Conceptualisation of a learning environment for programming through an analysis of the underlying research issues in teaching programming

Ioana Tuugalei Chan Mow
Computing department
National University of Samoa
Apia, Samoa
i.chanmow@nus.edu.ws

Wing Au; Gregory Yates
School of Education
University of South Australia
Adelaide, Australia
wing.au@unisa.edu.au; gregory.yates@unisa.edu.au

Abstract- **This paper provides insights into the underlying research on issues in teaching introductory programming at university which gave rise to the conceptualisation of the CABLE model – a learning environment for teaching computer programming trialed at the National University of Samoa over a period of 3 years. The paper describes why students find programming difficult. From analysis of the research, potential solutions are proposed. These solutions form the basis of recommendations for the conceptualization and establishment of a model of a learning environment called CABLE. Findings from the analyses of research on issues in teaching programming are also used as recommendations on methodology and implementation details of the proposed pedagogical model.**

Keywords-programming; computer programming; CABLE; collaborative learning; cognitive apprenticeship; modelling; metacognition; computer mediated communication

I. INTRODUCTION (HEADING 1)

"Computer science educators have shown growing concern over the difficulties with which novice Computer programmers learn programming principles. Computer programming is a challenging subject area which places a heavy cognitive load on students [1 - 3]. Most novice programmers have had little or no previous experience in programming and takes on average 10 years for a novice to be proficient in programming [4]. This paper describes why students find programming challenging and then recommends potential solutions from which the proposed pedagogical model CABLE is conceptualized. From these recommendations the components of CABLE are then proposed and put together to formulate the CABLE learning environment. This learning environment was then trialed over a period of 3 years in programming courses at the National University of Samoa.

II. ISSUES AND DIFFICULTIES IN LEARNING PROGRAMMING

A. *Cognitive requirements of programming*

Programming requires students to hold a wide range of information in working memory. These include the details of syntax and semantics specific to the programming language being used, some mental model of how to solve each problem, and the ability to differentiate between solving the problem and specifying the solution [5]. Computer programming also requires that the user be proficient in the use of (a) the development environment, (b) the programming language, and (c) compiler/interpreter, which are separate levels of the programming interface that the user must master in order to program successfully [5] ;[6]. Consequently, these demands would impose a heavy cognitive load on the student, making the learning of programming a complex and cognitively challenging task.

B. *Precision*

Another factor which contributes to the difficulty in learning programming is precision [7]. In programming, every eventuality and possibility must be taken into account and catered for. Such a level of precision is likely to be challenging for the novice programmer, and extremely demanding for those who are not accustomed to it.

C. *Mental Models*

The task of programming involves the construction of several mental models [8]. Firstly, computer programs are usually written as a solution to some problem, and it is obvious that an understanding or mental model of this problem domain needs to be established before there is any attempt to write an appropriate program [9]. The programmer must also develop a mental model of the program itself and how it will be executed [2];[5]; [9]. Learners are successful if they can construct viable models that match the design models of computer programming elements.

3rd Annual International Conference on Computer Science Education: Innovation & Technology (CSEIT 2012)

ISSN: 2251-2195
doi: 10.5176/2251-2195_CSEIT12.42

D. *Lack of Direct Manipulation*

A programming language differs from a typical user interface in that it is less immediate and more complex. Most programming languages require learners to carry out a number of actions in editing, compilation, and execution before a feedback is obtained. In most programming languages (with the exception of visual languages), only some statements in a programming language have clear and immediate feedback (mostly output statements), and the feedback is only immediate if the language is interactive. The actual programming model is not continuously presented to the learner at the interface and the learning process is interrupted. Furthermore, the situation in which the program is to be executed may not be available for inspection, because it may be in the future, or because the program may be applied to a greater range of data situations than are currently visible to the programmer. This lack of direct manipulation in programming systems has been identified as one of the factors contributing to the difficulty of programming, and a source of frustration to novice programmers [7]; [8].

E. *Hard Mental Operations*

The complexity of programming is a contributing factor to its challenging nature and "hard mental operations" refers to concepts which are complex and cognitively challenging [5]. In computer programming, these complex operations are almost unavoidable as there are many concepts in programming which are difficult for students to master, and which require them to think in unfamiliar and complex ways. Examples of hard mental operations for novice programmers include iterations, recursion and optimization problems. Formulation of effective instructional strategies such as scaffolding, coaching, and modeling [9]; [10] is needed to help students master such complex and mentally challenging operations.

