2018 HAWAII UNIVERSITY INTERNATIONAL CONFERENCES
STEAM - SCIENCE, TECHNOLOGY & ENGINEERING, ARTS, MATHEMATICS & EDUCATION
JUNE 6 - 8, 2018 PRINCE WAIKIKI, HONOLULU, HAWAII

USER INTERFACE EVALUATION: COMPARISON OF NOVICES TO UPPERCLASS COMPUTER SCIENCE STUDENTS

VANDEGRIFT, TAMMY
SHILEY SCHOOL OF ENGINEERING
UNIVERSITY OF PORTLAND
PORTLAND, OREGON
CHEN, TZU-YI
POMONA COLLEGE
CLAREMONT, CALIFORNIA

Dr. Tammy VanDeGrift
Shiley School of Engineering
University of Portland
Portland, Oregon
Dr. Tzu-Yi Chen
Pomona College
Claremont, California

User Interface Evaluation: Comparison of Novices to Upperclass Computer Science Students

Synopsis:

This study examines whether coursework in computer science changes how a student thinks about user interfaces. Our data collection consisted of asking experienced students to evaluate two on-line user interfaces for temperature conversion using prompts which differed only in the amount of context provided. Student responses were coded based on exhibited features believed to be used in expert evaluations of user interfaces, as well as whether they provided suggestions for improvement.

User Interface Evaluation: Comparison of Novices to Upperclass Computer Science Students

ABSTRACT

This study examines whether coursework in computer science changes how a student thinks about user interfaces. Our data collection consisted of asking experienced students to evaluate two on-line user interfaces for temperature conversion using prompts which differed only in the amount of context provided. Student responses were coded based on exhibited features believed to be used in expert evaluations of user interfaces, as well as whether they provided suggestions for improvement. These responses were then compared to results from the literature which analyzed novice evaluations of the same two interfaces.

We find that experienced students offered suggestions for interface improvements more often than did novices. Providing a context for the interface evaluation did not significantly impact the rate at which either novice or experienced students considered the purpose for the interface. However, experienced students were more likely to consider the user and they were more likely to evaluate the interfaces in terms of effectiveness, learnability, memorability, and interactivity. Experienced students, on average, used more criteria in their evaluations than did novices. Finally, whether or not a student had taken a software engineering class did not significantly affect responses. We conclude with thoughts on implications for the undergraduate CS curriculum.

1 INTRODUCTION

As humans interact with computers in increasingly varied and critical ways, training computer scientists capable of user-centered design becomes a more important task for educators. There are hints that user-centered design may play an important role in CS education, with some conjecturing that it is a "threshold concept" [12]. The ability to do user-centered design, however, likely first requires the ability to do user-centered evaluation. User-centered evaluation is also believed to be challenging.

For example, in [4] they compare four different methods for evaluation of a user interface for a pre-release version of a software product. The most effective was heuristic evaluation, a technique first described in [7] in which experts are asked to study an interface and to comment on it. But although effective, a drawback to its widespread adoption is noted in [4]: heuristic evaluation as they mean it requires UI specialists. The implication is that evaluators must have specialized training in understanding and applying important criteria for user centered design.

But as part of the commonsense computing project [2,5,6,10,11], there has also been prior work on how novice computing students approach user-evaluation tasks. In [1] researchers asked 149 computer science students who were in the first two weeks of introductory computer science classes at four institutions in the United States to evaluate two interfaces for converting temperatures between Celsius and Fahrenheit. Some students were given a prompt which mentioned a user Hal and the environment in which he would be using the interface; others were given a prompt that made no mention of context. Responses were coded for 11 criteria: six representing ways of considering usability, three representing ways of considering user experience, and two representing ways of considering context. They found that students "generally considered usability and user experience factors, but were less likely to consider context." In particular, they found no significant differences in the responses given when the prompt specifically mentioned a user and when it did not.

The study described in our paper bridges the gap between the study of novices in [1] and the study of UI experts in [4] by asking whether more advanced computer science students without specialized UI training give measurably different responses on the same user evaluation tasks studied in [1].

