Department of Computer Science,
University of Otago

Te Whare Wānanga o Otāgo

Technical Report OUCS-2001-07

Identifying the Danger Zones: Predictors of Success and Failure in a CS1 Course

Authors:

Nathan Rountree, Janet Rountree, and Anthony Robins

Status: submitted for review to *Inroads---the SIGCSE Bulletin*

Department of Computer Science,
University of Otago, PO Box 56, Dunedin, Otago, New Zealand

http://www.cs.otago.ac.nz/trseries/

Identifying the Danger Zones: Predictors of Success and Failure in a CS1 Course.

Nathan Rountree, Janet Rountree, and Anthony Robins

Department of Computer Science

University of Otago

{rountree,janet,anthony}@cs.otago.ac.nz

Abstract

We present the results of a survey which focuses on the backgrounds and expectations of a group of CS1 students in the first weeks of semester. When comparing their survey answers to their final grades on the course, we saw some surprising things: the group which indicated an intention to continue in computer science did no better than any other, and the strongest single indicator of success seems to be "expecting to get an A from the course."

In order to see if there were any particular combinations of answers that indicated success or failure, we ran a decision-tree classifier over the survey data. This resulted in the identification of differing "danger zones" for each year of study, in which we observed about twice the expected proportion of failing students.

1. Introduction

In the past decade there has been a rapid increase in the numbers of students enrolling in introductory programming papers. The CS1 paper at the University of Otago has expanded from just under 2% of 13000 students in 1995 to 3% of 16000 students (including summer school) in 2001. As a result, we now teach students whose educational background, reasons for taking the course, and levels of motivation are more widely varied.

COMP103 is the first year programming course offered by the Department of Computer Science at the University of Otago. A pass in COMP103 is mandatory for students intending to continue studies in either the Department of Computer Science or the Department of Information Science. COMP103 is also offered as an optional paper to non-computing majors.

In 2000 we changed our introductory programming language from Pascal to Java. Aside from the discussion this created about choice of language and what should be taught in the introductory curriculum, we started to wonder if there were any factors that could predict the pass/fail success of these students. For instance, did any previous course of study, or their level of enthusiasm for the taking the course instil a skill or attitude that could be seen as a factor in their success?

We held some expectation that students with a strong Math background would achieve higher grades than their peers as they would already have experience with computing concepts (such as functions and algorithms), and would already be in the habit of participating in weekly laboratory assignments (since COMP103 is held in the second semester, math students have already had one semester to get used to these conditions). We also expected that students who reported a higher level of desire to take COMP103 would achieve higher grades, due to their willingness to put in the necessary time and effort.

2. Background

There is a wide variety of studies concerned with the improvement of computer science education. For CS1, there are still questions about *what* to teach—what paradigm [7], which language [11], and which features of particular languages [18] are essential components of the curriculum? General issues include how to evaluate and measure programming skills [8], how to deal with plagiarism [9], and how to teach problem-solving skills and patterns [13]. Studies have also looked at the difficulties and consequences of the popular change in first language choice from procedural to object-oriented [2]. Initial interests for CS1 courses that use Java have tended to focus on the practicalities of teaching and resources available for establishing the course [1, 4, 14].

For some time, there have been studies of novice programmers which focus on the psychology of learning to program. These attempt to identify and describe the effect of cognitive style and personality, the ways in which expert and novice knowledge differs, and how novices construct mental models of problems and programs [3, 6, 15].

To our knowledge there are no published studies which investigate possible predictors of success and failure for novice students undertaking an introductory programming course. The study most closely related to our work was undertaken at IBM in New York [12], and looks at what factors may contribute to successful object-oriented learning. The authors studied a one week long Smalltalk course, followed by a smaller class using C++.

An important difference between the IBM study and our work is that our participants are absolute novices to programming (with only a few exceptions), whereas most of the IBM subjects were involved in software development as a profession. In addition to background questionnaires, the IBM instructors provided a subjective rating of the ability of their students at the end of the course, whereas the success of our participants is rated against their final course mark. Factors positively affecting the success of students in the IBM course include: greater experience of programming, recent participation in writing code, knowing more languages, and prior knowledge of an object-oriented, procedural, or functional language. Students who seemed to find object-oriented learning difficult included those who were viewed as dogmatic in their approach, or who spent excessive hours completing assigned exercises.

