Type 1: Using Instructional Design Techniques to Create Distance CS Education to Support In-Service Teachers

I. Project Goals and Outcomes

A. Introduction to the Problem and our Solution

In computer science, we do not currently have pedagogical practices that work well at a distance and for self-study. Our best practices in computing education are those that develop student's design ability (Felleisen, Findler et al. 2002; Findler, Clements et al. 2002; Caspersen and Bennedsen 2007; Caspersen and Kolling 2009) and that improve retention and broaden participation (Guzdial 2001; Guzdial 2002; Guzdial 2003; Sloan and Troy 2008; Simon, Kinnunen et al. 2010). All of these assume cycles of classroom lecture (often with learner engagement and live coding activities (Hake 1998; Caspersen and Bennedsen 2007; Simon, Kinnunen et al. 2010)) and student programming on homework assignments. Even materials designed for out-of-class study are just guides for student work in lab settings (Radenski 2006).

Student programming sessions can be inefficient with respect to learning goals (Long, Weide et al. 1998; Gray, Edwards et al. 2005; Edwards and Perez-Quinones 2008; Thornton, Edwards et al. 2008). Several studies have instrumented student programming environments to track their activity (Spacco, Strecker et al. 2005; Jadud 2006). Much of the students' time is spent on activities extraneous to their learning, such as struggling with programming language syntax (Jadud 2006) or making program permutations without insight into the meaning of the program permutations (Spohrer 1992).

Our inefficient practice for computer science teaching and learning is a direct impediment to our goals of creating more high school computer science teachers.

We are going to need to grow computer science teaching through in-service rather than pre-service teacher professional development (Guzdial 2010). The National Science Foundation has set a *"CS10K"* goal, of having 10,000 high school computer science teachers ready to teach the Advanced Placement Exam in Computer Science (AP CS) course in 10,000 high schools by the year 2015. In 2009, we had 2,019 high schools offering AP CS, which is a reasonably proxy for the number of AP CS teachers in the country (Ni and Guzdial 2011) since few schools have more than one AP CS teacher. We are unlikely to fill the gap from 2K teachers to 10K teachers using pre-service teacher education simply because of the need for teacher practicum and student teaching classrooms (Gal-Ezer and Stephenson 2009; Gal-Ezer and Stephenson 2010).

We have recently completed a study of students in an online computer science class (Benda, Bruckman et al. 2011). Students in on-line computer science courses tend to be "non-traditional," like in-service teachers. The adults in our study, as in others of online computer science courses (Kleinman & Entin, 2002), were working professionals with fragmented time available for their course. They often needed remedial help since their formal schooling had been unused for years. They had difficulty communicating with their teacher and others in their courses. We found that online students were unable to find the time to put in hours in front of a programming development environment.

In order to meet the goal of CS10K, we need a new, more efficient kind of computer science pedagogical practice.

Our current best practice is incompatible with the needs of working professionals such as in-service teachers. Fortunately, there is low-hanging fruit. Many of the research findings on

instructional design and in educational psychology on creating more efficient educational practices are rarely used in computing education. By creating new materials using these education research findings, we propose to create a new computer science distance learning approach that *is* compatible with developing in-service computer science high school teachers. We call our approach the ***examples+practice*** model, because we draw most significantly on the literature on creating effective **examples** (Sweller and Cooper 1985; Trafton and Reiser 1993; Atkinson, Derry et al. 2000) and on **practice** (exercises used to encourage reflection on examples and develop skills) based on cognitive tutors research (Anderson, Corbett et al. 1995; Cen, Koedinger et al. 2006; Aleven, Mclaren et al. 2009). We plan to structure examples in terms of the redundancy and modality principles of multimedia learning (Mayer 2009), and to choose both examples and practice to highlight the structure-behavior-function (SBF) aspects of programs (Goel, Rugaber et al. 2009; Vattam, Goel et al. 2009).

