Students' Alternative Standards for Correctness

Yifat Ben-David Kolikant
The Hebrew University of Jerusalem
Mount Scopus, Jerusalem
Israel, 91905
972-588-2056
yifatbdk@mscc.huji.ac.il

ABSTRACT
We examined students' definition of correctness as reflected by their decisions whether certain programs are correct. Using a questionnaire we found that students understand correctness as a relative property of the program and therefore might decide that a program is correct even when they evidence its incorrect behavior. We also found that students' definitions of systematic testing are inherently different from that of professionals, yet are consistent with their tolerance to errors.

Categories and Subject Descriptors
K.3.2 [**Computer and Information Science Education**]

General Terms
Human Factors, Verification.

Keywords
Correctness, conceptions, norms, practices.

1. INTRODUCTION
In this work we examine the students' standards for correctness as well as systematic testing. Our hypothesis is that students have different standards for good quality programs as well as for good work methods that are naturally correlated. This hypothesis is based on results from previous work where we investigated the classroom norms regarding algorithmic problems [2]. We found that computer science (CS) instructors deal with students whose experience in technology (mostly as users) influence their epistemology, learning trajectories, and hence their performances as programmers.

Specifically, students have different computer-science norms that govern their programming activities. For example, they are tolerant to errors, which are perceived as unavoidable part of the programming reality. Furthermore, thorough testing translates to execute your program for many non-systematically chosen input examples and hope for luck.

ICER'05, October 1–2, 2005, Seattle, Washington, USA.

Students' satisfaction with incorrect programs implies that they understand correctness in a different way than professionals. Professionals' definition for correctness is dichotomous, that is given a program and its goals if the program fulfills its goals for every legal input than it is correct (and of course it is incorrect otherwise). In contrast, we believed that students' understand correctness as relative. From that viewpoint, programs with incorrect I/O behavior can be relatively correct. Moreover, being tolerant to errors, these relatively correct programs might be considered as correct.

In order to examine our hypothesis we composed a questionnaire to which students at both high-school and college level responded. The questionnaire consisted of two parts. In part A, we inquired about work habits regarding testing and perceptions on these work habits. In part B, we described three incorrect programs and asked the students to decide whether these programs were correct, incorrect, and relatively correct.

Most students demonstrated confusion regarding correctness. Furthermore, they ascribe to correctness as a relative feature of programs. Moreover, in general relatively correct programs were also considered as correct in contrast to professionals whose conceptualization of correctness is dichotomist. The students' alternative conceptions on correctness settle well with their inadequate work habits regarding testing, their tolerance to errors, and their (faulty) belief that their testing is indeed systematic

2. BACKGROUND
2.1 The Professional Definition of Correctness
Joni and Soloway [7] bring the following definition for working programs: "a program is a *working program* if it exhibits correct I/O behavior for all input in the domain of the problems space." (p. 96) Yet, these programs might suffer from poor quality, that is, inefficient, illegible, non-modular, and not documented. Fleury [5] found that experts' goals when debugging is to "get the programs to work on all conceivable sets of data that a user might provide (p. 367)." Figure 1 summarizes the academic definition of correct programs.

> Program of a good quality = working program AND elegant program
>
> Working program = exhibits correct I/O behavior for **all** legal input
>
> Elegant program = efficient, legible, documented, and modular

Figure 1: The professional definition of correctness

The professional definition is settled with the rigorous methods for correctness verification used by professionals. Work done both in the academia and the industry is guided by long-term goals of research and design and therefore becoming professionals involves adopting norms of rigor as well as the ability and the propensity to invest efforts in tasks beyond making the program to work, such as careful testing of their products, analysis of errors, design of solutions, consideration of efficiency, and documentation.

2.2 Students' Values of and Methods for Testing

Students' inadequate work habits have been described by many researchers. For example, Edwards [4] complains that introductory computer science students rely on a trial and error approach for too long to fix errors. Furthermore, they tend to demonstrate careless attitude toward the quality of their programs. They conclude on correctness from executing a program merely once or twice, solely observing that the output shows no straightforward irregularity, or even worse, by compiling the program successfully. Furthermore, when an error is shown in the output, students merely try to switch around a few things in order to make it work, and finally, some students' work is oriented solely toward the goal of making the program give the correct answers to the instructor's input example.

In addition, Leventhal *et al*.[8] found that software testers exhibit positive test bias, that is, they have the tendency to test a hypothesis with data which is consistent with the hypothesis, rather than testing with data which is inconsistent with the hypothesis. However, advanced programmers performed better than beginners.