3. The Study

Data was collected during the second semester of 2000 and again in 2001. In the first week of laboratory sessions, Students were asked to complete an optional online survey which mainly consisted of multi-choice questions. Replies were collected via email. For detailed information about the content of COMP103, see [16].

Information was requested regarding:

- status: gender, age, enrolment status (part or full-time), year of study at university, intended major, how keen they were to take COMP103;
- background: what recent mathematics courses they had taken, whether they felt their strongest background was in humanities, science, or commerce subjects, whether they knew any programming language(s) already;
- expectations: how difficult they anticipated the course would be, what they expected of the workload, what grade they expected to achieve, whether they intended to enrol in second-year computer science courses.

Any replies which were duplicates or did not have a valid student ID number were excluded. In addition, replies have only been included from students who submitted their second assessed exercise and/or who attended the final examination for the course. This decision was made in order to exclude any early withdrawals from COMP103, but to include those students who made an early commitment to the course and withdrew later. The total number of valid replies for 2000/2001 totalled 472, out of a possible 748.

At the end of the semester, each student's reply was matched with their final mark out of 100. This mark reflects 30% worth of bi-weekly programming assignments, a single 20% mid-semester examination, and a 50% final examination. Students who answered the survey and completed the second laboratory exercise but were absent from the final exam have been counted as fails.

4. Results

In COMP103 a student who receives a mark of 50% or greater achieves a pass. However, we generally find that students who achieve a mark of 70% or greater are likely to be more successful in subsequent computing papers, so we also report results in this range. From the 472 valid survey answers, 73% of the students received a passing grade, while 45% of the total managed to score over 70%.

Table 1 compares the passing and over 70 proportions of students who chose to answer the survey with those who did not:

Table 1: χ^2-test result for "answered survey" vs. "did not answer"

	Answered	Did not Answer	χ^2	df	p
Pass	73%	55%	26.02	1	<0.001
Over 70	45%	29%	19.45	1	<0.001

We conclude that the group of students willing to participate in the survey is more likely to do well in COMP103.

Since all questions on the survey were multi-choice, each of them was subjected to chi-square analysis to determine if deviations from the expected pass rate or 70+ rate were statistically significant. With an increase in χ^2-value, the probability of the observed classes being independent of the explanatory variables diminishes. Table 2 summarises a comparison of the χ^2 values for each question in the survey, with respect to their pass rate and over 70 rate. Starred p-values are significant at 95% confidence and double starred values at 99% confidence.

Table 2: Summary of χ^2-test results on each survey question

Question	**Categories**	χ^2_{pass}	p_{pass}	χ^2_{70+}	p_{70+}
gender	male, female	0.04	0.85	0.03	0.87
age	16–18, 19–21, 22-24, 25+	14.74	0.002**	6.6	0.086
full-time status	part-time, full-time	0.01	0.93	0.56	0.46
year	1st, 2nd, 3rd, 4th+	14.22	0.003**	8.36	0.039*
major	comp sci, info sci, comp & info sci, other	7.06	0.070	14.50	0.002**
keeness	extremely, fairly, neutral, not	13.26	0.004**	19.15	0.001**
recent math	school, uni, other, none	15.21	0.002**	11.54	0.009**
background	humanities, science, commerce	11.68	0.003**	35.77	0.001**
know other language(s)	no, yes	8.04	0.005**	13.65	0.001**
expected difficulty	easier (than my other papers), the same, harder, unsure	9.51	0.023*	25.14	0.001**
expected workload	less (than my other papers), the same, more, unsure	7.87	0.049*	7.69	0.053*
expected success	A grade, B grade, C grade, unsure/prefer not to say	24.27	0.001**	44.51	0.001**
continuing in comp sci	yes, no, unsure	2.53	0.29	5.23	0.073

There are some interesting results here:

- Pass rate and 70+ rate seem independent of gender, of whether a student is full-time or part-time, and of whether a student is intending to continue in computer science. The last was quite surprising, since we expected that the intention to continue would correlate strongly with success.
- The 70+ rate seems to be independent of age range, but the pass rate is not.
- The pass rate seems to be independent of intended major, but the 70+ rate is not.

