Too Technical?

The paper by Kafai and Peppler (2011) touches upon a near and dear topic – why don't more people know (and enjoy) programming? It is alarming to think that "…programming as a form of creative media production moved from near-universal presence to extinct practice" (Kafai & Peppler, 2011, p.95). If "Software is eating the world" (Andresssen, 2011) – and every company is becoming a software company, this should be a critical societal concern.

Kafai and Peppler's paper introduces many possible ideas to help us identify why: too technical, focus on consumption rather than creation, "geek mythology", costs, cultural perceptions, a lack of qualified teachers, and so on. This lack of programming ability is cause for concern both for those seeking jobs and for those seeking employees. And, "This problem is even worse than it looks because many workers in existing industries will be stranded on the wrong side of software-based disruption and may never be able to work in their fields again. There's no way through this problem other than education, and we have a long way to go" (Andreesen, 2011).

The idea that the domain is "too technical" was called out both explicitly and implicitly in the article. For instance, when discussing the need for simple tools that eliminate "…thorny debugging processes and the risk of syntax errors" (Kafai & Peppler, 2011, p97). Would a comparable request be that we replace grammar, spelling and writing skills and ask that students only "write" by using magnetic pieces with pre-written words focusing only on rearranging the blocks? After all, languages (whether it English, French or Italian) are difficult and carry risk of misuse.

The "geek mythology" phrasing was new and interesting. A supporting paper from Margolis and Fisher (2003) not only identify characteristics of the mythology, but they highlight that for many people, the view of the "mythical" programmer isn't really representative: "Although male and female students provide similar descriptions of the typical computer science student, a larger number of students than we had expected (male and female) say this image of the computer science student "is not me." Sixty-nine percent of the female computer science majors as well as 32% of the men perceive themselves as different from the majority of their peers" (Margolis & Fisher, 2003, p.17).

Margolis & Fisher continue beyond just identifying the mythos and challenges and make recommendations. Interestingly, and connecting to this week's reading, they callout the need for broadening – broadening the view of what is computer science, broadening the view of what constitutes good programming, and overall broadening the view of the culture. I'd imagine that this broadening also has to include all the other connection points available to programming and the avenues of expression and DIY that multi-literacy affords.

References

Andreessen, M. (2011). Why software is eating the world. The Wall Street Journal, 20(2011), C2.

Kafai, Y. B., & Peppler, K. A. (2011). Youth, technology, and DIY: Developing participatory competencies in creative media production. Review of research in education, 35(1), 89-119.

Margolis, J., & Fisher, A. (2003). Geek mythology. Bulletin of Science, Technology & Society, 23(1), 17-20.