic approach was adopted in the design by integrating the following pedagogical and technological good practices and principles:

1. Adopt alternative pedagogical approaches to promote student learning through course and learning environment design: *constructive alignment* (Biggs, 1996) for course redesign and *conversational framework* (Laurillard, 2002) for learning environment design.
2. Apply technologies: e-learning platform (Moodle) for T&L and for collaboration (Dougiamas, 2004) and specially-designed programming learning tools such as program visualisation and animation tools (Norvell &Bruce-Lockhart, 2004; Sorva, 2012) to support T&L of programming.
3. Harness large class and programming teaching best practices: this includes infusing active learning like in-lecture live coding (Wulf, 2005; Rubin, 2013), pair programming (Hwang, Shadiev, Wang & Huang, 2012), team teaching (Hanusch, Obijiofor & Volcic, 2009) and using the service of senior students (Decker, Ventura & Egert, 2006). Continuous practical assessments (Prasad, 2006) including the less common ones such as reflective journals (Lee-Partridge, 2006) can be used to improve programming learning. Students' learning can further be influenced through embedded motivation strategies such as alignment between assignments with incremental grade improvement (Barros, 2010) and holistic assessment technique (Thompson, 2007).

3.1 Course redesign

Identification of instructional goals and writing performance objectives are amongst the preliminary activities of course design (Dick, 1996).We developed the cognitive domain intended learning outcomes based on a widely adopted computer science model curricula (ACM and IEEE, 2001). The affective domain outcomes envisaged introducing students to the soft (transferable) skills required by software engineers of the 21st century (McKinney & Denton, 2005).

We took a holistic design view in determining T&L activities and assessment tasks. As stipulated in the principle of constructive alignment (Biggs, 1996), T&L activities need to lead to attainment of the intended outcomes. The assessment tasks must measure the level of achievement of the learning outcomes. Accordingly, our design shown in Figure 1 includes educational activities which are considered to be fit for active learning within the context of large class teaching. The activities were teaching-team managed interactive large lectures, student-pair managed laboratory activities, student-pair managed assignment and project activities and mentor (senior student) supported self-managed reading, design, coding and reflection activities. The design envisaged student ownership of their learning through collaborative learning (such as in-class active learning, pair programming and group assignments) and inducing the practice of reflective learning.

In order to achieve this, the design incorporated teaching team members' task of reviewing and planning weekly activities and using the large class lecture to introduce new topics. The lecture classes had active learning components such as live coding and short student activities to bridge the concept building role of the lecture to the application-focused activities in pair programming, assignment and reflective activities of students. Senior students' role was providing support in the form of tutorial, technical and psychological advising and the teaching team responsibility was focused on resource development, collaboration, assessment, feedback and scaffolding support.

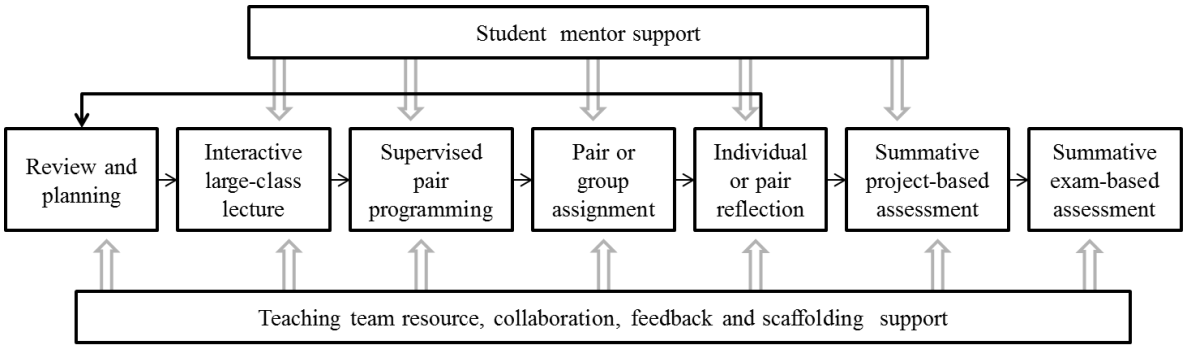

Figure 1: Instructional design for the case courses

3.2 Design of the learning environment

The course design in section 3.1 incorporated close interaction and collaboration within and amongst students, instructors (teaching team) and senior students. Engaging students in learning and assessment tasks was also taken as a mechanism for achievement of the courses' learning outcomes. Guided by constructivist pedagogical principles (Ben-Ari, 2001; Wulf, 2005), we adopted the following recommendations from the literature in the design of blended learning environment to support the course design:

- The need to present real-world problem cases with appropriate performance environment, problem representation and problem manipulation space (Jonassen, 1999).
- The need for interpretive and intellectual support such as related cases, information resources, cognitive tools, conversation and collaboration as well as social support tools (Jonassen, 1999).
- The need for continuous support in the form of guidance, coaching, scaffolding and reflection (Merrill et al. 2007).

