The International JOURNAL *of* LEARNING

Volume 14, Number 1

Some Pedagogical Considerations in Teaching First -Year Programming

June Mogensen and Tim Roberts

THE INTERNATIONAL JOURNAL OF LEARNING
<http://www.Learning-Journal.com>

First published in 2007 in Melbourne, Australia by Common Ground Publishing Pty Ltd
www.CommonGroundPublishing.com.

© 2007 (individual papers), the author(s)
© 2007 (selection and editorial matter) Common Ground

Authors are responsible for the accuracy of citations, quotations, diagrams, tables and maps.

All rights reserved. Apart from fair use for the purposes of study, research, criticism or review as permitted under the Copyright Act (Australia), no part of this work may be reproduced without written permission from the publisher. For permissions and other inquiries, please contact
<cg-support@commongroundpublishing.com>.

ISSN: 1447-9494
Publisher Site: <http://www.Learning-Journal.com>

THE INTERNATIONAL JOURNAL OF LEARNING is a peer refereed journal. Full papers submitted for publication are refereed by Associate Editors through anonymous referee processes.

Typeset in Common Ground Markup Language using CGCreator multichannel typesetting system
<http://www.CommonGroundSoftware.com>.

Some Pedagogical Considerations in Teaching First-Year Programming

June Mogensen, Central Queensland University, Queensland, Australia
Tim Roberts, Central Queensland University, Queensland, Australia

Abstract: This paper is addressed to educators and administrators planning to offer an introductory programming course (CS1) to first-year students, either as an entirely new offering, or as a replacement for, or modification of, an existing offering. The authors attempt to highlight some of the lessons learnt by educators in their various pedagogical approaches to generating interesting and engaging real-world experiences for the next generation of CS1 students. Ultimately, those charged with the responsibility of planning and developing CS1 should be guided by past theory and practice when selecting the best available options for their particular circumstances.

Keywords: Collaborative Learning, CS1, Extreme Programming, Group Learning, Pair Programming

Introduction

E DUCATORS CHARGED WITH the planning and development of the curriculum and delivery for a first-year computer programming course face many complex challenges. It is well-known that computer science suffers dramatically low retention rates, and has problems attracting and engaging a diverse audience, documented in the form of declining enrolment and completion by women and minorities (National Science Foundation 2000, National Centre for Education Statistics 2002), with overall failure rates in introductory computer science sometimes approaching 50% (Forte 2003).

Attempts to alleviate these problems have often led to heated discussion regarding the benefits of structured as opposed to object-oriented programming, or the best language to use (C++? Java? Visual Basic? Python?), or the appropriate time to cover parameter passing, or.... But it is at least arguable that more light, and less heat, would be generated by a greater concentration on aspects of the pedagogical approach to be employed. In this paper two such aspects - the use of pair and collaborative learning, and the selection of a suitable form of assessment - are considered in the light of the existing literature.

It needs to be stressed at the outset that the appropriate methods and solutions will depend upon contextual factors such as the type of program being offered, the backgrounds and prior experiences of the learners and the educators, and the resources available for development and delivery. Thus, no attempt is made to separate right from wrong, or even good practice from bad practice. Rather, the aim is

to present some examples of past practice, to form the basis for informed decisions about the teaching and delivery of CS1.

The use of Collaborative Learning

The theories of educationist Lev Vygotsky provide an underpinning for collaborative learning (Vygotsky, 1978; Doolittle, 1997). This paper is not an appropriate place to discuss the history or significance of collaborative learning in general, but the benefits of collaborative learning, in both face-to-face and online environments, are extremely well documented (eg Panitz & Panitz, 1998; Felder & Brent, 1994; Burdett, 2003). Specific collaborative learning techniques are detailed by Kagan (1994). Johnson, Johnson and Stanne (2000) have reported on a number of studies that have compared collaborative and cooperative approaches with other methodologies. A useful and up-to-date resource for educators interested in the use of online collaborative learning techniques in higher education can be found at Roberts & McInnerney (2007).