Inadequate work habits for testing and verification were also found by Iftikhar [6]. Moreover, Iftikhar found that students believed that these methods were systematic. Similarly Scott *et al* [10] found that university students (a) did not test their systems using the same tests as industry does, (b) the majority of students did not have the level of understanding required, and (c) students do not place the same value on software testing as industry does, and therefore, they are unlikely to test their systems with the same rigor that industry does. Finally, while students may have felt that their skills were aligned with industry, industry thought that their understanding was inferior.

Fleury [5] found that (college) students' viewpoint on programming is different than that of experts. Students' tend to avoid complexity, whereas experts, who have realized that avoidance of complexity is impractical, appreciate programs in which complexity is manageable. Consequently, students' definition of *easy programs* (to read, to modify, and so forth) are different than that of experts. Consequently, she recommended that instructors should be aware of students' tendencies, should provide specifics when demanding "easy-to-read" or "easy-to-modify" programs, and should provide learning opportunities for students that will lead the students to realize that avoiding complexity is impossible.

In previous work we explored classroom CS-norms [2]. We asked students to develop a program that calculates the numbers of $20 and $50 bills an ATM should give for a given amount. Most students were not able to solve the problem correctly. Most of them implemented an algorithm that divided the amount by 50 to calculate the number of $50 bills required, and then divided what was left by 20 to determine how many $20 bills were required; these algorithms did not handle correctly cases such as 80 and 160 because that would result with a reminder of 10 or 30.

Yet, most of the students considered their programs to be "mostly correct." They defended their approach by explaining that errorful programs are simply unavoidable. The following quotes are taken from that conversation.

Student: If you work very thoroughly, you will try more examples to check your program and see if it works.

Teacher: Can you be sure that the program is correct?

Students: The chances to succeed are higher this way.

Next, we explored CS teachers' conceptions of their students' work habits and performance in comparison to their educational goals. The teachers reported that while students accumulate factual knowledge, such as programming structures, they resist any attempt to change their work habits even though those habits lead them to poor performances [3].

Furthermore, not even when faced by situations where incorrectness was evident and work methods applied were useful, were the students convinced to abandon their methods. Instead they blamed anyone and anything else. The last complaint is consistent with the findings of Edwards [4]. This phenomenon was also observed by McCracken *et al* [9] who found that students who failed to develop a program blamed the conditions of the lab when in fact they never thought of the data structure needed to solve the problem.

Specifically, teachers complained that students sometimes are satisfied with success of compilation and that when testing involves a calculation to verify that the output displayed is indeed what should be, students do not perform the calculation, but rather, they merely check that the output seems reasonable. This behavior is consistent with the results of Edwards [4].

The quotes below exemplify these phenomena. The situation from which they were taken is a laboratory session in a course in concurrency for high-school level. The students were asked to develop a certain program (for further details, [1]).

Student: It [program] works. It prints some garbage at the top of the screen but that isn't important.

Teacher: [goes to student's computer] Show me.

Student: [executes the program and enters input; points to the screen] Here, it works.

Teacher: Is it the output you were expecting?

Student: I don't know.

2.3 The Notion of Relative Correctness

Note that the students quoted above did not care about the "garbage" displayed in addition to the output he expected. Also note that the students who produced incorrect programs for the ATM problem ascribe to these problems as *mostly correct*. These performances raised the hypothesis that students' understanding of correctness is different than the dichotomous definition professional posses.

In fact, we encountered situations where students decided a program is correct even though the program clearly did not fulfill its requirements. For example, we gave 138 high-school students who studied concurrency a synchronization problem that had two synchronization goals: (SG1) that three operations will be executed in a certain order and (SG2) that there will be no unnecessary other constraints. We provided the students with one correct solution and four incorrect solutions. The first two incorrect solutions did not fulfill SG1 and the second two fulfilled SG1 but did not fulfill SG2. Most students recognized that the first two solutions were incorrect, yet nearly a half of the students claimed that the two last solutions were correct although most of them clearly noticed that SG2 is not fulfilled. They claimed that SG2 was a nice-to-have feature yet not a crucial demand.

From other unreported observations we noticed that mostly, students were confused when a program produced the expected output but also unexpected output although they were familiar with the formal definition of a correct program. We concluded that students have different stable standards as to what constitute correctness (for further details, [1]).

