Linking Questions and Evidence

JOSH TENENBERG
University of Washington, Tacoma
and
ROBERT MCCARTNEY
University of Connecticut

This special issue features a set of papers recently published in the 3rd International Workshop on Computing Education Research (ICER'07). The papers were selected because they closely meet the publication criteria for ACM/JERIC: stemming from computing education practice, grounded in relevant literature, containing analysis of primary empirical data collected to support specific claims, and providing implications for practice. In addition, the diverse set of research methods used demonstrates how empirical methods can be chosen to serve the specific questions of interest, whether for purposes of research or for improvement of teaching practice. This set of papers can therefore serve as examplars and resources for the educator wishing to develop a wider range of tools for probing issues of teaching and learning within the classroom.

Categories and Subject Descriptors: K.3.2 [**Computers and Education**]: Computer and Information Science Education—*Curriculum*; *Computer science education*

General Terms: Experimentation, Human Factors

Additional Key Words and Phrases: Research methods, evidence, empirical methods, quasi-experiment

ACM Reference Format:
Tenenberg, J. and McCartney R. 2008. Linking questions and evidence. ACM J. Educ. Resour. Comput. 7, 4, Article 1 (January 2008), 7 pages. DOI = 10.1145/1316450.1316451. http://doi.acm.org/10.1145/1316450.1316451.

1. WHY THIS SPECIAL ISSUE?

In this issue, we feature a set of papers recently presented at the 3rd International Workshop on Computing Education Research (ICER'07) in Atlanta, Georgia. In this editorial, we provide our reasons for undertaking this special

Authors' addresses: J. Tenenberg, Institute of Technology, University of Washington, Tacoma, 1900 Commerce St., Tacoma, WA 98402; email: jtenenbg@u.washington.edu; R. McCartney, Department of Computer Science and Engineering, University of Connecticut, Storrs, CT 06269-2155; email: robert@engr.uconn.edu.

issue, a brief overview of the papers, and some implications of this set of papers for computing educators.

Our motivation for bringing these papers from a research audience to a teaching practitioner audience is twofold. First, these papers closely fit our criteria for publication in ACM/JERIC: They stem from computing education practice, are grounded in relevant literature, contain analysis of primary empirical data collected to support specific claims, and provide implications for practice. This issue should not be thought of as the Best of ICER—though we think that all of the papers are excellent—but rather as those ICER papers most relevant to ACM/JERIC's mission.

Our second reason for publishing these papers concerns the ways in which each of these papers uses evidence to support its claims, the focus of our editorial in the previous issue [McCartney and Tenenberg 2007]. In our tenure as editors-in-chief of ACM/JERIC, we have read dozens of papers submitted to the journal. Most of these papers appeal to a broad readership of practitioners, are carefully written, provide sufficient details for replication, and find support in the literature of research and/or practice. However, in the great majority of cases, these papers are returned to authors with requests for major revisions due to the absence or insufficiency of empirical evaluation. We thus hope that the papers presented here can serve as exemplars demonstrating to potential authors new ways in which evidence can be used to support claims about teaching and learning. In our descriptions of each paper, we thus highlight the empirical/evidential aspects of the research undertaken.

2. EXPERIMENTAL AND QUASI-EXPERIMENTAL DESIGNS

As scientists, many of us were trained in a positivist tradition for validating causal truth claims. The canonical form for doing so is the experimental design in which a sample is drawn from a population and each element of the sample is randomly assigned to a treatment or a control group. The variable of interest is measured after treatment and compared with the control group. There is an implicit *ceterus parabus* (all other things being equal) assumption, that is, that the treatment and control groups differ only by the treatment. By careful sampling and control, this reduces (though never eliminates) the possibilities for bias, strengthening our belief that the treatment has a causative effect on the outcome. These designs are not uncommon in the computing education literature. For instance, a review of a set of experimental designs undertaken to determine the causal effect of algorithm visualizations on student learning can be found in the meta-analysis by Hundhausen et al. [2002].

