**April 3,2018 Enoch > Peter & Rick**

Hi Peter,

As you may remember, I'm working on computational thinking and robotic coding badges for my master's project. As part of my evaluation plan, I have included a plan for expert review from a CT/coding expert, and I would love it if you would have time and be willing to do that for me.

I am developing 3-4 main badges along with accompanying instructional material for them. I am mainly interested in the efficacy of the badge requirements as an assessment of competency and the effectiveness of the instructional material in developing those competencies.

There would be 3-4 badges and probably 9-10 tutorial web pages. I anticipate it would take you only a few minutes to review and comment on each one.

Would you be willing to do act in this role?

If so, I can send you my badge drafts right now and my tutorial drafts as I complete them.

Let me know.

Thanks!

**April 3, 2018 Peter > Enoch & Rick**

Absolutely.

**April 3, 2018 Enoch > Peter**

Peter,

Thanks for agreeing to be my expert reviewer. Here are the badge drafts I would love your feedback on. Tutorials/Instructional Content to support these badges is forthcoming. You should have comment or edit access to all of the documents below, but let me know if you run into a problem. I would love for you to provide any comments you have using the comment feature of google Docs. The central question I would like you to evaluate is as follows:

Does submission evidence required by the badge provide adequate information to determine learner competency in the intended learning outcomes (ILOs)?

The ILOs are listed in the Badge Information section of the badge drafts. Of course, I would also love any other feedback you have!

Understanding Computational Thinking Badge

- Badge Draft
- Badge Submission Form/Worksheet
- Understanding Computational Thinking Quiz (Google Form)

Teaching Computational Thinking Badge

- Badge Draft
- Badge Submission Form/Worksheet
- Supplemental Video for Requirement 2 (You don't necessarily have to watch this whole thing, but if you can scrub through it and let me know if you think it's a good selection for the requirement, that would be great)

Teaching Early Coding - Level 1

- Badge Draft
- Badge Submission Form/Worksheet

Teaching Early Coding - Level 2

- Badge Draft
- Badge Submission Form

**April 4, 2018 Peter > Enoch & Rick**

OK, I've gone through and added comments to several of these documents. A few comments. First, Rick, please explain to me the role of rubrics in badging. I know you've discussed this before and I've heard your comments on rubrics, so I know there's some thought there about their place. However, the criteria for rating things is not clear and rubrics certainly have a place in making criteria clearer.

My other main comment is to better understand what computational thinking is, what problems it can help solve. There's a difference between scientific problems (looking for causality), engineering problems (looking for a workable solution) and computational problems (looking for an automated solution). CT loses its significance if it becomes all things to all people. This is unfortunately the route that almost all ideas take when they become popular. We want to show people how ubiquitous they are, so we start applying everything to that. In truth, there is still place for these other types of processes and I believe it behooves teachers to be able to distinguish and understand when something lends itself to a computational solution and when it does not (of course, scientific, engineering, and computational process can be mixed, but let's try to keep them separate right now for the sake of argument).

Finally, if you're ok with it, when you think you've gotten these to a reasonable point, I would love to have 3 other experts look at them who might be interested in promoting and using these badges themselves: Anne Leftwhich, Aman Yadav, and Jesús Moreno-León. All 3 are teacher educators focused on preparing CT-capable teachers at the elementary and middle-school levels and would be excellent

reviewers (and promoters) for these badges.  Having their stamp of approval would go a long way in drawing attention to these, methinks.

**April 4, 2018  Rick > Peter & Enoch**

Rubrics in badging …

You can have rubrics in badging if you want … we can use any assessments we want to and just issue a badge at the end. However, a couple of things:

- most of the badging platforms don't support rubrics. It's kind of an all or nothing. So you'd have to do the rubric in a different platform, like Canvas.
- in IPT 371 we've taken a mastery approach with the badges. Not all badging systems do this, but that's what we've done. So while they can get a B or C in the class, the badge means they mastered it all. But we let them resubmit if they want until they show mastery. So that's the main reason we haven't used rubrics.
- The other reason why we haven't used rubrics is because most of the skills in IPT 371 are pretty micro, and didn't seem to need rubrics. For example, if they are making a website and it says to add a blogging page to your site … well, you either got it or you don't. So it just never made much sense in our class.

That all being said, if rubrics make the most sense for the computational thinking badges, we could do that, but it'd be out of the norm for the class, and these are the reasons why.

I think your other comments are great.