We find that, overall, more experienced students are much more likely to discuss how effective a user interface is at accomplishing a particular task; they are also more likely to make a concrete suggestion about how to improve the interface. However, experienced students are only slightly more inclined than novices to consider the user or to consider the purpose of a software artifact in evaluating it. Furthermore, there is not a significant difference in interface evaluations between advanced students who have taken software engineering as compared to those who have not. This suggests incorporating more focused UI design and evaluation activities and assessments into the CS curriculum.

2 METHODS AND PARTICIPANTS

2.1 Method

As in [1], students were given electronic links to two user interfaces capable of converting temperatures from Celsius to Farenheit or Farenheit to Celsius. Figures 1 and 2 illustrate these interfaces, both of which were based on graphical versions provided by Raskin in [8].

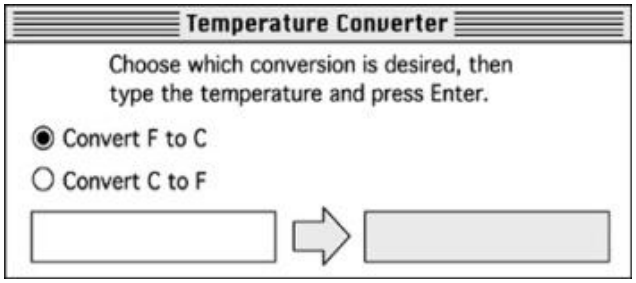

Figure 1: User interface A (text box).

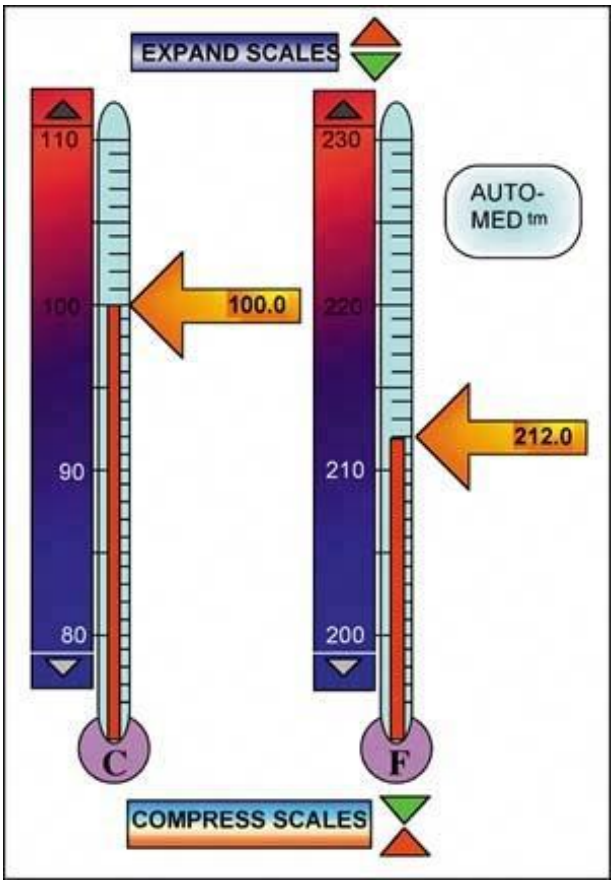

Figure 2: User Interface B (graphical).

Students were given one of two prompts, both asking them to "describe, compare, and contrast the interfaces". One prompt [Hal] gave additional context for how the interfaces would be used; the other [No Hal] simply asked students to do the evaluation. Here is the text from the study:

[**Hal Version**] Hal works at a computer, typing reports; he is occasionally interrupted by one or another of the researchers in his room, asking him to convert a temperature reading from degrees Fahrenheit (F) or Celsius (C) to degrees C or F, respectively.

Below are two possible interfaces (ways to interact with a program) for an application which converts temperatures between Celsius and Fahrenheit for Hal to use. Using complete English sentences, describe, compare, and contrast the interfaces. Before giving your answer, click here to try the prototypes.

[**No Hal Version**] Below are two possible interfaces (ways to interact with a program) for an application which converts temperatures between Celsius and Fahrenheit. Using complete English sentences, describe, compare, and contrast the interfaces. Before giving your answer, click here to try the prototypes.

