A blended learning approach for teaching computer programming: design for large classes in Sub-Saharan Africa

Tesfaye Bayu Bati[a]*, Helene Gelderblom[b] and Judy van Biljon[c]

[a] *School of Informatics, Hawassa University, Hawassa, Ethiopia;* [b]*Department of Informatics, University of Pretoria, Pretoria, South Africa;* [c]*School of Computing, UNISA, Pretoria, South Africa*

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 reports on an investigation into 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 was designed by integrating constructivist learning models of Constructive Alignment, Conversational Framework and the Three-Stage Learning Model. Design science research is used for the course redesign and development of the learning environment, and action research is integrated to undertake participatory evaluation of the intervention. The action research involved the Students' Approach to Learning survey, a comparative analysis of students' performance, and qualitative data analysis of data gathered from various sources. The paper makes a theoretical contribution in presenting a design of a blended learning solution for large class teaching of programming grounded in constructivist learning theory and use of free and open source technologies.

Keywords: programming; large class teaching; blended learning; constructivist learning

1. Introduction

The challenges of teaching and learning (T&L) programming are widely recognised, the consequences of which include high failure and dropout rates in introductory programming courses (Mendes, Paquete, Cardoso, & Gomes, 2012). Large class teaching of programming can exacerbate these challenges (Apple & Nelson, 2002). Students in large classes may feel isolated and anonymous, leading them to disengage and dissociate from attendance (Kerr, 2011).

Higher education continues to apply large class teaching to address the growing demand for access and to manage the rising financial constraints

*Corresponding author. Email: tesfayebayu@hu.edu.et

(Kerr, 2011). The stake is particularly high in Sub-Saharan Africa where there is a steady growth in tertiary education enrolment without having equivalent expansion of institutional capacities (UNESCO, 2010; Yizengaw, 2008).

Design science research (DSR) is integrated with action research to design and empirically evaluate a course and blended learning environment, which is referred to as an *intervention* hereafter. Blended learning combines face-to-face and information technology (IT) supported instructional activities (Hoic-Bozic, Mornar, & Boticki, 2009). The study was conducted in an Ethiopian university with a class of 216 students. The main research question was:

> How can a blended learning approach be used to improve large class teaching of programming?

The rest of the article is organised as follows. A review of literature is presented in Section 2. Sections 3 and 4 describe the intervention design and its implementation. The research design, the findings, discussion on the findings and the concluding reflection are presented consecutively from Section 5 through 8.

2. Related work and motivation

Our study concerns novice programming and draws on innovations from pedagogy, information technologies and large class teaching. Section 2.1 discusses novice programming while Sections 2.2–2.4 summarise pedagogical and technical strategies for improved T&L of programming.

2.1. Novice programming

Experience in the last 40 years shows that learning to program remained hard, and introductory programming courses endured high failure and drop-out rates (Mendes et al., 2012; Robins, Rountree, & Rountree, 2003). Recent large-scale projects of the McCracken group, the Leeds group, and the BRACE and BRACElet projects (Clear et al., 2011; Fincher et al., 2005) have examined novice programmers to understand their problems and experience in learning to program. This paper focuses on two main areas:

Knowledge and skills required for programming: novice students are found less successful at the programming task than their teachers' expectation (McCracken et al., 2001). Studies under the BRACElet project (Clear et al., 2011) investigated the underpinning skills and knowledge required for code writing. The result shows a positive correlation between mastery of code tracing and students' ability to reason about the code, and concluded that a combination of code tracing and code explanation skills is a strong predictor of performance on code writing.

The psychology of programming: programming as a cognitive activity demands different kinds of mental models – of a problem to be solved, its algorithmic representation, syntax of their code and their semantic equivalent (Robins et al., 2003). Further mental models necessary are that of the mechanics of debugging, editing, compiling and the internal behaviour of executing code. Sorva (2012) showed that novices encounter challenges in learning to program due to underdevelopment of those multi-faceted mental models. Robins et al. further noticed that students cannot frame the content knowledge they gained into a "chunk" of related knowledge (called a *schema)*, which is essential for problem-solving and code writing.

2.2. Constructivism and learner-centred assessment

Ben-Ari (2001) and Sorva (2012) promote *constructivist learning theory* for effective T&L of programming. Constructivists conceive learning as construction of knowledge through active engagement of the learners in solving authentic problems (Ramsden, 2003). This active engagement, i.e. student-centric instruction, can benefit students in developing mental models (Ben-Ari, 2001; Wulf, 2005).

Examples of constructivist pedagogical models are Mayes and Fowler's (1999) *Three-Stage Learning Model*, Laurillard's (2002) *Conversational Framework* and Biggs' (2003) *Constructive Alignment.* The former explains learning as a three-stage process: *conceptualisation* (creating initial exposure to a new concept), *construction* (applying new concept) and *dialogue* (conversing, reflecting and extending the new concept). The conversational framework takes learning as a dialogue in *discursive* (presentation of new concept), *interactive* (with tasks and resources), *adaptive* (putting ideas into practice) and *reflective* forms (reflecting on theories and practices). *Constructive alignment* is a tool for instructional design and involves defining and communicating intended learning outcomes (ILOs), planning T&L tasks that enable students to achieve ILOs and assessments that ensure achievement of preset ILOs. ILO specification and evaluation can be improved by applying learning taxonomies, such as Bloom's (Fuller et al., 2007) and SOLO (Biggs & Tang, 2007).

Thota and Whitfield (2010) integrated constructive alignment and SOLO for programming course design and assessment. They used the conversational framework to determine the roles and interactions in a T&L process. The three-stage model is to characterise information technologies and learning resources suitable for achieving student learning at each stage of the learning cycle (e.g. Hadjerrouit, 2005, 2008; Roberts, 2003).

Learner-centred formative assessment is an integral part of constructive alignment, and is intended to foster students' learning through frequent and prompt feedback (Webber & Tschepikow, 2013). Formative assessment has two functions: *feedback* (evidence about student learning) and *evaluation*

(judgement on learning) (Taras, 2005). Balanced achievement of these functions demands transformation from written examination dominance, which is common in Ethiopia (Bass, 2009) to more authentic, interactive and continuous assessments (Black & Wiliam, 1998).

2.3. *Information technologies for teaching and learning of programming*

Kerr (2011) and Cuseo (2007) argue that large class teaching causes decline of active student engagement and the quality of instructors' interaction with their students. The impact of these problems can be minimised by using learning management systems (Francis, 2012). Hoic-Bozic et al. (2009) and Azemi and D'Imperio (2011) demonstrated the use of e-learning communication (email and chat) and student support (discussion forum, grade notification and provision of interactive self-practice materials) to support engagement.

Various programming tools have also been designed to address the problem of mental model development described in Section 2.2. Program visualisation and simulation tools can be used to demonstrate execution steps and runtime behaviour of the program code (Rajala, Laakso, Kaila, & Salakoski, 2008). There are many program visualisation tools, including *UUhistle* for Python, *Jeliot3* and *JIVE* for Java, and *Teaching Machine* for C++. Empirical studies on integration of such tools illustrate a positive outcome in improving students' experience in introductory programming courses (Rajala et al., 2008; Sorva, 2012).

2.4. *Strategies for large introductory programming courses*

Computer science education literature does not sufficiently cover the nature and problems of large class T&L of programming (Apple & Nelson, 2002; Sheth, Bell, & Kaiser, 2013). The following instructional strategies are relevant to large class teaching of programming:

Engaging students for deep learning (Marton, Hounsell, & Entwistle, 1997): by infusing short active learning activities in the form of code walkthroughs, group code writing and code debugging that engage students during lecture and promote student-level collaborations (Wulf, 2005), and applying teaching methods that necessitate student energy, problem-solving and cooperative learning (Ramsden, 2003). Examples of engaging activities in programming are *in-lecture live coding* (Pears, 2010), collaborative learning, mainly in the form of *pair programming* (Hwang, Shadiev, Wang, & Huang, 2012), and use of *reflective journals* (Lee-Partridge, 2006).