Experimental control, however, is rarely possible within the real-world constraints that most educators face. Campbell and Stanley [1963] introduce the notion of the quasi-experiment that mirrors the experimental design but in which treatment and control are not (and usually cannot be) randomly assigned. This would happen, for example, when we teach 2 sections of the same course that vary only by a newly-introduced teaching innovation but where students self-select into one section or the other. One of the key contributions by Campbell and Stanley is to demonstrate how care in the design of quasi-experiments can reduce the threats to validity that can arise when randomization is not possible. Examples of this study design can be found in the work of Nagappan et al. [2003] concerning paired versus solo programming and in that of Fagin and Merkle [2003] concerning the effectiveness of robotics in the introductory programming course.

It should come as no surprise then that at JERIC we often receive papers with a comparative study design of treatment versus control to try to measure the efficacy of a teaching intervention. Many of these, however, suffer from the problems outlined by Campbell and Stanley [1963]: Numbers are too small to have sufficient statistical power; the two groups are composed of students with quite different characteristics prior to treatment that are not measured or adjusted for in the analysis; and the history of the groups differs for reasons other than the treatment (e.g., the groups are taught by different teachers), to name just a few. More than anything, this highlights the very real and pragmatic constraints under which most teaching practitioners operate. The prevalence of this study design, however, also indicates its hold on our collective perceptions of what constitutes evidence for pedagogical claims.

3. AN OVERVIEW OF THE PAPERS: MOVING BEYOND THE QUASI-EXPERIMENT

What might be most surprising about the papers that we present here is the absence of the experiment or quasi-experiment. Abandoning these designs as the sole basis for validating truth claims expands the set of questions that can be considered for investigation and requires a variety of methods for answering these questions.

Many of these questions, rather than being causal in nature, are descriptive, centered primarily on answering *what*, *how*, and *how many* kinds of questions. In this issue, Brian Hanks asks this type of question in his article "Problems Encountered by Novice Pair Programmers". Specifically, he asks for what kinds of errors are introductory programming students requesting assistance when they pair program and with what frequency? He thus leaves aside for future work (either by himself or others) the specific educational interventions that might reduce such errors. What is noteworthy about Hanks's study design is that he replicates the design of Robins et al. [2003]. Replication studies are particularly important (but all too rare) because they help to distinguish context from the phenomenon of interest. For the researcher, replication studies have the added advantage of coming with a data collection and analysis methodology predetermined. But by altering the study conditions—pair programming in his study versus solo programming in Robins et al.'s study—Hanks also implicitly asks for differences and similarities between student performance between the paired and solo programming conditions. The method for data collection and analysis is to systematically log each request for assistance during all labs throughout an academic term and code each of these into the predefined categories defined by Robins et al. Hanks finds that although paired students ask for less assistance, they nonetheless struggle with simple mechanical problems in the same proportion as solo programmers; pair programming is not a silver bullet.

The article “A Study of the Development of Students’ Visualizations” is another descriptive article. Authors Sajaniemi, Kuittinen, and Tikansalo inquire as to what visualizations students use for representing object-oriented program state during execution. Although there have been many studies involving students’ understanding of expert-generated visualizations (e.g., discussed in the meta-analysis cited earlier by Hundhausen et al. [2002]), Sajaniemi et al.’s is one of the few that examines student-generated visualizations. During three different laboratory sessions associated with a ten-week introductory programming course, the authors “asked students to draw pictures depicting program state at a specific point of program execution” of small Java programs. This data collection method directly serves the question of interest: If you want to study student visualizations of program execution, ask them to draw what they understand from a particular program as it is executing.

The analysis first involves the identification of the elements present in the drawing, using a researcher-defined coding scheme. Following this, the researchers scored the visualizations based on 1) their relationship to the software visualization tools that had been presented to students in the laboratory, 2) the form of the elements (e.g., the use of boxes, arrows, etc.), and 3) the semantic content of the elements (e.g., being student-identified as classes, objects, methods, method calls, etc.). This analysis leads the authors to make a number of inferences about the kinds of conceptions and *mis*conceptions that students have concerning object-orientation, and about the relationship of student visualizations with those presented to students by the software visualization tools used within the laboratories. And by tracking student representations over several weeks, they are able to track changes to representation that indicate both deeper learning and persistent misconceptions over time.