2.2 Participants and Institutions

The data used in [1] consists of responses collected in Fall 2008 from 149 novice computer science students recruited from Introduction to CS courses at four institutions in the United States. The new data analyzed here consists of responses collected in Fall 2009 from 30 experienced junior and senior computer science students from three institutions in the United States. While there is overlap in the institutions, the timeline means that no students participated in both studies. The students in the experienced study were given either the "Hal" or "No Hal" prompt depending on the last digit of their phone number; this "randomly" divided the participants into two groups. The students in the novice study were given a consistent prompt within their course section. Table 1 shows the study participants from the five different institutions.

Table 1: Study Participants

Institution	# Novices	# Juniors/Seniors
A. large public, East coast, USA	40 (20 Hal/ 20 No Hal)	----
B. small private, West coast, USA	28 (0 Hal / 28 No Hal)	6 (3 Hal / 3 No Hal)
C. small private, Midwest, USA	33 (0 Hal / 33 No Hal)	5 (1 Hal / 4 No Hal)
D. large public, Midwest, USA	48 (48 Hal / 0 No Hal)	-----
E. small private, West coast, USA	----	19 (9 Hal / 10 No Hal)

As part of the data collection, we asked the juniors and seniors to list all CS courses they had already taken: 11 of the 30 had already taken a software engineering course, and none of the 30 had taken a course in human computer interaction or user-centered design.

2.3 Data Collection and Classification

The task was given as a homework assignment during the first week of the term at each institution. Students completed the task electronically and submitted their answers through a SurveyMonkey survey. The researchers obtained IRB approval to collect this data from students.

Here are examples of responses from the "Hal" and "No Hal" evaluation tasks from experienced CS students:

Interface A is best if you have a specific temperature in Fahrenheit or Celsius and need to know the exact value of temperature in the other system. It's not very good at explaining the function detailing the switch from one system to the other. It is also very plain. Interface B is best if you want to see how temperatures in Fahrenheit relate to temperatures in Celsius; a move on the Celsius scale moves the Fahrenheit scale 9/5 as much, etc. Unfortunately if you are trying to find

a specific temperature, hope that the temperature you are trying to find lies within the default values, as the arrows for moving the scale up and down are un-intuitive and require multiple button presses. The interface should slide to adjust while the user is moving it. [JA02, Hal version]

Interface A is a simple interface with only two buttons and one box for the user to interact with. It is straight forward and provides easy use for quick and accurate conversions. Interface B is more colorful and significantly more complicated, as it has several buttons and two sliding temperature indicators. Although B looks better, it can not be accurately used to convert large numbers as easily as A can. [JC04, No Hal version]

To enable more accurate analysis, each response was divided into units. Ideally each unit captured a single UI evaluation idea, although occasionally for readability some units represented more than one concept. The researchers reviewed the division into units and discussed disagreements until they reached consensus. For example, JA02's text (above) divided into the following units:

- Interface A is best if you have a specific temperature in Fahrenheit or Celsius and need to know the exact value of temperature in the other system.
- Interface B is best if you want to see how temperatures in Fahrenheit relate to temperatures in Celsius; a move on the Celsius scale moves the Fahrenheit scale 9/5 as much, etc.
- It is also very plain.
- It's not very good at explaining the function detailing the switch from one system to the other.
- Unfortunately if you are trying to find a specific temperature, hope that the temperature you are trying to find lies within the default values, as the arrows for moving the scale up and down are un-intuitive and require multiple button presses. The interface should slide to adjust while the user is moving it.

After the units were identified, two researchers independently categorized each unit. When disagreements occurred, a third researcher broke the tie in the novice study. In the experienced student study, the two researchers argued to consensus when the unit was not coded consistently.

The categories that were used to code the units were divided into four broad categories: usability, user experience, context, and implementation. The initial set of categories was taken from a set of features believed to be used by experts when evaluating interfaces, according to Sharp [9], and are marked with a * in Table 2. Additional categories emerged as the analysis of novice responses was conducted, also shown in Table 2.