**April 4, 2018 Peter > Enoch**

Here's a video McKay put together to try to explain what I was talking about in regards to types of problems (scientific, engineering, computational).  It doesn't quite make the difference as clearly as I'd like, but I think it might help to get the ideas across.
Here you go!
https://youtu.be/JbpZDu4Mjss

**April 5, 2018 Enoch > Peter & Rick**

Thanks for your help with these!  Your input is very valuable and I think it will make for much stronger badges.  I do have a couple questions about the Computational Thinking badges.  I want to understand your concerns better before I make the adjustments you suggested.  (Sorry in advance for the long email, but I'm hoping this will really help me to crystallize my understanding of what we're looking for).

First, Do you feel strongly that the badges need to be assessed with a a task-specific rubric with a detailed rating scale?  Or is the checklist type of rubric that Rick has described sufficient?  What about the

possibility of assigning a point value to each requirement, for better weighting and to allow the reviewer to have a scale. For example, it might look something like this:

| | | |
|---|---|---|
| 1.1 | Coding Concepts (10 points) | Choose a block-based programming platform (e.g., Scratch, code.org) and build one or more block-based code sequences to show that you can apply ALL of the following coding principles (you may build these sequences either as part of an activity, or as a stand-alone project):<br><br>● Commands<br>● Loops<br>● Nested loops<br>● Events<br>● Conditionals<br><br>Take one or more screenshots of your sequence(s) and include them on your submission form along with annotations that describe which coding principles they demonstrate. |

The reviewer could reason that since there are 10 points and 5 coding concepts, the demonstration of each coding concept is 2 points, and award points accordingly. If only 4 concepts were accurately represented, they would get 8 points.

Second, Can you help me better understand the concern about types of problems and using specifically computational types of problems? Maybe a good article or two about this? I haven't run into this idea before and want to understand it better. Mostly what I have come across is the idea that you could approach ANY problem computationally. I really like the distinction between CT, the scientific method, and design thinking; but couldn't you argue that you could approach any problem in any of those ways, but that you would expect different types of results?

For example, in my current Understanding CT badge I pose the following scenario:

"As an educator in the 21st Century, you recognize that what you know is less important than your ability to learn and retain information. Knowledge that is “state of the art” or “best practice” in both pedagogy and your subject area may be quite different next year than they are today. As a conscientious educator, you feel that it is vital to stay abreast of new developments in your field. This is a worthy goal, but there are various obstacles to achieving it. How can you, as a 21st Century educator, use computational thinking to tackle this problem?"

I could approach the problem computationally and come up with a solution/algorithm that looks something like this:

**1.**
**2.**

| # | *Step Description* | *Agent* |
|---|---|---|
| 1 | Program Google Alerts to deliver news to me about [specific search strings] each Friday at 4:00pm | Me |
| 2 | Set aside 4:00-4:45 each Friday in Google calendar for professional development reflection & reading. Add a notification | Me |
| 3 | Google Alerts sends me a message. | Compute r |
| 4 | My calendar reminds take my professional reading time. | Compute r |
| | Write a reflection on how last week's best teaching practice went. | Me |
| 5 | Skim the articles and read most relevant. | Me |
| 6 | Choose a research-based best teaching practice to apply in my classroom the next week. | Me |
| 7 | Apply the teaching practice I chose | Me |

Approaching the problem in this way yields a solution that is automatic or partially automatic and can be "executed by a human or by a machine" or both (Wing, 2006).

I could also approach the problem scientifically and follow a process something like this:

| | |
|---|---|
| Question | What constitutes "best practice" in my field? |
| Research | Find evidence that X teaching practice yields high test results. |
| Hypothesis | If I successfully execute [X teaching practice] in my classroom, my students' test scores will improve. |

| | |
|---|---|
| Test | Execute X teaching practice. |
| Data Analysis | Compare my students' test results this year to those of last year. |
| Conclusions | My students test scores did not improve, so I should reconsider my implementation or abandon the practice. |

Approaching the problem this way yields hard data that supports my hypothesis. It does not, however, work very quickly and does not automate anything for me. If I report my findings, it also contributes to theory about good teaching practices.

Finally, I could approach the problem with design thinking and follow a process something like this.

| | |
|---|---|
| Empathize | I Observe, interview, and poll my students about what they would like to see change in the classroom. |
| Define | Based on insights from my students, I decide to design a teaching intervention that will better help students to develop much needed social skills. |
| Ideate | Possible Solutions:<br>● More socratic discussions<br>● Share writing more often<br>● Watch Ted Talks on communication |
| Prototype & Test | I decide to share writing more often in class. I execute a lesson in class where we share writing with each other and ask students what they think about it. I then refine my lesson plan and repeat the test iteratively until I'm happy with the intervention. |

Approaching the problem this way emphasizes creating a design that works for my situation, but may or may not take as much advantage of existing research, hard data, or automated methods.