Using support mechanism for improved class management: Ives (2000) recommends in-class activities with a lot of *exam-directed problems* to improve students' attendance. Student support can be facilitated through *team teaching* (Hanusch, Obijiofor, & Volcic, 2009), and the use of *undergraduate students* as assistants (Decker, Ventura, & Egert, 2006).

Aligning assessment activities: Barros (2010) tested *alignment between assignments* (connecting consecutive assignments in a way that the latter contains the whole or part of its predecessor) and *incremental grade improvement* (allowing students to improve their poor grades by achieving a threshold higher grade in the next assignment). Thompson (2007) applied *holistic assessment criteria* for assignments to engage students in self-monitoring and to enforce balanced programming skills development.

Creating closer relationships and a sense of community among students: by increasing student support services, introducing instructors who fulfil the role of communicators, enthusiastic engagement and a team spirit amongst tutors and instructors (Bryson & Hand, 2007).

3. Course and learning environment design

The case course was an introductory programming course taught through *imperative first* approach (Curricula, 2001). The course was redesigned by applying constructive alignment and Fuller et al.'s (2007) two-dimensional adoption of Bloom's taxonomy. The course's ILOs (Table 1) were drawn from the ACM/IEEE CS 2001 curriculum and the 2008 interim update (Curricula, 2001; McCauley & Mcgettrick, 2008). Corresponding aligned assessment activities are shown in the second column of Table 1.

The assessment activities were developed with an emphasis on learner-centred formative assessment, namely assignments, projects and journals. Additional assessment-related good practices such as alignment between assignments, incremental grade improvement (Barros, 2010) and online and face-to-face senior student support (referred to as student mentors) were incorporated. The formative assessment activities were evaluated using SOLO-based holistic assessment criteria adopted from Thompson (2007).

Table 1. Course ILOs and aligned assessment tasks.

No.	Intended learning outcomes	Assessment tasks
1.	Explain fundamental program constructs and programming concepts	Final exam; non-graded reading assignment
2.	Apply the techniques of structured (functional) decomposition for problem-solving and algorithm redevelopment	Graded and non-graded pair-based assignments
3.	Design and test algorithms for solving elementary problems	Group-based project
4.	Write, debug, trace and explain simple programs that implement designed algorithms, applying fundamental programming constructs and data objects	Individual and pair reflective journal
5.	Use teamwork techniques in problem-solving and program development	Final examination

Learning resource development follows the three-stage learning model of Mayes and Fowler (1999). Activity sequences among and within lecture, laboratory and independent and group work activities are planned and implemented based on the conception of learning in Laurillard's conversational framework. As depicted in Table 2, the learning process begins by presenting a new concept with narrative materials. In-class interactive exercises and live coding follow to help students apply the new concepts on authentic problem-solving (the construction phase), leading them to pair-programming laboratory activities where they apply IT tools. Assignments, including reflective journals, infuse dialogue among students and the teaching team. The blended learning environment of the course is shown in Figure 1.

The environment combined large class lectures, Moodle-based e-learning (Dougiamas & Taylor, 2002) and small-group laboratory and assessment activities. The three components were interrelated (shown with the bold arrows) with the Moodle system serving a central role for resource presentation, communication, collaboration and student support.

4. Context and path of implementation

Similarly to many institutions in Sub-Saharan Africa, Ethiopian universities face major challenges including a shortage of qualified faculty, poor infrastructure and facilities, and the consequent inability to meet increasing demands for access while maintaining quality (Yizengaw, 2008). Computer-based instruction is often affected by poor Internet connectivity and frequent disruption of electric power supply.

The educational culture in the country has inadvertently tended to promote shallow learning (Bass, 2009). Ashcroft and Rayner (2012) affirm an overwhelming focus on lectures followed by terminal examinations, and a trend to see teaching largely in terms of a transfer of knowledge from teacher to students. There are some arguments relating these trends to the prevalence of large class sizes (Tessema, 2009).

The case course defined in Section 3 is part of the Computer Science, Information Systems and Information Technology curricula of Hawassa University. English is the medium of instruction as is customary in Ethiopian secondary and higher education institutions.

There were 216 students (159 males and 56 females) who were registered for the course for the first time. The students had a similar academic background as they were all admitted directly from secondary schools based on a nationally administered entrance examination, and followed a nationally standardised preparatory school curriculum. However, there were variations in ICT skills and English language fluency, and a difference in the level of active participation in interactive sessions.

Table 2. Constructivist course design.

		Four forms of learning as dialogue that involves discussion, adaption, interaction and reflection (Laurillard, 2002)			
		Discursive: presentation of new concept	Interactive: interacting with teacher-constructed tasks	Adaptive: putting ideas into practice, modifying one's ideas and adapting ones actions.	Reflective: reflect of the learner's performance by both teacher and learner
Phases of Blending (Mayes & Fowler, 1999)	Conceptualisation (programming concepts)	Learners' experience: attending and apprehending Media form: *narrative:* slide presentation, code presentation, lecture note, web resources and library materials; *productive:* live coding with IDE	Learners' experience: discussing, coding, tracing and explaining Media form: *interactive and communicative:* in-class small-group collaboration with case problems Pair, group and self-practice with web-based questions (quizzes)		
	Construction (programming activities)		Learners' experience: designing and testing algorithm, experimenting programming concepts, structures, styles and standards; and, practicing program writing, debugging, testing and documenting Media form: interactive, communicative, adaptive and productive Pair and group problem-solving, coding and discussing web-based questions Pair and self-experimenting and practising with visualisation and tutorial tools (*Uuhistle*, *Python interactive tutorial*)		

	Four forms of learning as dialogue that involves discussion, adaption, interaction and reflection (Laurillard, 2002)			
	Discursive: presentation of new concept	Interactive: interacting with teacher-constructed tasks	Adaptive: putting ideas into practice, modifying one's ideas and adapting ones actions.	Reflective: reflect of the learner's performance by both teacher and learner
Dialogue (interaction, collaboration and discussion)		Pair and group graded and non-graded projects using adaptive medias (*IDLE/PyScripter, PyGame, Python Turtle, Tk/Tkinter and additional web resources*) Student support: facilitation and feedback by instructors; synchronous and asynchronous support by mentors	Learners' experience: presenting solutions, analysing and recognising higher order concepts and relationships, reciting learning experience Media form: productive, communicative and interactive structured reflective journal, verbal presentation on solutions and the development process; Web-based feedback and grade notification for submitted assignments; Online discussion forum for inquiries, grievance and feedback; Student mentors led ambiguity resolution (both in-person and online)	

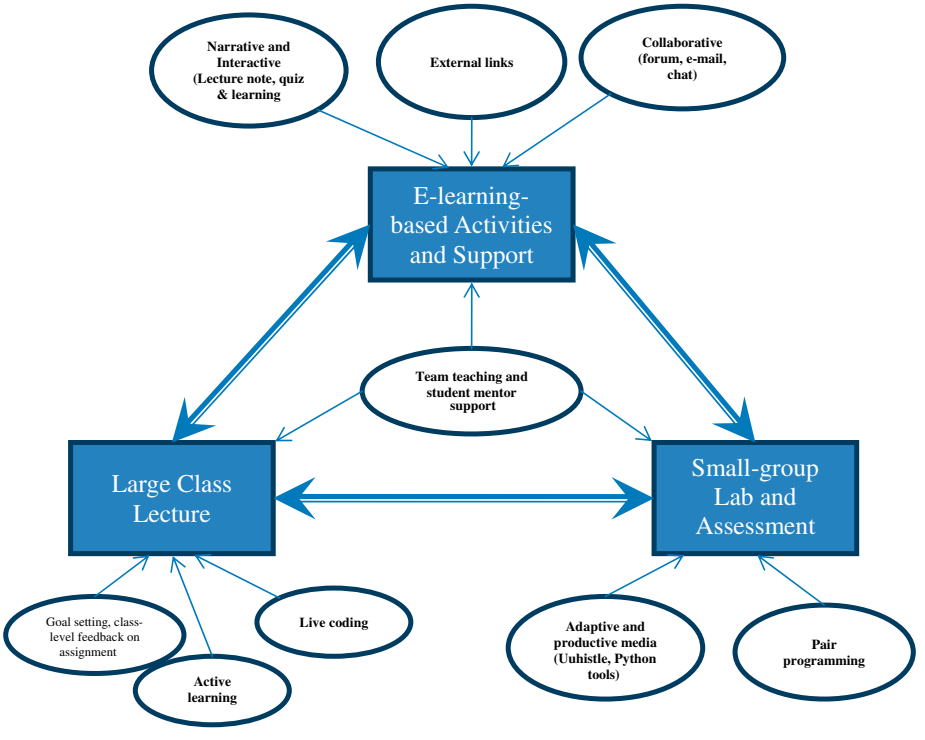

Figure 1. Integrated blended learning environment.

