

What I've read so far

Robyn A. McNamara

May 13, 2005

References

ACM/IEEE-CS Joint Curriculum Task Force (1991), 'Computing Curricula 1991', *CACM* **34**(6), 69 – 84. Summary of the full report on the computing curriculum. Talks about recommended content of computing courses; light on pedagogy. Useful as a frame of reference – when we talk about assessment, *this* is what's being assessed. (p68-tucker.pdf).

Allan, V. & Kolesar, M. V. (1997), 'Teaching computer science: a problem solving approach that works', *SigCUE Outlook* **25**(1-2), 2–10. Another approach to managing diversity of expectations and levels of ability and experience at introductory programming level. Evidence that even just application experience is significantly helpful with successful completion; evidently computer familiarization has a steeper learning curve than we usually like to pretend. Programming can be overwhelming to the unprepared, and they tend to drown in a welter of program syntax rather than coming to grips with analytical issues. Spreadsheets and puzzle-style problem-solving form the basis of a CS0 course, which seemed to induce definite improvement in student performance.

Applin, A. G. (2001), 'Second language acquisition and CS1: is * == **?', *SIGCSE* pp. 174–178. Comparison of cognitive aspects of learning to program versus learning a foreign language. One would predict that showing students an example of good code would help them write good code. The evidence seems to bear this prediction out.

Arter, J. & McTighe, J. (2001), *Scoring rubrics in the classroom: using performance criteria for assessing and improving student performance*,

Experts in Assessment, Corwin Press. Developing marking schemes, with particular attention paid to criterion-referenced schemes. Nothing especially IT-oriented, but thorough coverage of general assessment.

- Ashworth, P., Bannister, P. & Thorne, P. (1997), 'Guilty in whose eyes? University students' perceptions of cheating and plagiarism in academic work and assessment', *Studies in Higher Education* **22**(2), 187–203. The sociology of plagiarism. Gives a useful overview of social and psychological factors that influence students' attitudes to plagiarism.
- Astin, A. W. (1991), *Assessment for excellence: The philosophy and practice of assessment and evaluation in higher education*, Macmillan.
- Barker, L. J. & Garvin-Doxas, K. (2003a), 'The effect of institutional characteristics on participation of women in computer science bachelors degree programs', *ITiCSE* p. 242.
- Barker, L. J. & Garvin-Doxas, K. (2003b), 'Why project courses sometimes widen the experience gap among students', *ITiCSE* p. 258.
- Barker, L. J., Garvin-Doxas, K. & Jackson, M. (2002), 'Defensive climate in the computer science classroom', *SIGCSE* pp. 43–47.
- Barrett, R. & Cox, A. L. (2005), "At least they're learning something": the hazy line between collaboration and collusion', *JAEHE* **30**(2), 107–122.
- Barros, J. a., Estevens, L., Dias, R., Pais, R. & Soeiro, E. (2003), 'Using lab exams to ensure programming practice in an introductory programming course', *ITiCSE 2003* pp. 16–20.
- Beauboeuf, T. (2003), 'Why computer science students need language', *ITiCSE*.
- Ben-Ari, M. (2001), 'Constructivism in computer science education', *Jnl. of Computers in Mathematics and Science Teaching* **20**(1), 45–73. [constructivism.pdf](#) – A survey of attitudes to constructivism in the CS Ed literature. Includes comparative material on constructivism in the physical sciences and mathematics, where it is very popular and seems to be effective.

