# MAET Year Two: Final Reflection

Laurie E. Fernandez

Michigan State University

Masters of Art in Educational Technology

July 29, 2016

## Looking Back

It's summer! And you may be wondering why I am taking classes at Michigan State University instead of laying out at the beach. I have been wondering that myself. But, it's the sacrifice I am making to obtain that golden ticket, also known as, the Master of Arts in Education Technology. A group of 11 students, including myself, and two instructors set out on a two-week invasion of Erickson Hall where we tackled issues pertaining to how students learn, technology and leadership and educational research. We sure covered a lot of material in two weeks. Upon reflection it is amazing to me that so much of what we learned either effects the content we teach, the pedagogy or the way we teach, and most importantly, the technology we use. And through our TPACK lens, we know that considering a change in one area means that we need to consider how it affects the other two to optimally plan lessons that benefits a student's understanding of the material (Mishra & Koehler, 2009). I would like to journey back over these two weeks and reflect on what I have learned. More importantly, how I can use this information to be a better teacher using technology. How do I effectively teach technology with technology?

### Behaviorism

First, to be a more effective teacher, I need to know what learning is and what it looks like in my classroom. A common place to start is with Bloom's Taxonomy which has been used by generations of K-12 teachers. In 2006, it was revised by Lorin Anderson and David Krathwohl to reflect new core competencies associated with the integration of technology into the classroom and workplace (2006). But the concept is still the same. According to Bloom's Taxonomy, teaching facts to be memorized and assessed, now called *Remembering*, is the lowest level of learning. Psychologists described this type of learning as filling an empty child with knowledge and is associated with a learning theory called Behaviorism.

A traditional classroom is based on a behaviorist perspective where students are expected to memorize facts long enough to show how much they memorized on a conventional pencil and paper exam. Students who do well on the exam are positively rewarded with a good grade. Lee Shulman (1999) described three problems with this type of educational learning; *Amnesia* when students forget what they have learned, *Fantasia* when students don't realize that they misunderstand, and *Inertia* when

students know something but don't know what to do with it. Despite these drawbacks, there are aspects of the behaviorist perspective that are still relevant to effective teaching. And students still need to learn facts. Lee Shulman (1999) wrote, "You need facts to make sense; they are the basis for understanding, but they are never enough." I agree because it's important that I teach the facts about programming such as the intricate details of the syntax of C#. After all, there is no way to experience the fun of programming if a student can't write a syntactically correct program.

**Cognitivism**

If we continue to look at Bloom's Taxonomy, we see that the next level up is called *Understanding*. In programming, the syntax can be memorized, but without an understanding of programming concepts, students couldn't do anything with it. They need to know how to correctly write the code in an algorithmically correct order to successfully write a program. They need to know the when, where, why and how to use code to write a computer program that makes the computer complete a task. This requires students to have a deeper understanding of computer programming concepts. The study of how students understand is called Cognitivism.

With the cognitive learning perspective, psychologists are more interested in what is happening inside the mind of the learner. When a student is confronted with new sensory input, the information must be transformed into something that will fit in with the learners existing framework. Learning is a constructive process, building on what students already know. I was impressed by the teacher in the article called Teaching for Conceptual Change (Watson & Kopnicek, 1990). She taught her students about heat. But, instead of telling them what they needed to know, she let them actively construct an understanding of heat by allowing them to test their previous understanding of it, even if what they believed was erroneous.

There are several best teaching practices that come from the cognitive perspective such as building on the prior knowledge of students. According to Shulman (1999), "The most important single factor influencing learning is what the learner already knows". Another best teaching practice is that students cannot learn unless they are actively attending to new information. Cognitivists tell us that attention is crucial due to the limited processing abilities of the brain. Understanding takes "sustained

attention in order to study in enough depth" (Levstik & Barton,1997). Building a deeper understanding of computer programming requires my students to actively listen to and engage in my lessons. Unfortunately, it's possible that they could mindlessly copy code from my demonstrations or from someone else without listening to the explanation. As a result, students don't understand what they were typing. It's like knowing how to spell a word but not the meaning of it or how to use it in a sentence. Therefore, from a cognitive perspective, students will learn better if I assess student's prior knowledge and ensure that my students are focusing on the explanation of the code and not just typing it.

**The Creative Perspective**

To keep students engaged, I try to instill the feeling of fun in my classroom. I was glad to see that we discussed how creativity is an important aspect of effective teaching. Computer programming is a creative process and it can be fun but only if students understand what to write. Once they understand it, they can begin to apply the concepts in new and creative ways, writing any program they desire such as a game or a new cool app. It's interesting that at the top of Bloom's Taxonomy is Create. It's one of my top goals for my students; to be able to create a computer program or application.