Table 2: Expert criteria used for coding

Usability	*Utility: whether the interface provides what is needed by, and useful for, the user to accomplish the necessary tasks.
	*Effectiveness: a general goal capturing how good the interface is at allowing users to do what they need to do.
	*Efficiency: whether a user can carry out common tasks with a minimal number of actions.
	*Safe to use: how well the interface protects the user from errors and undesirable situations.
	*Learnability: whether the user can determine how to use the interface correctly to accomplish the necessary tasks.
	*Memorability: whether a user needs to repeatedly relearn how to accomplish specific tasks
User Experience	*Affect: the subjective quality of how interacting with and using the interface feels to a user.
	Visual Appearance: visuals of user interface without affect
	Interactivity: actions/controls for the user interface
Context	Considers user: the user is considered in the analysis
	Considers purpose: the purpose of using the interfaces is considered in the analysis
Implementation: how the interfaces may have been programmed	
Suggestion: indicates how the interface could be changed or improved	

For the junior/senior data, the researchers coded any unit that referred to a user specifically using the software as *considers user*. We were interested in knowing whether students explicitly thought about people (other than themselves) using and interacting with the interface. The terms GUI, UI, and user-friendly were not automatically coded as *considers user*, since we deemed those terms as jargon. We also re-coded the novice dataset for *considers user*, since the original coding of that dataset applied a stricter rule that required mentioning user goals.

We coded the novice dataset for *suggestion*, since it was not included in the original study [1] and yet emerged as potentially significant when examining the junior/senior responses. Table 3 shows example responses for categories for experienced students.

Table 3: Codes and example units

Criterion	Example
Utility	make it easy to convert in either direction [JA07]
Effectiveness	In interface B, you have to click to adjust the temperature on a thermometer, which is less accurate, [JA11]

Efficiency	and the user does not have to drag any icons to exact location on the GUI which might seem tedious and unnecessary. [JB04]
Safe to use	Both are silly in that they allow you to go below 0 on the Kelvin scale. [JA07]
Learnability	The controls to adjust the attributes of the application are neither explained nor clearly marked. For quite a while, I was not sure what exactly the unlabeled arrows were doing, nor what the buttons "expand scale" and "compress scale" were even supposed to do. I also could not easily devise what these things did by experimentation. [JC01]
Memorability	A also follows the simple "input what you know, output what you want to know" model of a black box program turning some information of yours into output containing information that you desire. [JC01]
Affect	The second interface, B, is more visual and more pleasant to work with if you feel like playing around. [JA10]
Visual Appearance	Interface B is a graphical interface where you slide the temperature on a thermometer up and down to have the value on the opposite thermometer shift to the converted value. [JB02]
Interactivity	I didn't like that I couldn't see a text cursor appear in the box when I clicked on it, nor could I highlight my old entry in order to delete it / start a new one. But the absence of a text cursor was the hardest of all, as I wasn't sure if I was ever in the box or not. [JA20]
Considers user	and could be used by someone who knows what they want and just needs a straight conversion. [JA22]
Considers purpose	Hal probably would too, for his purposes. [JA12]
Implement-ation	Secondly, there is no reason to force the user to press the enter key to perform the conversion: the result should update as the user types. Temperature conversion is computationally inexpensive and there is no reason to defer processing until the user finishes typing. [JA17]
Suggestion	B needs way more explanation and far fewer buttons. [JC03]

After coding every unit, we aggregated the codes to analyze which set of codes were found in each student's response. For example, student JB04 had units that were identified as {effectiveness, efficiency, considers the user, considers purpose, interactivity}. A student response could have anywhere from 0 to 13 codes. The results are presented at this level of analysis: a student response.

3 RESULTS

The following questions guided this study:

- RQ1: What expert criteria are used by experienced computer science students?
- RQ2: How do the criteria used by experience computer science students compare to those used by novices?
- RQ3: What is the effect of mentioning a user (Hal vs No Hal) on whether students consider context when evaluating a user interface?

- RQ4: What is the effect of taking a software engineering class on experienced students' responses?