III. THE CASE FOR A PEDAGOGICAL MODEL

Because of the complexity and abstract nature of computer programming, good instructional methods in teaching computer programming, is vital. This need for better instructional methods is also reinforced and reiterated by [11], [12] and [13]. Support for the need for focused research in the area of instructional approaches, is also evident in the final report of the Joint Task Force on Computing Curricula, IEEE Computer Society, Association for Computing Machinery, Computing Curricula,[14].Specifically,[14], highlighted the need for "a range of strategies that have been validated by practice", the need for "pedagogical innovation for continued success" and also encouragement for "continued experimentation" in this area.

From our current review of the literature, potential solutions have emerged which can be used in the construction of a pedagogical model and also provide justification of the need for such an innovation. These potential solutions and recommendations are listed below.

IV. POTENTIAL SOLUTIONS AND RECOMMENDATIONS FOR THE FORMULATION OF A PEDAGOGICAL MODEL

A. *Cognitive Apprenticeship*

The literature has established the cognitively demanding nature of computer programming [3]; [5]; [7], and with it the need for instructional strategies to accommodate or alleviate the heavy cognitive load. Studies have suggested cognitive apprenticeship and collaborative learning, as best practices for improving higher order learning and critical thinking skills [15- 16]. Cognitive apprenticeship is a model of instruction that involves the effective communication of domain knowledge in such a way that the students become aware of the thought processes involved in knowledge construction within that domain [17]. Cognitive apprenticeship is directed at teaching processes that experts use to handle complex tasks, and is characterized by a number of teaching methods [18]. The first of these, is modeling where the teacher models his or her thought processes in solving problems within a domain [19]. The second of these methods is guided practice or coaching where the student attempts to solve the problems for themselves with the support of the teacher to answer specific queries. A third method is scaffolding, where the teacher assists students to manage complex task performance and then gradually withdraws support from the student (fading). Other key components of this approach are articulation, where the student attempts to articulate their problem solving strategies; reflection, where the students are encouraged to reflect on how they approached tasks and solved problems, possibly by discussion with other students and, finally, exploration which is intended to encourage learner autonomy and problem formulation by students.

B. *Collaborative Learning*

Another instructional strategy that is gaining prominence as an effective teaching method is collaborative learning. There are many approaches to collaborative learning but all have the following characteristics in common [7]. It is a learning activity suitable for group work; it is small group based (usually 2-5); it has tasks which encourage cooperative behavior; it is characterized by student interdependence; individual student accountability and responsibility for task completion. The need for group support for students learning programming is well documented [20]. This is important for two reasons. Firstly, studies have shown the benefits of collaborative learning on learning computer programming [20]. Secondly, most programming in real world situations is done in teams and hence identifies the need for exposure of students to team work. The benefits of group work is also supported by studies on "pair programming1" [21];[22] and also the use of computer supported collaborative environments for programming [23].

[1] Writing the source code of a program in teams of two. Also called "peer programming,"

C. *Suitability of Cognitive Apprenticeship and Collaborative Learning as primary components of the Pedagogical Model*

The suitability of cognitive apprenticeship and collaborative learning as instructional approaches for teaching programming is premised on several factors. Firstly, these pedagogical practices are based on sound educational theories [16 - 17]. Underlying both collaborative learning and cognitive apprenticeship is Constructivism theory, Problem based learning and Vygotsky's social constructivism and the concept of guidance and collaboration in the zone of proximal development (ZPD). Secondly, both are supposed to stimulate cognitive and metacognitive processes in student learning. Thirdly, these practices allegedly promote social interactions which is claimed to foster the learning process [12];[19]. Finally, both practices are consistent with the Vygotskian notion of scaffolding, which is seen as vital in teaching of cognitively challenging tasks [10];[12].

From an analysis of research on areas of difficulty in the teaching of programming recommendations on instructional strategies which could overcome these challenges include teacher mediation, the use of scaffolding, coaching, articulation, modeling, visualization techniques, the use of the debugger facility, advanced organizers, multiple and worked examples. These recommendations all point to the suitability of cognitive apprenticeship and collaborative learning as components of the proposed pedagogical model.

