Select an EF challenge that you would like to address and briefly explain the challenge and why you chose to focus on it.

Introductory programming is a critically important course for students majoring in most engineering disciplines, but especially for electrical and computer engineering majors. Unfortunately, students often emerge from an introductory course lacking the independent problem solving skills they need to be successful in future courses. Their metacognitive skills in the domain of programming may be weak, as strategies that may have worked for them in other areas (such as working textbook problems to assess their skills) are not necessarily applicable to programming courses. This is particularly true for hardware-focused programming courses such as embedded systems and hardware description languages, as these fields have not developed a wide selection of textbooks with assignments students can use for self-assessment.

One major roadblock to metacognition is student reliance on help from peers and tutors. In other courses such as math and science, most students have, over time, developed metacognitive strategies that allow them to participate in group study while maintaining accurate self-assessment of their own skills. However, this does not seem to translate to programming courses. Interactions with peers and tutors often result in the peer or tutor supplying critical insight for solving a programming problem, which could be anything from providing the initial design plan to finding bugs in code. The student is deprived of the chance to develop these, and is left without means to self-assess their abilities.

Identify the type of product/support that you are developing and explain why this support mechanism is appropriate for addressing the EF need and is appropriate to your professional setting.

University tutors are experienced programmers, but they have little to no pedagogical training. It is therefore not surprising that when a student presents a problem, the tutor shows them how to solve it. The tutors are interested in seeing their students succeed, so I believe they would welcome guidance that would help them help the student in a way that builds the student's problem solving abilities and self-assessment skills.

I chose to create a tutoring "tip sheet" that applies the Landmark Non-Directive Coaching model (Dahlstrom-Hakki, 2016) to the tutoring process. A previous study of executive function coaching directed towards students with learning disabilities and ADHD finds that coaching helps students by "providing information, offering options for strategies and skills, and asking questions that prompted them to create their own solutions and take increased responsibility for their own actions" (Parker & Boutelle, 2009). Many of our students do have diagnosed learning disabilities or ADHD, but I believe that this benefit could extend to all students as a Universal Design practice. The improved self-awareness and self-determination reported by the students in the study aligns with the goals of academic tutoring.

Cite and briefly explain the theory or approach that is guiding the development of your product/support.

The suggestions in the tip sheet reframe the role of the tutor from a problem-solver to a coach asking curious questions, informed by the tutor's experience with programming challenges. The tutor asks the student to visualize the operation of the program and reflect on contexts in which similar problems have been encountered. Strategies such as solving a simpler problem and building to a larger problem, using tools effectively, and creating checklists for troubleshooting are shared by the tutor for the student to use to solve problems.

The tutor is supplying *metacognitive modeling*, identified in the context of another STEM discipline (biology) as: "Showing students explicitly how you, as a biologist, think procedurally in solving a problem—how you start, how you decide what to do first and then next, how you check your work, how you know when you are done" (Tanner, 2012). This metacognitive modeling is being done in a context which is new to the student, and the student needs this guidance to develop metacognitive skills that apply to this new discipline. Multiple articles have described metacognition strategies as "not generic across subjects, and attempts to teach them as generic can lead to failure to transfer" (National Research Council, 2000) and they are therefore "most effective when it is adapted to reflect the specific learning contexts of a specific topic, course, or discipline". (Chick, n.d.) The observation that the metacognitive strategies for other STEM disciplines developed over a student's educational career may not transfer to programming is therefore not surprising, and supports from tutors and instructors are all the more important. A more powerful strategy would be to share a version of this tip sheet with instructors and students as well as tutors, since awareness of non-directive coaching approaches and metacognitive modeling could potentially improve all of the learning environments, from the classroom to the tutoring office to the student lounge.

References

Chick, N. (n.d.). *Metacognition*. Retrieved November 10, 2018, from Vanderbilt Center for Teaching and Learning: https://cft.vanderbilt.edu/guides-sub-pages/metacognition/

Dahlstrom-Hakki, I. (2016). An Overview of Non-Directive Coaching at Landmark College. Landmark College Institute for Research and Training. Retrieved from MPDL 631 - Understanding and Supporting Diverse Learners, Canvas site.

National Research Council. (2000). *How People Learn: Brain, Mind, Experience, and School: Expanded Edition.* Washington, D.C.: National Academies Press.

Parker, D. R., & Boutelle, K. (2009). Executive Function Coaching for College Students with Learning Disabilities and ADHD: A New Approach for Fostering Self-Determination. *Learning Disabilities Research & Practice, 24*(4), 204-215.

Tanner, K. D. (2012). Promoting Student Metacognition. *CBE Life Sci Educ., 11*(2), 113-120.