Figure 2 depicts the implementation path. It was started with an expert panel review. The role and composition of the expert panel is discussed in Section 5. Student mentors (14 in total) were trained on techniques of providing student support by a pedagogical expert. Students were introduced to the course's proposed T&L and assessment mechanisms, the expected role and responsibilities of students and instructors, and the embedded research and ethical issues.

The principal component of the intervention is the *weekly educational and research activities* that ran for 12 consecutive weeks. The constructivist course design (Table 2) was enacted within the context of the course T&L environment (Figure 1). There were some changes and refinement made during the course of the enactment based on student feedback, continuous evaluation of the process by the teaching team (in weekly meetings) and feedback from student mentors (after evaluation of every assignment). The enacted educational activities are discussed below.

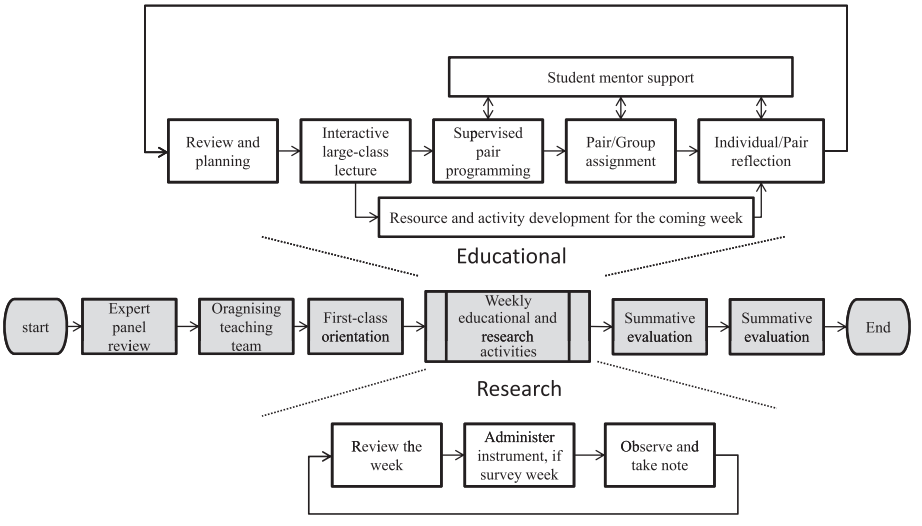

Figure 2. Implementation activity flow diagram.

4.1. Interactive large class teaching with a team-teaching strategy

Two instructors (one with a second degree and 10 years of teaching experience and a junior with a first degree and one-year experience) offered interactive group lectures that lasted 1.6 h. The instructors shared the responsibility of lecturing and class management. The lecture included concept presentation, demonstration of worked-out solutions with screen projection, live coding with students participating in joint code development and in-class, small-group problem-solving activities. Instructors watched students and provided individualised and team-based help during the problem-solving sessions. The activities served as bridge between the lecture, laboratory and self-practice activities by transiting from *discursive* form of communication during the lecture into *interactive* and *adaptive forms* (see conversational framework in Table 2).

4.2. Pair programming with web-based problems, visualisation and production tools

Student pairs were formed randomly by instructors. Female students were paired separately to encourage engagement. Laboratory sessions involved pair programming using questions from the e-learning portal, i.e. students alternated between *passenger* and *navigator* roles. The Python visualisation tool *UUhistle,* Internet-based interactive tools and Python production tools like *PyGame, Turtle* and *Tk/Tkinter* were incorporated into the laboratory problems. The role of the teaching team (the instructors plus a technical assistant) was provision of whole-class, personalised and ad hoc support.

There were six laboratory sessions per week (2.5 h each) for six different student groups.

4.3. Independent and group-based student activities with interactive resources and student support

Online quizzes and reading directions were regularly posted on the e-learning portal for independent and group learning. To minimise the anonymity problem of large class education, different schemes of student support were concurrently facilitated. The main schemes were student mentors' casual support and tutorials; online forums with active engagement of instructors and student mentors; and female student support by senior female students. Required but non-graded end-of-chapter assignments were also used for formative assessment and feedback by student mentors.

4.4. Constructively aligned assignments with e-learning support

Three sets of learner-centred assessment activities (pair projects) that coincided with three core portions of the course (sequence, selection and repetition with user-defined data structures) were planned to foster students' learning through frequent and prompt feedback. The pair projects were followed by the graded *reflective journal* assignments (Section 2.4). The reflective journal was based on a template and guidelines from Curtin University of Technology (Curtin University, n.d.). One assignment and linked reflective journals were skipped due to time pressure from other courses. Progressive engagement of students was motivated through application of *alignment between assignments* and *incremental grade improvement* as recommended by Barros (2010) and discussed in Section 2.4. The evaluation was based on *holistic assessment criteria* adopted from Thompson (2007).

4.5. Summative assessment – comprehensive group projects and written examination

The summative assessment contributed 62% of the total mark – 10% for code comprehension (reading), 17% for code generation (writing) projects and 35% for the final written examination. The comprehension project exposed students to game programming with *PyGame* from an open source eBook (Sweigart, 2009). Students were required to understand the program logic and organisation of the code by reading the documentation and executing the code. The code generation project prompted students to develop their own Python program of corresponding scope and apply basic software engineering principles. The two projects were group-based, combining male and female students (six to eight students), and were evaluated through oral presentation and question-answering (facilitated by at least two instructors).

The BRACElet project approach (Section 2.1) was adopted for the final examination by taking established questions and assessment procedures from published sources (Lopez, Whalley, Robbins, & Lister, 2008; Shuhidan, Hamilton, & D'Souza, 2009). The aim was to facilitate comparative analysis of students' performance. Our students were familiarised with the nature and composition of the final examination through model questions. Examination papers were evaluated by independent evaluators that excluded the course instructors.

Figure 3 summarises the instructional and assessment sequence of activities embedded in the educational activities outlined above. Both the intervention and its implementation were focused on addressing the main problems of large class teaching: *student anonymity*, *reduced engagement* and *instructors' assessment load*. The solutions enacted were (1) synchronising lectures, laboratory classes and assignments by applying the principle of constructive alignment, (2) applying diverse student support (online and face-to-face by instructors and students mentors) and (3) maintaining an assessment strategy that is timely and relevant but not excessively time consuming (through applying holistic assessment criteria, building division of labour in assessment evaluation and using e-learning for feedback and collaboration).