- Ben-Ari, M. & Sajaniemi, J. (2004), 'Roles of variables as seen by CS educators', *ITiCSE 2004* pp. 52–56.
- Beyer, S., Rynes, K., Perrault, J., Hay, K. & Haller, S. (2003), 'Gender differences in computer science students', *SIGCSE 2003* pp. 49–53. An attempt to discern reasons for underrepresentation of women in computing courses by surveying CS-major and non-CS-major attitudes to computers and computing. Significant differences were found in computer confidence.
- Booth, S. (1989), 'Learning to program: a phenomenographic perspective', *ACTA Univ. Gothenburg Studies in Educational Science* .
- Borich, G. D. & Tombari, M. L. (2004), *Educational assessment for the elementary and middle school classroom*, Pearson Prentice Hall.
- Bowden, J. (1984), 'Students' approaches to learning: Influence of assessment in first year', *University of Melbourne Gazette* **40**(4), 10.
- Bridges, P., Bourdillon, B., Collymore, D., Cooper, A., Fox, W., Haines, C., Turner, D., Woolf, H. & Yorke, M. (1999), 'Discipline-related marking behaviour using percentages: A potential cause of inequity in assessment', *Journal of Assessment and Evaluation in Higher Education* **24**(3), 285–300. Raw marks assigned by markers in disciplines where there is a lot of marker latitude have a differently-shaped distribution to those assigned by markers in disciplines where there is little latitude: paradigm examples being history versus mathematics.
- Brown, J., Andreae, P. & Biddle, R. (1997), 'Women in introductory computer science: Experience at Victoria University of Wellington', *SIGCSE* pp. 111–115. Describes measures taken at Victoria University, Wellington, to address high rates of attrition and poor results among female computer science students. Changes were made to many aspects of the program, but female success and attrition rates did not seem to be significantly affected.
- Brown, S. & Glasner, A. (1999), *Assessment Matters in Higher Education: Choosing and Using Diverse Approaches*, Oxford University Press. Caulfield General collection 378.1662 B879A.

- Bucci, P., Long, T. J. & Weide, B. W. (2001), ‘Do we really teach abstraction?’, *SIGCSE* pp. 26–30. Better pedagogy can improve students’ abstract reasoning ability.
- Burton, R. F. (2005), ‘Multiple-choice and true/false tests: Myths and misapprehensions’, *JAEHE* **30**(1), 65–72.
- Byrne, P. & Lyons, G. (2001), ‘The effect of student attributes on success in programming’, *ITiCSE* pp. 49–52. Gender, prior experience, learning style and prior academic performance. No really strong conclusions could be drawn, although study did show evidence that people with one particular learning style seem drawn to study computing.
- Califf, M. E. & Goodwin, M. (2002), ‘Testing skills and knowledge: introducing a laboratory exam in CS1’, *SIGCSE 2002* pp. 217–221.
- Carbone, A. J. & Kaasbøll, J. J. (1998), ‘A survey of methods used to evaluate computer science teaching’, *ITiCSE* pp. 41–45. Overview of methods used to evaluate new teaching strategies. Useful bibliography. (p41-Carbone.pdf).
- Carter, J., English, J., Ala-Mutka, K., Dick, M., Fone, W., Fuller, U. & Sheard, J. (n.d.), ‘How shall we assess this?’.
- Carter, J. & Jenkins, T. (99), ‘Gender and programming: What’s going on?’, *ITiCSE*. Gender differences in approaches to learning: extra, voluntary tutorial classes were offered, and women took them up to a much greater extent than did men, leading to a presumably erroneous perception that the teaching staff were more willing to help female students. Learning outcomes were “credible, if unspectacular”. Interesting for future study on psychological and sociological factors affecting uptake of extra tuition etc. Female students achieve results on a par with males but seem less confident.
- Chamillard, A. & Braun, K. A. (2000), ‘Evaluating programming ability in an introductory computer science course’, *SIGCSE 2000* pp. 212–216.
- Chamillard, A. & Joiner, J. K. (2001), ‘Using lab practica to evaluate programming ability’, *SIGCSE* pp. 159–163. Case study: how to run pracs while avoiding the usual pitfalls of cheating, the Friday effect etc. and

ensuring statistical validity. These pracs are only run for 180 minutes per semester and are summative, and are therefore more like Monash's prac exams than Monash's pracs. (p159-Chamillard.pdf).