The article “Uncovering Student Values for Hiring in the Software Industry” by Chinn and VanDeGrift, though descriptive in nature, contrasts with those just discussed in focusing on student beliefs absent an implicit normative standard, rather than on student misconceptions and errors in comparison to expert performance. Chinn and VanDeGrift inquire into the values that computer science students hold in a hypothetical scenario in which they are asked to make hiring decisions. Presented with a set of detailed descriptions of 4 job candidates for both a software development and a program manager position for an imaginary software company, each student in selected courses was asked to make hiring decisions and provide a rationale.

The authors undertook a data-driven content analysis in order to answer specific questions about the candidates chosen, the criteria students used in making these choices, and whether students changed their hiring choices following an in-class small-group discussion. This content analysis involved coding each of the student-generated rationales into a set of 25 emergent categories, such as “technical ability”, “organizational skills”, and “commitment to company values”. Each of these categories was then grouped into 1 of 5 broad areas: technical skills, soft skills, personal traits, previous experience, and company/job considerations. Having developed these categories and broad areas, the authors provide the statistical distribution of student responses for each of these categories. This distribution summarizes the students' valuing of these different characteristics for each of the job positions. By undertaking this study, the authors reveal what are often tacit dimensions of student attitudes about the characteristics that they perceive future employers will value.

The Simon and Hanks article "First Year Students' Impressions of Pair Programming in CS1" is likewise focused on student beliefs. While there have been many studies comparing various aspects of student performance between paired and solo programmers in introductory programming courses, this is the first that we know of where the researchers inquired into student perceptions of the experience of working in pairs. The data collection method used was semi-structured interviews. Students at 2 universities in which the first programming course uses pair programming followed by a second programming course that uses solo programming were interviewed during their second course. The researchers asked students to describe pair programming, talk about the valuable and unsatisfactory pair programming experiences, and compare their paired to solo programming experiences.

Simon and Hanks's analysis is structured around each of their main questions, and is primarily thematic, that is, they uncovered common themes among student responses that exhibited an underlying similarity. Overall, students found considerable benefit from working in pairs, including mutual support and help when they were stuck. But there were also negative aspects to the pair programming experience, including the difficulties that arise from different skill levels within the pair and the almost unanimous belief that students do not understand their code as thoroughly when pair programming.

The article by Yarosh and Guzdial begins with a particular teaching innovation—the use of a context-rich *media computation* approach to the introductory data structures (CS2) course—and asks about how students experience this course. In this course, data structures are not presented as simply context-free abstractions but as concrete tools used for manipulating the sounds and images of a realistic animated movie sequence that provides the contextual backdrop for the entire course. In this article, the authors are interested in the impact of the contextual nature of this course on student perceptions of the course and their learning.

The data collection method involves a sequential use of both quantitative and qualitative techniques. An initial survey was given at the halfway point in the academic term. Because surveys typically entail check-box style answers to constrained choice questions, they can be administered to a large number of people and analyzed at low cost; yet they rely on the researchers's preconceptions about which factors are important to the students. The authors then used this survey to identify particular questions about which they wanted to probe more deeply. This involved semi-structured interviews administered to only a small subset of the students. This depth requires a higher cost in data analysis but results in a richness of data that suggests new causal hypotheses. Finally, these interview hypotheses were translated into a survey that was

Table I. Summary of Research Questions and Methods

Article	Question	Context	Data collection method	Data analysis method
Hanks	For what kinds of problems do students seek assistance?	Lab exercises in intro Java programming course	Naturalistic: Logged requests for assistance	Code and count
Sajaniemi et al	What visualizations do students use for representing OO program state	Intro Java programming course	Elicited: Drawn pictures	Element, structure, & semantic analysis
Chinn and VanDeGrift	What criteria do CS students use in making hiring decisions	Computer Ethics and Software Dev. courses	Naturalistic: Written rationales	Data-driven content analysis
Simon and Hanks	How do students experience pair programming?	Intro programming sequence (CS1 and CS2)	Elicited: Semi-structured interviews	Data-driven thematic analysis
Yarosh and Guzdial	How does a "contextual" approach affect student perceptions?	Intro data structures	Elicited: Surveys, semi-structure interviews	Mixed method: statistical, thematic

administered to all students to determine the extent to which these beliefs were shared by the entire study population.