3. METHODOLOGY

3.1 Research Tool

We distributed a questionnaire among 24 high-school students and 16 college students when they finished their CS studies. The responses were anonymous and individual.

The questionnaire consisted of two parts. Part A was designed to gain information on students' practices, norms, and perceptions regarding testing and verification. It consisted of the following five statements, each of which associated with evidence found in previous works. The students were instructed to mark one of the following options for each statement: (0) I disagree, (1) I agree, and (2) otherwise. There was room left for comments:

A.1 I executed a program I had written many times and got valid output, therefore I know that my program is correct

A.2 I wrote a program that computes a complicated calculation. When I **test** the program, I sometimes do not calculate (manually) the expected output, but rather satisfied if the output displayed looks reasonable

A.3 There are cases that I am sure that a program I wrote is correct and then I am satisfied with compiling it (with no executions)

A.4 When I test a program I systematically verify that I checked all the possible input examples

A.5 There is always the possibility that there is an input example for which the program does not work that I did not find

Statements A.1, A.2, and A.3 describe verification methods that we observed in classes and are considered to be inadequate by professionals. Statement A.4 measures the conceptions of students on their methods as being systematic, specifically, that all the legal input is covered; whereas statement A.5 measures whether their methods are actually systematic.

Part B was designed to gain information about students' standards of correctness, in particular to examine whether previous evidence on perceiving correctness as relative apply here too. We gave three assignments. In each assignment we described an incorrect program and an output displayed from which one can know that the program is incorrect. In all the assignments, the mistake was that unexpected output was displayed in addition to expected output. These assignments were based on situations we encountered in our observations. For each program we gave three statements, to each of which the students had to mark if they agree, disagree, or otherwise, and we gave room for comments.

The statements were as follows, (a) the program is correct, (b) the program is incorrect, and (c) the program is correct if the output does not distract from getting the required information (for assignment 1 we used a different version: the program is correct for most cases).

In assignment 1 the students were given a simple if-then program code whose goal was to display output according to the value of input X. The program was incorrect because for one group of inputs it produced the expected value, yet in addition it produced one more unexpected output. The students were also given testing results of input examples taken from this group and other groups.

Assignment 2 was phrased as follows:

You developed a very complicated program that should display hundreds of outputs. The program displayed all the output you expected to get but also in the end displayed one output item that does not suit the program requirements.

Finally, assignment 3 was as follows:

You developed a program that produces information about your family at your request. When you gave your family data and asked for the names of all your cousins, the program displayed the names of all your cousins but in addition, in the end you got the name of one of your uncles.

3.2 Data analysis

We first read the explanations of students who checked the "otherwise." If the explanation revealed a clear tendency toward agreement or disagreement we changed the response code accordingly. That way we had under "otherwise" only students who could not decide whether they agree or disagree.

3.2.1 Part A

For every statement we calculated the percentage of students who agreed with it and checked its statistical significance. We also calculated the distribution of students who value their testing as systematic and at the same time reported on non-systematical methods.

3.2.2 Part B

All programs given were incorrect. Therefore ideally all students should have disagreed with the statements 'the program is correct' and agreed that the program is incorrect. In addition, since there is no such concept "relatively correct" in the curriculum students should have disagreed with this statement. Therefore this response is ranked 1.

The next best response (rank 2) was given to students who understood that the program is incorrect, yet also agreed that it is

relatively correct. The worst responses would be of students who agreed the program was incorrect. We distinguished between responses where students also agreed that the program is relatively correct and those where students disagreed on that. Yet, we ranked both these responses 3. Finally, the rank 4 was given to indecisive responses, that is, responses where "otherwise," was checked, as well as responses in which students either agreed or disagreed that a program was both correct and incorrect. The ranks are summarized in the left column of Table 2.

We calculated the distribution of the responses for each group (college and high-school) and conducted statistical analysis to examine whether these groups are significantly different. In addition, we conducted correlation analysis among the statements of each assignment.

4. FINDINGS

4.1 Part A: Work Habits and Conceptions

Table 1 present the students' responses to part A. Each row presents the percentage of students who agreed with one statement. The two leftmost columns indicate the statement, the next column indicates the percentage of agreement among high-school students, the fourth is for the college students, and the rightmost column presents the percentage of agreement among the entire sample population.