- Clarke, V. A. & Teague, G. J. (1994), 'A psychological perspective on gender differences in computing participation', *SIGCSE* pp. 258–262. Factors influencing gender bias in Australian students' decisions to study computer science and to persist with that study in post-secondary education. Male students are advantaged by open, solo, unstructured lab classes and multiple-choice testing; female students are advantaged by closed, structured, groupwork lab classes and extended written exam questions. Both genders are effectively disadvantaged by the inaccurate portrayal in the media of professional computing as a solitary activity, when in reality most jobs in IT are profoundly human-centred.
- Cohoon, J. M. (1994), 'Departmental differences can point the way to improving female participation in computer science', *SIGCSE* pp. 198–202. CS departments at different universities have different female attrition rates, suggesting that institutional factors apply which we may be able to control. The reasons for this are not fully explored here, but could be enlightening if studied further.
- Curtis, K. K. (1982), 'Computer manpower: Is there a crisis?', Web page, National Science Foundation. Understaffing in the academic computer science community, presumably due to aggressive cherry-picking and headhunting by industry. Note the date: episodes of perceived academic/industry recruitment imbalance seem to have recurred several times in the young history of computing.
- Daly, C. & Waldron, J. (2002), Introductory programming, problem solving and computer assisted assessment, *in* '6th Annual International CAA Conference', pp. 95–107. Case study on the use of electronically-assessed prac exams for introductory programming. Includes a nice lucid explanation of the problems that appear to be endemic to introductory programming skills assessment – evidence suggests that students are coming out with a good grasp of the syntax of their programming language, but without enough problem-solving or analytical skills to be able to design and construct solutions to specified problems.

- Daly, C. & Waldron, J. (2004), 'Assessing the assessment of programming ability', *SIGCSE 2004* pp. 210–213.
- Dalziel, J. (1998), 'Using marks to assess student performance: Some problems and alternatives', *Journal of Assessment and Evaluation in Higher Education* **23**(4), 351–366. Problems with giving numeric marks for assessment. Some intriguing ideas about the process by which a qualitative assessment becomes a quantitative mark. Occasional excursion into philosophy and semantics of numerals versus numbers. (numeric-marks.pdf).
- de Raadt, M., Toleman, M. & Watson, R. (2004), 'Training strategic problem solvers', *SIGCSE* **36**(2), 48–51. Explicit teaching of problem-solving methodologies to overcome the tendency of programming courses to focus on (industrial) programming language syntax. A position paper rather than an experimental report; a preliminary study is discussed which demonstrated that introductory programming students' problem solving skills are poor, but that is only one data point.
- Decker, R. & Hirshfield, S. (1994), 'The top 10 reasons why object-oriented programming can't be taught in CS1', *SIGCSE 1994* pp. 51–55.
- del Río, A. C. (2004), 'How *not* to go about a programming assignment', *SIGCSE Bulletin* .
- Denning, P. J. (1999), 'Our seed corn is growing in the commons', Information Impacts online magazine: http://www.cisp.org/imp/march_99/denning/03_99denning.html. The tragedy of the commons as it applies to the supply of high-quality computer science graduates for research and higher education. When industry desperately needs qualified computing professionals, it skims the cream from computer science departments; this affects the supply of well-qualified and well-taught graduates in later years.
- du Boulay, B. (n.d.), *Studying the Novice Programmer*, Lawrence Erlbaum Associates, chapter Some Difficulties of Learning to Program, pp. 283–299. Conceptual errors that new programmers make. Now looking a little dated, but a good basis for further reading.
- Edwards, S. H. (2004), 'Using software testing to move students from trial-and-error to reflection-in-action', *SIGCSE* pp. 26–30.

- Everitt, B. S. (1993), *Cluster analysis*, 3 edn, Edward Arnold.
- Fleury, A. E. (2000), 'Programming in Java: student-constructed rules', *SIGCSE 2000* pp. 197–201.
- Fox, R. (2001), 'Constructivism examined', *Oxford Review of Education* **27**(1), 23–35. A sceptical review of constructivism. Debunks radical constructivism and social constructivism, and finds the rest more enlightening than the most naive positivist models, but not much more.
- Gagné, E. D., Yekovich, C. & Yekovich, F. (1993), *The Cognitive Psychology of School Learning*, 2nd edn, HarperCollins. Gippsland 370.152 GAG.
- Ginat, D. (2001), 'Misleading intuition in algorithmic problem solving', *SIGCSE* pp. 21–25. Students have a tendency to leap intuitively to an erroneous conclusion based on a faulty analogy, and then selectively perceive evidence supporting their conclusion and disperse or explain away evidence contradicting it. Difficulty in dispelling incorrect ideas if they have been enshrined in an internally-consistent construct.
- Ginat, D. (2003a), 'The greedy trap and learning from mistakes', *SIGCSE 2003* pp. 11–15. Naïve approaches to problem-solving and algorithm design lead to learners having misplaced confidence in the correctness of the greedy algorithm that they had proposed.
- Ginat, D. (2003b), 'The novice programmer's syndrome of design-by-keyword', *ITiCSE 2003* .
- Ginat, D. (2004), 'Do senior CS students capitalize on recursion?', *ITiCSE 2004* pp. 82–86.
- Gold, J. R., Jenkins, A., Lee, R., Monk, J., Riley, J., Shepherd, I. & Unwin, D. (1991), *Teaching geography in higher education: A manual of good practice*, Blackwell, chapter 8: Assessing students.
- Goldweber, M. e. a. (1997), 'Historical perspectives on the computing curriculum', *ITiCSE '97 Working Groups and Supplemental Proceedings* pp. 94–111. ITiCSE Working Group on Historical Perspectives in Computing Education. Urges caution in following trends in computing curricula, as many do not stand the test of time. Discusses pedagogical