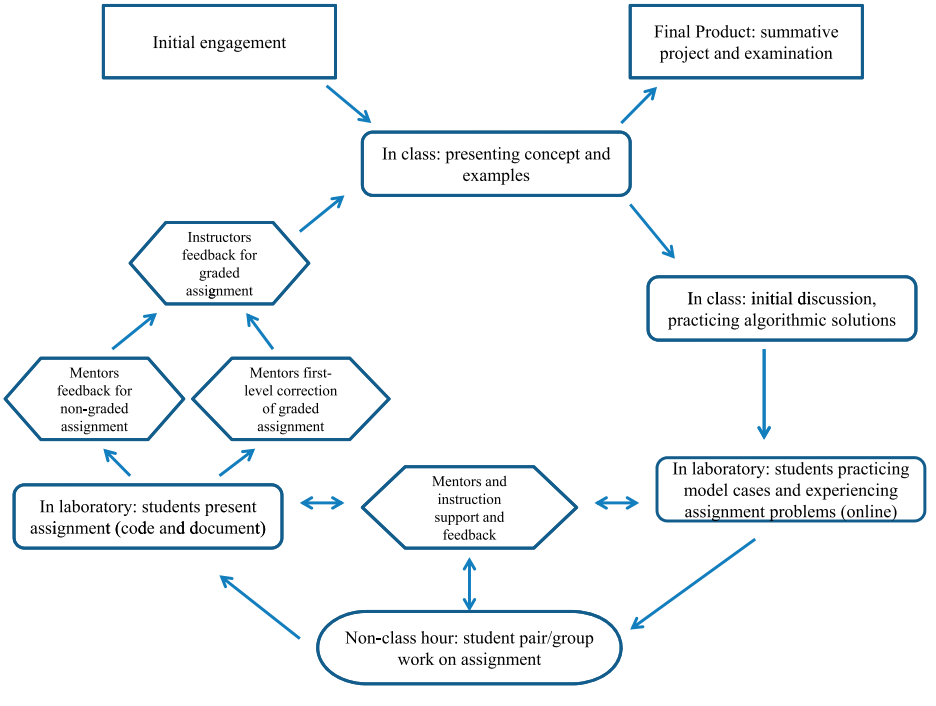

Figure 3. Instruction and assessment cycle (based on Bledsoe, 2011).

5. Research design

Our study is guided by the research question: *how can a blended learning approach be used to improve large class teaching of programming?* Theoretically, the study aimed at establishing a course and learning environment design and enactment guidelines for large class teaching of programming. The practical goal was to examine the impact of blended learning on students' learning of programming.

Laurillard (2012) argues that design science enables instructors to draw upon existing theory to drive new knowledge about T&L practices. DSR, a research method in design science, targets generation of design knowledge relevant for practitioners (Hevner, March, Park, & Ram, 2004), and therefore DSR was selected to develop, implement and evaluate the intervention.

DSR is a research method with three interwoven cycles (Hevner et al., 2004). It begins with a *relevance cycle* of literature review and context analysis to develop conceptual framework for the research. The second cycle, *design and development*, is an iterative design, development and formative evaluation of an artefact or intervention. Artefact evaluation can be done in an artificial (laboratory) or naturalistic context (Venable, 2006). The design and development cycle is followed by a *rigour/theory-building* cycle that targets generation of design principles. The cyclic procedure, however, is not always linear, and overlapping as well as going backward and forward between the cycles is not uncommon.

This article covers the first two DSR cycles. A literature review (Sections 1 and 2) was used to select and integrate technology and pedagogy for the intervention design (Sections 3 and 4). Action research was integrated with DSR for evaluation. Scholars promote cross-fertilisation of DSR with action research for evaluating artefacts in an organisational context (Sein, Henfridsson, Purao, Rossi, & Lindgren, 2011) and to improve theoretical abstraction and knowledge generation from DSR. Beck, Weber, and Gregory (2013) suggest that the latter goal is achieved through the *critical reflection* and *learning* steps of action research.

We adopted the definition of action research given by Melrose (2001). Citing Kemmis and McTagger, Melrose (p. 161) defined action research as *a form of collective, self-reflective enquiry undertaken by participants in social situations.* Newton and Burgess (2008) and Melrose suggest that there are three distinct approaches to action research with corresponding different goals and purposes of the inquiry. The three action research types are *technical* (for knowledge generation), *practical* (for improvement of practice) and *emancipatory* (influencing change or providing conditions for emancipation). We applied the technical mode because of the intended aim of generating knowledge on the design of blended learning and finding theoretical explanation of its application to large class teaching of programming.

The action research was conducted with a *mixed-methods sequential explanatory design* (Ivankova, Creswell, & Stick, 2006). That is, by undertaking two cycles of action research whereby the first cycle evaluates the *effectiveness* of the intervention with a predominantly quantitative data collection and analysis. The second cycle explores *theoretical explanations* through more qualitative means.

We report here on the first cycle action research conducted between October 2011 and February 2012 with the design specified in Section 3. The core research group of the study were the three authors, faculty members and officials from the hosting institution. As a PhD study by the first author, he took the lead responsibility under the direction of the supervisors (second and third authors). Selected senior Computer Science and Education faculties and a representative from the academic quality assurance unit of the host university served as members of the expert panel (Figure 2). The panel had the authority to enforce educational quality, and to monitor and affirm progress made with their pre- and post-implementation evaluation.

The intervention design (Section 4) requires participation of many additional stakeholders whose roles fall under *mini-project research groups*, i.e. the members might not persistently engage in the process (Melrose & Reid, 2000). The group members include course instructor(s) who are assigned to handle a course for a semester and student mentors recruited to provide student support. The groups provided formative evaluation from the perspectives of their engagements: instructors and technical assistants by continuous observation, note-taking and making ongoing evaluative meetings (weekly); student mentors by informal observation of students' engagement and attitude.

Evaluation of the success and impact of blended learning intervention is often contentious, partly due to the diverse drivers that encourage integration of IT into education and students' experience (Ginns & Ellis, 2007). Ginns and Ellis identified quality T&L, flexibility, skill development and access as some of the main motivations for adoption of ICT. With a focus on learning quality, we have used learning effectiveness and student satisfaction from the Sloan Consortium's five pillars of quality of online education (Lorenzo & Moore, 2002) as two means to evaluate the success of our blended learning intervention. The first round action research evaluated the effectiveness *of the intervention* by analysing the holistic nature of the learning experience through students approach to learning survey (Biggs, Kember, & Leung, 2001) and conducting comparative achievement tests. Sub-Sections 5.1 through 5.3 present the data collection and analysis in the first cycle.

5.1. Approach to learning survey

The theory underlying the students' approach to learning (SAL) survey suggests that students take different approaches to their learning. The two broad

approaches are *deep learning* (necessary for higher order learning) and *surface learning* (a kind of rote learning for scoring marks) (Marton & Säljö, 1976). The choice of one of the approach depends upon interplay of many factors explainable with Biggs' 3P Model (Biggs & Tang, 2007). Biggs and Tang developed a *revised two factor study process questionnaire* (R-SPQ-2F) for SAL survey. The instrument has 20 five-point Likert scale items.

We conducted R-SPQ-2F survey to determine the level of influence of our intervention on SAL to program. It was administrated online twice early in the semester (before) and again at the end (after), which is referred to as a *contextual approach*. The response rate for the first round was 178 (77%) and 122 (53%) during the second. The second survey was affected by frequent power interruptions during the survey week. Cronbach's α values for scale reliability in this study were at an acceptable level of .79 for the *before* and .85 for the *after* surveys. Data analysis included 66% of the first and 51% of the second round responses after data cleaning.

The analysis was done according to a procedure proposed by Biggs et al. (2001) for contextual approach administration. The class-level mean score of the students' responses was computed separately for the *before* and *after* surveys. The difference between the *after* and *before* mean scores is used to determine the success rate of the intervention. There is no clear guidance on how to interpret R-SPQ-2F mean scores. Hamm and Robertson (2010) adopted simple criteria of the greater the difference, the greater the strength of the preference; and conversely the smaller the difference, the weaker he preference for either learning approach.

5.2. Comparative performance test

Assessment is one of the main challenges of large class teaching (Ward & Jenkins, 1992). Ward and Jenkins, for example, raised the workload and the reliability and validity issues when assessment procedures are diversified in response to the demands of large class teaching. Our assessment design involved usage of student mentors and holistic criteria for faster evaluation.

Two different analyses were done on assessment activities. First, performance of our students was described with descriptive statistics, and the interrelation between different assessments was determined with correlation analysis. The correlation was established with an open source package called PSPP (PSPP, n.d.). The correlation helps to examine the evaluative role of formative assessments (vis-à-vis examinations), besides their feedback role. As in Rajalingam and Oo (2011), the correlation can also be used to measure the academic progress of students. Important for the Ethiopian context, where terminal examinations are emphasised (Bass, 2009), positive correlation with written examination helps open discussion on learner-centred assessments.