Daigle and Doran (1998) set out to determine if active learning and collaboration could promote 'fact base' comprehension as a basis for attaining the higher levels of knowledge described in Bloom's Taxonomy (Bloom, 1956). Their observations included that students took a more active role in their own learning; facts were learned more quickly and outside of the traditional classroom learning environment, lecture time was reduced; other life skills (such as working in teams) were acquired; and, the approach was considered to be 'learning level independent'.

Extreme Programming (XP), in which all code is written collaboratively with a partner, and its perceived benefits, were the subject of research by Williams & Kessler (2000). There was a significant degree of resistance to this idea, due to the fact that programming had always been a solitary pursuit, but the success reported both anecdotally and statistically from pair teams showed it to be a *'powerful technique for generating high-quality software products'*. They followed this study with a trial at the University of Utah (Williams & Kessler, 2000), which confirmed that pair-learning enabled students to learn new languages better and faster than was possible by an individual. They also found that the lecturer's workload was reduced because pairs were more likely to find sources other than the lecturer to answer their questions, and there were only half as many assignments to mark. Student success and, consequently, morale was also higher in paired teams.

Their findings were confirmed by Yang (2002), whose concluded from pair-learning experiments conducted at North Carolina State University in 2001, for his Master's thesis, that:

- *Students who followed the pair learning style performed better on the programming assignments and were more likely to get a higher score on the examinations.*
- *A higher percentage of the students who utilized the pair learning style succeeded in the CSI class by completing the course with a grade of C or better.*
- *The teaching assistants in the pair learning section had reduced workload, since students in the pair learning section were more self-sufficient.*
- *Paired students demonstrated higher-order thinking skills than the students when compared with the students who worked alone.*
- *Students in the pair learning section were more likely to think far beyond the programming assignment to applying their knowledge in more challenging programming contexts.*
- *Paired students collaborated more extensively with each other; and collaboration is an important skill for programmers in industry. (Yang 2002).*

LeJeune (2003) examined previous works on collaborative learning and considered that although CSI was an ideal environment to introduce it, some previous attempts had failed because they failed to appropriately address all five essential elements: a) a common problem b) small group interaction c) collaborative behavior d) positive interdependence and e) individual and group accountability and responsibility.

In his own experiments, he encouraged competitive behaviour between different groups, but collabora-

tion between group members. He gave different groups responsibility for researching different areas, all of which were important in solving problems, so that each group needed to exercise positive interdependence to access knowledge from other groups. Being responsible for each other's participation and contributions also fostered student interdependence. Peer pressure was a natural catalyst for individual responsibility and accountability, and having an examination at the end of the semester further enhanced this. No statistics were gathered on this work due to the small size of the classes, but:

'...anecdotal evidence and personal observations suggest that this approach was generally successful'. LeJeune (2003)

In their research on the subject, Hanks, McDowell, Draper & Krnjajic (2004) gave the same programming assignment to both pair and individual learners, to assess performance. They were not confident that paired students' programs were designed better or showed greater understanding of concepts. However, they found sufficient evidence to suggest *'the benefits of paired programming outweigh its costs'*. Paired students' programs had greater functionality, the students were more confident and satisfied in their work, and they were more likely to fulfil course requirements for programming assignments.

Student culture and the way it inhibits collaborative learning were the subject of a study by Waite, Jackson & Leonardi (2004). They found 7 areas that had the potential to derail pair learning:

- students preferred to work alone;
- procrastination on assignment work by some members left little time for discussion;
- experimentation with code, rather than striving first to understand the problem;
- non-acceptance of the process – seeing assignments as products rather than as opportunities to learn;
- combativeness where students see their own view clearly, but have difficulty seeing the relevance of other points of view;
- lack of supportive behavior towards other team members; and
- lack of passion in participants.

They suggested ways in which this culture might be overcome, and observed that resistance decreased over time, leading to better performance and increased student satisfaction.

In fact many colleges, e.g. University of Wisconsin (2004) and the University of Sydney (Barrie, Crowe & Kamvounias, 2002), actually advise students how to form study groups, and some even offer

the services of course coordinators to help with contact details for fellow students.

Why then is collaborative learning not more widely practiced in CS1? Some courses do use collaborative techniques, but these are still very much in the minority. The majority of courses assume individual learning, but it is common for students to form informal study groups, which may then present themselves as cases of plagiarism.