- Laurillard's (2002) conception of T&L as interpersonal and internal dialogue between teachers and students that occur at two different levels: at the level of description of the topic and at the level of activities in task environment. The learning resources which support learning as a dialogue include narrative, interactive, communicative, adaptive and productive media forms (Laurillard, 2002).
- The potential of free and open source information technology tools and blended learning platforms to support the type of T&L environment specified above. The tools identified were the *Moodle* integrated learning management system (Dougiamas, 2004) and *Teaching Machine*for program visualisation (Norvell & Bruce-Lockhart, 2004).

Figure 2 presents the overall framework of our learning environment design. The proposed learning environment incorporated three forms of T&L activities: a large class lecture in an auditorium (for 200+ students), a small-group laboratory (maximum 40 students), and e-learning-based support. The three T&L modalities were set to be interactive (shown with bi-directional tick arrows). A passage through the face-to-face lecture and the blended laboratory and e-learning activities was assumed to complete a cycle of programming learning involving new concept building, applying the concept learned and reflecting on the experience.

The large class environment (with one LCD projector and a laptop for teachers) was intended for the team-teaching facilitated concept-building activities. The innovative elements of the lecture were setting and communicating goals and making the lecture class more interactive by inclusion of active learning activities and participatory live coding. The oval-shaped nodes in Figure 2 show the innovative elements introduced.

The small-group laboratory was used for supervised pair-programming activities. In addition to course management systems (ours is Moodle-based), different automated tools were integrated to assist students to coup up with the challenging aspects of novice programming (Section 2.2).

The e-learning-platform had a dual role. It was used as a resource repository, i.e. to support the lecture, assessment and pair-programming activities by distributing notes, case projects, and assessment and laboratory guide questions. Over and above this, the e-learning system was used for supporting student-to-teacher and student-to-student collaboration.

The pivotal elements of the learning environment were the teaching team and senior students (shown in the centre of the diagram in Figure 2). Besides the traditional role of teaching, the two groups combined were set to empower students' ownership of their learning through explicit teaching and guided transition in the learning process. The main strategies used were provision of structured learning resources (notes, laboratory practice and assignment problems), use of criteria-referenced assessment; continuous support and timely feedback.

There were many challenges encountered during implementation. These are covered in Section 4 under an ADR based evaluation of the course and learning environment design.