We propose to create these materials and evaluate them with in-service high school teachers learning computer science. We will evaluate them against three measures, which will form our ***three research questions***:

1. Do teachers learn computer science using the **examples+practice** model? We will evaluate the learning using a variation of an existing validated test of introductory CS knowledge which is programming language independent (Tew and Guzdial 2010; Tew 2010; Tew and Guzdial 2011).
2. Do teachers become more efficient at programming tasks after using **examples+practice**? We plan to take baseline measures during Year One of teachers in our existing summer and in-service professional development workshops, on the amount of productive time spent on programming assignments. We will compare to the same programming tasks in Years Two and Three for our distance students.
3. How do we apply and adapt instructional design practices for computer science? For this question, we propose a design-based research approach (Cobb, Confrey et al. 2003; Lamberg and Middleton 2003; Clements 2007). The instructional design and educational psychology literature on which we are basing **examples+practice** has rarely been used in computer science instruction. Our plan is to develop our materials begin careful observations of their use, including detailed interviews, in order to understand how the materials are being used, if they are being used the way we predicted, and if our learners are finding them useful and satisfying. We are relying on the fact that our learners, being post-graduate adults who specialize in learning, will able to reflect on and report to us the strengths and weaknesses of our materials. *An outcome of this project will be published design guidelines on creating effective distance learning materials for computer science based on instructional design and educational psychology literature.*

B. Problem: Needs of In-Service Computer Science Teachers

One of the greatest needs for our existing computer science teachers in the United States is certification and adequate professional development towards that certification (Gal-Ezer and Stephenson 2009; Gal-Ezer and Stephenson 2010). Few states offer computer science certification (Ericson, Armoni et al. 2008), and the result is teachers have their certification in something else (most likely Business in the US (Wilson, Sudol et al. 2010)) and are unlikely to have much (if any) computer science preparation pre-service (Ni and Guzdial 2011).

The list of what high school computer science teachers need to know is long (Gal-Ezer, Hazzan et al. 2009). Certainly, they need to know computer science, including some programming, which is part of Level 2 and Level 3 of the *ACM K-12 Model Curriculum* (Tucker, Deek et al. 2003). Our CS10K goals are only achieved by teachers knowing the *Computer Science: Principles* content (Astrachan, Barnes et al. 2011; Astrachan, Cuny et al. 2011), which includes a significant programming

component. In addition, teachers need to know *how to teach* the content, which is called *pedagogical content knowledge* (Shulman 1986; Shulman 1986; Saeli 2009). Teachers also need to know how to diagnose student errors (both in code and in understanding) and identify appropriate interventions to support learning.

Results of Prior Work: We have three on-going grants in which we have offered professional development and studied the factors influencing teacher success in in-service professional development: *Georgia Computes!; Disciplinary Commons for Computing Educators (DCCE);* and *Operation: Reboot*. Through these projects, we have learned how to create professional development that has impact, about the challenges for in-service teachers in taking professional development, and what the current state-of-the-art is in distance CS education.

Georgia Computes! is a National Science Foundation *Broadening Participation in Computing (BPC)* Alliance (Grant #0634629, 0940394) whose goal is to broaden participation by improving computing education at all levels of the pipeline across the state of Georgia. GaComputes works from elementary and middle school (e.g., YWCA after school workshops and Girl Scout activities and summer camps) to motivate interest in computer science, through high school (e.g., teacher professional development and lending libraries), and undergraduate (e.g., workshops on curriculum that engages and motivates a more diverse student body) (Ericson, Guzdial et al. 2005; Ericson, Guzdial et al. 2007; Bruckman, Biggers et al. 2009). The GaComputes Alliance includes the Georgia Department of Education, University System of Georgia, and youth-serving organizations such as YWCA and Girl Scouts of Greater Atlanta. In GaComputes, we have tried to measure impact across the pipeline. For example, teachers at 38% of Georgia high schools have participated in our professional development. Those schools produce 58% of students from Georgia in introductory computing courses (CS1 and CS2) in a survey of Georgia public universities, and those schools produce significantly more women and under-represented minority CS students than the other 62% of Georgia schools.