Table 1: Percentage of students who agreed with statements of Part A

Statement		High school N= 25	College N = 15	Total N= 40
A.1	many executions	50%	50%	50%
A.2	reasonable output	33%	69%	48%
A.3	Solely compiling	42%	31%	37%
A.4	systematic verification	71%	75%	72%
A.5	Errors are possible	54%	81%	65%

Fifty percent of both groups agreed with statement A.1, that they can conclude correctness from merely executions on many input examples. While we would have expected them to at least comment on the need for systematic choice of the examples, the few comments we got from those who disagreed with that statement reveal that the disagreement is rooted with the use of the word 'many,' or in one of the students' words: "not many [input examples], one or two examples work for me," while we would have expected the students to stress that many examples yet not systematically chosen (to represent the entire input) does not make a sufficient method.

Similarly, a non-negligible number of students (42% of high school students and 31% of college students) reported that they are sometimes satisfied with mere compilation and that they sometimes do not verify that the displayed output is indeed correct but rather are satisfied if it looks reasonable (33% of the high school students and 69% of college students).

Interestingly, many students in both groups believe that they are systematic concerning correctness verification. The responses to this statement were found to be statistically significant ($r=.05$, $p<0.05$). Nonetheless, a distinguishable number of these students also agreed on performing the verification methods educators and CS professionals consider as inadequate as displayed in Table 2.

Table 2: Percentage of "systematic" students' (who agreed with Statement A.4) agreement with statements of Part A

Statement		Percentage of students who agreed		
		High School N=18	College N= 11	Total N=29
A.1	many executions	44%	36%	41%
A.2	reasonable output	22%	36%	28%
A.3	Solely compiling	33%	27%	31%
A.5	Errors are possible	44%	45%	45%

Note that both the high school and the college groups have a significant number of "systematic" students who agreed that they apply inadequate methods for testing (statement A.2 and A3.) In addition, the fact that 41% of these "systematic" students conclude correctness when they get expected output behavior for *many* input examples implies that students have a different conception of the meaning of being systematic where quantity covers for analysis. This hypothesis is strengthened by the fact that 45% of the "systematic" students agreed that there is always a possibility that the program does not produce correct output for *the entire* input domain. We concluded that students ascribe to their non-systematic methods as systematic.

4.2 Part B: the Definition of Correctness

The distribution of students' response to the three assignments in part B is presented in Table 3. The responses of the two groups in comparing proportion differences were found to be significantly different. Specifically, we checked the difference in proportion hypotheses analysis p1-p2 between the high-school group (n=24) and the college group (n= 16) for every statement in part B. We found that excluding two statements (statement c in assignment 1 and statement a in assignment 2) the difference in answers proportion between the groups is statistically significant ($alpha \leq .01$). Therefore, we present the results of each group separately.

Each row presents the frequency of a specific response among each of the groups. The responses rank is presented in the left column (a detailed explanation about the different ranks is provided in section 3.2.2). The second column presents the response to the three statements that comprise an assignment according to the following order from left to right: the program is correct, the program is incorrect, and the statement on relatively correctness. We coded the responses as follows: students' agreement to statement is marked by "√," and disagreement is marked by "X." The third column and the fourth column present the percentage of students who gave this response in the high-school group and the college group respectively.

4.2.1 Misjudgment of correctness

The distribution of the responses (Table 3) is characterized by different performances between the two groups as well as different responses to each assignment in each group. This fact is prominent given the fact that all the programs given were incorrect and the incorrect output behavior had the shared characteristic of extra output.

Interestingly, the portion of high school students who responded correctly (rank 1) was greater than that of the college group for all the assignments: 38% vs 31% in assignment 1, 58% vs 19% in assignment 2, and 83% vs 44% in assignment 3. However, in both groups there was a distinguishable portion of students who failed to recognize that the program was incorrect.

First, in assignment 1, none of the high school students agreed that the program is correct; in contrast, 25% of the college students thought that the program is correct (rank 3) and 19% were indecisive, that is, 41% of the college students could not tell the program was incorrect, even though they were given the fairly simple program code, the exact program goals, and the input example that reflects the incorrect I/O behavior.

Additionally, in responding to assignment 2, 13% of the high school students decided the program described was correct (and 16% more were not certain. Even worse, 38% of the college students decided that the program was correct and 30% more were indecisive.