D. *Effective Teacher Mediation*

Effective teacher mediation is necessary for improved programming ability [24 - 25]. Effective teacher intervention takes the form of: (a) selecting and creating tasks designed to achieve educational goals; (b) focusing students' attention on particular aspects of their experience; (c) providing formal mathematical language for the mathematical concepts; (d) emphasizing planning for algorithm development, (e) suggesting paths to pursue; (f) providing metacognitive prompts and asking higher-order questions; (g) facilitating disequilibrium using computer feedback as a catalyst; (h) providing tailored feedback regarding students' problem-solving efforts; (i) discussing errors and common misunderstandings; (j) continually connecting the ideas developed to those embedded in other contexts; (k) providing modeling and coaching; and (l) promote both student–teacher and student–student interaction.

E. *Visualisation Techniques*

Programming involves the loss of direct manipulation of data and subsequently the lack of immediate feedback [5]. These two consequences indicate a need for learner support in the form of visualization tools or techniques [26 - 27]. An example of visualization tools is the use of the debugger features and the variable stack feature of JBuilder, the integrated developer environment (IDE) used in programming courses at NUS. Visualization techniques can also be provided in the form of coaching, modeling and scaffolding processes of the cognitive apprenticeship approach. Hence, the visualization of program execution via the step through features of the JBuilder debugger, the visualization of the variable stack are all useful features of the JBuilder environment which provide learner support.

F. *Strategies for Building Mental Models*

The need for students to build good mental models of both the problem and its solution [5] has been identified as crucial in facilitating the transition from a novice to an expert. Again, this points to the suitability of cognitive apprenticeship as the idea of the expert modeling concepts to the novice and the use of coaching, scaffolding, articulation, feedback and reflection, are all conducive to building of good mental models in a novice or apprentice. Other recommended strategies include the use of advanced organizers, visualization techniques, multiple and worked examples

G. *Technological Solutions*

The use of the computer as a communications tool in the form of computer-mediated communications (CMC), such as e-mail and websites has revolutionized the delivery of instruction and provides an effective learning environment [28]. The three main advantages to online learning are that students (a) can choose their own pace of study, (b) are able to organize their own schedule and not be tied to time and place, and (c) can choose a pace of study independent of others. Advantages of computer mediated communication (CMC) in e-learning are as follows. Firstly with CMC learners acquire knowledge within the context in which it is used. Secondly the use of CMC eliminates the problem of transfer of knowledge from the context of learning to the context of practice – situated learning. The third advantage is the motivation provided by the presence of a diverse real audience instead of just the teacher as in a conventional classroom. Yet another advantage of CMC is the provision for a more diverse range of people to interact.

For the proposed model, CMC techniques such as e-mail, discussion forum, online notes, interactive quizzes and bulletin boards can be used to electronically implement key aspects of cognitive apprenticeship such as scaffolding, coaching, modeling, and reflection. The use of a hybrid or blended model is ideal as it uses online techniques to supplement face-to-face interactions [28]. The hybrid model is preferred over the online model as it is then possible to maintain face-to-face communications but at the same time utilize online techniques to provide learning independent of time and place [28]. Since the intention is to integrate online learning techniques as part of the proposed instructional model, factors which may affect its effectiveness are (a) the cost and access to the technology, (b) the reliability of the technology, and (c) prerequisite skills, such as ability to use the online software and good skills in reading and writing.

V. Definition of the Pedagogical Model "CABLE"

The main argument of this paper is that effective instruction is crucial for improving student performance in computer programming. It is therefore appropriate at this stage to define the CABLE model and its components. Although it is based on the cognitive apprenticeship approach, the CABLE model is further enhanced by incorporating the principles and practice of collaborative learning. Furthermore elements of this cognitive apprenticeship-based approach will be implemented electronically by means of e-mail, bulletin board, online notes and worked examples. This is referred to as tele-apprenticeship. Hence CABLE is a hybrid model which uses both face-to-face and online mode of delivery. The components of CABLE are as follows:

- Cognitive apprenticeship (Modeling, Coaching, Scaffolding, Articulation, Reflection)
- Computer-mediated communication(E-mail, Bulletin board, Online resources)
- Collaborative learning

A theoretical model for teaching programming has now been established by combining a selection of best practices from the literature. It is important to realize however, that this is a theoretical model and the next step in the process would be to define the implementation details or how this theoretical model can be put into practice. The following section will describe each of the components of CABLE and how these components will be implemented.