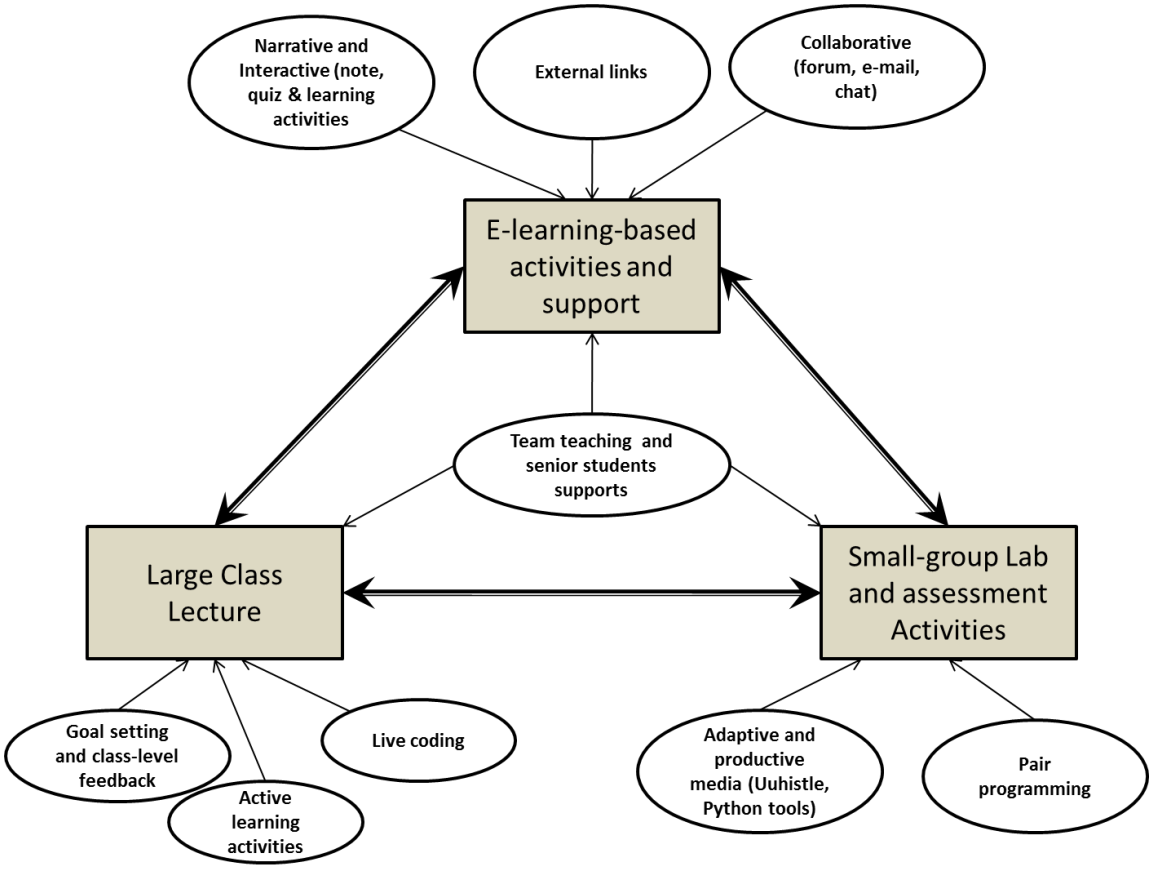

Figure 2: Integrated support structure for blended-learning of programming

4 RESEARCH DESIGN

As discussed in the previous sections, this paper reports on a study that examined the use of blended learning to address the challenges of introductory programming in large class context. The study had a dual goal of addressing the practical problem(s) of learning to program in large class and explaining the outcomes achieved through identification of course and learning environment design principles.

Reeves, Herrington and Oliver (2005) and Laurillard (2012) promote design research based research approaches for this kind of study. ADR, as one variant, combines features from design research (for its focus on building an artefact or intervention) with action research (for its in situ and participatory evaluation) (Baskerville, Pries-Heje & Venable, 2009). The course and learning environment design was presented in Section 3. It was implemented with a Python-based programming course in the first semester of 2012/13 academic year as the first round ADR cycle. The predominantly quantitative result from the first cycle was reported in Bati, Gelderblom and Van Biljon (2014). The second cycle, which is partially reported in this paper, was mostly qualitative and intended to explain the quantitative result from the first cycle. It was conducted with and advanced programming course in C++ in the second semester of the academic year. A total of 219 students were registered all of whom were participants in the first round ADR. The learning environment and course activities were revisited to address gaps identified during the first cycle.

A focus group discussion (FGD) with students was one of the data collection method used to answer the sub-question "*how do the different components of the intervention contribute to support large class teaching of programming.*"The focus group participants (N=13, 7 males and 6 females) were selected on the basis of their examination result of the first programming course. The participants were chosen from lower performing (below the first quartile) and top performing (above the third quartile) students. The two groups met separately in FGD led by independent modulators (or with a

limited involvement of the researcher). Guiding questions which were passed through the ethical approval procedure of University of South Africa (UNISA) were used.The four main FGD questions were:

1. What were the positive aspects of the T&L approach that helped you learn better?
2. What aspects weren't so good?
3. What aspects used to keep you engaged in the course? What deterred you significantly?
4. What improvements do you suggest?

The FGD events were audio-recorded and the facilitators took notes which were used in the data analysis. The data analysis was done following the *grounded action research* procedure (Baskerville & Pries-Heje, 1999). Grounded action research is proposed to improve the rigor of theory-formulation in action research by using units of analysis and techniques from grounded theory such as constant comparative method and different levels of coding (open, axial and selective). The constant comparison method is a process whereby each interpretation and finding is compared with existing findings as it emerges from the data.

The RQDA library of the open source statistical package R (Huang, 2012) was used to facilitate the coding procedure. The FGD data was first transcribed verbatim and the text was fed to the RQDA library. See the excerpt in Figure 3. The coding result is presented in Section 5.