A dramatic drop in the rate of “WDF” (Withdrawn/Fail) grades was noted by Chase and Okie (2000) over a three-semester period of trialing two strategies, peer instruction and cooperative learning, with female students having the highest improvement. They noted particularly more regular attendance and more active participation in class discussion and activities. Cooperative learning techniques were also studied by Beck, Chizhik & McElroy (2005), who also noted significant, positive effects for students, particularly for women and minority students.

At the Queensland University of Technology, an online, interactive, collaborative learning environment was developed and trialed with twelve students. Truong, Bancroft & Roe (2002) reported on the work, stating that:

‘... collaboration work benefits problem solving efforts, even for relatively non-complex and small problems. It enhances confidence in the solution and enjoyment of the process.’

They intended to incorporate extra functionality to allow students to work in groups in future versions of the software to gain the added benefits of working collaboratively.

A course designed to address challenges faced by both teachers and students in early CS1 courses was reported on by Barg, Fekete, Greening, Kay, Kingston & Crawford (2000). They aimed to give students a more real-world experience in their learning activities – one based on teamwork and broad problem-solving skills. They wanted to foster such goals as:

- an integrated curriculum;
- competence in programming in an object-oriented language;
- enhancement of program quality;
- production of low running time programs;
- competence in the use of recursion;
- encouraging students to become self-directed learners;
- fostering higher level thinking to solve problems;
- understanding and valuing working as a team member;
- an ability to communicate both orally and in writing; and

- gaining experience in planning projects. (Barg, Fekete, Greening, Kay, Kingston & Crawford, 2000).

After reviewing the literature, they decided that the approach of Problem-Based Learning (PBL) would most suit their learning and outcome objectives.

True PBL moved away from the typical small ‘problems’ that were currently in vogue, to solving large, real-world problems. This often meant that the solution involved disciplines other than CS, and which were unfamiliar to students. Students were not told how to solve the problem, but given access to resources to help them clarify the problem and how it might be resolved. They worked cooperatively, often with the aid of a tutor who was not necessarily an expert in the field, but rather a facilitator of the learning process. This approach was also in line with the current pedagogical move away from the teacher being the ‘font of all knowledge’.

The authors devoted class-time to generic problem solving skills, which they defined as:

- defining a learning plan;
- brain-storming to get started on a problem;
- reflection;
- articulation of problems and solutions;
- self-assessment; and
- practice in active listening and other communication skills.

Analysis of the outcomes of the first semester’s trial of PBL, against a control group being taught under the traditional system, showed no significant difference in performance of the two groups, however as the exam had been set by traditional lecturers, this outcome was viewed as positive for the PBL trial group. Another benefit of PBL was that plagiarism was no longer an issue.

Surveys taken at the end of the semester indicated that of those who completed the questionnaire, a majority of both PBL and traditional learners found the work meaningful but PBL learners’

‘emotional response to the course was more positive than that of students trained in traditional learning methods’ (Barg et al, 2000).

Follow-up assessments of course participants confirmed the authors’ belief that PBL fostered all the goals they desired, which, valuable as they may have been to students in many other aspects of their studies, are perhaps even more valued by future employers. They also believe students’ programming skills were very much enhanced in comparison with those doing traditional CS1 courses.

Greening, Kingston and Crawford (1996) were also advocates of PBL. Their report on a trial PBL

course conducted in 1996 in parallel with a conventional CS1 course concentrated on two concerns: could weaker students learn equally well with the PBL format as with current methods, and would PBL impose on already stretched resources of the faculty? They noted that with the two courses being run in parallel, students would compare what was being taught each week, and some in the PBL class were concerned that they were not being taught the same topics. They did not easily accept moving out of their comfort zone (being passive learning). This became less of a problem as students came to understand the philosophy of PBL.

No evidence was found to suggest weaker students could not perform equally well in either discipline, and they concluded that costs of both systems were comparable. Results were sufficiently encouraging to continue the work, although the authors recognised that a number of mistakes had been made in the original trial and these would be addressed in future trials.

The Selection of an Appropriate Form of Assessment

Examinations provide very artificial environments in which to assess student learning. It is hard to equate the experience of the examination room with any real-world work situation. Unless appropriately formulated, the temptation can be to test memory rather than practical programming skills.