VI. Implementation Details of CABLE

A. Modelling

One of the main distinguishing features of cognitive apprenticeship is modeling. Within CABLE, the teacher being the expert will provide modeling through a variety of techniques. These include: (a) demonstrating object-oriented programming concepts and skills, (b) creating programs with classes and objects using certain problem solving heuristics, and (c) demonstrating how to model JAVA applications by stepping the students through the component processes. In practical terms, modeling will be achieved by demonstrating concepts on the blackboard, and by live demonstrations where the lecturer uses a computer to create a program within the integrated developer environment (IDE) while the students observe the process through a data projector display.

B. Coaching

A second component of the CABLE approach is coaching. This would be achieved in several ways: by the lecturer giving expert coaching in class, by means of expert help via e-mail, and also peer coaching from other students as they collaborate in certain programming activities. Coaching would also be facilitated by means of interactive online tests [29].Students could test their level of knowledge and skills by taking these tests and by clicking on a button, the test would be graded and instant feedback of their test score will be returned.

C. Use of the Debugger

Another feature which facilitates coaching and modeling is the use of the debugger utility of JBuilder, the integrated developer environment for creating Java programs. The coaching or modeling process is made more transparent by the use of a debugger within the JAVA editor JBuilder. The debugger allows students to step through any JAVA program and allows the student to see the sequence of execution of the JAVA program. More importantly it facilitates tracing of the values of the attributes in a program as the program executes. Hence, the debugger not only provides valuable feedback to the student in the location and causes of program errors or bugs, but also provides the learner with a visualization of the process of execution of a computer program [6]; [30].

D. Contextualised Learning

Yet another feature of the cognitive apprenticeship approach is situating abstract tasks in authentic contexts. In the proposed research, JAVA programming will be taught within the context of the systems development life cycle so that students can see the process steps as integrated in a larger context but can at the same time still focus on the individual activity. Hence students would be given a problem in the form of a requirements document and they would provide the solution by proceeding through the various phases of the systems development life cycle just as they would in a real life situation or the workplace. Situated learning would also be facilitated by means of collaboration in pairs for modeling activities as in real life activities of software development are actually carried out by a team of developers. To increase the potential for transfer of problem solving skills across a diversity of situations, students need to see and identify similarities between problem contexts and the application of approaches. Within the proposed model, this will be achieved by the lecturer modeling the heuristics of instantiating objects across a variety of situations and by emphasizing the similarities between problem contexts and the application of common approaches.

E. Articulation and Think Alouds

Another important feature of cognitive apprenticeship is identifying the processes of the task and making them visible to the students. In CABLE this would be achieved by the lecturer modeling his thought processes by thinking out aloud. The students would also be encouraged to think out loud as they step through such processes as instantiating objects. This will also be facilitated by means of posting online notes and online sample solutions on the class website. [31] maintain that familiarity with the language of a discipline and academic genre is an important and defining factor in students' ability to read and write appropriately within the discipline. Research conducted by [32] on effective instructional approaches in computer programming have shown that modeling and articulation of the expert's thinking processes and the expert's

use of language is very important and conducive to learning computer programming. Hence, the articulation process is very important in that it forces the issue of the correct use of expert language or programming language by the students. The lecturer would gradually withdraw the guidance (fading) when students demonstrate they could now step through the process with confidence. The learners would then be given more complex tasks.

F. *Individualised Feedback*

In the CABLE approach, feedback is structured and is given on a weekly basis. Feedback will be provided by an online system where the lecturer provides individualized feedback via individualized e-mails sent to and received from each student. On a weekly basis, the students are expected to send an e-mail to the lecturer, which answers several questions. The first question requires them to describe what activities or topics they had done during the week and to indicate how they felt about their progress. The second and third question requires them to describe any areas they are having problems with, asking any questions they needed answers to. The last question requires the student to reflect upon what they have learnt and how useful they thought what they had learnt would be to them. The lecturer would then respond to each student via e-mail and it was hoped that this communication would not only provide remediation, but also encourage students to reflect on their work. From the individualized feedback, the lecturer could gauge areas most students were having problems with and use it to post some frequently asked questions (FAQs) and their solutions on the class web-site, providing further feedback and guidance to students in the class.