approaches informed by the diverse intellectual background that computing brings together: mathematical, engineering/design, art, science. Sound basis in education theory.

Götschi, T., Sanders, I. & Galpin, V. (2003), ‘Mental models of recursion’, *SIGCSE 2003* pp. 346–350.

Greening, T. (2000), ‘Pedagogically sound responses to economic rationalism’, *SIGCSE* pp. 149–156.

Gunn, F. & Macdonald, I. (1997), ‘Developing a culture of teaching at first year university level: A training course for postgraduate student tutors that made a difference’, *ASERA* .

Hagan, D. & Sheard, J. (1996), ‘The value of discussion classes for teaching introductory programming’, *ITiCSE* pp. 108–111. Tutorial classes help marks in C++ programming. Caveat: tutor facilitates discussion among students (i.e. I’m not at all sure that the standard Clayton third-year chalk-and-talk is doing anywhere near as much good). Includes down-to-earth discussion of what to do in your tute class that is sadly inapplicable to Clayton tutors, as we’ve got way too much to write on the whiteboard to waste time getting the students to talk. And I’m probably going to have to rewrite this annotation before it goes live. [jsighj \(p108-Hagan.pdf\)](#).

Haines, C. (2004), *Assessing students’ written work*, Routledge/Falmer.

Hansen, M. R., Kristensen, J. T. & Rischel, H. (1999), ‘A theory-based introductory programming course’, *Frontiers in education* **1**, 11–25.

Hazzan, O. (2003), ‘Computer science students’ conception of the relationship between reward (grade) and cooperation’, *ITiCSE* **178–182**.

Holmboe, C., McIver, L. & George, C. (2001), Research agenda for computer science education, *in* ‘Proc. PPIG’, Vol. 13.

Hopper, M. (1998), ‘Assessment in WWW-based learning systems: Opportunities and challenges’, *Journal of Universal Computer Science* **4**(4), 330–348. Technical requirements for online assessment. Open standards and flexible architectures are a must, as no single product will be able to

cover all the potential requirements for future assessment activities. Beware “checklist evaluations” in software evaluation just as in student assessment.

- Howland, J. (1997), ‘It’s all in the language: Yet another look at the choice of programming language for teaching computer science’, *Journal of Computing in Small Colleges* **12**(4), 58–74. Factors affecting choice of language for introductory programming courses – author barracks for Scheme.
- Hsi, S. & Soloway, E. (n.d.), ‘Learner-centred design: Addressing, finally, the unique needs of learners’, *SIGCHI workshop*. Advances in computing speed have led us to a situation in which we can design software not only to be usable, but to be *learnable*. Workshop.
- Jacobson, N. (2000), ‘Using on-computer exams to ensure beginning students’ programming competency’, *SIGCSE 2000* **32**(4), 53–56.
- Jarc, D. J. (n.d.), ‘Ada, Pascal’s replacement for introductory courses in computer science’, pp. 126–134. Photocopy obtained from Linda McIver.
- Jones, E. L. & Allen, C. S. (2003), ‘Repositories for CS courses – an evolutionary tale’, *ITiCSE* pp. 119–123.
- Kane, M. T. (2001), ‘Current concerns in validity theory’, *Journal of Educational Measurement* **38**(4), 319–342.
- Kay, D. G. (1996), ‘Bandwagons considered harmful, or The past as prologue in curriculum change’, *SIGCSE Bulletin* **28**(4), 55–58. A plea for moderate conservatism in the introductory programming curriculum. Trends for XYZ-Early can bring XYZ into the course before there is any pedagogical justification for doing so, possibly at the cost of having to sacrifice essential concepts that are not central to ideology. (kay-bandwagons.pdf).
- Kay, D. G. (1998), ‘Large introductory computer science courses: Strategies for effective course management’, *SIGCSE* pp. 131–134. Challenges imposed by large enrolments and strategies for coping with them. Challenges include dissociation and hence attrition, and the potential for inequity in marking and teaching when the class has to be split up