In my spare time, I am very creative. I like to decorate my home and I quilt. I have been a crafter and sewer since I was little. It's the creative side of me that makes me a good teacher (Henriksen, Mishra, 2013). I like to try new things and create an atmosphere where students aren't afraid to fail. I encourage creativity. If I know one of my students is interested in drawing, I help them find a connection between drawing and programming. It's very motivating. An example of creativity in my classroom is the very first programming lesson I teach where students experience the amount of detail that goes into writing code. Their assignment is to build something with Legos®. But, building the structure isn't the ultimate goal. It's writing detailed instructions so someone else can build that structure using their instructions.

**Motivation**

Creating an atmosphere of fun and creativity is very motivating for students. It's what drives my students to pay attention to my demonstrations. Furthermore, students learn better when they have an internal locus of control or feel that success is the result of effort and hard work (Rotter, 1966). In other

words, they are motivated to learn because they know that they are in control of their own learning and therefore are engaged in the lessons. I can help develop a more internal locus of control by guiding their focus on their potential to learn, teaching them to enjoy the challenge and showing them that learning computer programming is hard work and takes effort but is achievable (Dweck, 1999). I also need to convince them that computer programming is not just for computer nerds.

I like the description of motivation by Mihaly Csikzentmihalyi (2010). He described it as "flow" when someone who is so totally involved in their work that they lose all track of all time and space. I often have this feeling when I am writing a program that is just a bit challenging. I know that I can do it but it will take some effort to get the program working. And I really look forward to that moment when the program runs successfully. "Flow" as described by Csikzentmihalyi is when the work is just difficult enough to make it challenging but not impossible and students feel they have the skills to accomplish the work. Students can feel anxious or worried if they don't have the skills to complete a challenging task. Boredom and apathy can happen if the challenge is too low for their skill level. The take away from this is that I need to continue moving the challenge level of the work I assign to help students feel motivated and in the "flow" so they can find the fun in computer programming as I do.

**Summary**

The two weeks in class felt like a whirl wind. We learned a lot, more than I can cover in this reflection. I learned that it's important to teach programming syntax but I need to continue to ensure that my students understand programming so that they can be creators of games or cool new apps. I learned that students need to build on prior knowledge and why it's important that students engage in my explanation of programming concepts, not just mindlessly copy what I write. I learned that students need the programming skills for assignments that are just a bit challenging to keep them motivated. You may have noticed that most of what I learned was not directly related to technology but to the pedagogy I use to teach computer programming. Looking through the TPACK lens, I now need to consider the technology that will work with my content and these new pedagogical ideas to create lessons that optimally benefit my students and make me a better teacher.

## Looking Forward

On the first day of class, students walk into my classroom eager to make the video games they love to play. They are excited and motivated to learn. It's quite a shock when they learn what it takes to write a program. Their first programming lesson teaches them how to write code that displays "Hello World" on their monitors. As I am demonstrating this lesson on the first day, I patiently wait for them to get excited about what they have done. "Isn't that cool?" I ask them. "Imagine what you can do with this?" But I can see the looks on their faces. I can see students thinking, "When do we make a video game?"

Attention and motivation are crucial in my classroom. Without it, students can mindlessly copy what I demonstrate without knowing what they are typing. I usually don't find out about it until about two weeks into the semester when I start getting a sense of who is actually understanding how to program and who is not. I start to see plagiarism or disgruntled students. I start to get the question, "How can I do this assignment when you haven't taught us how to write it?" I start to panic and ask myself, "Now what?" Do I start over from the beginning? Do I keep going hoping the stragglers will catch up? Or, do I help each of them individually? I need to find out sooner when a lesson hasn't hit its mark because moving on to more complicated concepts is almost impossible for these students. There are couple of ways I would like to do this.

### Concept Maps

To begin with, I would like to start the semester by assessing my student's prior knowledge. I have several ideas I would like to try. I already do a pre-assessment. But currently, I only use this to compare to the post-assessment to show growth. This may be a good place to start but I would like to do a little more. My idea is to begin the semester with a concept map activity. It will serve two purposes. One, show me what the students know and bring out any misconceptions they may have. And two, It's also a good activity to introduce students to the big picture. I sometimes forget to do this and I always appreciate it in classes that I attend. To learn more about concept maps, I will tap into the enormous amount of teaching experience available at my school by asking for help in my learning cohort. At KCTC, we have formed learning groups that meet to discuss educational issues and to learn new things from

fellow teachers. My cohort could help me learn how to teach with a concept map as well as help me choose whether to use technology for this or not.