Table 3 summarises those questions where certain responses deviated from the expected pass rate or 70+ rate. For each possible answer in each multi-choice question, we report the percentage of students who chose that answer and passed, and the proportion that got over 70%. To give a sense of the prevalence of particular answers, the percentage of the class who chose each answer is reported in the final column. Results of a question are only included if the corresponding p-value from Table 2 is below 5%.

Table 3: Pass-rate and 70+ rate for each survey question ($p < 0.05$ only)

		Passed (%)	Over 70 (%)	Group Size (%)
Entire sample:		73	45	100
Age:	16–18	80		36
	19–21	72		40
	22–24	54		11
	25+	74		12
Year of study:	first year	79	48	63
	second year	62	33	19
	third year	61	39	9
	fourth year plus	72	53	9
Intended major:	computer science		55	39
	information science		35	31
	comp & info sci		39	8
	other		42	22
Keenness:	extremely keen	85	62	23
	fairly keen	69	42	54
	neutral	78	38	16
	not keen	60	27	6
Recent mathematics:	school	68	36	16
	university	78	50	61
	other	81	52	6
	none	58	32	17
Background:	humanities	61	28	18
	science	79	57	56
	commerce	70	31	25
Know a language:	no	71	41	82
	yes	86	63	18
Difficulty:	easier than my other papers	80	69	10
	about the same	81	54	32
	harder than my other papers	68	35	49
	undecided	69	42	10
Workload:	less than my other papers	64	46	6
	about the same	79	52	42
	more than my other papers	71	39	47
	undecided	58	42	5
Expected grade:	A grade	90	70	26
	B grade	69	37	47
	C grade	63	24	8
	not sure/prefer not to say	64	38	18

5. Discussion

Age:

Two things are surprising here: the unusually low pass rate of the 22–24 group, and the unusually high pass rate for 16–18. This overturned an expectation that more mature students tend to do better at computer science papers. It is possible that the 22–24 group is just old enough with respect to the New Zealand school system not to have had sufficient prior computing experience; however we consider this unlikely since a basic computing paper (COMP101) is a prerequisite to COMP103.

Year of study:

The results associated with this question were also interesting; clearly students in their second or third year of study at university do less well in COMP103 than students in their first or fourth year. Since COMP103 is a prerequisite for continuing in computer science or information science, we can reasonably expect that second-year or third-year COMP103 students who list computer or information science as their only major are either "re-starting" their degree or switching majors. Of the 64 students in this situation, 38% failed. Non-computing majors in their second or third year are presumably taking COMP103 to enhance their computing skills or to "fill-in" their points in order to complete a degree; there were 37 students in this situation and 47% of them failed. Although we would need to look at individual student records to confirm this trend, we feel that this indicates that treating an introductory programming paper as a "filler" may not be a good idea.

Intended major:

Note that the pass rate has too high a p-value to be statistically significant. There is however a 20 percentage point difference between the proportion of information science students achieving over 70% and the proportion of computer science students. We should point out that although COMP103 is a prerequisite for continuing in both departments, there is very little material in the course that is specific to information science.

Keenness:

As we expected, that group of people who considered themselves to be extremely keen to participate in COMP103 had a much higher pass rate than those who were not. Less expected was the difference in pass rate between those who were "fairly keen" (69%) as opposed to those who were "neutral" (78%). It would seem as though being willing to "go all out" leads to success, but being just slightly reserved in your enthusiasm (enough to tick "fairly keen" rather than "extremely keen") is worse than reporting no feeling at all.

Recent mathematics:

Although COMP103 has no specifically mathematical content found above year 10 in high-school, it does seem as though ability in programming is suggested by ability in math. However, the disparity between school math and university math is interesting. Perhaps university students have simply had more practice at working mathematically, or perhaps the type of student who is inclined to take a math course at university is more inclined to do well in COMP103. We suspect that it is the work habits instilled by university math courses (small weekly assignments and laboratory sessions) that provide the advantage rather than simply the mathematical skills learned in the course.

Background:

Both pass rate and number of students achieving over 70% are lower for students who consider themselves to have a humanities background. This was of some concern, considering our effort to run a course that did not rely on students' level of mathematical skill. We suspect that there is a large difference in the style of COMP103 from what humanities students are used to: 24 programming assignments over 13 weeks rather than (for example) 3 essays, and a strong sense of "right answer" and "wrong answer" (a program either produces the correct output or it doesn't) rather than "better answer" and "less good answer". As anyone who has

programmed is aware, the rewards of the task and the frustrations inherent in it are strongly polarised; thus there is a very different sense of what you have achieved and when you have achieved it. This may lead some students to an incorrect assessment of what they need to do to be successful in the course.