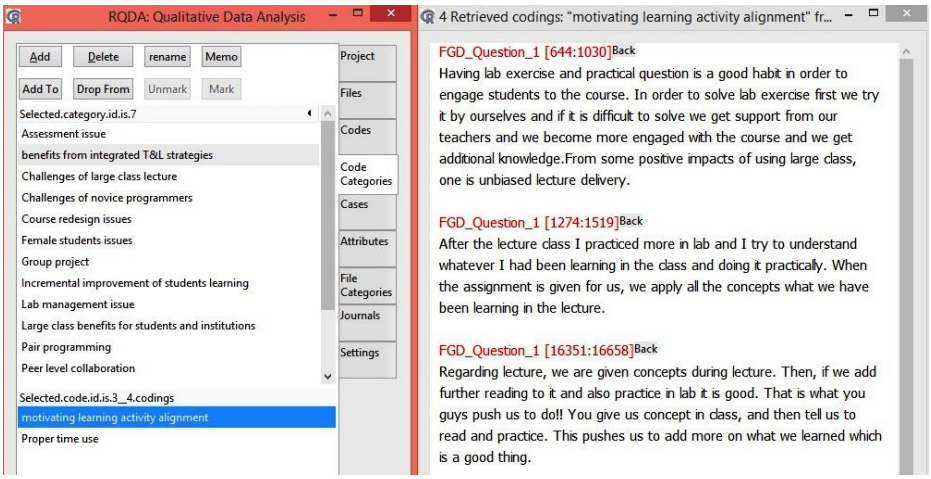

Figure 3: RQDA-based analysis of the research data

5 RESULTS

The FGD question had a thematic focus in the following three areas:

- large class T&L of programming
- technology integration, and
- use of integrated T&L approaches.

Initial coding of the data was made by reading the transcribed documents statement-by-statement in the RQDA's File window. The principle of constant comparison was used to identify 44 codes (basic concepts) from the data. The *Code* column of Table 2 contains a partial list. A constant comparison made amongst the 44 initial codes led to generation of 14 provisional code categories, three of which are shown in the third column of the table.

The code categories emerged range from challenges related to large class programming T&L (4 categories), instructional design and student support (5) to benefits and positive outcomes of the intervention (5). The code categories were refined by rechecking them against the source documents.

As the FGD had a specific thematic focus, which were set by the guiding questions (see Section 4), the emerging code categories had a natural alignment with them. The more important results from the grounded action analysis were the *properties* (or the attributes of the emerged code categories) and their *dimensions* (or range of values that the properties can assume). For example, as shown in the last two columns of Row I in Table 2, the challenges of large class T&L can be characterised by its impact on students and teachers; and student-related challenges include lack of interactivity and difficulty to actively engage in large class activities. Such relationships emerged through examining texts linked to the codes from the source documents (FGD text).

Table 2: partial list of code, code categories and their properties

No	Code	Code Category	Property	Dimension
I	Large lecture: difficult and uneasy to ask question	Challenges of large class lecture	Student challenges	Ranging from lack of interactivity causing tiredness and boredom to loosing comfort to engage in discussion and to ask question.
	Large class and class management issues			
	Lecture "moves" fast		Teacher challenges	Difficulty to manage divergent student preference (or styles of learning) such as interest to move faster while others opt for a slower move.
	The long time the lecture takes (2.5 to 3h)			
II	Learning resources (online) are not used by students	Technology integration	Mechanism, strategy and trajectory	Ranging from timely building of technology use skills, timey availability of resources and activities and systematic engagement (from lecture throughout assignments and quizzes.)
	Pre-posting of educational resources			
	Learning resources: timely delivery is useful			
	Early exposure to ICT skills were advantageous			
	Integration of technologies		Positive impact	Motivation and level of confidence (trust on teachers and learning environment, relinquished initial frustration); incremental change in student engagement (in projects and assignments, spending time in computer laboratories, in pair programming activities)
	Motivation by online resources and Internet			
	Integration of technology as motivation factor			
	The online environment is very useful			
III	Pair programming: lack of support	Pair and group work	Conditions for effective pair and group work	Considering students' attitude or behaviour (pairing strategy); integrating student support; timeliness and appropriateness (relevance and fitness in level of difficulty)
	Pairing: selfish students are not cooperative			
	Experience in pair programming positive		Benefits of pair programming	Motivating experience; knowledge sharing amongst students; improved team work skills
	Pair programming: useful for support to each other			
	Pair programming: improved in the second semester			

Our findings below summarise such relationships derived from the grounded action research. The blended learning intervention was described in Section 3 by categorising it into course design (learning outcomes, T&L activities and assessment tasks) and blended learning environment (large class lecture, pair-programming based laboratory and e-learning support). The findings were summarised under the three themes of the FGD.