As part of GaComputes, we began a study of students taking a set of CS courses on-line at one of our partner institutions, Columbus State University (Benda, Bruckman et al. 2011). These courses are now part of a computer science teaching endorsement program, the first such program in Georgia (Whitehead, Ray et al. 2011). Our participants, like most students taking CS courses on-line at a distance (Kleinman and Entin 2002; Mashiko, Morita et al. 2002; Ragonis and Haberman 2003), are stressed for time, which influences their studies. As one participant described:

> *"I had my few afternoon hours that I could work on the stuff, but it all just boiled down to me not having time for my family when I was taking the courses. I think the bottom line was with my family structure, I shouldn't have taken more than one course at once." […] "sometimes I felt like I wasn't putting enough into one class because I was putting so much into the other class." […] "Then I had to put more time into the family, because I didn't put in as much as I should have, but I still had to put time in for them."*

In particular, our participants told us about how they were stumped by syntactic errors, how to get help, and challenges with remembering long-ago mathematics courses:

> *Andrew - "I said one time that I couldn't get this mathematical problem to work. His response was, "I'm not going to teach you algebra." So if you get one little piece or spacing wrong, it doesn't work."*
>
> *John – "There were times that it would take me hours to find one comma out of place, or find that one something that was wrong, so I didn't mind sticking with it but it just got to the point where I just didn't get it."*

DCCE is an NSF CPATH (CISE Pathways to reinvigorate undergraduate education) grant (#0829601) that brings together AP CS high school teachers and CS1/CS2-teaching undergraduate faculty for one Saturday a month for an academic year. The goal of the project is to improve teacher quality through reflection and creating a community of teachers (Ni and Guzdial 2011; Ni, Guzdial et al. 2011; Ni, Tew et al. 2011). Through DCCE, we have come to see how in-service high school

teachers are isolated, lacking CS background, and desperately in need of community—even more than other STEM teachers. Because so few states offer certification for CS teachers (Ericson, Armoni et al. 2008; Gal-Ezer and Stephenson 2009; Gal-Ezer and Stephenson 2010), a community of peers and role model teachers creates a sense of belonging, increases the odds of being retained as a teacher, and generates interest in improving as a teacher (Ni and Guzdial 2011; Ni, Guzdial et al. 2011; Ni, Tew et al. 2011). The need for in-service professional development is particularly explicit in this comment from a DCCE participant:

> *[Becky]: "I struggle with giving everyone the material and being able to explain it to everyone…I struggle with how to be creative with the programming. I have a problem with trying to make the programs have meaning to them...It is hard to teach. It's hard knowing how to teach it, how to give it to them…It's hard to explain…I would have to definitely update my skills. I would have to do something because I don't know if it's old age or what. When I look at kids' codes, they think I should know it as soon as I look at it. For the longest time I thought I should, but I don't have to. I have to study it just like they do."*

Operation: Reboot (BPC #0940932) retrains unemployed IT workers as high school teachers (Betts 2009). Each IT worker is paired with an in-service high school teacher who wants to learn more computer science. By pairing, each learns from the other, and each pair is mentored by an expert high school CS teacher. Through Operation:Reboot and GaComputes, we have been developing expertise in using the current state-of-the-art in in-service professional development tools, including Elluminate, video examples, digital repositories, and shared content management systems.

C. Proposed Solution: Research-Driven Examples+Practice CS Education

An enormous body of literature exists on how people learn and how best to design instruction to support learning (Donovan, Bransford et al. 1999). We are proposing to draw on a selection of this literature that is most promising in terms of addressing in-service teacher needs in learning computer science.

i. Worked Examples

One of the most promising findings in educational psychology in terms of improving efficiency of instruction (i.e., achieving the same learning goals in less time) is the use of *worked examples* (Sweller and Cooper 1985). The hypothesis behind worked examples is that much of the learning in STEM leads to cognitive overload because we ask students to solve problems at the same time as they are learning the basic principles (Sweller 1988). Worked examples have students learn by studying completely worked-out problems, *and then* tackling problems to solve. At least one review of the literature suggests that studying worked examples is far more successful than more problem-solving oriented approaches (Kirschner, Sweller et al. 2006).

