

Students, They Are A Changin'

Assoc. Professor Lewis E. Hitchner

Dept. of Computer Science

California Polytechnic State Univ.

San Luis Obispo, CA 94307

hitchner@csc.calpoly.edu

<http://www.csc.calpoly.edu/~hitchner>

Students are changing!

In my opinion, based upon observation of my students in recent years, the latest generation of university students in general and Computer Science students in particular exhibit these characteristics to a greater extent than earlier students:

- weaker mathematics and science background
- weaker problem solving skills
- weaker study skills and time management skills
- weaker ability to understand abstractions in solutions for general problems
- stronger preference for concrete solutions for specific problems
- decreased intellectual curiosity

So, "What else is new?", you say. This trend has existed for quite a while.

Here's my opinion of what's new about the changes occurring in the current generation of college students. They are the "Nintendo Students", and the fast food students, and the e-commerce students. I've borrowed the first term from Col. Richard Satava, US Army, who has been an invited speaker at many Virtual Reality conferences over the past 15 years. Col. Satava, an MD with a specialty in surgery, coined the term "Nintendo Docs" to refer to recent medical school graduates who are very much at ease with playing video games, using computers, and using other digital technologies -- technologies their predecessors and teachers in the medical profession are not typically at ease with, and in many cases technologies they are not knowledgeable about or even somewhat intimidated by. Dr. Satava uses this term in invited speeches he gives when promoting the use of Virtual Reality, tele-operations, and related technologies for use in battlefield medical operations. He emphasizes that the "Nintendo Docs" embrace such technology not only for military applications, but also for routine medical practice. He also emphasizes how few medical school professors and how few practicing medical doctors are prepared or able to teach and work with "Nintendo Docs".

Today's "Nintendo Students" are similar. They have grown up at ease with digital technologies. However, Dr. Satava's "Ninentdo Docs" moniker references the positive aspect of a new generation. There is a negative aspect as well. "Nintendo Students" are also accustomed to services, delivered via high technology, that provide a high response rate and often require little planning and input preparation by the requester: fast food, bank ATMs and credit cards, computer video games, surfing the Web, music and videos on demand, and most recently e-commerce. Some students even consider electronic mail a bit antiquated since chat rooms and instant messaging are more responsive -- better instant gratification. Computer science students are at the forefront of this generational change due to their keen interest in digital technologies and their extensive experience in using computers.

Ah yes, computers -- "the bane of Computer Science education"! "What's that?", you say. What could be

bad about computer technology, the technology from which we teachers and courseware developers all derive our livelihoods, our professional *raison d'etre*, and our passion to impart our knowledge to our students?[1]

Today's Computer Science students, in fact most high school and college age students, often use learning strategies that they've developed due to years of using computers, video games, and other quick-response-little-forethought-required services. They sometimes learn, or favor learning, according to the following principles:

- prefer very quick, brief, and visual answers to problems and information queries
- resort to trial and error problem solving techniques
- accept non-linear, hyper-linked, sometimes discontinuous paths as a good way to explore information
- prefer solving specific problems that have specific, concrete solutions
- expect teachers to provide explicit directions so that students need merely "follow directions"

They do not always learn, or are adverse to learning, according to the following principles:

- seek lengthy answers to problems and information queries, especially if the response is slow or consists of verbose text
- rely upon a methodical problem solving technique
- accept linear, beginning-to-end, connected paths as a good way to explore information
- enjoy finding general, abstract solutions that apply to a category of problems
- are willing to discover how and what to learn, without explicit directions from a teacher

These learning preferences are promoted by, and likely caused by, their experiences using computers:

- Desktop GUI's entice students to expect quick solutions to problems with little thought needed about the best input needed to get a result.
- Mistakes in commonly used desktop software often have little consequence because of "undo" commands and the ease of trying an alternative with one or two mouse clicks. For some students the only user mistakes that do have "significant consequences" are those that occur in fantasy worlds, such as getting "killed" in a video game.
- Lengthy online or printed manuals, and even online help pages that are longer than normal, are perceived as tedious and boring to read when all you want is the one, explicit, short answer that fixes your one, immediate, small problem.
- The World Wide Web and hypertext links, as well as popular print and video media, influence students to shun methodical approaches to problem solving and to favor jumping around or taking a serendipitous route to reach a goal.
- Popular computer software typically provides a specific solution for a specialized problem rather than a general purpose solution. In spite of the popularity of Linux, Unix's philosophy of simple, general purpose software tools (software filters) is becoming a less known, less appreciated principle of software development. In end user software applications CISC is in and RISC is out!
- Students are accustomed to accepting short hints from an online Wizard or an Office Assistant offering the default choice or a small number of most likely (in the mind of the software vendor) choices of program options.
- Paper and pencil solution plans and testing are dreaded, but mouse clicks and web forms that produce instant responses are embraced (as long as there aren't too many long dialog boxes).
- Distance learning and other online learning techniques that allow one to learn when, where, and at

whatever pace one desires are becoming more popular, especially when examinations contain a significant portion of true-false or multiple choice questions.

In my opinion a consequence of these life experiences and learning preferences of current students is that "higher level" learning -- categories above the Knowledge and Comprehension Levels of Bloom's Taxonomy[2]: Synthesis, Analysis, and Evaluation, and perhaps even Application -- are declining in value. Students have less exposure to learning at the higher levels. Many high tech businesses, that are the future employers of our graduates, are more focussed on finding solutions to immediate, specific problems -- problems that frequently can be solved by reusing or adapting a specific (concrete) solution to a previous problem -- than on developing general (abstract) solutions.

So, where do we Computer Science educators and courseware developers fit in this situation? Are we part of the problem? Or, are we part of the solution?

- Do we help our students learn to find solutions to general problems by studying specific, concrete examples that collectively lead to understanding abstractions? Or, do we allow them to believe that every problem has a specific solution that they can memorize or look up in a book?
- Do we encourage our students to learn factual knowledge, and then use well defined methods for applying it to alternative problems, evaluating and analyzing other solutions, and synthesizing new solutions?
- Are we guilty of not preventing them from following their natural tendency to believe that it is more important to know how to debug compiler syntax errors than it is to know how to build alternative implementations of an abstract data type and when and why it is best to use each alternative?
- Do we reinforce their belief in learning specifics over general principles by "teaching" specific versions of a specific software suite in our classes and textbooks (have you had an entering freshman ask you what version of the JDK you "teach" in your CS1 course yet?)?
- Do we push our students to find the quickest answer to the weekly programming assignments and, because we "don't have enough time" to cover all the required course topics, forget about teaching them to methodically plan their solution before they "write code" or to reflect on the successes and failures of their solution compared to alternatives after the assignment has been graded?

If we have the wrong answer to most of these questions, then I believe we are as much a part of the problem in the "Nintendo Student" age as we are a part of the solution. My position for this workshop is that:

- Due to the fact that computing applications are becoming more complex and more critical to the normal functioning of society, and
- Due to the fact that computing technologists of the future (our students of today), who are limited to solving problems based upon knowledge of solutions to previously solved, specific problems, will be poorly prepared for evaluating, analyzing, and synthesizing solutions to new, more general and more complex computing problems, and
- Due to the fact that Computer Science students are at the forefront of changes in life experiences that affect their learning skills because of their role as "Nintendo Students" and as leaders in using computing technology,
- Computer Science educators and courseware developers should refocus our teaching efforts by employing methods that promote learning problem solving techniques over learning solutions to a few specific problems. We should be willing to give up some "topic coverage" and focussing on specific technologies and specific applications of technology. We should be leading the way in high

technology education towards spending more time in the early years helping students learn general computing problem solving techniques so that they can cover the topics, learn new technologies, and learn new applications of technology -- after they've left the university and have begun their lifelong learning.

[1] See "If computers in school are the answer, are we asking the right question?", one of a series of full page ads placed in the New Yorks Times by the Turning Point Project, a coalition of 80 non-profit organizations seeking to draw attention and produce action on a small set of critical social issues, education being one of their primary ones. <http://www.turnpoint.org> and <http://www.turnpoint.org/technoad1.pdf>

[2] Bloom, B.S., *Taxonomy of Educational Objectives*, New York, Longman, 1956.