5.1 Large class lecture

Introduction of innovative practices was one of the key goals of the intervention (Section 3). The students 'experience in the two semesters suggests persistence of some large class challenges and difficulties. The feedback from the students generally adhered to the challenges outlined in Table 1.The challenges were partially caused by teacher factors (level of readiness for learner centred and large class education).Some conventional class management tools such as signing an attendance sheet were regarded as time-consuming and unproductive.

There were also design elements that were positively recognised for addressing large class issues. Two of the widely appreciated areas are enriching large class lecture by inclusion of in-class active learning activities and in-advance posting of learning resources (notes, laboratory class guiding questions and assignments). Small-class active learning activities such as short algorithm writing, tracing a given code fragment, or live coding were described as "mind-catching" and "entertaining" by some FGD participants. One FGD participant recounted the value of building concepts before and in-class with a follow-up reading and practicing to “push [students]to add more on what [they] learned”. Another one suggested that she went to class after reading the day’s lecture material and used her class to “broaden [her] learning". Overall, the fact students completed their two semester programming courses in a class of 200+ without much complaint and a fair degree of satisfaction suggest the chosen large class strategy a positive effect.

5.2 Technology integration

Two lines of technologies were integrated in the design: e-learning and programming tools (as discussed in Section 3). The internet was used as a main source of resources and additional references. Students reacted positively on integrated use of technologies. The reflection include: use of specialised programming tools (e.g. visualisation tools) to know how the “program works internally” and learn about internal process such as “how a memory [is] allocated, why it is [used, and how they hold] garbage values when …initially reserved”. One informant claimed that “the biggest advantage of the online tool was getting teachers online to support us on demand… and …[s]ome students also post interesting things which made us stay on the course portal longer”.

The students identify prerequisites for better integrated use of technologies. The important ones include building students' technology use skills right from the beginning of their university education; making necessary resources available in a timely manner and aligning technologies with course activities.

Infrastructure was a problematic area. Among the challenges were laboratory management (provision of laboratory facilities and ensuring their level of accessibility, security and academic use) and insufficient technical and administrative support. Frequent power cuts and unexpected interruption of e-learning and other Internet services were amongst the deterring factors.

5.3 Use of an integrated approach

We used the phrase *integrated approach* to refer to the following course and learning environment design components: -

- pair programming based laboratory and assessment activities;

- team teaching and student support by senior students;
- alignment between lecture, laboratory activities and assignments as well as between assignments; and in each of these,
- application of technologies at different levels of T&L, assessment and collaboration activities.

Pair programming

Students found pair programming advantageous to become more familiar with the course. Working in pairs served as motivator and influenced students to support to each other and to build teamwork skills. The following observations were recorded: “doing in pair helps to share our knowledge among ourselves”, “supporting to each other”, and “to learn [in] collaboration”.

The challenges with pair programming include the level of pair commitment, appropriateness of pairing strategy (criteria-based vs. random pairing by instructors), level of student support made available, and the supportive role of resources (how relevant and fit are the learning resources and level of access to laboratories).

Aligned assessment and learning activities

Students valued the dual value of having frequent assessment activities for accumulating good marks and learning from the process. There was appreciation for the following components: assignments that are properly linked with in-class and in-laboratory activities and presentation of sample projects together with evaluation criteria of the assignments.

Group assignments became problematic when the group size increased (students preference was a size of 5 or 6). Additionally, students advised to consider their workload (vis-à-vis their total courses), to ensure that the assessment evaluation procedure differentiate individual-level contribution of teamwork and to punish those involved in plagiarism.

Team teaching and student support

Students reflected on the relevance of cohesive student support, i.e., incorporating continuous, on-demand and target oriented support by teaching team, senior students and by peers. Technical and management support related to the e-learning platform, campus network and computer laboratories were also found essential.