Rather than using assessment of written code in an introductory CS course, Lister (2001) suggested that students should be examined by multiple-choice questions. He believed that marking schemes for assignments based on writing large amounts of code were *'generous to the point of being inconsistent with the subject objectives'*.

He concluded, *'CS1 cannot produce accomplished programmers'* but stated that it is essential for the first course to have clear objectives and an assessment that tests those objectives. Objectives he set for his trial included bringing all pupils to a uniform level of performance on minimum essentials, and addressing only the lower levels (knowledge and comprehension) of Bloom's taxonomy. Multiple-choice questions seemed a natural way to achieve these objectives.

Traynor and Gibson (2005) investigated the use of automatically generated multiple choice questions and automatic assessment of students' responses to these questions. They set down three requirements for their auto-generated questions to clearly define what questions could be generated, while at the same time ensuring quality questions were generated. They found this was achievable using two approaches: a) template-based mutations (TBM) and b) a predefined

syntax tree (PST). In TBMs, the lecturer supplies a suitable suite of problems, each one consisting of *'a piece of code and a question regarding its behaviour'*. PSTs involve

'using randomization techniques for creating short pieces of code.... and making decisions based on input parameters about how the program will be developed'.

Results of their trials suggested that template questions achieved higher quality results than did the syntax tree, but required additional work by the lecturer. Future work will compare the re-developed system against traditional assessment methods.

Smith (2004) stated that passively learning facts by rote to pass exams has been shown to be ineffective in preparing students for their working lives, where the other 95% of their education takes place. He aimed to assist students to become critical thinkers, capable of independent group projects. To achieve this, he placed much emphasis on weekly assessment, wherein students worked in fourperson teams, each person assuming a different role in the team each week. His conclusion was that, although it may initially appear daunting, such an assessment regime benefited students and teachers alike because of the constant feedback on performance. And finally, he noted that all students improved their learning processes, not just the better ones.

The problem of students being unable to develop straightforward programs after introductory CS programs was addressed by Daly & Waldron (2004). They considered that the assessment methods currently in use (programming assignments and written exams) failed to reveal the extent of the problem. They proposed introducing a lab exam to assess programming and problem solving ability, which could be assessed automatically and provide instant feedback. A significant drawback with this type of assessment is the number of computers required to be in use simultaneously.

Overall, lab exams were found to be *'more accurate assessors of programming ability than traditional methods...'*. Staff could measure the effectiveness of their strategies by accurately measuring the programming ability of students. Students reacted positively to the change, realising that the only way to pass was to learn to program correctly.

A theory on using a participatory exam process to make the course more attractive to students, while also increasing knowledge retention, was put forward by Wu, Bieber, Hihltz & Han (2004). Constructivism and collaborative learning underpinned the process. The authors believed that examinations were a very important assessment tool, but that because of their influence on the way students learn, the format of

such exams needed to be such that they fostered achievement of educational goals.

The participatory exam process involved students in both setting the exam questions and assigning a result for fellow students' work. The authors were greatly encouraged by the results of this work, seeing a future where *'students are confident and skilled in self- and peer-assessment as part of their learning process'*.

Assessment based on computer-based exams was examined by Barros, Estevens, Dias, Pais & Soeiro (2003), in their efforts to increase motivation and reduce student attrition rates. They were so encouraged by the success of the work, that they have stated their intention to investigate an automatic marking system for the computer-based exams.

Carbone, Hurst, Mitchell & Gunstone (2000) regarded assessment of the students' competency as being of paramount importance in any redesign of the CS1 course, and advocated that students not be expected to write complex code too soon, stating:

'Often smaller tasks can still be used to develop the students' interest and understanding of the subject.'

The early work of Berglund and Daniels (1997) examined how changing methods of assessment changed the way students work and affected their quality of education. The research was carried out with a target group of sixty engineering students, but involved some common CS subject matter such as algorithms and data structures and computer architecture. Changes to assessment included replacing the final exam with weekly assignments, introducing seminars as a method of examination, or, where the final exam was retained, placing more emphasis on the students' competency in understanding and their ability to analyse and synthesize.