3.1 Expert Criteria

Table 4 shows the percentage of novices and experienced computer science students who considered each category. Except for the updated coding of the *considers user* and *suggestions* categories, the percentages for CS1 students are those reported in Bouvier *et al.* [1].

Table 4: Novice and experienced student use of criteria

evaluation criteria		**# subjects (%) CS 1**	**# subjects (%) Experienced**
Usability	Utility	103 (69.1%)	18 (60.0%)
	Effectiveness	65 (43.6%)	23 (76.7%)
	Efficiency	80 (53.7%)	14 (46.7%)
	Safe to use	8 (5.4%)	3 (10.0%)
	Learnability	32 (21.5%)	15 (50.0%)
	Memorability	0 (0.0%)	2 (6.7%)
User experience	Affect	38 (25.5%)	11 (36.7%)
	Visual appearance	110 (73.8%)	26 (86.7%)
	Interactivity	74 (49.7%)	21 (70.0%)
Context	Considers user	63 (42.3%)	19 (63.3%)
	Considers purpose	35 (23.5%)	12 (40.0%)
Implementation		19 (12.8%)	1 (3.3%)
Suggestion		3 (2.0%)	5 (16.7%)

For all but the utility, efficiency, and implementation categories, the percentage of experienced students considering the category was larger than for the set of novices. Applying the N-1 chi-squared test for two proportions, the categories with p-values less than .05 to reject the null hypothesis that the proportions are the same are as follows:

- Effectiveness: p-value .001
- Learnability: p-value .0013
- Memorability: p-value .0015
- Interactivity: p-value .0427
- Considers user: p-value .0357
- Suggestion: p-value .004

Another perspective on the responses is given by Figure 3, which shows the percentage of students who applied different numbers of criteria in their responses. The average number of criteria used for experienced students was 5.7 and for novices, 4.2.

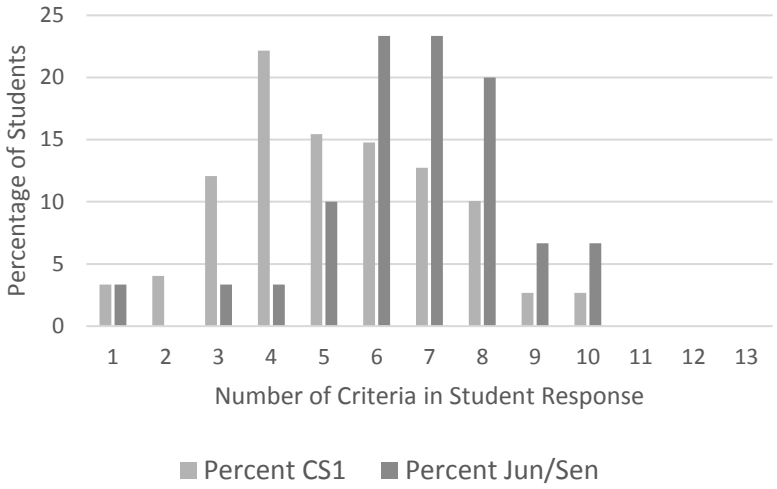

Figure 3: Distribution of the number of criteria used in evaluation task

3.2 Considering Context

We gave two versions of the prompt to see whether mentioning a user and context would lead students to clearly consider context in their response. The two relevant categories are "Considers user" and "Considers purpose". Table 5 shows the results for these two categories for the four populations.

Table 5: Hal vs No Hal tasks with respect to context

	Considers user	Considers purpose
Novice (Hal), N = 68	26 (38.2%)	14 (20.6%)
Novice (No Hal), N = 81	37 (45.7%)	21 (25.9%)
Experienced (Hal), N = 13	7 (53.8%)	6 (46.1%)
Experienced (No Hal), N = 17	12 (70.6%)	6 (35.3%)

We looked at the context category holistically by merging the user and purpose criteria. Of the experienced students, 9 considered user and purpose, 10 considered the user without the purpose, and 3 considered purpose without the user. In total, 22/30 (73.3%) experienced students considered context. Of the novices, 26 considered user and purpose, 37 considered user but not purpose, and 9 considered purpose but not the user. Of the novices, 72/149 (48.3%) considered context. A significantly different proportion (chi-squared 6.2644, p-value .0123) of experienced students consider context versus novices.