The critical aspect for us is that worked examples approaches have led to more efficient learning (Sweller and Cooper 1985). Sweller & Cooper showed that they could teach just as much algebra in half as much time with worked examples as a traditional, problem-solving approach. Similar results have been found in other domains (Atkinson, Derry et al. 2000).

Instructional designers and educational psychologists know a lot about how to construct examples that ease learning and later problem-solving. Atkinson et al (2000) have published a thorough review of the existing literature on designing examples. For example, Catrambone has shown that structuring the examples in terms subgoals and labeling those subgoals has a dramatic impact on students' ability to re-use (transfer) the knowledge in other problem-solving settings (Catrambone 1994; Catrambone 1994; Catrambone 1995; Catrambone 1996).

Worked examples have rarely been used in computer science education. Pirolli used worked examples with great results for teaching recursion (Pirolli 1991; Pirolli and Recker 1994; Bielaczyc, Pirolli et al. 1995). Clancy & Linn taught Pascal with text case studies around the same time (Clancy

and Linn 1992). Guzdial & Kehoe developed a case library of design examples (Guzdial and Kehoe 1998). Students who used the library saw many more designs and did significantly better on later design problems. Dorn found that informal learners learned normative computer science from studying examples with embedded CS definitions and explanations (Dorn 2010).

A major challenge with a worked examples approach is *covering the space*. What examples and how many do we need? We propose to use an approach from cognitive tutor development (Anderson, Corbett et al. 1995; Aleven, Mclaren et al. 2009). We will define our learning outcomes and a cognitive model for the skills needed to achieve those outcomes, then define the practice tasks, through to subgoals and subfactors, and provide examples for each subgoal. By tracking our examples down to the subfactor level, we can then use techniques inspired by cognitive tutors research to track learning at the subfactor level and improve instruction based on that performance (Cen, Koedinger et al. 2006; Koedinger and Stamper 2010). By using Catrambone's subgoal approach to modeling, we believe that we can avoid the expense and complication of developing the cognitive model down to a production rule level (Catrambone 1998).

An important empirical question is *how many examples to provide for each goal and subgoal*. By tracking student error and time to completion, we can estimate student understanding, as in cognitive tutors(Cen, Koedinger et al. 2006; Koedinger and Stamper 2010). Students who already know the content can push ahead further, and others can be given more examples. Pirolli (1991) used eight examples for each of the primitives of his LISP-like language as part of his training materials, which provides us a starting point for number of example-practice pairs needed for each subfactor.

ii. Practice on Structure, Behavior, and Functions

Examples are most effective when they are interleaved with ***practice***, the opportunity to demonstrate the knowledge in the example. These practice sessions are focused on recall and the simplest problem-solving, e.g., "Write a LISP function that adds the numbers 3 and 2. You should call a particular LISP function with the arguments 3 and 2" (Anderson, Corbett et al. 1987). Pirolli's recursion lessons interleaved examples and practice, where students would write small recursive functions (Pirolli 1991). Trafton and Reiser (1993) found that the most effective learning comes from interleaving one example and one practice exercise on that example content.

We propose to provide practice opportunities for students that will support them in learning from the examples, drawing on best practices for student assessment in computer science. The goal is to encourage students to think about the examples. Some of these may be multiple choice questions, (Lister, Adams et al. 2004; Tew and Guzdial 2011), but more will be fill-in-the-blank code problems, like Problets (Krishna and Kumar 2001; Naps, Roessling et al. 2005), JavaBat (Roessling, Joy et al. 2008), or PracticeIt! (Reges and Stepp 2010). Practice exercises will also involve *self-explanation* activities (e.g., explaining to oneself what an example means) which have been shown to improve student learning from examples (Bielaczyc, Pirolli et al. 1995), even if that self-explanation merely involves a prompt to take notes (Sandoval, Trafton et al. 1995; Trafton and Trickett 2001).