6 DISCUSSION

The results presented in Section 5givea mixed response to the guiding research question: *how do the different components of the intervention contribute to support large class teaching of programming?* The study was initiated with a blended student-centred teaching and learning environment design (Section 3). The design incorporated an integrated teaching strategy that combined large class lectures, pair programming based laboratory practices, and follow-up assignments and projects.

There were positive student reflections about each component of the intervention (separately and in integration): e-learning, visualisation tools, course design with a synergy amongst large-size lecture, laboratory practices and continuous assessment activities. The students' reflection demonstrates that large class lectures can have a positive influence when it becomes more activity-based and synergy with the other components is pedagogically established - like lecture-laboratory-assignment alignment. Laboratory sessions can enhance students' learning of programming with the use of tailor-prepared practical and assignment activities with additional feature of criteria-reference assessment evaluation and in-advance availability of the resources. The other area recognised for improved

student engagement was the student support and collaborative practices for which the e-learning platform can play a decisive role.

Each design component also had shortcomings. Large class lectures were not interactive enough to keep students engaged throughout. Pair programming practices were hampered by pair-level commitment which was influenced by the pairing strategy. Students did comment on balancing their workload against their available time and on the importance of limiting the group size for group-based projects.

We also gained a learned experience from participating in the research process. Figure 4 summarises the learning trajectory. We feel more confident that a student-centred and integrated T&L approach can help students to learn programming in large classes. It also became clear to us that other factors (perceived inaccurately or not perceived at all during the design) can pose problems. Some of these student and teachers factors are under-preparedness for student-centred learning, and inexperience in large-class T&L and management strategies. The process improvement discourse throughout the implementation was the mechanism used to address the observed gaps (such as student disengagement and learning resources usability). Managing the students and the course activities through process improvement encouraged principled engagement of the students, and for us, this was the formative educational gain we had as university professors. We have become convinced about the necessity of a process refinement through continuous evaluation and adjustment on teaching-team, learning resources and technology roles and appropriation of instructional strategies.