**Design a New First Lesson**

Writing their first program can seem a little daunting to some students. So to begin with, I want to design a better first lesson that uses the skills they already have as well as being just a little bit challenging. I already have an idea to try. I would like to demonstrate a program that has already been written, give students a copy of the code and allow them to play around with it and modify it to see what happens. It will give students the freedom to explore without the fear of failing. It would also help those students who have a difficult time keeping up with typing. I really want students to hang on to that excitement and motivation they had when they first entered my classroom. To learn more about designing a good first lesson, I plan to connect with other high school computer teachers. I might be able to find a Twitter feed or a blog. I'll have to do some researching on the internet to find one.

**Developing an Internal Locus of Control**

Students are motivated to learn when they are in control of their own learning. Some teachers believe this means that the curriculum is available online where students work through the lessons at their own pace. But I am not a fan of this type of learning. To me, it means something different and there are several things I plan to do that support my student's internal locus of control. One is to offer options to students, maybe allow them to choose between different types of assignments. Another, is to be careful using praise. Praising students for their ability and not their effort may get the opposite reaction. The last idea I have is to help students help themselves. I am usually too quick to show students where they have made a mistake when they struggle to write a program. Instead, I would like them to struggle with it for a little while longer and then start giving them clues where to look. I plan to scaffold their ability to solve their own problems. To implement this idea, I would have to jump right in and do it. This is one of those things that I can't learn from a book. I have to learn by doing.

**Purchasing a Digital Collaboration System**

My students live in a connected world. They communicate with friends often, both in person and on their mobile devices. They play games together and share information about how the games are

played. But during school, they work independently writing individual programs. We know that children learn best when they take part in joint activities with teachers and more knowledgeable peers (Levstik & Barton, 1997). Creating a real work environment in my classroom would help students better understand the purpose for what they are learning. I am currently working with my administration to purchase the Extron TeamWorks System which is a digital collaboration system for my classroom. It will allow students to work together and collaborate on projects. They will be able to discuss what they are doing with a team on a shared monitor and ask their fellow teammates for help when they need it. If I'm not able to get it this year or next, I'll keep working at it. I already have a group of Extron support personnel to help me build this system and help me learn to teach with it.

**Summary**

The real value of adding technology to improve instruction is to start with the big ideas of computer programming. What is it that I actually want to them to know? Ok, I know that they need to learn syntax and variables and conditionals and looping. But, let's look at even bigger ideas. Computer programming gives students a process that allows them to think of complex problems in a way that can be resolved by a computer. It teaches them to think. It shows them that they have the power to create. It's a problem solving approach that they can bring to any field they choose to go into.

**References**

Anderson, L. W., & Krathwohl, D. R. (Eds.). (2001). A taxonomy for learning, teaching and assessing: A revision of Bloom's Taxonomy of educational objectives: Complete edition, New York : Longman.

Dweck, C. (1999). Caution: Praise can be dangerous. American Educator, 23(1), 4-9.

flowinstitute. (2010, October 20). Mihaly Csikszentmihalyi – Flow [Video file]. Retrieved from https://www.youtube.com/watch?v=JjliwSJGDiU

Henriksen, D., & Mishra, P. (2013). Learning from creative teachers. *Educational Leadership.* 70(5). Available at: http://www.ascd.org/publications/educational-leadership/feb13/vol70/num05/Learning-from-Creative-Teachers.aspx

Levstik, L. S., & Barton, K. C. (1997). The theory behind disciplined inquiry. In Doing History: Investigating with children in elementary and middle schools (pp. 9-16). Mahwah, New Jersey: Lawrence Erlbaum Associates.

Mishra, P., & Koehler, M. J. (2009). Too Cool for School? No Way! Learning & Leading with Technology.

Rotter, J. B., (1966). "Generalized Expectancies for Internal Versus External Control of Reinforcement." *Psychological monographs* 80 *ProQuest.* Web. 27 July 2016.

Shulman, L. (1999). What is learning and what does it look like when it doesn't go well. *Change, 31*(4), 10-17.

Watson, B., & Kopnicek, R. (1990). Teaching for Conceptual Change: Confronting Children's Experience, Phi Delta Kappan (pp. 680 – 684).