Finally, in responding to assignment 3, 83% of the high-school students gave the correct response (rank 1), 12% decided the program was correct, and only 5% were indecisive. In the college group 26% decided the program was correct and 24% were indecisive. Thus, the responses of both groups to assignment 3 were better than the responses to assignment 2 despite the fact that the I/O incorrect behavior was rather similar: after a display of a sequence of correct output one "extra" incorrect output was displayed. The only difference in the assignments was the 'cover story' for the programs. In assignments 2 the goal was abstract whereas in assignment 3 the program processed the data of the students' family. This implies that there are subjective factors that influence students' tolerance to incorrect I/O, namely to their standards of correctness.

4.2.2 The notion of relative correctness

The correlations between the agreement responses to statements regarding relative correctness and agreements that the program was correct or incorrect were as follows: In the high-school group there was no significant correlation among the responses to assignment 1. In assignment 2 and assignment 3 there was a strong positive correlation between those who agreed that the program is relatively correct and those who agreed that the program is correct (0.518 in assignment 2 and 0.676 in assignment3) as well as a strong negative correlation between agreement to the statements of relative correctness and agreement to the statement that the program was incorrect (-0.715 in assignment 2 and -0.676 in assignment 3).

Table 3: The students' responses to the assignments in part B

Rank	Responses			High school, N=24			College, N=16		
	Correct	Incorrect	Relative	Assignment 1	Assignment 2	Assignment 3	Assignment 1	Assignment 2	Assignment 3
1	X	√	X	38%	58%	83%	31%	19%	44%
2	X	√	√	50%	13%	0%	25%	13%	6%
3	√	X	√	0%	13%	8%	25%	25%	13%
3a	√	X	X	0%	0%	4%	0%	13%	13%
4		Indecisiveness		12%	16%	5%	19%	30%	24%
				100%	100%	100%	100%	100%	100%

In the college group we found that in assignment 1 there was a strong positive correlation (0.505) between students who agreed the program is relatively correct and students who agreed the program is correct, yet there was no significant correlation between the responses to the relative correctness and incorrectness. In assignment 2 we found a positive non-significant correlation (0.41) between the responses to relative correctness and correctness. Finally, in assignment 3 we found a significant correlation (0.610) between the students who agreed that the program is relatively correct and those who agreed that the program is correct as well as strong negative correlation (-0.595) with those who agreed that the program is incorrect.

These correlations point out that the notion of "relative correctness" is common among students and that it sometimes completely overlaps with the concept of correctness (unlike the dichotomy presented by Joni & Soloway, 1986). Furthermore, these findings support our hypothesis that students conceptualize correctness as essentially relative.

The student's comments support the hypothesis too. While students clearly understood that the I/O behavior is not optimal their explanations reveal that their red line for correctness is different than that of the dichotomous definition:

"The program (assignment 1) fulfills its requirements even though it prints unnecessary output."

"The program (assignment 2) is not perfect but it works and that's what counts."

"The program (assignment 3) is correct but it is not finished."

Furthermore, we would have expected that students would include the logical mistake (in the nested if-sentences) that causes the incorrect I/O behavior of assignment 1. However, there were no references to the structure of the program; instead students quantified the relative part of the input to which the program produced the expected output as exemplified in the quotes above.

5. DISCUSSION

5.1 Students' Work Habits and Standards

A significant number of students agreed that "When I test a program I **systematically verify** that I checked all the possible input examples." However, many of the same students agreed at the same time that there is still a possibility that there is an input example that they did not cover at all. Furthermore, many of them agreed that they sometimes use verification methods professionals would not consider as neither systematic nor adequate, such as that they do not calculate the expected output but rather estimate the reasonability of the displayed output or even worst that they sometimes do not even execute the program once. Finally, about half of the students of both groups reported that they conclude on correctness when the program works for *many* input examples.

The discrepancies between students' inadequate methods and their positive conceptions of these methods are consistent with other works described above [5, 6, 10] as well as with our previous insights regarding students' (mis)conceptions of *thorough testing* [3]. We, thus, concluded that students have different standards of what constitute systematic examination which governs their performance (Figure 2).

Testing = Systematic examination of input examples

WHERE systematic = all input examples I could think of,

Examination = (sometimes) estimation of output reasonability

Figure 2: Students' conceptions of testing

5.2 Students' Definition of Correctness

Our conclusions regarding students' understanding (namely definitions) of correctness are summarized in figure 3.