Although students had worked harder, and achieved deeper understanding of the work than previous students, generating the different assessments resulted in a very much increased workload for staff. However, the authors believed that the teachers' workload would approach previous levels as they gained familiarity with the system. The experience showed that changing the way students are assessed, changes the way they learn and has the potential to improve the quality of their education.

A computer-assisted formal programming examination for first year CS students was developed by English (2002). At the time, there were concerns over increasing student numbers and the accompanying increased marking workload for instructors. Plagiarism was also a problem with assignment work. Their solution was to use an on-line programming exam, which they believed fulfilled many needs, such as allowing a large variety of question types,

making allowances for special needs students, and addressing health and safety issues, whilst also automating the marking system to some extent. Trial results were very positive from both students' and teachers' viewpoints – students seeing the on-line exam as working well and being fair, while marking time was reduced from around ten to fifteen hours to just two hours.

At The Educators' Symposium, Edwards (2003), proposed a totally test-driven program right from the beginning of CS1. The key element of this style of program was *'comprehensive, automated evaluation of student work'*. He believed that this "rapid, concrete and useful feedback" had the potential to cause a paradigm shift in student behaviour.

Carter, English, Ala-Mutka, Dick, Fone, Fuller & Sheard (2003) examined current assessment practices and how respondents viewed the new technology of Computer Assisted Assessment (CAA). The study was undertaken at a time when class sizes were increasing and beginning to cause an undue burden on instructors charged with preparation and assessment of exam papers. Because CAA was relatively new, it was most easily applied to multiple-choice assessments. The authors considered that first year CS was an obvious place to use CAA because of the large numbers of students, and the fact that many skills assessed are lower-level skills such as remembering, understanding and application of knowledge. They considered that programming would actually benefit from 'little and often' assessment and that plagiarism could be reduced by individualizing tests.

The major perceived benefits of CAA were speed, consistency and availability, and increased amount, quality and variety of feedback to students. There were also benefits to teachers, such as allowing more time for advanced support, and reuse and sharing of resources between teaching staff. The authors considered the cost of such a system to be comparable to costs incurred for staff training, and that this could be reduced by developing a reusable question bank.

CS1 assignments that relied heavily on screen animations and user interaction to generate interest and enthusiasm amongst CS1 students were described by Astrachan and Rodger (1998). The classes and code for the animations and image display were provided to allow students to focus on learning C++. They believed this led to "better comprehension and mastery of the material in our courses". Building on this work, which used screen animation and user interaction to generate interest and enthusiasm with CS1 students, Wicentowski and Newhall (2005) found that assignments based on image processing were an effective pedagogical tool for teaching a range of topics, including multi-dimensional arrays, function call semantics and modular design, and that student reaction to them was overwhelmingly posit-

ive. Students were more engaged in the learning process and became more confident and capable programmers.

Whitfield (2003) discussed '*the implementation of course embedded assessment in CS1*'. In this, each assignment is assessed '*using criteria that are linked to course outcomes*'. CS1 outcomes, strategies to achieve these outcomes, and methods to assess their effectiveness, were developed. These CS1 outcomes were separate from, but touched on, outcomes developed for graduates of the CS degree. Assessment was still by assignment, but the first two looked at correct syntax, correct output, proper documentation and indentation, while the last two included problem solving skills (as demonstrated by the correctness of the algorithm), critical thinking skills (selection of conditional and repetition constructs) and communication skills (documentation and output structure).

This system did not create a lot of additional work for instructors, but it was felt that having several different kinds of assessment was very useful in identifying students' strengths as well as their weak areas. The students tested were found to be learning the language, but not developing problem skills sufficiently, a problem that was to be addressed by CS1 instructors after the trial period was concluded.

Conclusion

Courses in computer science typically suffer from above average failure rates, and consequent high levels of attrition. Discussions regarding the teaching and delivery of CS1 typically focus on language issues, perhaps to the detriment of pedagogical considerations, which may be more important. Educators looking at providing the best possible learning environment for their first-year students should consider, amongst other things, the use of extreme programming techniques; the potentially very positive role to be provided by the use of collaborative learning; and in particular, the use of pair programming; and also perhaps consider a higher usage of problem-based learning methodologies. When these are considered alongside the possibilities provided by different forms of assessment, with a perhaps greater concentration on using assessment for formative, rather than for purely summative, reasons, the opportunities for improvement in teaching CS1 are many.