Our challenge for our student audience (in-service teachers) is to come up with effective exercises and practice problems that will help our teachers learn the kinds of knowledge that they might learn from writing programs in a development environment. What do students learn when they are programming? What kinds of knowledge are required and used when programming? Ashok Goel and his colleagues have been studying designers in a variety of domains (including mechanical engineering and computer science) (Goel, Rugaber et al. 2009) and describe designers' knowledge in terms of:

- *Structure*: The obvious static components of the design. In software design, these are the statements of the program. Structure knowledge is used in writing code (Lister, Fidge et al. 2009).

- *Behavior*: How the components in the structure interact in the design. In an algorithm to sort an array, a loop, some assignment statements, and a conditional test are the components that result in a behavior such as a bubble sort or a heap sort. Behavior knowledge is used in debugging and tracing code (Lister, Adams et al. 2004; Whalley, Lister et al. 2006; Venables, Tan et al. 2009).
- *Function*: What the design does. In software, this is the purpose or meaning for a particular program or software component.

Programming requires specification of structure, based on understanding the behavior of individual statements, in pursuit of some function. Research on SBF models find that novices understand structure well, function less well, and behavior weakest of all (Liu, Hmelo-Silver et al. 2005; Hmelo-Silver, Jordan et al. 2008). We see parallels in the computer science literature. Pirolli (1991) found that students found it easier to learn from structure-focused examples than behavior-focused examples. A series of studies in Australasia have found that students cannot easily discern function from a given piece of code and find tracing very difficult (Lister, Adams et al. 2004; Whalley, Lister et al. 2006; Lister, Fidge et al. 2009; Venables, Tan et al. 2009).

Recent research in science education has found that students learn systems especially well through SBF-informed hypermedia examples (Liu, Hmelo-Silver et al. 2005; Liu, Marathe et al. 2006; Hmelo-Silver, Jordan et al. 2008; Sinha, Gray et al. 2010; Vattam, Goel et al. 2010). In comparison to traditional biology examples which are based on structure (e.g., "Here is the cardiovascular system"), SBF-informed hypermedia starts from function (e.g., "How does oxygen enter the animal?") and leads through behavior (e.g., "The blood carries the oxygen from the lungs throughout the body.") Students using SBF-informed hypermedia examples actually develop more elaborated understanding than students studying with more traditional examples.

Computing educators use visualizations to describe behavior in student programs (Hundhausen, Douglas et al. 2002). While these visualizations describe behavior, a weak spot in student learning, algorithm visualizations don't consistently lead to learning. What makes them most effective is including *practice* with the visualizations, e.g., making students create their own visualizations, tracing visualizations, and including visualizations as part of code writing (Hundhausen and Brown 2007). A new class of visualization tools include the ability to ask students to do tracing of the visualization themselves, with feedback on errors (Sorva and Sirki 2010).

An obvious question is: Why develop a new **examples+practice** approach when we already have cognitive tutors that work so well for teaching programming (Sleeman 1986; Anderson, Corbett et al. 1995)? There are two reasons.

- A cognitive tutor uses a cognitive model, that structures a set of exercises with a student model (e.g., a behavioral trace) that directs feedback when students work on the exercise. Tracing student activity is expensive to implement (Anderson and Corbett 1993) even with new tools supporting example tracing (Aleven, Mclaren et al. 2009). We are proposing to develop a cognitive model, but to use many examples and well-designed practice *without* a behavioral trace. *We are proposing that multiple, targeted examples with practice feedback can guide learning as well as individualized tracing*. We hope to develop the instruction more easily (e.g., without the technical expertise required for tutoring authoring), in less time, and with lower cost than a complete tutor. That's an empirical summative question. In the proposed work, we are doing the formative work to create the approach.
- Second, it is not clear that we know how to develop a cognitive tutor to teach teachers. Teachers need to develop knowledge about being a teacher (e.g., pedagogical content knowledge, diagnosing and addressing student problems) that we do not yet know how to teach via a tutor (Koedinger 2011). An example-based approach has worked for teaching pedagogical content knowledge (Mouza 2010). Worked examples plus practice has the greatest potential for designing instruction for computer science teachers.