Is there something I'm missing here? Are there problems that really could not be approached computationally?

Thanks!

**April 24, 2018 Enoch > Peter**

I never heard back from you on this last email. Potentially it was just too much to respond to in email form? Would it be a possibility to set up a video chat with you sometime in the next couple days? I want to start sending you my tutorial material, but want to be sure to have the badge (assessment) nailed down first so I know what I'm designing to. Getting your feedback on the questions in this last email would be extremely helpful.

Thanks!

**April 25, 2018 Peter > Enoch**

So sorry, Enoch. I read this and thought I'd responded to it. I realize now that wasn't the case (I must've read it on my phone when I was out and about and thought, "here's what I'll say when I sit down at a computer," but never did it. I'll go through these now and get you the feedback.

See replies below

On Thu, Apr 5, 2018 at 7:12 PM, Enoch Hunsaker <enoch.hunsaker@gmail.com> wrote:
Peter,

Thanks for your help with these! Your input is very valuable and I think it will make for much stronger badges. I do have a couple questions about the Computational Thinking badges. I want to understand your concerns better before I make the adjustments you suggested. (Sorry in advance for the long email, but I'm hoping this will really help me to crystallize my understanding of what we're looking for).

First, Do you feel strongly that the badges need to be assessed with a a task-specific rubric with a detailed rating scale? Or is the checklist type of rubric that Rick has described sufficient? What about the possibility of assigning a point value to each requirement, for better weighting and to allow the reviewer to have a scale. For example, it might look something like this:

| | | |
|---|---|---|
| 1.1 | Coding Concepts (10 points) | Choose a block-based programming platform (e.g., Scratch, code.org) and build one or more block-based code sequences to show that you can apply ALL of the following coding principles (you may build these sequences either as part of an activity, or as a stand-alone project):<br><br>• Commands<br>• Loops<br>• Nested loops<br>• Events<br>• Conditionals<br><br>Take one or more screenshots of your sequence(s) and include them on your submission form along with annotations that describe which coding principles they demonstrate. |

The reviewer could reason that since there are 10 points and 5 coding concepts, the demonstration of each coding concept is 2 points, and award points accordingly. If only 4 concepts were accurately represented, they would get 8 points.

> I would look at this from two perspectives: (a) teacher/grader, and (b) student. Rick mentioned that you use a mastery approach in the class, which I think is fine. However, how does one know when mastery has been reached? The clearer you can be, the more obvious it is for both the reviewer and the student to determine that mastery has been reached. In some cases, that means add point values, but in other cases, points just add another layer when the task itself is evidence enough. For example, I might say, "run a 40-yard dash in under 5 seconds." I don't need any extra points to know how well I've done. In fact, I'll know exactly how close I am and how much I still lack by looking at my performance. In other cases, it's less clear.
>
> In this case, if someone were to use each of these five elements, I could simply look for their presence and know that they've been used. So, points may not be necessary. The fact that you ask people to annotate what they've done further acts as evidence that the student understands how they've used each of these elements.
>
> The one problem is that you should probably remain consistent across all your rubrics in the same course. So, if there are some activities that require a point value to help both the teacher and student know what "mastery" consists of, you may need to add it to all. In that case, you'd need to clarify the point breakdown in activities like this one that are comprised of multiple parts. For example, you might say, "2 points for each principle demonstrated." The benefit with that is that you might be able to award a point for its presence and another for its judicious use, which takes the task a bit farther.

Second, Can you help me better understand the concern about types of problems and using specifically computational types of problems? Maybe a good article or two about this? I haven't run into this idea before and want to understand it better. Mostly what I have come across is the idea that you could approach ANY problem computationally. I really like the distinction between CT, the scientific method, and design thinking; but couldn't you argue that you could approach any problem in any of those ways, but that you would expect different types of results?

For example, in my current Understanding CT badge I pose the following scenario:

"As an educator in the 21st Century, you recognize that what you know is less important than your ability to learn and retain information. Knowledge that is “state of the art” or “best practice” in both pedagogy and your subject area may be quite different next year than they are today. As a conscientious educator, you feel that it is vital to stay abreast of new developments in your field. This is a worthy goal, but there are various obstacles to achieving it. How can you, as a 21st Century educator, use computational thinking to tackle this problem?"