The research highlighted in this paper will, hopefully, serve as a starting point for educators interested in refocusing attention away from the continuing contentious debates surrounding the merits of particular languages, and back on to some of the more fundamental, and perhaps ultimately more important, pedagogical issues.

References

- Astrachan, O. and Rodger, S.H. (1998). Animation, visualization, and interaction in CS 1 assignments. *Proc. SIGCSE Technical Symposium on Computer Science Education*, Atlanta, Georgia, USA, 30(1):317 – 321, ACM Press.
- Barg, M., Fekete, A., Greening, T. Hollands, O., Kay, J., Kingston, J.H. & Crawford, K. (2000). Problem-based learning for foundation computer science courses. *Computer Science Education*, 10: 109-128, Swetz & Zeitlinger
- Barrie, S., Crowe, C. & Kamvounias, P. (2002). Using groups in teaching and learning. Retrieved on 10.05.06 from <http://www.ecu.edu.au/conferences/herdsa/main/papers/nonref/pdf/SimonBarrie.pdf>
- Barros, J.P., Esteveens, L., Dias, R., Pais, R. & Soeiro, E.(2003). Using lab exams to ensure programming practice in an introductory programming course. *Proc. Annual Conference on Innovation and Technology in Computer Science Education*, Thessaloniki, Greece, 35(3):16 – 20, ACM Press.
- Beck, L.L., Chizhik, A.W. and McElroy, A.C. (2005). Cooperative learning techniques in CS1: design and experimental evaluation. *Proc SIGCSE Technical Symposium on Computer Science Education*. St. Louis, Missouri, USA 37(1):470-474, ACM Press.
- Berglund, A. and Daniels, M. (1997): Improving education quality, a full scale study. *Proc. SIGCSE Technical Symposium on Computer Science Education*, San Jose, California, 29(1): 330 – 334, ACM Press.
- Bloom, B.S. (ed.) (1956): *Taxonomy of Educational Objectives, the classification of educational goals – Handbook I: Cognitive Domain*, New York: McKay.
- Burdett, J. (2003). Making Groups Work: University Students' Perceptions. *International Education Journal* Vol 4, No 3, 2003.
- Carbone, A., Hurst, J., Mitchell, I. & Gunstone, D. (2000): Principles for designing programming exercises to minimise poor learning behaviours in students. *Proc. Australasian Conference on Computing Education*, Melbourne, Australia, 26-33, ACM Press.
- Carter, J., English, J., Ala-Mutka, K., Dick, M., Fone, W., Fuller, U. & Sheard, J. (2003): How shall we assess this? Annual Joint Conference Integrating Technology into Computer Science Education. *ITiCSE 2003 Working Group Reports*, 35(4): 107 – 123.
- Chase, J.D. and Okie, D.G. (2000): Combining cooperative learning and peer instruction in introductory computer science. *Proc. SIGCSE Technical Symposium on Computer Science Education*, Austin, Texas, United States, 32(1):372-376, ACM Press.
- Daigle, R.J. and Doran, M.V. (1998). Facilitating Bloom's level one through active learning and collaboration. *J. of Information Systems Education*, 9(3): 3 - 6.