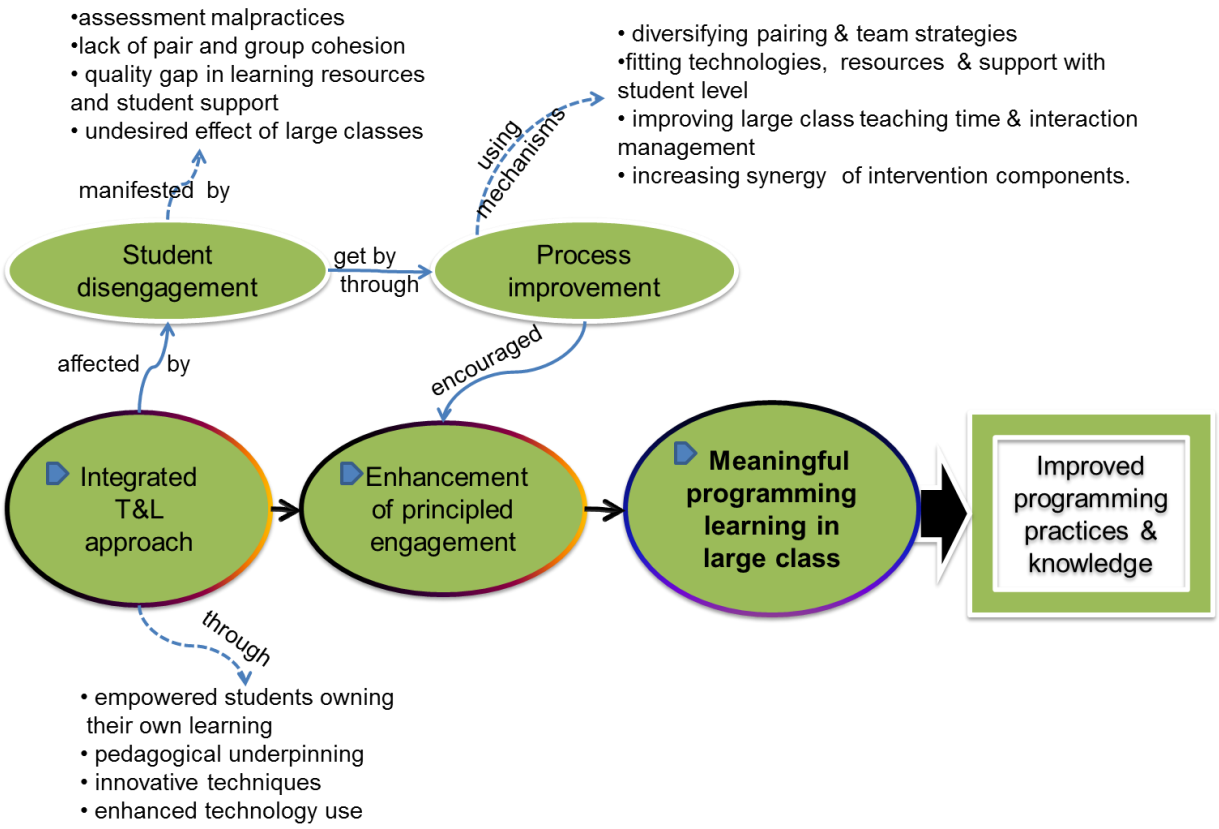

Figure 4: Trajectory of blended T&L of programming as experienced at Hawassa University

Overall, we found that the year-long practice was supportive in promoting *principled engagement of students.* The resultant impact was encouraging for students to create *meaningful learning of programming* in a large class *as advocated by* Mayer (2002).

7 CONCLUSION

This paper presented students reflection on blended learning of programming in a large class context. The course and learning environment design and implementation had a constructivist pedagogical underpinning and integrated large class and programming teaching best practices.

The grounded action research evaluation of the FGD data showed that students had a mixed perception about the intervention. The results suggest that students can learn programming within a large class when instructors consistently apply lecture-laboratory-assignment alignment, in-lecture active learning, and student-support services. Learning resources (technological tools included) and collaborative activities and their timeliness are additional factors. The roles of senior students as mentors and teaching in team are considered to have positive influence.

The students' reflection also pointed out areas of improvement. Unidirectional and longer duration in lecturing are counterproductive to student engagement. Students also become increasingly passive when the learning resources are perceived less fit for purpose and the learning environment and infrastructure are not accessible or lack the expected level of administration. Engagement in assignment and assessment activities can also vary with the extent of student support provided and the quality of assignment evaluation and feedback.

REFERENCES

ACM and IEEE (2001). Computing Curricula 2001: Computer Science. http://www.acm.org/education/curric_vols/cc2001.pdf

Altbach, P.G, Reisberg L., & Rumbley, L. E. (2009). Trends in global higher education: tracking an academic revolution). A report prepared for UNESCO 2009 world conference on higher education. http://unesdoc.unesco.org/images/0018/001831/183168e.pdf.

Ben-Ari, M. (2001). Constructivism in Computer Science Education. *Journal of Computers in Mathematics and Science Teaching*. 20(1), 45-73

Barros, J. P. (2010). Assessment and grading for cs1: towards a complete toolbox of criteria and techniques. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research* (pp. 106-111). ACM.

Baskerville, R., & Pries-Heje, J. (1999). Grounded action research: a method for understanding IT in practice. *Accounting, Management and Information Technologies*, *9*(1), 1-23.

Baskerville, R., Pries-Heje, J., & Venable, J. (2009, May). Soft design science methodology. In *proceedings of the 4th international conference on design science research in information systems and technology* (p. 9). ACM.

Author 1, Author 2, &author 3. (2014)

Biggs, J. (1996). Enhancing teaching through constructive alignment. *Higher education*, *32*(3), 347-364.

Brown, J., & Baatard, G. (2008). Teaching strategies in large class programming courses. *ECU Publications Pre. 2011*

Chamillard, A. T., & Merkle, L. D. (2002). Management challenges in a large introductory computer science course. In *ACM SIGCSE Bulletin* (Vol. 34, No. 1, pp. 252-256). ACM.

Decker, A., Ventura, P., & Egert, C. (2006). Through the looking glass. *SIGCSE Bull.*, 38(1), 46.

Dick, W. (1996). The Dick and Carey model: Will it survive the decade? *Educational Technology Research and Development*, *44*(3), 55-63.

Dougiamas, M. (2004). Moodle: A virtual learning environment for the rest of us. *TESL-EJ*, *8*(2), 1-8.

Gomes, A., & Mendes, A. J. (2007). Learning to program-difficulties and solutions. In *International Conference on Engineering Education–ICEE* (Vol. 2007).

Hanusch, F., Obijiofor, L., & Volcic, Z. (2009). Theoretical and Practical Issues in Team-Teaching a Large Undergraduate Class. *International Journal of T&L in Higher Education*, *21*(1), 66-74.