Second, the performance of our students was compared with students from other institutions using examination questions drawn from multinational and multi-institutional computer science education research projects (Lopez et al., 2008; Shuhidan et al., 2009). The evaluation criteria and data presentation format used in the source literature are adopted for the comparison purpose. The aim was to get an indicator of our students' comparative performance within limits of the time, context and possibly procedural differences.

5.3. Qualitative data

Qualitative text analysis method was used to analyse descriptive data from students' reflective journals (from two rounds of submissions), researchers' memos and transcriptions of expert panel, and instructors' and students' meetings.

We merged action research with analysis techniques from grounded theory as proposed by Baskerville and Pries-Heje (1999) for data analysis. Data from students' reflective journals were analysed through coding procedures from grounded theory: concepts in the data were identified and named, and then grouped at a more abstract level into categories applying the constant comparison methods from Glaser and Strauss (1967). We supplemented the emerging result with content analysis of textual data from the researcher's memos and transcriptions of the expert panel and participant meetings, as recommended by Elo and Kyngäs (2008) and Srivastava and Hopwood (2009). Through memoing during coding, the process eventually developed in abstraction to find relationships between the categories identified, and finally a story (or core category) was developed as a generalised relationship. Coding was done with a qualitative data analysis (RQDA) library of the open source statistical package, R (Huang, 2012).

6. Findings

We first present the results of the R-SPQ-2F questionnaire survey, along with the correlation analysis made in Section 6.1. The inter-assessment analysis is presented in Section 6.2 and the comparative performance analyses in Section 6.2. The results of the qualitative analysis are summarised in Section 6.3.

6.1. R-SPQ-2F questionnaire

Table 3 presents the results of the R-SPQ-2F survey. The class-level mean score for the first round administration (*before*) for the deep approach (DA) was 39.27 and 40.9 in the second round (*after*). The DA's mean score difference was 1.58. The surface approach's (SA) mean score was 27.26 *before*

Table 3. Result of the SAL survey.

Mean score	*N*	DA	SA
Before	119	39.27	27.26
After	62	40.9	27.08
Diff. (after – before)		1.58	−.18

and 27.08 *after* with a marginal mean score difference of −.18. This result suggests some positive change towards a deep approach and a modest decrease in the SA at class level.

A correlation analysis made between the R-SPQ-2F and students' final examination result is presented in Table 4. The analysis was performed on the results of 34 students who had completed both the *before* and *after* R-SPQ-2F surveys. The DA and SA values for the correlation were calculated by subtracting the before score from the after score of each student.

There was a fair correlation between the DA and students' examination result ($r(32) = .359$, $p = .019$). This suggests that students who adopted a DA performed relatively better. In contrast, the negative correlation between SA and the examination ($r(32) = -.026$, $p > .5$) established no substantial relationship.

6.2. Inter-assessment analysis

Table 5 presents the descriptive statistics of the students ($N = 214$) in the different assessment activities – two coding assignments (Proj1; Proj2), two reflective journals (RJ1; RJ2), summative code reading and writing projects (Proj3-R and Proj3-W), and a final examination. Table 5 shows that the mean score of the students in every assessment activity is above the passing cut-point of 50% of the weight allocated.

Figure 4 shows the performance trend across all the assessment activities. The box plots (the marks in the inter-quartile ranges) for the learner-centred assessments were smaller and have shorter whiskers than the final examination – showing a concentration of marks around the median. The final examination has a larger box plot with longer whisker indicating a large performance divergence in the examination, with larger standard deviation.

Table 4. Correlation between SAL and performance in examination.

	DA	SA	Final exam
DA	1		
SA	.022	1	
Final exam	.359*	−.026	1

*A fair positive linear relationship.

Table 5. Descriptive statistics on students course results.

Assessments	Proj1	RJ1	Proj2	RJ2	Prog3-R	Proj3-W	Exam
Weight	12%	6%	12%	8%	10%	17%	35%
Mean	10.93	5.40	10.35	7.15	8.70	14.46	18.29
Stand. deviation	.98	.43	.80	.62	.64	1.35	6.83

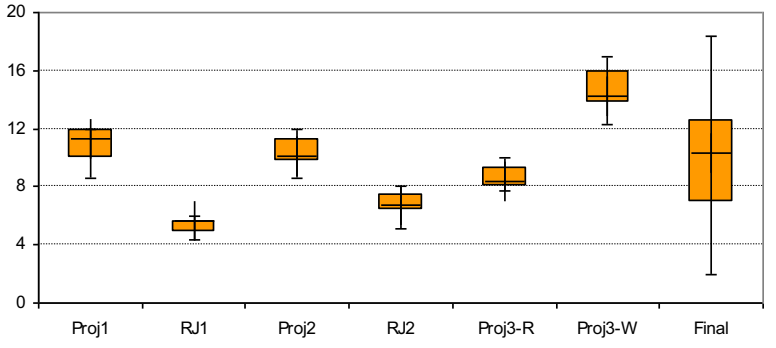

Figure 4. Students' result in assessment activities with a scaled-down final result (to 20%).

Correlation among the assessment activities (Table 6) overall was positive but weak. A moderate positive correlation was found between the examination (weight = 35%) and total marks ($r(214) = .726$, $p < .001$), indicating the discriminatory role of the examination. Fair correlation is found between the code-writing project (*P3-W*) and the final examination ($r(214) = .326$, $p < .001$), which can be explained in terms of the focused and interactive evaluation and feedback described in Section 4.

6.3. Comparative performance

The performance of our students was compared with that of students from two tertiary institutions as a measure of the impact of the intervention. The

Table 6. Correlation analysis among assessment activities.

Assessment activity	Proj1	RJ1	Proj2	RJ2	P3-R	P3-W	Final
Proj1	1						
RJ1	.014	1					
Proj2	.228	.159	1				
RJ2	.108	.460	.215	1			
P3-R	.100	.107	.153	.197	1		
P3-W	.164	.096	.171	.172	.350	1	
Final	.182	.227*	.166	.210*	.138	.326*	1

*Fair or moderately strong correlation.

two institutions are located in developed countries and are believed to have better institutional capacity than Hawassa University (HU). We chose the two universities because of the availability of the question papers and students' results in the literature with sufficient description of the evaluation procedure (Lopez et al., 2008; Shuhidan et al., 2009). The comparison was made between multiple choice, tracing and *explain in plain English* questions.

6.3.1. *Comparison with multiple-choice questions*

Eleven multiple-choice questions from Shuhidan et al. (2009) were used for the comparison. The result in the source document was presented in terms of Lord's *level of difficulty* scale (Lord, 1952). Lord's scale considers a question as *easy* if 85% or more students selected the correct response. A question has *medium* difficulty if 51–84% of the students picked the correct response and *hard* if 50% or less answered correctly. Table 7 displays the result from the two institutions ($N = 220$ for the other institution).

The results reflect similar student response levels for five of the eleven questions (45.45%). HU results were one level of difficulty lower than the other university in the remaining six questions, with a relatively low percentage of correct responses for four questions: 8, 15, 18 and 19.

6.3.2. *Comparison with tracing and "explain in plain English" questions*

Lopez et al. (2008) was consulted for the comparison of the tracing and explain in plain English questions. Lopez et al. presented their students' performance as mean score for each question category.

The aggregated mean score for two tracing questions (a *while* and *for* loop with *nested conditional*) was 2.40 out of five in Lopez et al. ($N = 38$) and 2.08 ($N = 216$) for HU. As one of the three "explain in plain English" questions was not included in Lopez et al., we compared their aggregated mean score for the three questions (which is 3.20 out of eight) with a scaled-up mean score for two questions of HU (3.12 out of eight). The mean score difference was .32 for tracing and .08 for the "explain in English" questions in favour of the other university.