"Coaching" Tips for Digital and Embedded Programming Tutors

Programming classes aim to prepare students to be independent problem-solvers. However, many students don't know where to draw the line when it comes to getting help with programming classes—and the wrong kind of help can prevent students from developing their problem-solving skills. Students also need to be able to assess their own skill in applying the course material, in order to prepare for exams. Here are some tips on how to respond to the most common student issues in a way that "coaches" students into finding their own solutions.

I don't know where to start!

- *Let's read through the lab assignment together, stopping when we get to a term that is unfamiliar*.
    - Ask curious questions as you read, to make sure the student understands the assignment.
- *Draw a picture of the program's operation on your hardware.*
    - Have the student make a list of the steps the program should take. Also list the hardware setup tasks the student will need to perform in hardware and software.
- *Explain to your hardware what it should be doing, using your own words.*
    - Ask the student to identify parts of the pseudocode that they already know how to write in code.
- *Could we do a small part of this lab? A simpler version of the assignment? What small parts of this lab could you do right away? Now that you've done this mini-assignment, can you think of anything that you covered in class related to <additional actions>?*
    - Students can often extrapolate a mini-assignment into the full assignment by thinking of the changes they would need to make. Simpler code is also easier for the student to debug.

Does this program look correct?

- *Give me the basic idea of what your overall program is doing.*
- *What is this part of your program doing?*
- *Why did you decide to use <technique, construct> here?*
    - These curious questions will help the student reflect on choices made. It's likely that you will now identify some problems with the overall approach.
- *What would happen if…*
    - Present situations that would reveal errors in the code.
    - If the problem is fixable within the current structure of the code, ask the student what change could be made to deal with the problem. If it requires a large rewrite, it's best to reveal that up front.

This program isn't doing anything/isn't doing what it is supposed to do.

- *Let's start at the beginning of the code and walk through it, thinking about what the processor would do. Tell me what the processor would do here.*
    - Many times, students do not realize the time scale of their instructions, and walking through it in the context of realistically timed inputs will help the student discover mistakes and deepen the student's understanding of processor operation.
- *Do you have a diagram or description of the required hardware connections? Let's go through the required connections, making sure we understand them and that they have been set up correctly.*
    - It's probably a good idea to go through hardware connections right when you begin troubleshooting. The student should tell you what hardware connections are required; don't offer up hardware solutions if they have not been provided.
    - If there is a missing or incorrect connection, try using directed questions to help the student discover the error—don't start changing connections yourself.
    - Discuss ways to discover hardware mistakes, by considering output or measuring voltages/connectivity.
- *Do you have access to a debugger or system viewer? Do you know how to set breakpoints/step through code? Where do you think the problem might be?*
    - If available, a debugger tool is extremely useful for independent learning in programming courses. Students should learn how to use the debugger efficiently to look into the operation of their code. Tutoring time spent on teaching students how to use the debugger will pay off in the future.
    - For hardware description languages, the viewer that allows students to see the inferred hardware is critical for their understanding of their own design as well as the discovery of errors. Checking this viewer should be one of the first troubleshooting steps. Tutoring time spent analyzing the inferred hardware in the viewer and identifying signs of mistakes will help the student develop independent problem solving skills.
- *This project requires that you follow steps in a design flow. What are they? Let's go through the steps and make sure each one was completed.*
    - The student should make a record of the design flow steps, and use the checklist every time they complete a design until it becomes second nature.

My code won't compile. I have an error message.

- *Have you seen this error message before? What was wrong when you saw this before?*
- *Let's look at the message to see if there are any hints or keywords that might point us to the problem. Does the error message include any names of your variables or functions?*
    - Certain errors pop up frequently, especially for students who are learning new syntax. With the student, develop a list of common situations that could cause this error.

- Be sure that the student knows that one code error can cause many error messages, and until the student has developed expertise in reading error messages, the code should be recompiled after each fix.

Am I ready for the exam?

- *Do you have a practice exam? Do you have a list of topics covered?*
  - If the instructor does not offer a practice exam, the student needs to ask the instructor about the types of questions that will appear on the exam. It's important that the student knows whether the exam will focus on problem solving/coding from scratch, hardware function, or if there will be particular skill problems (number representations, timing diagrams, etc).
  - If there is no practice set, try to come up with some practice problems that the student can work on which cover these topics at a realistic level of difficulty.
  - Suggest that the student form a study group, in which every student comes up with 2 practice problems that the rest of the group must solve.
- *What resources are you allowed to use on the exam? Are you solving problems without using resources like notes that won't be allowed on the exam? Are you checking the answer key as you work the problems?*
  - Students will often consider themselves prepared if they understand the answers to practice problems. This is a major error in their self-assessment of their preparedness for exams. Students must be able to solve problems, coming up with their own original solutions, without help or disallowed resources.
  - Give the student some time to work through a practice problem independently before jumping in to help. Share your assessment of the student's preparedness in a positive way, such as "I think we need to work some more on <topic>" or "You'll need to do <task> without help on the exam, so let's try another one of these practice problems".