The authors found that 70% of students perceived the media computation context to increase the interest of the course, though only 45% of students thought that this context increased the course's usefulness. The authors also suggest that the narrative context of the animation project provided structure and coherence that had positive learning benefits to many students.

4. EVIDENCE REDUX

As can be seen from this summary of the articles, there is a tight coupling between the research questions of interest to the researchers and the data collection and analysis methods used to investigate the questions. If we want to understand student beliefs, then we ask them; if we want to understand student representations then we elicit them; and if we want to determine the problems that lead students to ask for assistance, then we meter these. In all cases, the question drives the method, subject to researcher constraints on time, energy, access to data, skill, and the other pragmatic tradeoffs associated with an empirical approach. This general pragmatic approach to investigate questions of teaching and learning is more fully elaborated in Fincher and Petre [2004]. Table I summarizes the set of studies described by these articles and highlights the relationship of the question to the data collection and analysis methods.

Our focus on evidence in the current and previous editorials represents our concern with scholarship in teaching and learning. What are the implications for the educator who wants to improve practice? And how does this reflect back on the authoring of ACM/JERIC articles? First, we hope that these articles demonstrate that the experiment and quasi-experiment are not the only possible forms for investigating teaching and learning within the discipline. In addition, what some of these studies show is that teachers already have ready-to-hand a rich set of data that they can use to probe issues of teaching and learning: the assignments that students hand in (as in the Chinn and VanDeGrift article), and their observable behavior (as in the Hanks article). These data are referred to as naturalistic in Table I. Making effective use of these naturalistic data requires their systematic collection, analysis appropriate to the data (such as counting, categorizing, and thematizing), and an explicit reflection on the implications of the analysis for teaching the course.

But not all questions are answerable by naturalistic data. Some data need to be explicitly elicited from students, such as the drawings in the Sajaniemi et al. article and the surveys in the Yarosh and Guzdial article. Though not in the toolkit that most computer science educators develop during graduate school, the use of surveys, interviews, and other elicitation methods can be incrementally learned by educators in professional practice. As with other skills (such as programming) their mastery requires use and critical reflection. Our hope is that the set of articles presented here can serve as a set of examplars and resources for the educator wishing to develop a wider range of tools for probing issues of teaching and learning within the classroom.

REFERENCES

CAMPBELL, D. AND STANLEY, J. 1963. *Experimental and Quasi-Experimental Designs for Research.* Rand McNally.

FAGIN, B. AND MERKLE, L. 2003. Measuring the effectiveness of robots in teaching computer science. *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education (SIGCSE), ACM SIGCSE Bull. 35,* 1.

FINCHER, S. AND PETRE, M. 2004. *Computer Science Education Research.* Routledge Falmer, London, UK.

HUNDHAUSEN, C., DOUGLAS, S., AND STASKO, J. 2002. A meta-study of algorithm visualization effectiveness. *J. Visual Lang. Comput. 13*, 3, 259–290.

MCCARTNEY, R. AND TENENBERG, J. 2007. Why evidence? *J. Educ. Resour. Comput. 7,* 3, 1.

NAGAPPAN, N., WILLIAMS, L., FERZLI, M., WIEBE, E., YANG, K., MILLER, C., AND BALIK, S. January 2003. Improving the CS1 experience with pair programming. In *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education (SIGCSE), ACM SIGCSE Bull. 35,* 1.

ROBINS, A., ROUNTREE, J., AND ROUNTREE, N. 2003. Learning and teaching programming: A review and discussion. *Comput. Sci. Educ. 13,* 2, 137–172.

Received October 2007; accepted October 2007