among different tutors. Strategies cover both in-class and out-of-class methods, and also methods for keeping up good communications with the other teaching staff (tutors/demonstrators) to ensure consistency of marking. (p131-Kay.pdf).

Knox, D. & Woltz, U. (1996), ‘Use of laboratories in computer science education: Guidelines for good practice. Report of the Working Group on Computing Laboratories’, *ITiCSE* pp. 167–181. How to design good prac materials. Includes material on pedagogy and a little on assessment. Important reading for anyone interested in lab-based assessment. (p167-Knox.pdf).

Kuechler, W. L. & Simkin, M. G. (2003), ‘How well do multiple choice tests evaluate student understanding in computer programming classes?’, *Journal of Information Systems Education* **14**(4), 389–399.

Kuittinen, M. & Sajaniemi, J. (2004), ‘Teaching roles of variables in elementary programming courses’, *ITiCSE 2004* pp. 57–61.

Kushan, S. B. (1994), ‘Preparing programming teachers’, *SIGCSE 1994* pp. 248–252.

Lister, R. (2001), ‘Objectives and objective assessment in CS1’, *SIGCSE* pp. 292–. Begins with a plea for better understanding and explication of objectives prior to assessment, and then segues to a description of MCQs Done Right. Does not address the “prior problem”, which is that a student who knows nothing at all can expect to get 25% on a MCQ exam, leading to mark inflation. Interesting idea: CS1 cannot lead to proficient programmers and does not do so anywhere, so why make them write code at all?

Lorenzen, T. (1981), ‘The case for in class programming tests’, *SIGCSE* **13**(3), 35–37. A very early plea for class-marked pracs and prac exams to cut down on the practice of “collaborative development” of student assignments while still allowing the development of a cooperative spirit in labs.

Lui, A. K., Kwan, R., Poon, M. & Cheung, Y. H. Y. (2004), ‘Saving weak programming students: applying constructivism in a first programming course’, *SIGCSE Bulletin* **36**(2), 72–76. Using constructivist principles

to design teaching strategies in order to lower attrition and build success in the Open University context, where students are not subject to rigorous entry requirements. Includes a fascinating analysis of the factors that make weak students weak. “Analogy considered harmful”: because all analogies are imperfect, presentation of an analogy may lead some weak students to construct an inaccurate mental model, and weak students are not sufficiently emotionally or intellectually resilient to go through the process of model reconstruction or adaptation. Even technical terms should not be taught until after the relevant concept has been demonstrated enough times for the learner to have modelled it, as jargon can confuse through semantic collisions.

- Lynch, K. & Markham, S. (2003), ‘The winds of change: Students’ comfort level in different learning environments’, *ITiCSE* pp. 70–73.
- Macdonald, I. & Gunn, F. (1997), ‘A model for change in tertiary teaching: Combining education research, tutor training, and reflective practice to create Teaching Communities’, *ASERA* .
- Macdonald, I., Mitchell, I., Gunn, F. & Carbone, A. (1996), Helpless, isolated and underpaid: Turning computer science demonstrators into teachers, *in* ‘Proc. Australasian Science Education Research Association’.
- Maheshwari, P. (1996), Teaching programming paradigms and languages for qualitative learning, *in* ‘Proc. Second Australasian Conference on CS Education’, pp. 32–39.
- Marshall, S. J. & Sonenberg, L. (1996), Computer science 100 and 200 level curriculum evaluation one year on, i.e. “Is the curriculum working?”, Technical Report 96/44, University of Melbourne. A review of the University of Melbourne introductory Computer Science curriculum, including student perceptions but not including success rates etc. Valuable data on the students’ perception of what’s going on in teaching – which should never be confused with the staff’s perception of what’s going on. What actually *is* going on is likely to be different again.
- Mayer, R. E. (1988), *Studying the Novice Programmer*, Lawrence Erlbaum Associates, chapter The Psychology of How Novices Learn Computer Programming, pp. 139–159. Comparison of techniques for teaching programming concepts.