Knowing a programming language already:

Unsurprisingly, the 18% of students who claimed to already know a programming language had a much higher success rate. However it is interesting to note that 12 of those 84 students still failed the course: we conclude that knowing a programming language is no guarantee of success in an introductory programming course. We also note that where students listed the programming languages they knew, we saw everything from "C, C++, Forth, M68k assembler" to "macros in MS Office."

Expected difficulty, Expected workload:

The last three questions all indicate that students are remarkably good at making their own assessment of the challenges facing them and how well they are likely to react to them. In general, students who thought the course would be more difficult for them did less well than those who thought it would be about the same as their other courses. This leads us to believe that it is a mistake for students to overestimate or underestimate the course; it is probably of similar difficulty to other first-year courses but has a differing mode of execution.

Expected grade:

This is clearly the strongest single indicator of success: students who expect to get an A and are willing to say so are far more likely to be successful. This suggests to us that a positive attitude is more important than having the right background, and that students are fairly good at estimating their own ability.

6. Danger Zones and Success Zones

We thought it would be interesting to determine if there was any combination of answers which would suggest that a student was more likely to fail COMP103, or more likely to get over 70%. To this end, we ran a decision tree inducer over the survey data, first with pass/fail and then with over 70/under 70 as class labels. The classification software was based on IBM's SPRINT algorithm [17], with the Minimum Cost Complexity pruning method from CART [5]. Paths from the root of the tree to "fail" leaves can be seen as answers which put a student into a "danger zone" where the likelihood of failing the course is about twice as high as usual.

Figure 1 shows an example tree induced on the group of students in fourth year or above. The left path of each decision node is followed if the student checked that answer, the right path if not. The values at leaf nodes indicate the predominant class at that leaf, the percentage of examples at the node that are in fact that class, and the number of examples at the leaf.

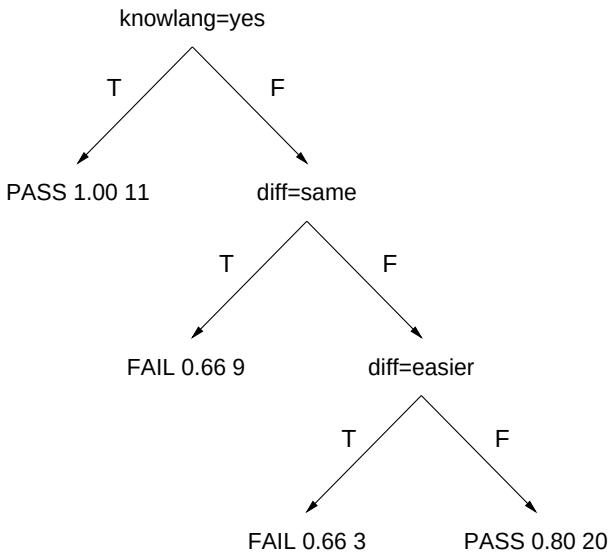

Figure 1: A decision-tree showing clusters of high fail-rate above fourth-year.

The danger zones identified by the decision tree are clearer if taken by year of enrolment:

First year:

Not looking for an A and
Not majoring in cosc or info and
Not “extremely keen” or “neutral”
Group size: 38, Pass: 53%, Fail: 47%, Over 70: 26%, Under 70: 74%

Second year:

Not looking for an A and
No recent university or “other” math
Group size: 29, Pass: 38%, Fail: 62%, Over 70: 17%, Under 70: 83%

Third year:

Background not science
Group size: 20, Pass: 40%, Fail: 60%, Over 70: 15%, Under 70: 85%

Fourth year:

Don’t know a programming language and
Think difficulty will be same or easier
Group size: 12, Pass: 33%, Fail: 67%, Over 70: 17%, Under 70: 83%

Taken together, the “danger zone” has 99 student examples, of whom 56 failed the course (56.5% instead of the expected 27%).

The first-year danger-zone students are characterised by a much higher uncertainty about whether they ever wish to do more computer science (50% instead of 35%, suggesting that the intention to proceed in computer

science may be significant for this group, if not for second-years and above). While some students change their minds about half way through the course as they discover they really enjoy programming, those who still don't intend to continue are unlikely to put in the time and effort required to pass the course.