Table 7. Comparison performance.

Other university ($N = 220$)	Q. no.	2	3	4	7	8	11	15	17	18	19	20
	Difficulty*	M	M	M	E	E	H	M	E	M	M	M
Hawassa university	Correct (%)	74	68	82	59	59	49	43	73	43	27	55
($N = 216$)	Difficulty	M	M	M	M	M	H	H	M	H	H	M

*E = Easy, M = medium, H = hard.

6.4. Qualitative indicators

Students' reflections in two rounds of reflective journal submissions were analysed applying the coding mechanism typically associated with the grounded theory method. The first round ($N = 216$) was an individual assignment while the second was pair-based ($N = 108$) and submitted three weeks after the first. The grounded theory analysis led to emergence of the following provisional relationships, which shall be further developed with a predominantly qualitative research in the next cycle of action research.

6.4.1. First-time programmers begin with a plenitude of challenges

Challenges that emerged are misconception and misinformation, insufficiencies in academic background, novelty of programming concepts and practices, and the problems of learning context and culture. Students were led to think of programming as difficult due to misconceptions (e.g. not distinguishing programming skills from basic ICT skills) and misinformation by senior students about its difficulty. Lack of basic exposure to ICT and programming concepts in secondary school was a typical shortcoming. Students' perception of programming as "conceptual" and reading as a main learning strategy put students at risk during the coding project. This is aggravated by lack of practicing problem-solving on paper and poor habit of laboratory use. Psychologically, students were strained by stringent program errors and a feeling of dependency syndrome – believing that they are not capable of programming, and hence to depend on others to complete their assessments.

6.4.2. Integrated approach to programming leads to improved student engagement

Integrated support by instructors and student mentors helped students to develop persistence in engagement. The progressive improvement was supported by pair programming (mainly for quick debugging), mentor support (to boost morale and casual tutoring) and reasonably timely evaluation and feedback. The second round reflective journals demonstrated clear progression in students' engagement. There were positive developments in independent and group programming culture (e.g. practising during off-class time, building teamwork practice and using diversified information sources) and in the level of confidence in their ability to program. For example, one student group reported: "after accomplishing the second assignment, we [have] develop[ed] our skills, confidence, [and] positive attitudes".

The challenges and progressive improvement in students' engagement emerged from the content analysis of textual data from instructors' memos and transcriptions of meetings of instructors and student mentors. The summative group-based project (code reading and writing), which was

conducted two weeks before the end of the semester, demonstrated the progress achieved at the time of course completion. Four instructors involved in the evaluation reported strong familiarity of students with basic software engineering principles, presentation and ICT skills, understanding of complex programs and improved group-level engagement.

The visualisation tool (UUhistle) was not mentioned in the reflective journals. Course instructors also observed gaps in supporting less-engaged students, and in the timeliness and quality of assignment evaluation. There was also no strong evidence on the benefit of female-only pairing strategy in improving their engagement. Female students reported that their pair practice was affected by lack of access to computer and unsuitable computer laboratories (e.g. other students taking their reserved computers and feeling unsecure in overcrowded laboratories). Shortcomings observed in the summative project were existence of dependency problem (plagiarism both from Internet sources as well as from their classmates) and unbalanced team-level participation, though both were at a reduced scale compared to previous assignments.

7. Discussion

The first cycle action research evaluated the effectiveness of the blended course and learning environment designed through a DSR process. The design integrated constructivist learning theories and instructional models, information technologies and large class teaching strategies. The action research examined the transformative role of blended learning in the context of the challenges facing large class teaching of programming.

Table 8 summarises the research and its findings that show mixed results on the impact of the intervention. We discuss the contrasting findings with a focus on two related areas: (1) determining the effectiveness of the intervention and (2) identifying practice improvement and design refinements necessary for the next cycle of action research.

7.1. Intervention effectiveness

Learning effectiveness is one of the core criteria for measuring the success of blended learning intervention (Lorenzo & Moore, 2002).

The findings from the achievement tests (descriptive statistics, inter-assessment correlation and comparative performance analysis) produced mixed results. The class-level mean score of the course is above the pass mark of 50% and the weak but positive correlation among assessment activities suggest that the intervention had a positive impact on learning. The R-SPQ-2F survey results, namely a slight move towards deep learning and a minor decline in students' SA to learning, provide further quantitative evidence for the positive impact on learning effectiveness. The performance of

Table 8. Summary of the research findings.

Research problem	Intervention and implementation	Embedded research	Findings
T&L of programming in large classes addressing difficulty in novice programming, and concerns in large class teaching: student anonymity, low student–instructor interactions, class management and decline in assessment quality	Pedagogically underpinned blended learning that integrate large class and programming teaching best practices: • Constructively aligned course design • Learning environment design that integrates technologies (e-learning and program visualization), and large class strategies (team teaching, students mentoring, holistic assessment and collaboration) using conversational framework and the three-stage learning model as integration framework	Measuring intervention effectiveness through: • SAL survey with R-SPQ-2F • achievement test (correlation analysis of students' in assessment results; comparing performance based on selected common examination questions) • grounded theory analysis of qualitative data	• Positive but small move towards deep learning and slim decline in surface learning in a before/after R-SPQ-2F survey • Satisfactory achievement in assessment results (mean score above passing cut-points of 50% of allocated weight). Marks in learner-centred activities were more right-skewed and concentrated with small inter-quartile ranges. Higher deviation in examination results (bigger inter-quartile range, longer whiskers in Figure 3 and larger standard deviation) • Positive but weak correlation among assessments, mainly amongst the learner-centred assignments and projects • Fair correlation between deep learning and examination marks

Research problem	Intervention and implementation	Embedded research	Findings
			• Modestly comparable performance between our students and students from other universities with the limitations described in Section 5.2. As in the other university, our students' mark in tracing and *explain in plain English* question was very low • Improving persistence of students' engagement in programming with declining but observable large class and programming challenges – disengagement of some students, low-level use of visualisation tool, problem in quality and timeliness of evaluation and feedback

our students compared to that of students from other international institutions is promising when taking the context into account.

The qualitative data analysis revealed progressive improvement in engagement and level of confidence of our students in their programming ability, and integration of software engineering principles. The impact of the integrated approach to learn programming, i.e. combination of interactive lecture, use of technology, student mentors, pair-programming and instructor support, was evident from the students' reflective journals.

The positive quantitative and qualitative findings of this study offer a response to the research question by confirming that this design can be used to improve large class T&L of programming.

Design and implementation shortcomings are evident in the low correlation between the students' scores of the different assessments, and in their poor performance in the tracing and *explain in plain English* questions. The evaluative role of the learner-centred assessments was also comparatively poor, when compared to the higher discriminatory role of the final examination. These shortcomings inform design refinement as discussed in the next section.

7.2. Design refinement guidelines

The findings have motivated changes to the intervention design. The major changes necessary are:

(1) *Enhancing technology integration:* the impact of visualisation tools in the course was not significant. Further improvement is needed to better integrate visualisation and programming tools with assessment activities.
(2) *Improving learner-centred assessments:* immediate feedback on assessment promotes learning (Taras, 2005). The second action research needs to reduce delays in feedback time and to improve quality of assessment, which demands more facilitation and assessment role of instructors.
(3) *Engaging disengaged and less-engaged students:* providing personalised student support (Klem & Connell, 2004) and increase the evaluation of individual contributions in team-based assignments and projects (Hayes, Lethbridge, & Port, 2003).

8. Conclusion and future work

In this article, we have presented a course and learning environment design that addresses the problems of large class teaching, which is a common practice in many higher education institutions in Sub-Saharan Africa. Our study describes a technology-enhanced, large class education solution for

the widely recognised problems of lack of skilled instructors, and resource constraints associated with higher education in the region. The course and blended learning environment design was informed by theories and practices from education, blended learning and computer science education. The focus was on the problematic area of large class teaching of programming.