*In summary, the **examples+practice** approach will provide examples that identify structure, behavior, and function components, and practice will exercise student understanding of all three components.* Research in instructional design suggests that this should be successful, but has not yet been tried in computer science.

iii. Using Multiple Modalities

Computer science is a text-heavy discipline. Visualization research attempts to leverage students' ability to process images for learning computing (Naps, Cooper et al. 2003), but it has been surprisingly unsuccessful (Hundhausen, Douglas et al. 2002). Educational psychology literature suggests that the intuition to mix modalities is good, but visualization plus written text isn't necessarily going to help. Rather, mixing *audio* with visual information is more likely to provide more information to students without cognitive overload (Mayer 2009). There is a significant body of literature which suggests that people learn more deeply from visual information and narration than visual information with text. Thus, written text explaining textual programs may be leading to cognitive overload, making it harder for students to understand the content. Educational psychologists believe that humans have both an auditory channel and a visual channel for receiving information, and by providing narration over text we optimize use of those channels (Mayer and Moreno 1998). Of particular interest for our in-service teachers, audio can lead to greater efficiency in instructional time (Solomon 2005).

In our examples, we propose to make heavy use of audio explanation for program text (e.g., perhaps in the form of video), rather than more textual explanations. The approach of replacing audio narration for explaining textual information has been used to create better explanation of geometry proofs (Mousavi, Low et al. 1995). Using audio narration for improving computer science education is novel.

iv. Supporting Communications

A common complaint of participants in our study of distance CS education classes is the lack of communication between the student and peer students, and between the students and the teacher (Benda, Bruckman et al. 2011). Our DCCE participant teachers frequently talk about the isolation that they feel (Ni and Guzdial 2011; Ni, Guzdial et al. 2011; Ni, Tew et al. 2011). Communications needs to be part of any professional development effort for computer science teachers.

We plan to provide two kinds of supports in our instructional materials. We will explore mechanisms for encouraging discussion such as our *anchored discussions* that we have built and used successfully in the past (Guzdial 1997; Guzdial, Hmelo et al. 1997; Guzdial 2001), with a particular focus on low-cost and time efficiency for in-service teachers-as-students (Guzdial, Rick et al. 2000; Rick, Guzdial et al. 2002). Anchored discussions link discussion formats to examples or practice assignments, to create a focus for discussions, which has been shown to create more learning-effective, on-topic discussions than unanchored discussions (Guzdial and Turns 2000).

To solve the problem of student isolation and student-teacher disconnect, we are particularly concerned with a sense of *awareness* that would drive use of the communication.

- First, the teacher needs to know where students are. We propose the creation of a *teacher dashboard*, driven by performance on the practice and indexed by learning goals and subgoals. We are interested in providing similar kinds of insight as the cognitive tutor "skillmeters" (Anderson, Conrad et al. 1993; Anderson, Corbett et al. 1995). The teacher will be able to see at a glance how the class is doing, and how individuals are doing, which encourages a sense of activity and where the class may be challenged.
- Second, the students need to have a sense of where they are with respect to the others. We propose the creation of individual and group "skillmeters," so that students can have a sense that they are making progress, and so is their whole cohort. We believe that real benchmarks like these will provide students, especially teachers who may worry that

they "can't" learn computer science (Fisher and Margolis 2003), with a sense of how well they are really doing.

v. Scenario of Examples+Practice CS Education

To make more concrete our plans, consider this scenario. Angela is a teacher learning *Computer Science: Principles*, and is progressing in her studies. She's working on simple functions: Calling functions, manipulating the results, and using those results in a call to another function.