The second-year danger zone suggests that a transition into COMP103 from another program is more difficult if the student has not experienced a course in the manner of university mathematics. Further, if the student is not committed to getting an A in the course, the chances of failing seem to be higher than the chances of passing.

Third-year students may be changing major or program, or taking COMP103 to fill in points towards their current program. Without a background in science (by third-year, we can assume this means university science papers) more students in this category will fail rather than pass.

Finally, fourth-year students who do not already know how to program and assume they will find this course no harder than anything else they have done are making a tactical error. Previously we argued that COMP103 shouldn't be seen as more difficult than other courses, but this may be a dangerous point of view for fourth-year students.

The one clear success zone uncovered by the decision-tree software agrees with the contingency tables, taking the two most significant attributes and combining them:

Success zone:

Expect an A and
Background is science
Group size: 84, Pass: 88%, Fail: 12%, Over 70: 76%, Under 70: 24%

An even better success zone is seen if we restrict the zone above to students between the age of 16 and 18:

Success zone:

Expect an A and
Background is science and
Age is 16 to 18
Group size: 31, Pass: 100%, Fail: 0%, Over 70: 90%, Under 70: 10%

7. Limitations

Our decisions about what constitutes a "pass" or a "good" student are tied to a specific course at a specific university with students primarily educated in New Zealand schools. However the COMP103 course is based on a popular, standard textbook [10] and our graduates have had notable success in the international job market, in postgraduate work, and in the ACM Programming Competition. Hence we believe that the trends we see here regarding attitude, self-assessment, and educational culture are generalisable to a wider community than our own.

Our course has evolved in response to student reaction over the last two years, so some patterns seen in the 2000 intake may not be present in the 2001 course and vice versa. Similarly, the population has changed somewhat: the first time COMP103 was taught with Java generated more interest from senior undergraduate and postgraduate students than we would normally expect. However we note that the patterns presented in this paper are only those which can be validated against *both* semesters surveys, so we have some confidence that we may see them again in coming years.

Making the survey voluntary resulted in a self-selected sample that consisted of more than half the class, but also a clearly more successful group. However, we felt it was more important to have students answering the questions honestly rather than feeling coerced into providing information that they may not wish to share. We are making an assumption that the majority of students who answered the survey did so honestly, since there seemed to be genuine interest in providing the department with helpful information.

8. Conclusion and Future Work

We have presented the results of a survey taken over two semesters of an introductory programming course. The purpose was to see if there were factors independent of students' previous academic performance that influenced their success in the paper. We found that the strongest single indicator of success was the grade the student *expected* to achieve at the beginning of the course. The questions on expected grade, anticipated difficulty, and anticipated workload indicate that students have a strong sense of how well they are likely to do within the first two weeks of the semester. Other factors that are related to success include whether the student thinks his/her background is science, commerce, or humanities; whether they have recent university math experience; and what year of study he/she is in—but not always displaying the relationships we might expect.

To identify clusters of failing students and students who achieve over 70%, we ran a decision-tree classifier over the survey data, resulting in simple rules to describe "danger-zones" for each year of study and a single obvious success zone. Although the danger zone exemplars include 21% of the survey answers, they account for nearly half of all the failing students. Danger zone students have about twice the failure rate of the whole sample. The single strongest combination of answers for predicting a grade over 70% was expecting an A and having a "science" background; when combined with the raw survey results, we conclude that attitude and type of work habits are the strongest predictors of success that we can see without accessing a student's academic record.

Our results suggest several ways in which we might improve COMP103 before we even begin to consider curriculum or teaching methodology. Students should be told at the beginning of the course that they are remarkably good at assessing their own ability—that what they suppose they will get in this course will correlate strongly with what they will actually get. Coming in with the right attitude seems to be very important not only for doing well, but even for simply passing the course.

Students should be given a clear message that taking introductory programming as a "filler" course is potentially a tactical error. We believe that the only good reason to take this course is because you want to learn how to program, and that the chances of failing the course increase dramatically when this is not true.

We need to be very precise at the beginning of the course about why COMP103 is likely to be different to other courses the participants may have done. It seems that the time commitment to laboratory sessions and programming assignments, the grappling with programming concepts, and the sharp distinction between an assignment that works and one that doesn't can come as an unpleasant surprise to some students.