The course and learning environment design and the action research evaluation in a semester-long implementation has produced the following positive results: some changes in SAL towards deep learning, modestly comparative student scores with students from foreign universities, and good-quality group projects as judged by instructors and confirmatory reflection of the students.

The contribution was to develop and evaluate a contextualised lesson on the design of a blended learning solution for large class teaching of programming grounded in constructivist learning theory and use of free and open source technologies. We have extended the experience of Marsh, McFadden, and Price (2003) and Francis (2012) in application of blended learning for large class teaching by empirically testing them in programming education. The experience of Thota and Whitfield (2010), Hadjerrouit (2005, 2008) and Brabrand (2008) was also expanded by incorporating components to make them applicable to large class teaching. The lesson includes application of educational models (constructive alignment, three-stage learning model and conversational framework) that integrate best practices in large class and programming teaching such as pair programming and holistic and aligned assessments.

The first cycle of action research raised some issues to be addressed in the next cycle and future research. The fair correlation between the final code-writing project (done with oral presentation and question answering) and the examination points to an alternative strategy for improved quality of assignment evaluation. Shortcomings to be addressed in future work include support for unengaged students, poor integration of visualisation tools and low correlation among assignment results. Hence, design improvement required in the next round include better alignment of technologies with T&L and assessment activities, diversification of evaluation strategy for immediate feedback and enhancement of support for struggling students.

Lorenzo and Moore (2002) proposed learning effectiveness, student satisfaction, faculty satisfaction, cost-effectiveness and access as five criteria for impact assessment of blended learning. Our study in the first cycle considered learning effectiveness only. The next cycle will focus on student satisfaction and analysis of the learning effectiveness from students' perspective. Future research in the area can replicate the design and the design guidelines generated from our study applying one or more of the Lorenzo and Moore's measures.

Acknowledgement

This work was partially supported by the NORAD project of Hawassa University from the third phase of a Norwegian Government-supported project.

References

Apple, D. K., & Nelson, D. (2002). Identification of non-success factors in a large introductory computer science course and constructive interventions for increasing student success. In *32nd Annual Frontiers in Education, 2002. FIE 2002*. (Vol. 1, pp. T1G-1). Boston, MA: IEEE.

Ashcroft, K., & Rayner, P. (2012). The purposes and practices of quality assurance in ethiopian higher education: Journey, adaptation and integration. *International Journal of Business Anthropology, 3*, 19–35.

Azemi, A., & D'Imperio, N. (2011). New approach to teaching an introductory computer science course. In *Frontiers in Education Conference (FIE), 2011* (pp. S2G-1). Rapid City, SD: IEEE.

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). Koli: 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.

Bass, J. M. (2009). Empathetic consultancy: A reflective approach to ICTD. In *Proceedings of the 10th International Conference on Social Implications of Computers in Developing Countries*. Dubai: Dubai School of Government.

Beck, R., Weber, S., & Gregory, R. W. (2013). Theory-generating design science research. *Information Systems Frontiers, 15*, 637–651.

Ben-Ari, M. (2001). Constructivism in computer science education. *Journal of Computers in Mathematics and Science, 20*, 45–73.

Biggs, J. (2003). *Aligning teaching for constructing learning*. Higher Education Academy. Retrieved from http://www.heacademy.ac.uk/resources/detail/resource_database/id477_aligning_teaching_for_constructing_learning

Biggs, J., Kember, D., & Leung, D. Y. (2001). The revised two-factor study process questionnaire: R-SPQ-2F. *British Journal of Educational Psychology, 71*, 133–149.

Biggs, J., & Tang, C. (2007). *Teaching for quality learning at university: What the students does* (3rd ed.). Maidenhead: Open University Press.

Black, P., & Wiliam, D. (1998). Assessment and classroom learning. *Assessment in Education: Principles, Policy & Practice, 5*, 7–74.

Bledsoe, K. E. (2011). Managing problem-based learning in large lecture sections. *Bioscience Education, 18*. doi:10.3108/beej.18.1

Brabrand, C. (2008). Constructive alignment for teaching model-based design for concurrency. In *Transactions on petri nets and other models of concurrency I* (pp. 1–18). Heidelberg: Springer Berlin.

Bryson, C., & Hand, L. (2007). The role of engagement in inspiring teaching and learning. *Innovations in Education and Teaching International, 44*, 349–362.

Clear, T., Whalley, J., Robbins, P., Philpott, A., Eckerdal, A., & Laakso, M. (2011). Report on the final BRACElet workshop. *Journal of Applied Computing and Information Technology, 15*(1). Retrieved March 9, 2014, from http://www.citrenz.ac.nz/jacit/JACIT1501/2011Clear_BRACElet.html

Curricula, C. (2001). *Computer science*. Final Report. The joint task force on computing curricula. IEEE Computer Society and Association for Computing Machinery.

Curtin University. (n.d.). *Structured reflection*. Retrieved September 30, 2011, from http://learningcentre.curtin.edu.au/skills/structured_reflection.cfm

Cuseo, J. (2007). The empirical case against large class size: Adverse effects on the teaching, learning, and retention of first-year students. *The Journal of Faculty Development, 21*, 5–21.

Decker, A., Ventura, P., & Egert, C. (2006). Through the looking glass: Reflections on using undergraduate teaching assistants in CS1. *ACM SIGCSE Bulletin, 38*, 46–50.
Dougiamas, M., & Taylor, P. C. (2002). *Interpretive analysis of an internet-based course constructed using a new courseware tool called Moodle*. Paper presented at the 2002 International Conference of the Higher Education Research and Development Society of Australasia (HERDSA). Perth: HERDSA.
Elo, S., & Kyngäs, H. (2008). The qualitative content analysis process. *Journal of Advanced Nursing, 62*, 107–115.
Fincher, S., Lister, R., Clear, T., Robins, A., Tenenberg, J., & Petre, M. (2005). Multi-institutional, multi-national studies in CSEd research: Some design considerations and trade-offs. In *Proceedings of the First International Workshop on Computing Education Research* (pp. 111–121). Seattle, WA: ACM.
Francis, R. W. F. (2012). Engaged: Making large classes feel small through blended learning instructional strategies that promote increased student performance. *Journal of College Teaching & Learning (TLC), 9*, 147–152.
Fuller, U., Riedesel, C. G., Thompson, E., Johnson, C. G., Ahoniemi, T., Cukierman, D., … Hernán-Losada, I. (2007). Developing a computer science-specific learning taxonomy. *ACM SIGCSE Bulletin, 39*, 152–170.
Ginns, P., & Ellis, R. (2007). Quality in blended learning: Exploring the relationships between on-line and face-to-face teaching and learning. *The Internet and Higher Education, 10*, 53–64.
Glaser, B. G., & Strauss, A. L. (1967). *The discovery of grounded theory: Strategies for qualitative research*. New Brunswick: Aldine Transaction.
Hadjerrouit, S. (2005). Learner-centered web-based instruction in software engineering. *IEEE Transactions on Education, 48*, 99–104.
Hadjerrouit, H. (2008). Towards a blended learning model for teaching and learning computer programming: A case study. *Informatics in Education, 7*, 181–210.
Hamm, S., & Robertson, I. (2010). Preferences for deep-surface learning: A vocational education case study using a multimedia assessment activity. *Australasian Journal of Educational Technology, 26*, 951–965.
Hanusch, F., Obijiofor, L., & Volcic, Z. (2009). Theoretical and practical issues in team-teaching a large undergraduate class. *International Journal of Teaching and Learning in Higher Education, 21*, 66–74.
Hayes, J. H., Lethbridge, T. C., & Port, D. (2003). Evaluating individual contribution toward group software engineering projects. In *Proceedings 25th International Conference on Software Engineering* (pp. 622–627). Portland, OR: IEEE.
Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly, 28*, 75–105.
Hoic-Bozic, N., Mornar, V., & Boticki, I. (2009). A blended learning approach to course design and implementation. *IEEE Transactions on Education, 52*, 19–30.
Huang, R. (2012). *RQDA: R-based qualitative data analysis*. R package version 0.2-3. Retrieved from http://rqda.r-forge.r-project.org/
Hwang, W. Y., Shadiev, R., Wang, C. Y., & Huang, Z. H. (2012). A pilot study of cooperative programming learning behaviour and its relationship with students' learning performance. *Computers & Education, 58*, 1267–1281.
Ivankova, N. V., Creswell, J. W., & Stick, S. L. (2006). Using mixed-methods sequential explanatory design: From theory to practice. *Field Methods, 18*, 3–20.
Ives, S. M. (2000). *A survival handbook for teaching large classes*. University of North Carolina. Retrieved from http://teaching.uncc.edu/learning-resources/articles-books/best-practice/large-classes/large-class-handbook
Kerr, A. (2011). *Teaching and learning in large class at Ontario Universities: An exploratory study*. Toronto: Higher Education Quality Council of Ontario.
Klem, A. M., & Connell, J. P. (2004). Relationships matter: Linking teacher support to student engagement and achievement. *Journal of School Health, 74*, 262–273.
Laurillard, D. (2002). *Rethinking university teaching* (2nd ed.). London: Routledge.
Laurillard, D. (2012). *Teaching as a design science: Building pedagogical patterns for learning and technology*. New York, NY: Routledge, Taylor & Francis Group.