G. *Metacognition*

A second differentiating factor between CABLE and the traditional approach is the cultivation of metacognition. This can be facilitated by encouraging students to reflect on their progress, problems encountered, what they had learnt, the usefulness of what they had learnt and also by the articulation of their thinking processes in the form of "think-alouds".

H. *Collaborative Learning*

The third differentiating factor between the CABLE and the traditional model is the incorporation of elements of collaborative learning. With CABLE, coaching and mediation would also be provided by a more capable peer as the students are paired, with the more capable student collaborating with the weaker student in carrying out their programming tasks in class. The value and effectiveness of collaborative learning, in fostering learning, is well documented in numerous studies [33].

In addition to all the recommended components of the CABLE model, the course reader was also revised to provide additional assistance for the students. The notes were structured so that Java programming was situated as part of the systems development life cycle for the development of systems ("programming in the large"). Secondly, the notes and exercises had been structured to encourage the articulation of steps in the Java activities such as those of instantiating an object. This was to encourage students to use some problem-solving heuristic to arrive at a solution. This was also aimed at facilitating the students learning of the syntax and semantics of Java.

VII. IMPLEMENTATION

This paper has described how from analysis of the research on programming, potential solutions were proposed which form the basis of recommendations for the conceptualization and establishment of a model of an effective learning environment called CABLE. Findings from the analyses of research on issues in teaching programming were also used as recommendations on methodology and implementation details of the proposed pedagogical model.

In this paper, the transition from theoretical model to practical approach is established with the definition of the implementation details of the CABLE approach. The CABLE environment was evaluated in a series of field trials (Projects 1, 2, and 3).). In Project 1 and Project 2, the CABLE approach was evaluated by contrasting it with the traditional didactic approach to university instruction. The main aim of Project 3, however, was to evaluate the effectiveness of the learning environment, based on self reporting by students on their levels of engagement.

Students in both the CABLE and traditional groups were given the same set of instructional materials in their JAVA training, and participated in similar lectures and practicum classes. The main differences between the two approaches were: (a) the use of structured and individualized feedback in CABLE by means of an email help desk, (b) the provision of a rich meta-cognitive experience through articulation and the use of "think alouds", feedback from the lecturers, careful scaffolding in terms of questions posed throughout the study materials, and (c) structured collaboration in class projects and activities.

In Project 1 and Project 2, the effectiveness of the CABLE approach was evaluated using a post-test on computer programming skills and problem solving skills. Student attitudes towards the CABLE approach were evaluated using a post-study questionnaire, test scores, student interviews, and weekly feedback from the online helpdesk. In Project 3, levels of student engagement were evaluated by self reporting measures.

In Project 1, the results indicated that the CABLE group performed better in the post-test than the traditional group. Initial analysis based on the post-test indicated that CABLE had advantaged high-ability more than low-ability students. However, a more sensitive measurement using hierarchical regression analyses indicated that CABLE had a positive effect on achievement irrespective of ability level. The treatment effect was significant and independent of initialability level and gender.

Additional analyses, based on classifying items into recall and elaborative categories revealed that the positive

impact of the CABLE treatment was far stronger in the case of examination items demanding of high-level problem-solving cognition. This outcome indicates that the CABLE experience could lead to improved problem-solving and critical thinking skills. This supports earlier findings by Snyder Farrell and Baker (2000), and Hogan and Tudge (1999), on the positive effects of cognitive apprenticeship on problem-solving. Hence, the results of Project 1 support the hypothesis that the CABLE approach has positive effects on student achievement.

The results of Project 2 showed that students in the CABLE group scored more highly in the post-test than those in the traditional (non-CABLE) group. Hence these results support the hypothesis that students exposed to CABLE outperformed those taught in the traditional university mode. No significant differences could be discerned between CABLE and non-CABLE groups in student attitudes to the learning environment (as indexed on *PAS*), as both groups exhibited high levels of liking for their course instruction. In terms of collaborative learning, students in the CABLE group showed high levels of positive affect for collaborative learning, and with the high-ability group showing a stronger preference. The results were very similar to those in Phase 1. High levels of liking for the online environment was evidenced in the responses in the questionnaire, responses in student interviews, and the evaluation using the Triple P framework. There were no differences in the responses between ability groups. Evaluation using the Triple P framework indicated that in terms of the level of growth of the online community, the online community in this phase had progressed to stage three of the five stage model (Salomon's stages of growth) where students were involved in information exchange using e-mail and the discussion forum. Hence, the results support the hypothesis that students exposed to CABLE exhibited positive feelings towards the online aspects of the environment.