Correct program = working program

Working program = exhibit reasonable I/O for many legal inputs

Reasonable output = mostly correct but also incorrect output OR output that looks like what one would expect* the program to display

*expectations vary according to subjective factors // or tolerance oward the unexpected varies according to subjective factors

Figure 3: the students' definition of correctness

First, the fact that many students decided that the program of assignment 1 was correct even though they were given the simple code, the goal, and an input example that reflects the incorrect I/O behavior implies that students' standards for a "working" program do not necessarily stress the requirement for correct I/O behavior for *all* the input of the problem space. Instead they soften this requirement to a correct I/O behavior for *most or many* parts of the input domain. This definition reflects their tolerance to errors.

In addition, the students' responses in assignment 2 and assignment 3 were quite different despite the fact that the descriptions of the programs' goals and the incorrect I/O behaviors were similar. Most students recognized that the program that concerned their families was incorrect while they tolerated a similar incorrect I/O behavior in assignment 2 that concerned unknown calculations. This implies that students' standards of correctness are influenced by subjective factors unlike the nature of professionals' understanding of correctness. This insight is important because these standards govern our performance when we encounter unexpected output; probably, students would have behaved differently if they encounter the situations described in these two assignments.

A possible explanation is that students judge the quality of the programs from a user point of view, and thus, a meaningless sequence of numbers would not be damaged by an additional number, whereas a mistake about your own family is noticeable and intolerable.

Most importantly, students' tolerance to errors is coherent with their (mis)judgment of programs that have incorrect I/O behavior as correct, what cause this inadequate knowledge to be robust to teachers' instruction.

Finally, the existence of the notion of relative correctness was found to be evident. This notion is compatible with students' preference to develop programs hands-on the computer by writing the first idea they have in mind and iteratively test it by executions and refine it locally as well as their acceptance of errors as almost unavoidable (in relatively simple program).

The fact that the performance of the college students was poorer than that of the high-school students raises a worrying issue. Arguably, college students gained experience of developing complicated software and thus realized that error-free programs is rare. Besides, much best-sold software produce sequences of versions where each version addresses errors found in the former version. Yet, this does not explain the fact that so many students did not recognize that the short program in assignment 1 was incorrect. Is it possible that while in high-school to some extent students' understandings of correctness intersect with the black-or-white definition but when they move on and gain more formal experience their attitude changes? In the future we plan to expand the investigation to include university level, to include a larger number of students, and to inquire about students' attitude toward different incorrect I/O behaviors, such as missing output items.

6. CONCLUSIONS

We found that students' definitions for correctness and systematic testing are different than those of professionals. Students' understand correctness as a relative property of the program and tolerate errors. This definition settles well with their definition of systematic testing that overlaps with work methods that professionals consider inadequate.

7. REFERENCES

[1] Ben-David Kolikant, Y., & Ben-Ari, M. (Submitted). Fertile Zones of Cultural encounter.

[2] Ben-David Kolikant, Y, & Pollack, S. (2004). Establishing computer science professional norms among high-school students, *Computer Science Education*, 14, 1, 21-35.

[3] Ben-David Kolikant, Y., & Pollack, S. (Submitted). Negotiating professional norms with informally technology experienced students: what goes wrong?

[4] Edwards, S. H. (2004). Using software testing to move students from trial-and–error to reflection-in-action. *SIGCSE`04,* March, Norfolk, Virginia, 26-30.

[5] Fleury, A. E (1993). Students beliefd about Pascal programming, *Journal of Educational Computing Research*, 9(3), 355-371.

[6] Iftikhar, B. (2004). Including Validation, Verification, and Debugging techniques in UTCS Curriculum, http://www.cs.utexas.edu/users/almstrum/cs370/iftikhar/final.html.

[7] Joni, S., & Soloway, E. (1986). 'But my program runs.' Discourse rules for novice programmers, *Journal of Educational Computing Research,* 2(1), 95–125.

[8] Leventhal, L. M., Teaslley, B. E., & Schertler Rohlman, D. (1994). Analyses of factors related to positive test bias in software testing, *International Journal of Human Computer Studies*, 41,717-749.

[9] McCracken, W. M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Ben-David Kolikant, Y.,Laxer, C., Thomas, L., Utting, I., Wilusz, T., 2001. A multi-national, multi-institutional study of assessment of programming skills of first-year computer science students. *SIGCSE Bulletin*. 33 (4), 125-140.

[10] Scott, E., Zadirov, A., Feinberg, S., & Jayakody, R. (2003). *Proceedings of Informing Science Educational Technology Education Joint conference*, Pori, Finland, 957-967.

8. ACKNOWLEDGMENTS

I thank Sarah Polack, Iris Ben-David Hadar, and Moti Ben-Ari for commenting on earlier drafts of this manuscript.