- McCracken, M. (n.d.), ‘Assessment of programming skills of first year CS students: Do they really know how to program?’, Working group proposal for ITiCSE 2001.
- McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B.-D., Laxer, C., Thomas, L., Utting, I. & Wilusz, T. (2002), ‘A multi-national, multi-institutional study of assessment of programming skills of first-year CS students.’, *SIGCSE 2002* pp. 125–140.
- McGettrick, A. & Mansor, N. (1999), ‘Standards and levels: a case study’, *Journal of Assessment and Evaluation in Higher Education* **24**(2), 131–140. Suggests a framework for deciding what standards of ability or competence should be required for qualification at different levels. Context: British higher education system. Touches on the interrelationship between University and the broader community; assessment as quality assurance.
- McInnis, J. R. & Devlin, M. (2002), ‘Assesing learning in Australian universities’, Centre for the Study of Higher Education, Melbourne University. Practical suggestions for assessment design in Australian tertiary education. Covers groupwork, plagiarism, international students, large class assessment, standards, and online assessment. See also <http://www.cshe.unimelb.edu.au/assessinglearning>.
- McKenna, P. & Laycock, B. (2004), ‘Constructivist or instructivist: pedagogical concepts practically applied to a computer learning environment’, *ITiCSE 2004* pp. 166–170.
- McKeown, J. & Farrell, T. (1999), ‘Why we need to develop success in introductory programming courses’, *CCSC Central Plains Conference*. Community, industry and academic needs for skilled graduate programmers will not be met unless and until we start emphasising student development over testing: deemphasising the weeding-out of the unsuccessful students and reinforcing the idea that success is, and should be, attainable to anyone who is really willing to achieve it. (CPCCCSCpaper.htm).
- Meeden, L., Newhall, T., Blank, D. & Kumar, D. (2003), ‘Using departmental surveys to assess computing culture: Quantifying gender differences in the classroom’, *ITiCSE* pp. 188–192.

- Morgan, C. K., Watson, G. K., Roberts, D. W., McKenzie, A. D. & Cochrane, K. W. (2004), ‘Scholarship neglected? How levels are assigned for units of study in Australian courses’, *JAEHE* **29**(3), 283–298.
- Oosterhof, A. (1994), *Classroom applications of educational measurement*, Macmillan.
- Pane, J. F., Ratanamahatana, C. & Myers, B. (2001), ‘Studying the language and structure in non-programmers’ solutions to programming problems’, *Intl. Jnl. of Human-Computer Studies* **54**(2), 237–264. So-called “natural programming”. Possibly not of great utility in the computer industry, but potentially very enlightening to anyone investigating the way people must adapt mentally to programming in the traditional style. Our brains do not seem very well-adapted to the kind of reasoning that we have to do in order to program; good programming languages are not similar to natural human language. This is an attempt to construct a kind of computing creole, with the expressive power of programming language but the internal flexibility of a human language.
- Pane, J. & Myers, B. A. (2000), ‘The influence of the psychology of programming on a language design: Project status report’, *Proc. PPIG* **12**, 193–208. Progress report on the design of a programming environment (it goes beyond language) to match natural human expectations of computer behaviour. Still relevant to my work on assessing skills with traditional programming languages, because their background section points at likely areas of misconception that assessment can target.
- Paxton, M. (2000), ‘A linguistic perspective on multiple choice questioning’, *Journal of Assessment and Evaluation in Higher Education* **25**(2), 109–119. What price discourse in the community of practice? This paper makes the point that if multiple-choice testing is the sole means of assessment, then it will be possible for graduates to emerge fully-qualified without ever having engaged in the discourse of their discipline. The use of MCQs ameliorates some of the disadvantages experienced by non-native speakers of the language of instruction, but at the cost of introducing new ones. Furthermore, it is not clear that the knowledge gained in the study for a MCQ test transfers to real-world situations: correlations between performance on MCQ tests and essays are not as high as one would wish.