We have begun work on a formal prediction model based on the survey data that uses the "danger zone" decision trees to aid in model selection, and logistic regression to estimate the importance of each independent variable. Initial results suggest that we should be able to achieve a prediction accuracy of at least 83% (a 10 percentage point improvement on a naive classifier).

We are working concurrently on the issue of distinguishing problems associated with program comprehension as opposed to program creation. We intend to process the "danger zone" students' examination results to try to see if there are specific "sticking points" in the curriculum, and to try to see if these are related more strongly to program comprehension or creation.

Finally, we need to make it clear to our students that to succeed in learning to program, you need to be *striving* to get an A. Author Terry Pratchett has commented several times that really successful writers "...have to *want* to write. Too many people want to *have written*." We believe that there is a similar problem in learning to program: many people would like to have the skill, but find the mental attitude required to gain it is hard to sustain. Our results suggest that a positive attitude is the most important factor.

Acknowledgments

This work has been supported by internal University of Otago Research into Teaching grants. We are also grateful for the support of the other members of the COMP103 teaching team, especially Sandy Garner and Mike Atkinson.

References

[1] Andreae, P., Biddle, R., Dobbie, G., Gale, A., Miller, L., and Tempero, E. Experience Teaching CS1 with Java, *Journal of Computer Science Education*, 14, 1&2, 2000, 19–28.

[2] Biddle, R. and Tempero, E. Java Pitfalls for Beginners, *SIGCSE Bulletin*, 30 (2), 1998, 48–52.

[3] Bishop-Clark, C. Cognitive Style, Personality, and Computer Programming, *Computers in Human Behaviour*, 11 (2), 1995, 241–260.

[4] Boszormenyi, L. Why Java is Not My Favorite First-Course Language, *Software – Concepts and Tools*, 19 (3), 1998, 141–145.

[5] Brieman, L., Friedman, J., Olshen, R., and Stone, C. *Classification and Regression Trees*. Wadsworth, 1984.

[6] Burkhardt, J., Detienne, F., and Wiedenbeck, S. Mental Representations Constructed by Experts and Novices in Object-Oriented Program Comprehension, *INTERACT' 97*, 1997, 339–346.

[7] Deek, F., Kimmel, H., and McHugh, J. Pedagogical Changes in the Delivery of the First-Course in Computer Science: Problem Solving, then Programming, *Journal of Engineering Education* , 87 (3), 1998, 313–320.

[8] Dunsmore, A., and Roper, M. A Comparative Evaluation of Program Comprehension Measures, Technical Report EFoCS-35-2000, 2000, University of Strathclyde.

[9] Joy, M., and Luck, M. Plagiarism in Programming Assignments, *IEEE Transactions on Education*, 42 (2), 1999, 129–133.

[10] Koffman, E., and Wolz, U. *Problem Solving with Java*, 1999, Addison-Wesley.

[11] Kolling, M. The Problem of Teaching Object-Oriented Programming, Part 1: Language, *Journal of Object-Oriented Programming*, 11 (8), 1999, 8–15.

[12] Liu, C., Goetze, S., and Glynn, B. What Contributes to Successful Object-Oriented Learning, *OOPSLA '92*, 1992, 77–86.

[13] Reed, D. Incorporating Problem-Solving Patterns in CS1, *SIGCSE Bulletin*, 30 (1), 1998, 6–9.

[14] Reges, S. Conservatatively Radical Java in CS1, *Proceedings of the Thirty-First SIGCSE Technical Symposium on Computer Science Education*, 2000, 85–89.

[15] Rist, R. Program Structure and Design. *Cognitive Science*, 19, 1995, 507–562.

[16] Robins, A., Rountree, J., and Rountree, N. *My Program is Correct but it Doesn't Run: a Review of Novice Programming and a Study of an Introductory Programming Paper*. Technical Report OUCS-2001-06, 2001, University of Otago.

[17] Shafer, J., Agrawal, R., and Mehta, M. SPRINT: A Scalable Parallel Classifier for Data Mining, *Proceedings of the 22nd VLDB Conference*, 1996, 544–555

[18] Stephenson, C., and West, T. Language Choice and Key Concepts in Introductory Computer Science Courses, *Journal of Research on Computing in Education*, 31 (1), 1998, 80–95.