- Daly, C. and Waldron, J. (2004). Assessing the assessment of programming ability. *Proc. SIGCSE Technical Symposium on Computer Science Education*, Norfolk, Virginia, USA, 36(1): 210 – 213, ACM Press.
- Doolittle, P. (1997). Vygotsky's Zone of Proximal Development as a Theoretical Foundation for Cooperative Learning. *Journal on Excellence in College Teaching*, 8 (1): 83-103.
- Edwards, S. (2003). Rethinking computer science education from a test-first perspective. *Companion of 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages and Applications*, Anaheim, Ca., USA, 148-155, ACM Press.
- English, J. (2002). Experience with a computer-assisted formal programming examination. Retrieved on 12.03.06 from <http://www.iticse2002.dk/conference/pday1.shtml>
- Felder, R. M. & Brent, R. (1994). Cooperative learning in technical courses: procedures, pitfalls, and payoffs. Retrieved on 12.03.06 from http://www.ncsu.edu/felder_public/Papers/Coopreport.html
- Forte, A. (2003). Programming for communication: overcoming motivational barriers to computation for all. *Proc. IEEE Symposia on Human Centric Computing Languages and Environments*, Auckland, NZ, 2003. 295-286.
- Greening, T., Kay, J., Kingston, J. Crawford C. (1997). Results of a PBL trial in first-year computer science. *Proc. Australasian Conference on Computer Science Education*, The University of Melbourne, Australia, 201 – 206, ACM Press.
- Hanks, B., McDowell, C., Draper, D. & Krnjajic, M. (2004). Program quality with pair programming in CS1. *Proc. SIGCSE Conference on Innovation and Technology in Computer Science Education*, Leeds, UK, 36(3): 176 – 180, ACM Press.
- Johnson, D., Johnson, R. & Stanne, M. (2000). Cooperative learning methods: A meta-analysis. Retrieved on 25.11.06 from <http://www.co-operation.org/pages/cl-methods.html>
- Kagan, S. (1994). Cooperative learning resources for teachers. San Clemente, CA: Resources for Teachers.
- LeJeune, N. (2003). Critical components for successful collaborative learning in CS1. *J. Computing Sciences in Colleges*, 19(1): 275 – 285, Consortium for Computing Sciences in Colleges.
- Lister, R. (2001). Objectives and objective assessment in CS1. *Proc. Thirtysecond SIGCSE Technical Symposium on Computer Science Education*, Charlotte, North Carolina, USA, 33(1): 292 – 296, ACM Press.
- National Center for Education Statistics. Beginning Post Secondary Students (BPS) Longitudinal Study 1996-2001 (2002): Retrieved on 18.03.06 from <http://nces.ed.gov/pubsearch/pubsinfo.asp?pubid=2002171>
- National Science Foundation (2000). Women, minorities, and persons with disabilities in science and engineering 2000, Arlington, Virginia. Retrieved on 12.03.06 from <http://www.nsf.gov/sbe/srs/nsf00327/pdfstart.htm>.
- Panitz, T. & Panitz, P. (1998). Ways To Encourage Collaborative Teaching In Higher Education. in *University Teaching: International Perspectives*, James J.F. Forest edition, 1998, Garland Publishers: New York.
- Roberts, T.S. and McInnerney, J. (2007). Online Collaborative Learning in Higher Education. Retrieved on 10.03.07 from <http://clp.cqu.edu.au>
- Smith, P.D. (2004): *A process education approach to teaching math & computer science*. Retrieved on 18.04.06 from <http://www.pcrest.com/psmith.htm>
- Traynor, D and Gibson, J.P. (2005). Synthesis and analysis of automatic assessment methods in CS1: generating intelligent MCQs. *Proc SIGCSE Technical Symposium on Computer Science Education*. St. Louis, Missouri, USA 37(1): 495-499, ACM Press.
- Truong, N., Bancroft, P. & Roe, P. (2002). *ELP – a web environment for learning to program*. *Proc. Australasian Computer Science Conference on Research and Practice in Information Technology*, 16: 255 – 264, Australian Computer Society, Inc. Referenced on 14.06.07 from <http://www.ascilite.org.au/conferences/auckland02/proceedings/papers/082.pdf>.
- University of Wisconsin. (2004). Request for starting a study group. Retrieved on 12.03.06 from <http://www.uwm.edu/Dept/TARC/Application/studygroups.html>
- Vygotsky, L. (1978). Social development theory. Retrieved on 12.03.06 from <http://tip.psychology.org/vygotsky.html>.
- Waite, W.M., Jackson, M.H. & Leonardi, P.M. (2004). Student culture vs group work in computer science. *Proc. SIGCSE Technical Symposium on Computer Science Education*, Norfolk, Virginia, USA, 36(1): 12 – 16, ACM Press.
- Whitfield, D. (2003). From university wide outcomes to course embedded assessment of CS1. *J. of Computing Sciences in Colleges*, 18(5): 210 – 220, ACM Press.
- Wicentowski, R & Newhall T. (2005). Using image processing projects to teach CSI topics. *Proc. SIGCSE Technical Symposium on Computer Science Education*, St. Louis, Missouri, USA 37(1):287-291, ACM Press.
- Williams, L.A. and Kessler, R.R. (2000). All I really need to know about pair programming I learned in kindergarten. *Communications of the ACM*, 43(5): 108 – 114, ACM Press.
- Wu, D., Bieber, M., Hihltz, S.R. & Han, H.J. (2004). Constructivist learning with participatory examinations. *Proc. 37th Annual Hawaii International Conference on System Sciences*, Track 1, Vol 1, IEEE Computer Society.
- Yang, K. (2002). Pair learning in undergraduate computer science education. *Laboratory for Collaborative System Development at NC State University*, Retrieved on 14.02.06 from <http://www.lib.ncsu.edu/theses/available/etd-05222002-130217/>