3.3 Impact of Coursework

Table 6 looks at the effect of taking a software engineering class on responses from experienced computer science students. There are no statistically significant differences at the p=.05 level.

4 DISCUSSION

We started by asking whether students who have taken multiple computer science courses are more likely than novices to apply expert criteria to the evaluation of user interfaces. The data collected gives insight into both student evaluation of user interfaces and the undergraduate computing curriculum.

Table 6: Experienced students' criteria divided by those who took Software Engineering vs those who did not

evaluation criteria		**# subjects (%) No SE**	**# subjects (%) Took SE**
Usability	Utility	5 (45.5%)	13 (68.4%)
	Effectiveness	8 (72.7%)	15 (78.9%)
	Efficiency	5 (45.5%)	9 (47.4%)
	Safe to use	0 (0.0%)	3 (15.8%)
	Learnability	6 (54.5%)	9 (47.4%)
	Memorability	1 (9.1%)	1 (5.3%)
User experience	Affect	5 (45.5%)	6 (31.6%)
	Visual appearance	9 (81.8%)	16 (84.2%)
	Interactivity	6 (54.5%)	15 (78.9%)
Context	Considers user	6 (54.5%)	13 (68.4%)
	Considers purpose	5 (45.5%)	7 (36.8%)
Implementation		0 (0.0%)	1 (5.3%)
Suggestion		2 (18.2%)	3 (15.8%)

We found that advanced students are more likely than novices to apply a broad range of criteria. This suggests juniors and seniors have developed a more holistic view of user interface evaluation.

Advanced students are also more likely to consider most individual criteria including, for example, learnability and memorability. The memorability numbers are too small to draw statistically supported conclusions. However, 50% of experienced students and just 21% of novices evaluated the interfaces with how easy/hard it is for a user to understand how to use the interface. Again, we do not know which experiences (other than more CS courses) contribute to this result. Perhaps experienced CS students have seen/used a larger variety of software and their interfaces, some of which are difficult to learn. Surprisingly, both novice and experienced students were less likely to consider the user when given the Hal scenario. This may be because the user in question, Hal, was already identified.

Our results give insight into the kinds of evaluation criteria that are reinforced by the "standard" undergraduate curriculum in computer science. We see how experienced computer science use more criteria and more often include the user in the evaluation task. Also, experienced users evaluated the effectiveness of the interfaces more often than novices. This could be the result of having more software development experience, since the novices had little to no training even in programming at the time the data was collected. Data structures courses emphasize ways of effectively organizing data for efficient storage and retrieval; taking that course could provide insight into how well an interface does what it is designed to do. We saw that taking a software engineering class does not impact the frequency with which criteria are used. This suggests that students need training specifically in UI design and evaluation since more general SE courses do not necessarily lead to more developed UI skills.

Implications for computing education include more focused UI design and UI evaluation activities and assessments into the CS curriculum. The ACM guidelines for Computing Curricula 2013 include 4 units (tier-1, required) of foundations of HCI, including usability heuristics and usability testing [3]. User interface evaluation tasks, similar to those used in this study, could be incorporated into existing software engineering courses. This could also shed light into a conjecture that user-centered design is a "threshold concept" in computing [12].

4.1 Limitations of the Study

This study has limitations with respect to research design. Due to limited student access, the researchers collected data from five different institutions, three of which were used as data collection sites for the CS1 and experienced student populations. A better study design could look at the same student over time (CS1, junior year, senior year) to study the progression of user interface design evaluation skill over time. The CS 1 population includes all students taking CS 1, which includes students who may or may not continue with more computer science courses. The populations differ in that respect: the experienced set of students were computer science majors. Additionally, at the time of data collection in 2009, enrollment at two of the data collection sites was low. Finally, the experienced study data was collected at small, private institutions in the USA. These results may not generalize to populations at different types of institutions.