The results of Project 3 indicated that based on ratings of 7 variables, that CABLE classes appeared to be more strongly motivating than non-CABLE classes. There were significant differences between the students in the two treatments in terms of sense of reward and stimulation.

In summary the key findings of the CABLE trials are as follows:

Finding 1: *The CABLE model can be implemented as a viable instructional model in the teaching of Java programming. This confirms the possibility of making the transition from an instructional model of CABLE to a viable learning environment.*

Finding 2: *CABLE is a viable instructional model which can be introduced into the normal conduct and administration of university programming courses, without apparent, detrimental consequences.*

Finding 3: *Students exposed to CABLE evidenced increased achievement on Java programming scores relative to those taught in the traditional (non-CABLE) mode.*

Finding 4: *There were no significant differences in student attitudes towards the learning environment, between students taught with the CABLE model and those taught in the traditional university mode of instruction.*

Finding 5: *Students taught programming in CABLE showed positive attitudes towards the collaborative elements of CABLE.*

Finding 6: *Students exposed to CABLE reported positive evaluations towards the online learning elements of CABLE.*

Finding 7: *Students taught under CABLE reported higher levels of mental engagement when compared to students taught using traditional methods of course instruction*

The actual trialing and evaluation of the CABLE environment is the detailed subject of other papers: i) Chan Mow, Au & Yates, 2004; ii) Chan Mow, Au & Yates, 2006; and iii) Chan Mow, Au & Yates, 2008. The ultimate goal of designing a learning environment such as CABLE, informed by recent learning theories and design, is that it is hoped that an effective learning environment will lead to improved teaching and learning of computer programming within the university context.

REFERENCES

[1] Chan Mow, I.T (2006). The Effectiveness of Cognitive Apprenticeship based Learning Environment (CABLE) in Teaching Computer Programming. Unpublished PHD dissertation, University of South Australia

[2] Phit-Huan Tan, Choo-Yee Ting and Siew-Woei Ling (2009) Learning difficulties in programming course: Undergraduates' perspective and perception. Proceedings of IEEE International. Conference on Computing Technology and Development, 2, pp: 42-46.

[3] Garner, S. (2006) Cognitive load reduction in problem solving domains, Edith Cowan University, 2006.

[4] Winslow. L.E. (1996) "Programming pedagogy- a psychological overview". SIGCSE Bulletin, Vol 28, pp.17–22.

[5] Pane J.F, Myers, B.A.& Ko A. (2004) Natural programming languages and environments. Communications of ACM 47(9),pp. 47-52

[6] Bruce-Lockhart,M.P, Norvell T.S.& Cotronis, Y.(2007). "Program and algorithm visualization in engineering and physics". Electronic Notes in Theoretical Computer Science, vol 178, pp. 111-119.

[7] Blackwell, A.(2001). "First steps in programming: a rationale for attention investment models". Presented at IEEE Symposia on Human-Centric Computing Languages and Environments. Arlington, VA, pp.2-10.

[8] A.K.,Lui,A.K., Kwan,R., Poon, M, Cheung,Y.H.Y.(2004). " Saving weak programming students: applying constructivism in a first programming course". SIGCE Bulletin, 36(2).

[9] Dickey, M. D. (2008). Integrating cognitive apprenticeship methods in a Web-based educational technology course for P-12 teacher education. Computers and Education, 51(2),

[10] Winnips,J.C.(2001). "Scaffolding by design. a model for www learner support". Unpublished PHD dissertation, Netherlands :University of Twente.

[11] Salomon,G., Perkins,D.N. & Globerson,T. (1991) Partners in cognition: Extending human intelligence with intelligent technologies. Educational Researcher, 20(3), pp.2-9.

[12] Blair,A. & Hume, T. (1994). An Exploration of the Application of Constructive learning Techniques to Software development using Object orientation as a Vehicle. Paper presented at CTI Annual Conference. Retrieved March 12, 2003, from http://www.ulst.ac.uk/cticomp/therhume.html

[13] Cheng,W.F.J.(2010) Teaching and Learning to Program: A Qualitative study of Hong Kong sub degree students Unpublished PHD dissertation, University of Sydney.