About the Authors

June Mogensen

Central Queensland University, Australia

Tim Roberts

Central Queensland University, Australia

THE INTERNATIONAL JOURNAL OF LEARNING

EDITORS

Mary Kalantzis, University of Illinois, Urbana-Champaign, USA.

Bill Cope, University of Illinois, Urbana-Champaign, USA.

EDITORIAL ADVISORY BOARD

Michael Apple, University of Wisconsin-Madison, USA.

David Barton, Lancaster University, UK.

Mario Bello, University of Science, Technology and Environment, Cuba.

Robert Devillar, Kennesaw State University, USA.

Manuela du Bois-Reymond, Universiteit Leiden, Netherlands.

Ruth Finnegan, Open University, UK.

James Paul Gee, University of Wisconsin-Madison, USA.

Kris Gutierrez, University of California, Los Angeles, USA.

Roz Ivanic, Lancaster University, UK.

Paul James, RMIT University, Melbourne, Australia.

Carey Jewitt, Institute of Education, University of London, UK.

Andreas Kazamias, University of Wisconsin, Madison, USA

Peter Kell, University of Wollongong, Australia.

Michele Knobel, Montclair State University, New Jersey, USA.

Gunther Kress, Institute of Education, University of London.

Colin Lankshear, James Cook University, Australia.

Daniel Madrid Fernandez, University of Granada, Spain.

Sarah Michaels, Clark University, Massachusetts, USA.

Denise Newfield, University of Witwatersrand, South Africa.

Ernest O'Neil, Ministry of Education, Addis Ababa, Ethiopia.

José-Luis Ortega, University of Granada, Spain.

Francisco Fernandez Palomares, University of Granada, Spain.

Ambigapathy Pandian, Universiti Sains Malaysia, Penang, Malaysia.

Miguel A. Pereyra, University of Granada, Spain.

Scott Poynting, University of Western Sydney, Australia.

Angela Samuels, Montego Bay Community College, Montego Bay, Jamaica.

Juana M. Sancho Gil, University of Barcelona, Spain.

Michel Singh, University of Western Sydney, Australia.

Richard Sohmer, Clark University, Massachusetts, USA.

Pippa Stein, University of Witwatersrand, South Africa.

Brian Street, King's College, University of London, UK.

Giorgos Tsiakalos, Aristotle University of Thessaloniki, Greece.

Gella Varnava-Skoura, National and Kapodistrian University of Athens, Greece.

Cecile Walden, Sam Sharpe Teachers College, Montego Bay, Jamaica.

Nicola Yelland, Victoria University, Melbourne, Australia.

Wang Yingjie, School of Education, Beijing Normal University, China.

Zhou Zuoyu, School of Education, Beijing Normal University, China.

Please visit the Journal website at <http://www.Learning-Journal.com> for further information:

- ABOUT the Journal including Scope and Concerns, Editors, Advisory Board, Associate Editors and Journal Profile
- FOR AUTHORS including Publishing Policy, Submission Guidelines, Peer Review Process and Publishing Agreement

SUBSCRIPTIONS

The Journal offers individual and institutional subscriptions. For further information please visit

<http://ijl.cgpublisher.com/subscriptions.html>. Inquiries can be directed to

subscriptions@commongroundpublishing.com

INQUIRIES

Email: cg-support@commongroundpublishing.com