- Pears, A. N. & Daniels, M. (2003), 'Structuring CSEd research studies: Connecting the pieces', *ITiCSE* pp. 149–153.
- Pittenger, D. J. (1993), 'The utility of the Myers-Briggs type indicator', *Review of Educational Research* **63**(4), 467–488.
- Postema, M. & Markham, S. (2002), 'Student satisfaction: a method for exploring quality factors within computing educations', *The New Zealand Journal of Applied Computing and Educational Technology* **6**(1), 51–59.
- Powers, K. & Powers, D. T. (99), Making sense of teaching methods in computer education, *in* '29th ASEE/IEEE Frontiers in Education Conference'. Constructivism and all that.
- Ramalingam, V., LaBelle, D. & Wiedenbeck, S. (2004), 'Self-efficacy and mental models in learning to program', *ITiCSE 2004* pp. 171–175.
- Randall, C., Price, B. & Reichgelt, H. (2003), 'Women in computing programs: Does the incredible shrinking pipeline apply to all computing programs?', *ITiCSE*.
- Reges, S. (2003), 'Using undergraduates as teaching assistants at a State University', *SIGCSE 2003* pp. 103–107.
- Rowell, G. H., Perhac, D. G., Hankins, J. A., Parker, B. C., Pettey, C. C. & Iriarte-Gross, J. M. (2003), 'Computer-related gender differences', *SIGCSE 2003* pp. 54–58. Gender-related differences in attitudes to computing and information technology among undergraduates at Middle Tennessee State University. Survey found more females enjoy using computers, although more males enjoyed programming; there were no significant differences in computer confidence or positivity of attitudes toward careers in computing, suggesting that differential gender attrition has wider causes than is generally believed.
- Ruehr, F. & Orr, G. (2002), 'Interactive program demonstration as a form of student program assessment', *Journal of Computing Sciences in Colleges* **18**(2), 65–78.
- Sambell, R. & McDowell, L. (1998), 'The construction of the hidden curriculum: Messages and meanings in the assessment of student learning',

Journal of Assessment and Evaluation in Higher Education **23**(4), 391–402. Case studies of hidden curriculum effects in “innovative assessment” programs.

- Sanders, I. & Mueller, C. (2000), ‘A fundamentals-based curriculum for first year computer science’, *SIGCSE* pp. 227–231. Curriculum design to overcome incorrect expectations of CS1 — many students equate it with programming or even basic computing proficiency, and end up demotivating themselves and their peers. Approached by adding more theory at early stages of the course. Context of post-apartheid South Africa, so some racial equity considerations prevail as well (white students are usually more affluent and therefore more likely to have had some programming experience at home).
- Sheard, J. & Dick, M. (2003), ‘Influences on cheating practice of graduate students in IT courses: what are the factors?’, *ITiCSE* pp. 45–49.
- Sheard, J., Dick, M. & Markham, S. (2002), ‘Cheating and plagiarism: perceptions and practices of first year IT students’, *ITiCSE 2002* pp. 183–187. An attempt to measure the incidence of cheating among IT undergraduates at Monash and Swinburne. Exam cheating is both frowned upon and rare, but less-serious breaches of plagiarism policy are very common (85.4% of Monash students!) despite the students’ knowledge of the code of conduct. Furthermore, few students would be willing to do in a cheat if they spotted one.
- Shiel, B. (1981), ‘The psychological study of programming’, *Computing Surveys* **13**(1), 101–120.
- Snyder, B. R. (1973), *The Hidden Curriculum*, MIT Press. Institutional structure affects how students approach learning, often in unintended ways. Gippsland 378 SNY.
- Stegink, G., Pater, J. & Vroon, D. (1999), ‘Computer science and general education: Java, graphics and the Web’, *SIGCSE* pp. 146–149. Case study: introductory programming with JaDE (Java Development Environment), web-based courseware instead of book. Students seem to enjoy the course and it attracts women to CS. No numbers, alas. (p146-Stegink.pdf).

