Contemporary Issues in Teaching and Learning with Technology

Jerry P. Galloway
Texas Wesleyan University, USA
University of Texas at Arlington, USA

INTRODUCTION

To speak of contemporary issues in instructional technology is like counting wave crests in a stormy ocean: they are changing quickly all the time. New technologies and new issues present themselves daily. Educators struggle with both the instructional integration of computing and developing the skills and knowledge necessary to use technology effectively (Lipscomb & Doppen, 2005). Why, after over 30 years of having computers in schools, are educators still having such difficulties?

Today's population is much more accustom to electronics, yet knowledge is weak, concepts are misunderstood, and the difficulties of teaching with technology seem as serious and convoluted today as ever before. The great physicist and thinker, Richard Feynman, offered some critical comments about the challenges of educators. "What happens is that you get all kinds of statements of fact about education, about sociology, even psychology — all kinds of things which are, I'd say, pseudoscience" (Feynman, 1999, p. 242). Today, we understand "more about education [but] the test scores are going down…we just don't understand it at all. It just isn't working" (p. 243). Being critical of how the scientific method is applied to education, Feynman's comments highlight how the study of teaching and learning yields limited or questionable results. Teacher trainers take their best guess on how to prepare teachers to use technology.

BACKGROUND

Educational computing is a relatively new discipline compared to mathematics and science. While the earliest uses of computers might have been by departments of mathematics, it quickly became important for virtually all teachers to become computer literate. But what exactly that entails was not exactly clear (Galloway, 1985) for learning and in society (Beaty & Tucker, 1987).

Microcomputer technology, primitive by today's standards, lacked user-friendly applications, any sort of consistent user interface, or easy-to-use telecommunications and interconnectivity. There was an early division between those who learned to program computers vs. those who focused more exclusively on applications software. Conceptual development, improvement of problem solving, and higher-order thinking skills in computing have been directly linked to the inclusion of Logo programming (Allen, 1993; Battista, 1994; Borer, 1993; Dalton & Goodrum, 1991) and BASIC programming (Overbaugh, 1993). Yet, in spite of an overwhelming need to operate early microcomputers through programming, educators focused instead on the actions and procedural tasks of specific applications (Galloway & Bright, 1987).

With this as a foundation, decades of training have followed in which educators have tried to master new devices and software. So, how long does it take to reach a point of nationwide competency, to develop the protocols of effective use, to establish the knowledge of how best to learn computing? Compared to centuries of science and mathematics, perhaps our 30-plus years do not seem so long.

EDUCATORS LEARN COMPUTING: A PROBLEM OF PERSPECTIVE

Our collective perspective on what it means to learn computing affect what goals we pursue and how we proceed. For example, the use of rubrics or portfolios were not commonly emphasized in education 30 years ago. Today, they are an accepted or at least popular tool for preparing educators (Galloway, 2006; Rural School and Community Trust, 2001). Does this represent progress or perhaps just a symptom of changing fads? Is this a function of real knowledge or mere opinions? This is again reminiscent of a Feynman (1999) criticism, as he suggests that professionals 30 years ago have as much right to a correct opinion as we have today, "to equally unscientifically come to a conclusion" (p. 243)—even if wrong.

Preparing Teachers

It is unlikely that educators younger than their mid-40s graduated high school without having computers in their education. There has been, since the late 1970s, a continual focus on the needs of teachers to learn and adapt to a technology-based profession.

Our attempt over the years to change educators into computer-literate professionals essentially failed. Many will argue the point, as clearly there are countless success stories. But, with the exception of the techies and innovative pioneers, educators across the profession a generation ago did not, have not changed their basic approach to integrate technology.

Compared to in-service classes, college courses, training, or other options, an overwhelming majority of teachers maintain that their primary methods of learning computing was through self-study and personal experimentation (Galloway, 1997). It can be argued that teachers must assume a responsibility for advancing their technological knowledge and be engaged learners.

When taking a computer class, one must go beyond the prescribed activities. For example, it is not likely that one would be assigned the experience of losing a file or opening a file with the wrong program. These frustrations can be a very necessary part of learning. Far too often educators are passive and restrict their involvement to occasional and discrete enrichment offered through someone else's initiative. Delays and intermittent and partial commitments inhibit learning.

As an analogy, when this author was young, rock-n-roll music was still the choice of the young, but grandfather did not relate and found it quite distasteful. In elevators in 1968, one would hear music from Lawrence Welk and such. This author believed that if elders could simply understand and learn about rock-n-roll and what the artists were attempting to express musically that society could change and the music would be accepted. Today, one is likely to hear McCartney, Dillon, The Beatles, Buddy Holly, or many of the other artists that were objectionable in those earlier years. One might think that, indeed, things changed.

However, the point is that this did not occur because the elders were influenced or convinced. The younger, rock-n-roll generation did not change anyone. The elders were not convinced. No metamorphosis occurred. The young simply grew older and brought their music with them. As the elders died off, the young with a new culture replaced the old.

The same seems true for the computer-using generation. Our efforts a generation ago were ineffective. We have simply waited around while a new generation grows older bringing their technology-based lifestyle with them. Until our children have time to take their place, today's teachers are still introduced to computing as beginners.

Training vs. Education

What do current educators expect from computer training? If we accept that it is difficult to teach someone who does not want to learn, what do students expect from their training? Unfortunately, the most popular notion in instructional technology is that teachers are to be trained, not educated. More than mere semantics, teaching tends to emphasize *showing teachers how to use* technology — rather than facilitating insight, understanding, and conceptual development. In-service programs and college curricula emphasize only what teachers are expected to use rather than what might develop good concepts. Omitting programming is a classic example where teachers as end users of software never see the construction process or design methods behind what they are supposed to learn. Today's design tools (for Web pages and such) are a modern example of where these issues still apply.

Teaching for conceptual understanding and higher-order thinking skills should not be only a part of teaching programming (Tu & Falgout, 1995), but also a fundamental goal of instruction for beginners in computing. Skills and even performance standards can still fail to generate important understandings, perspectives, concepts — integrated knowledge — that all contribute a fundamental and critical basis for problem solving and adaptability.

Focusing on conceptual development will still involve procedures and tasks just as focusing on discrete skills will likely yield some insights and discoveries. But instruction should yield a more complete, fundamental understanding of computing. Most programs and perspectives fail to recognize this important viewpoint and instead pursue skills and competencies to the detriment of understanding, insight, and problem solving.

It is common in other disciplines to speak of *education* rather than *training.* Conceptual development is often the primary focus in the study of science (Trumper, 1997). Even when the preparation of teachers is described in terms of training, science concepts are emphasized, not skills (Thompson & Schumacher, 1995). In spite of the procedures and skills inherent in science and mathematics, students are guided toward the development of a conceptual understanding as they are educated — not trained.

A training model targets activities and the software teachers will use. Much like an airline reservations clerk must learn the keystrokes and procedures for prescribed tasks, educational computing is similarly conceived. An education model, on the other hand, calls for activities and experiences that will yield a deeper kind of learning. Keystrokes and software familiarity would be incidental to the more important yield of experiences, much like those in science and mathematics, that develop understanding, concepts, problem solving, and critical thinking skills. An education-based program would provide experiences because of their educational value regardless of whether they are part of an anticipated skill set. Skill sets, tasks, and the procedural rituals of training will inevitably change and evolve far beyond the scope of any training experience.

Student teachers can be part of the problem as they, very often, prefer the training model. Contrary to any real value or longevity of such an approach, a more involved education presents an undesirable challenge. They prefer to simply be shown what to do. Guided tasks, prescribed procedures,