4.2 Future Work

There are many ways in which this work could be extended. A future study could examine whether *any* modification of the prompt leads to students consistently evaluating the interface for a specific purpose. This might additionally give insight into how educators could scaffold such evaluation tasks to encourage students to consider context when doing interface evaluation. Such UI evaluation tasks could also be used for pre-assessment and post-assessment in a software engineering or Human Computer Interaction course.

Future studies could look at different populations. One study might use students who had taken a UI or HCI class and compare those results to the ones in this paper. Another might use software engineering professionals as subjects; results would be informative as to how and whether general experience in the field affects the criteria used in UI evaluation.

5 CONCLUSIONS

This study examined differences in task performance for two populations: novices in CS 1 and junior or senior-level computer science majors. The user evaluation task required no formal CS training, which presented an opportunity to see how true novices compare with students who have had more than two years formal experience in a computer science program. We found that experienced students are more likely to consider the user and evaluate interfaces for effectiveness, learnability, memorability, and interactivity. Experienced students are also more likely to offer suggestions for improvement. While this result is positive in that CS students have demonstrated more skill than novices, it also reveals opportunities for curriculum development focused more directly on UI design and UI evaluation.

ACKNOWLEDGMENTS

Thanks to Andy Carle for helpful discussions and to Gary Lewandowki for collecting data. Thanks to our students for their insight and data. This work was supported in part by the National Science Foundation under grants DUE-0736343, DUE-0736700, DUE-0736572, DUE-0736738, DUE-0736859, and DUE-0736958. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation.

REFERENCES

[1] Bouvier, D., Chen, T.-Y., Lewandowski, G., McCartney R., Sanders, K., and VanDeGrift, T. User Interface Evaluation by Novices. In *Proc. ACM ITiCSE '12.* ACM Press (2012), 327-322.

[2] Chen, T.-Y., Lewandowski, G., McCartney, R., Sanders K., and Simon, B. Commonsense computing: using student sorting abilities to improve instruction. In *Proc. SIGCSE '07*, ACM Press (2007), 276–280.

[3] Computer Science Curriula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. http://www.acm.org/education/CS2013-final-report.pdf. 2013.

[4] Jeffries, R., Miller, J. R., Wharton, C., and Uyeda, K. User Interface Evaluation in the Real World: A Comparison of Four Techniques. Proc. ACM SIGCHI '91. ACM Press (1991), 119-124.

[5] Lewandowski, G., Bouvier, D., Chen, T.-Y., McCartney, R., Sanders, K., Simon, B., and VanDeGrift, T. Commonsense understanding of concurrency: computing students and concert tickets. *Communications of the ACM*, 53(7): 60-70, 2010.

[6] McCartney, R., Bouvier, D., Chen, T.-Y., Lewandowski, G., Sanders, K., Simon, B., and VanDeGrift, T. Commonsense computing (episode 5): algorithm efficiency and balloon testing. In *Proc. ICER '09*, ACM Press (2009), 51-62.

[7] Nielsen, J., and Molich, R. Heuristic evaluation of user interfaces. Proc. ACM SIGCHI '90. ACM Press (1990), 249-256.

[8] Raskin, J. The Humane Interface: New Directions for Designing Interactive Systems. Addison-Wesley Professional, 2000.

[9] Sharp, H., Rogers, Y., and Preece, J. Interaction Design: beyond human-computer interaction, 2nd edition. Wiley, 2007.

[10] Simon, B., Bouvier, D., Chen, T.-Y., Lewandowski, G., McCartney, R., and Sanders, K. Commonsense computing (episode 4): Debugging. Computer Science Education, 18(2):117-133, 2008.

[11] Simon, B., Chen, T.-Y., Lewandowski, G., McCartney, R., and Sanders, K. Commonsense computing: what students know before we teach (episode 1: sorting). In Proc. ICER '06, ACM Press (2006), 29–40.

[12] Zander, C., Boustedt, J., McCartney, R., Mostrom, J.-E., Sanders, K., and Thomas, L. Student transformations: are they computer scientists yet? In Proc. ICER '09, ACM Press (2009), 129-140.