Hornsby, D. J., & Osman, R. (2014). Massification in higher education: large classes and student learning. *Higher Education*, 1-9.

Huang, R. (2012). RQDA: R-based Qualitative Data Analysis. R package. http://rqda.r-forge.r-project.org/

Iipinge, S. M. (2013). Challenges of large class teaching at the university: implications for continuous staff development activities. http://people.math.sfu.ca/~vjungic/Lipinge.pdf

Ivankova, N. V., Creswell, J. W., & Stick, S. L. (2006). Using mixed-methods sequential explanatory design: From theory to practice. *Field Methods*, *18*(1), 3-20.

Jonassen, D. H. (1999). Designing constructivist learning environments. *Instructional design theories and models: A new paradigm of instructional theory*, *2*, 215-239.

Kay, D. G. (1998). Large introductory computer science classes. *Proceedings of the Twenty-Ninth SIGCSE Technical Symposium on Computer Science Education -SIGCSE '*98. doi:10.1145/273133.273177

Laurillard, D. (2002). *Rethinking University Teaching. A conversational framework for the effective use of learning technologies*. London: Routledge.

Laurillard, D. (2012). *Teaching as a Design Science: Building Pedagogical Patterns for Learning and Technology*. Routledge, Taylor & Francis Group. 7625 Empire Drive, Florence, KY 41042.

Lee-Partridge, J. E. (2006). Using Reflective Learning in an Introductory Programming Course. In *Emerging Trends and Challenges in Information Technology Management*. DOI: 10.4018/978-1-59904-019-6.ch023.

Mayer, R. E. (2002). Rote versus meaningful learning. *Theory into practice*,*41*(4), 226-232.

McKinney, D., & Denton, L. F. (2005). Affective assessment of team skills in agile CS1 labs: the good, the bad, and the ugly. *ACM SIGCSE Bulletin.* 37(1), 465-469.

Merrill, M. D., Barclay, M., & van Schaak, A. (2007). Prescriptive principles for instructional design. *Handbook of research for educational communications and technology*, 173-184.

Norvell, T. S., & Bruce-Lockhart, M. P. (2004). Teaching computer programming with program animation. In *Proceedings of the 2004 Canadian Conference on Computer and Software Engineering Education*.

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., ... & Paterson, J. (2007). A survey of literature on the teaching of introductory programming. In *ACM SIGCSE Bulletin,* 39(4), 204-223.

Prasad, C. (2006). A SWOT analysis of introducing practical assessments in introductory programming. *New Zealand Journal of Applied Computing & Information Technology*, *10*(1).

Prosser, M., & Trigwell, K. (2013). Qualitative variation in approaches to university T&L in large first-year classes. Higher Education, 67(6), 783–795.

Reeves, T. C., Herrington, J., & Oliver, R. (2005). Design research: A socially responsible approach to instructional technology research in higher education. *Journal of Computing in Higher Education*, *16*(2), 96-115.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, *13*(2), 137-172.

Rubin, M. J. (2013). The effectiveness of live-coding to teach introductory programming. In *Proceeding of the 44th ACM technical symposium on Computer science education* (pp. 651-656). ACM.

Sarpong, K. A. M., Arthur, J. K., & Owusu, P. Y. (2013). Causes of Failure of Students in Computer Programming Courses: The Teacher–Learner Perspective. *International Journal of Computer Applications*, *77*.

Sorva, J. (2012). *Visual program simulation in introductory programming education*. Doctoral dissertations 61/2012, Aalto University publication series

Teferra, D., & Altbachl, P. G. (2004). African higher education: Challenges for the 21st century. Higher Education, 47(1), 21–50

Thompson, E. (2007). Holistic assessment criteria: applying SOLO to programming projects. In *Proceedings of the ninth Australasian conference on Computing education-Volume 66* (pp. 155-162). Australian Computer Society, Inc..

Unesco (2010) Trends in Tertiary Education : sub-Saharan Africa. UIS Fact Sheet, December 2010, No. 10. http://www.uis.unesco.org/Library/Documents/fs10-trends-tertiary-education-sub-saharan-africa-2010-en.pdf

Wulf, T. (2005). Constructivist approaches for teaching computer programming. In *Proceedings of the 6th conference on Information technology education* (pp. 245-248). ACM.

Sein, M. K., Henfridsson, O., Purao, S., Rossi, M., & Lindgren, R. (2011). Action Design Research. *Mis Quarterly*, *35*(1).