- Stein, L. A. (1998), ‘What we swept under the rug: Radically rethinking CS1’, *Computer Science Education* **8**(2), 118–129. Design for a first-year programming course focussing on software as collections of interacting threads. Java with parallelism early; first-years coming to grips with designing distributed systems. Apparently they can do that over at MIT. (rug.html).
- Struyven, K., Dochy, F. & Janssens, S. (2005), ‘Students’ perceptions about evaluation and assessment in higher education: A review’, *JAEHE* **30**(4), 325–341.
- Tam, M. (2000), ‘Constructivism, instructional design and technology: Implications for transforming distance learning’, *Educational Technology and Society* **3**(2), 50–60. A misconstrual of constructivism: I disagree that constructivist educational design is necessarily devoid of educational goals and objectives. Furthermore, you can be an educational constructivist without being an ontological relativist. Constructivism and objectivism are not mutually exclusive in my experience. After all, if you completely deny the “object under study”, you’re not really left with a basis on which to construct anything.
- Thomas, L., Ratcliffe, M., Woodbury, J. & Jarman, E. (2002), ‘Learning styles and performance in the introductory programming sequence’, *SIGCSE 2002* pp. 33–37.
- Thomas, P. (2003), ‘The evaluation of electronic marking of examinations’, *ITiCSE* pp. 50–54.
- Thomson, K. & Falchikov, N. (1998), “‘Full on until the sun comes out’: the effects of assessment on student approaches to study”, *Journal of Assessment and Evaluation in Higher Education* **23**(4), 379–390. Tight deadlines mean shallow learning even if they are due to poor time management on the part of the students.
- Trakhtenbrot, M. (2003), ‘Analysis of typical misconceptions in a theoretical CS course, and how to address them in e-learning’, *ITiCSE* p. 241.
- Trochim, W. (n.d.), ‘An introduction to concept mapping for planning and evaluation’, Web page.

- Waller, W. A. (1994), ‘A framework for CS1 and CS2 laboratories’, *SIGCSE 1994* pp. 198–202.
- Werner, L. L., Hanks, B., McDowell, C., Bullock, H. & Fernald, J. (2005), ‘Want to increase retention of your female students?’, *Computing Research News* **17**(2).
- White, G. (1993), ‘Standardized mathematics scores as a prerequisite for a first programming course’, *Mathematics and Computer Education* **37**(1), 96–104.
- Wiggins, G. (1998), *Educative assessment: designing assessments to inform and improve student performance*, Jossey-Bass.
- Wills, C. E., Deremer, D., McCauley, R. A. & Null, L. (1999), ‘Studying the use of peer learning in the introductory computer science curriculum’, *Computer Science Education* **9**(2), 71–88. Report on an intercampus project on introducing peer learning at first- and second-year level CS courses. Includes an overview of potential benefits and risks.
- Wilson, B. C. & Shrock, S. (2001), ‘Contributing to success in an introductory computer science course: A study of twelve factors’, *SIGCSE* pp. 184–188. A study of factors influencing final mark in CS1. Gender, previous programming experience, previous internet experience, previous computer gaming experience, previous productivity software experience, attribution of success to luck, self-efficacy, comfort level, encouragement from others, work style preference, mathematical background. Comfort level, mathematical background, and attribution to luck were found to be predictive. (p164-Wilson.pdf).
- Woit, D. & Mason, D. (2003), ‘Effectiveness of online assessment’, *SIGCSE 2003* pp. 137–141. Auto-marked prac exams as a means of ensuring programming competence in practical programming-oriented courses at CS2 level. Compares different combinations of marked and voluntary labs, with and without a mid-semester test. Attention needs to be paid to motivational factors when the weekly assignments are voluntary!
- Yokomoto, C. F. (1995), ‘What pre-exam and post-exam quizzes can tell us about test construction’, *ASEE/IEEE Frontiers In Education* .

Zachary, J. L. (1994), 'Tutorial-based teaching of introductory programming courses', *SIGCSE 1994* pp. 136–140.