[14] The Joint National Task Force on Computing Curricula Report.(2013) IEEE Computer Society, Association for Computing Machinery, Computing Curricula 2013, Computer Science, Strawman Draft Report

[15] Parham,R.J.(2003). "An assessment and evaluation of computer science education". Journal of Computer Science in Colleges, 19(2), pp.116-127.

[16] Brown, F. A. (2008). Collaborative learning in the EAP classroom: Students' perception. [Online] Available: www.esp-world.info/Articles_17/issue_17.htmL.

[17] Brill, J. M., & Galloway, C. (2007). Perils and promises: University instructors' integration of technology in classroom-based practices. British Journal of Educational Technology, 38, 95-105.

[18] Moursound,D.G. (2002) Increasing your expertise as a problem solver: Some roles of computers. Eugene, OR: ISTE. Copyright (C) David Moursund. Retrieved August 11th, 2004 from http://www.uoregon.edu/~moursund/PSBook1996/chapter_9.htm.

[19] Jarvela,S. (1998) "Socioemotional aspects of students learning in a cognitive apprenticeship environment". Instructional Science, 26, pp.439-471.

[20] Kolling, M. & Rosenberg,J. (2001)"Guidelines for teaching object orientation with java". SIGCSE Bulletin, 33(3), pp.33-36.

[21] Carmichael,H.W., Burnett,J.D.,Higginson,W.C., .Moore,B.G. & Pollard,P.J. (1985). "Computers, children and classrooms: a multisite evaluation of the creative use of microcomputers by elementary school children". Toronto, Ontario, Canada: Ministry of Education.

[22] Williams,L., Wiebe,E., Yang,K., Ferzli,M. & Miller,C. (2002)"In support of pair programming in the introductory computer science course", Computer Science Education, 2, pp.197-202.

[23] Calvani, A., Fini, A., Pettenati, M. C., Sarti, L., and Masseti, M. (2006). Design of collaborative learning environments: bridging the gap between cscl theories and open source platforms, In Journal of e-Learning and Knowledge Society

[24] Pedroni,M.(2003) "Teaching introductory programming with the inverted curriculum approach", Diploma thesis, Department Computer Science, ETH Zurich. [Electronic Version].

[25] Clements,D.H. (1999) "The future of educational computing research: the case of computer programming, information technology in childhood education", pp. 147-179. Retrieved Jan 2nd , 2004 from http:// investigations.terc.edu/relevant/pdf/EducationalComputing.pdf .

[26] Brusilovsky,P. & Spring,M. (2004) "Adaptive, engaging, and explanatory visualisation in a C programming course", ED-MEDIA'2004 – World Conference on Educational Multimedia, Hypermedia and Telecommunications,eds L. Cantoni & C. McLoughlin, Lugano, AACE, 21-26 June 2004,Switzerland, pp. 1264-1271.

[27] Yuen,A.H.K. (2006) "Learning to program through interactive simulation",Educational Media International, 43(3), pp. 251-268.

[28] Wang, F. K. and Bonk, C. J. (2005). A design framework forelectronic cognitive apprenticeship. Journal. Asynchronous. Learning. Networks.5(2) http://www.sloan-c.org/publications/jaln/v5n2/v5n2_wang.asp).

[29] Schank, R.C., Berman,T. & McPherson,J. (1999) Learning by doing. In C. M. Reigeluth (Ed.), Instructional design theories and models: A new paradigm of instructional theory, pp. 161–181. Mahwah, NJ: Lawrence Erlbaum.

[30] Chmiel ,R. &.Loui,M.C. (1998) "Debugging: from Novice to Expert" : Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2004, Norfolk, Virginia, USA, ACM 2004, ISBN 1-58113-798-2.

[31] Lea, M., & Street, B. (1998) Student writing in Higher Education: an academic literacies approach. Studies in Higher Education, 23(2), pp.157–172.

[32] Tholander,J.,Rutz,F.,Karlgren,K. & Ramberg,R. (1999) "Design and evaluation of an apprenticeship setting for learning object-oriented modeling". In: Cumming, G., Okamoto, T., & Gomez, L., (Eds.), Proceedings of The International Conference on Computers in Education, Chiba, Japan, Nov. 1999.

[33] Johnson,D.W. & Johnson,R.T. (1999) "Learning together and alone: cooperative, competitive, and individualistic learning" (5th ed.). Boston: Allyn & Bacon.