Lee-Partridge, J. (2006). Using reflective learning in an introductory programming. In *Emerging trends and challenges in information technology management: 2006 Information Resources Management Association International Conference* (Vol. 1), Washington, DC, USA, May 21–24, IGI Global.
Lopez, M., Whalley, J., Robbins, P., & Lister, R. (2008). Relationships between reading, tracing and writing skills in introductory programming. In *Proceedings of the Fourth International Workshop on Computing Education Research* (pp. 101–112). New York, NY: ACM.
Lord, F. M. (1952). The relation of the reliability of multiple-choice tests to the distribution of item difficulties. *Psychometrika, 17*, 181–194.
Lorenzo, G., & Moore, J. (2002). *Five pillars of quality online education: The sloan consortium report to the nation*. Retrieved from http://sloanconsortium.org/publications/books/pillarreport1.pdf
Marsh, G. E., McFadden, A. C., & Price, B. J. (2003). Blended instruction: Adapting conventional instruction for large classes. *Online Journal of Distance Learning Administration, 6*(4). Retrieved from http://www.westga.edu/~distance/ojdla/winter64/marsh64.htm
Marton, F., Hounsell, D., & Entwistle, N. (1997). *The experience of learning*. Edinburgh: Scottish Academic Press.
Marton, F., & Säljö, R. (1976). On qualitative differences in learning: I-outcome and process. *British Journal of Educational Psychology, 46*, 4–11.
Mayes, J. T., & Fowler, C. J. (1999). Learning technology and usability: A framework for understanding courseware. *Interacting with Computers, 11*, 485–497.
McCauley, R., & Mcgettrick, A. (2008). *Computer science curriculum 2008: An interim revision of the CS 2001, a report from the interim review task force*. ACM.
McCracken, M., Wilusz, T., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., … Kolikant, Y. B.-D. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *ACM SIGCSE Bulletin, 33*, 125–180.
Melrose, M. J. (2001). Maximizing the rigor of action research: Why would you want to? How could you? *Field Methods, 13*, 160–180.
Melrose, M., & Reid, M. (2000). The daisy model for collaborative action research: Application to educational practice. *Educational Action Research, 8*, 151–165.
Mendes, A. J., Paquete, L., Cardoso, A., & Gomes, A. (2012). Increasing student commitment in introductory programming learning. In *2012 Frontiers in Education Conference Proceedings* (pp. 1–6), October 3–6. Seattle, WA: IEEE.
Newton, P., & Burgess, D. (2008). Exploring type of education action research: Implications for research validity. *International Journal of Qualitative Methods, 7*, 18–30.
Pears, A. N. (2010). Enhancing student engagement in an introductory programming course. In *Frontiers in Education Conference (FIE) 2010 IEEE* (pp. F1E-1). Washington, DC: IEEE.
PSPP. (n.d.). *PSPP statistical package*. Retrieved from http://www.gnu.org/software/pspp/
Rajala, T., Laakso, M. J., Kaila, E., & Salakoski, T. (2008). Effectiveness of program visualisation: A case study with the ViLLE tool. *Journal of Information Technology Education, 7*, 15–32.
Rajalingam, S., & Oo, Z. (2011). *Finding the correlation between formative and summative assessments by Spearman's correlation coefficient: A case study*. Retrieved from http://espace.library.curtin.edu.au/R?func=dbin-jump-full&local_base=gen01-era02&object_id=174728
Ramsden, P. (2003). *Learning to teach in higher education* (2nd ed.). London: Routledge Falmer.
Roberts, G. (2003). Teaching using the web: Conceptions and approaches from a phenomenographic perspective. *Instructional Science, 31*, 127–150.
Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education, 13*, 137–172.
Sein, M., Henfridsson, O., Purao, S., Rossi, M., & Lindgren, R. (2011). Action design research. *MIS Quarterly, 13*, 37–56.

Sheth, S., Bell, J., & Kaiser, G. (2013). A competitive-collaborative approach for introducing software engineering in a CS2 class. In *2013 IEEE 26th Conference on Software Engineering Education and Training (CSEE&T)* (pp. 41–50). San Francisco, CA: IEEE.

Shuhidan, S., Hamilton, M., & D'Souza, D. (2009). A taxonomic study of novice programming summative assessment. In *Proceedings of the Eleventh Australasian Conference on Computing Education-Volume 95* (pp. 147–156). Darlinghurst: Australian Computer Society.

Sorva, J. (2012). *Visual program simulation in introductory programming education* (Doctoral dissertation 61/2012). Aalto University. Retrieved from http://urn.fi/URN:ISBN:978-952-60-4626-6

Srivastava, P., & Hopwood, N. (2009). A practical iterative framework for qualitative data analysis. *International Journal of Qualitative Methods, 8*, 78–84.

Sweigart, A. (2009). *Invent your own computer games with python*. CreateSpace. Retrieved from http://inventwithpython.com/

Taras, M. (2005). Assessment – Summative and formative – Some theoretical reflections. *British Journal of Educational Studies, 53*, 466–478.

Tessema, K. A. (2009). The unfolding trends and consequences of expanding higher education in Ethiopia: Massive universities, massive challenges. *Higher Education Quarterly, 63*, 29–45.

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). Darlinghurst: Australian Computer Society.

Thota, N., & Whitfield. (2010). Holistic approach to learning and teaching introductory object-oriented programming. *Computer Science Education, 20*, 103–127.

UNESCO. (2010). *Trends in tertiary education: Sub-Saharan Africa*. UIS Fact Sheet, December 2010, No. 10. Retrieved from http://www.uis.unesco.org/FactSheets/Documents/fs10-2010-en.pdf

Venable, J. (2006). A framework for design science research activities. In *Emerging trends and challenges in information technology management: 2006 Information Resources Management Association International Conference,* Information Resources Management Association, IGI Global, Washington, DC, USA, May 21–24, 2006 (Vols. 1 and 2, pp. 184–187).

Ward, A., & Jenkins, A. (1992). The problems of learning and teaching in large classes. In G. Gibbs & A. Jenkins (Eds.), *Teaching large classes in higher education: How to maintain quality with reduced resources*. London: Psychology Press.

Webber, K. L., & Tschepikow, K. (2013). The role of learner-centred assessment in postsecondary organisational change. *Assessment in Education: Principles, Policy & Practice, 20*, 187–204.

Wulf, T. (2005). Constructivist approaches for teaching computer programming. In *Proceedings of the 6th Conference on Information Technology Education* (pp. 245–248). New York, NY: ACM.

Yizengaw, T. (2008). *Challenges of higher education in Africa and lessons of experience for the Africa-US higher education collaboration initiative: A synthesis report for the Africa-U.S. higher education initiative*. Retrieved July 21, 2013, from http://www.aplu.org/NetCommunity/Document.Doc?id=1183