I could approach the problem computationally and come up with a solution/algorithm that looks something like this:

| # | *Step Description* | *Agent* |
|---|---|---|
| 1 | Program Google Alerts to deliver news to me about [specific search strings] each Friday at 4:00pm | Me |
| 2 | Set aside 4:00-4:45 each Friday in Google calendar for professional development reflection & reading. Add a notification | Me |
| 3 | Google Alerts sends me a message. | Compute r |
| 4 | My calendar reminds me to take my professional reading time. | Compute r |
| | Write a reflection on how last week’s best teaching practice went. | Me |
| 5 | Skim the articles and read most relevant. | Me |
| 6 | Choose a research-based best teaching practice to apply in my classroom the next week. | Me |

| | | |
|---|---|---|
| 7 | Apply the teaching practice I chose | Me |

I love the example of using different methods to approach the problem. I think you're right that the same problem could be approached in a variety of different ways and that each would yield different (though complementary) results. I guess my concern has been that I see us saying, "CT! CT! CT!" to teachers and some people are turning everything into it, and in some cases diluting what it means. There are those (e.g., Peter Denning) that have been very reticent to call the patterns like the one above an "algorithm," when in truth they are "heuristics." In our efforts to simplify things as teachers, we sometimes oversimplify and then the very thing we are promoting begins to lose some of its meaning.

I believe that many many problems that aren't currently approach computationally could be. As for a good article on this, you're in luck, b/c someone just presented on this recently. Check out this article: https://link.springer.com/chapter/10.1007%2F978-3-319-71483-7_15 on solving everyday challenges in a computational way of thinking. I'm a believer in that not every problem should be only approached computationally, just as not every research question should only be approached with a strict scientific method, and that doing so leaves you open to potential blind spots.

Probably the best solution is to help your students understand what it means to approach a problem computationally. Your side-by-side examples of the same problem approached from different perspectives is probably a great way to do that.

**April 25, 2018 Peter > Enoch**

One more thing, on the first item. If none of your badges use a point breakdown or rubrics, then I'd just figure out how to make the evidence speak for itself on each thing they're supposed to do. It's best to maintain consistency across the badges, I think. And it's probably possible to make the manifestation of each a good piece of evidence itself.

**April 25, 2018 Enoch > Peter**

This is great, thank you!

After reading through your feedback, the Standl article you sent, and learning more about algorithms vs. heuristics, I think I understand a little better your concern about the example in the worksheet.

My problem still is this:

Since my audience for this badge is teachers, I want the examples to resonate with them and their real world. I believe that this will help teachers to represent CT as valuable to their students in a more authentic way. The problem I'm having is thinking of (or finding) educational problems that lend themselves to a strict algorithmic solution, mostly, I think, because teaching is a human endeavor, and humans will not always behave in strictly predictable ways. I can think of a number of educational problems in which a heuristic (such as the one I described above) can provide enough scaffolding (and even some degree of automation) to a solution, but also build in enough flexibility to be a viable solution for an educational problem.

So, here are my questions:

- Does producing a heuristic rather than an algorithm (which seem to have a lot of overlap) really constitute something completely different than CT, even if other elements (e.g., decomposition, abstraction) remain intact?
- 
- Can you think of any educational problems you have been able to solve using a true algorithm/computational thinking? Am I barking up the wrong tree in trying to get them to think about these problems computationally? The alternative, I suppose, would be trying to get them to think about CT in the context of a discipline they are trying to teach as teachers, rather than the discipline of teaching itself. Do you think that is a better approach?

**April 25, 2018 Peter > Enoch**

Enoch,

Great discussion. Initially, I was going to say, "definitely have them do it for their own content areas b/c that not only helps them to see the value of solving a problem in a computational way, but gives them materials they can turn around and use in their classroom." But then as I walked to the computer to respond to this, I thought of a bevy of teacher examples. Here's a common teacher problem that you might solve using computational thinking:

1. As a teacher, you send out a newsletter to students' parents each week with class updates. You love to personalize these letters so that parents can see what their own child has done and how s/he's doing in the classroom, and issue a personal challenge based on each student's individual weekly goals. Parents love it, too, and have expressed how much they appreciate the personal touch and feedback. The problem is that it takes so much time to create a personalized letter for 30 students each week. If you don't find a better solution, you may have to send out a more generic letter.
    1. Computational solution:
        1. Decompose: You look at the problem and realize there are several parts to this: (a) collecting individual student data, (b) updating the newsletter with class-level info, (c) customizing specific parts of the letter, and (d) sending each letter out to the correct parent(s)/guardians.
        2. Pattern-finding: There are specific parts to the letter that you realize repeat. For example, each week, you report on goals, challenges, and successes for

individuals as well as the class. You also report on upcoming homework, which is the same for everyone.

3. Algorithm: Even though your letters are personalized, they follow a specific formula. In fact, your letter is replete with variables where you simply replace one students' name, parents, goals, etc. with another's. You realize your entire newsletter is basically a simple algorithm. [show example paragraph]
4. Automation: You don't necessarily consider yourself a tech pro, but you remember in your awesome IP&T course at BYU that you learned about spreadsheets and mail merges. A mail merge allows you to craft a message and to replace certain variables with information stored in a google sheet. You can't remember how to do the merge off-hand, but you google it and find the "Mail Merge with Attachments" free add-on. You use this to create a personalized letter that, when you export, will automatically be sent to the address you have in your spreadsheet for each parent.
5. Analysis/Evaluation: After sending out the letters, you wonder why you didn't approach things computationally earlier. You now have a more personalized letter and it takes 1/3 the time (or less) than it used to. What's more, automating this solution has given you more time to work on the interpersonal aspects of teaching. You realize you can take it a step further, though; you've been entering students' goals into the spreadsheet from your personal discussions with them. Instead, you realize that you can send a google form to students, and automatically merge their answers from the spreadsheet that reports the form data. So, each week on Thursday, you now have students write a brief reflection (reporting their email and name, so they can get credit and you can connect it to your data). You merge this data with the rest of your spreadsheet data and create an even more customized letter without increasing your overhead. You may even have time to catch a show this weekend!

The curious thing about this example is that we taught it to students in the very course that you're currently working with 20 years ago. The mail-merge happened in MS Word and MS Excel (which is still possible and works well).

Without going into detail on the solutions, here are 11 other potential teacher problems that lend themselves to a computational solution. Many of these come from personal experience, either as a teacher or as a parent observing my kids' teachers' solutions.

1. Trying to understand student performance trends.
2. Item analysis on a recent test (use CT to create an algorithm to automate the reporting of each item, identify problematic items, and improve the test)
3. Collecting and displaying examples of student work over time.
4. Regularly gathering, filtering, and sorting current events related to specific units so that your materials are relevant and up-to-date
5. Documenting your pre-school students' work across the year
6. Writing reminder emails ahead of time that can be sent out to help students with upcoming assignments and tests (see Google boomerang add-on)

7.
8. Making randomized group assignments to put students into teams, but making sure that everyone gets to work with everyone else at least once throughout the year.
9. Providing students with practice tests/quizzes that pull from a bank of items so that the quiz is dynamically generated each time.
10. Setting up a classroom point system that will alert you and the students where the class is at and what is lacking in order to reach the goal.
11. Keeping your online materials synced with your personal computer (basically, this is what Github is for).
12. Collecting, analyzing, and displaying the results of a weekly survey so that you can do "just-in-time" teaching and know what to focus on in class that day.
13. [BONUS: many things that can be solved using IF This Then That (IFTTT): check out http://www.appsinclass.com/ifttt.html]

You can see a lot of these things have to do with management tasks, but that's precisely what CT can be helpful with—repetitive, predictable tasks. But this gives us more time to do what humans do well—social interaction. As Gary Kasparov noted in a TED talk, working with automated, intelligent systems allows us to be more human, not less (https://www.ted.com/talks/garry_kasparov_don_t_fear_intelligent_machines_work_with_them).

You'll also note that most work best with a solution that requires the teacher to know a little bit of programming. We can't just tell teachers that they've already been doing CT and don't have to learn anything new. They need to learn the language of automation. In most of these cases, it's actually rather simple automation with languages that they might already be familiar with or might quickly learn (e.g., google sheets, gmail, IFTTT), but being familiar with computing concepts will (hopefully) help them to not fear figuring out the syntax for whatever language will help them to solve their problems.

Hopefully, this gives you some teacher ideas. Basically, if you ever find yourself thinking, "there's got to be a more efficient way to do this," then that's probably a good problem for a CT-based solution, IMO.

**April 26, 2018 Enoch > Peter**

Wow! Peter, this is fantastic. This will make this badge much richer. Is it ok if I use/adapt your examples in the badge and/or tutorial?

**April 26, 2018 Peter > Enoch**

Absolutely