- **Structure example+practice:** She watches a video of someone explaining and using a function that retrieves the red channel from pixels of a picture, reduces the red by 50%, then saves the new red value back to the pixel. After watching this video, she is asked to complete a single line of code in a function that retrieves the blue channel and then sets a new blue channel. She is supposed to increase the blue by 20%. At first she misses the syntax, but a problem-specific error message corrects her. The second time, she uses the constant 0.20, decreasing blue by 80%, and a problem-specific mathematics explanation corrects her. She gets it right the third time. That took her about 20 minutes, which she had time to do in the morning. She sets the materials aside, and returns to it that afternoon at lunch.
- **Behavior example+practice**: She then sees an animation of the execution of the function reducing blue by 20%. Midway through the animation, the execution stops, and she's prompted to fill in the values of the variables after the next three statements execute.
- **Function example+practice**: She's now shown a new function (which increases green by 20%), and she's shown three outputs of the program, and she's asked to identify the right output and to self-explain what the function does.
- **Subgoal PCK**: Later that evening, Angela sees a video describing the misconceptions that students often have in dealing with calling functions. She is prompted to update her on-line teacher notebook with her self-explanations on how to recognize those misconceptions and how she might help students to learn a better conceptualization.

Because she has trouble with the first structure example+practice, she's given some additional examples on that same learning outcome. She notices that her skillmeter shows that she's doing really well on the subgoal for calling functions, and she sees that only one other member of her cohort is further than her on this set of examples+practice. Her professional development workshop leader sees an update on the class dashboard on how everyone is doing.

vi. Future Development of the Proposed Work

Our overall goal with this research direction is *to create a new medium for computing education*. Static textbooks, Powerpoint, or even PDFs and other e-book formats cannot adequately convey program behavior. Algorithm visualizations are insufficient without student activity. We aim in this effort to create a new medium, built on the best of educational psychology and instructional design, to better support learning about computer science. The closest tool to what we are proposing is the Open Learning Initiative (OLI), whose tools were used to improve the efficiency of a statistics course by 50% (Lovett, Meyer et al. 2008), but has rarely been used for CS, lacks the CS-specific features we describe, and has mostly been used for hybrid classes, not distance-only. By starting with in-service teachers, we have an audience that is particularly time-constrained and yet is more metacognitively aware than other students (e.g., K-12 students) and thus can serve as design informants.

Once we have this medium, we plan to develop it for other purposes. Certainly, we want more teachers to use it, even informally. We can also use this medium for younger students. For example, the *AP Computer Science: Principles* will need a textbook, since a textbook is required for Advanced Placement course audits across all domains (Board 2011). We will have created *Computer Science:*

Principles content in our new medium for teachers, by the end of this project. A natural next step is to explore developing these materials into a free, on-line textbook for high school students studying AP *Computer Science: Principles*.

If the CS10K goal is successful, we can anticipate that around 8,000 of the 10,000 high school teachers teaching AP CS:P will be relatively new computer science teachers. They will likely never to have taught AP CS previously. The communications and "dashboard" mechanisms that we propose will be useful scaffolding for new teachers. We are particularly envisioning a mechanism that might use the teacher dashboard as an index into a digital library of classroom practices, such that the right in-class activity might be suggested to the new teacher that would best serve the learning needs of the students. We can imagine the system suggesting, "65% of your students are having difficulty understanding how processes can be distributed across a network. Here is a great in-class activity to improve learning on that outcome..." We could enhance the proposed project by linking learning outcomes from the cognitive model to recommended in-class activities, so that problems with the outcomes could lead to suggestions for the new teacher.

II. Evaluation Plan

Our evaluation is organized around our three research questions and the issue of underserved minorities (which is treated separately in a later section of the proposal). "Students" here refer to our student population, in-service high school teachers. We will seek our participants through our existing ICE@GT channels (e.g., through mailing lists and Georgia Department of Education), and also through the Georgia CSTA Chapter and the Federation of Disciplinary Commons (see *Implementation Plan*). We will offer incentives to be part of our data collection process.

A. Do students learn computer science using the examples+practice model?

We plan to develop instruments to measure teacher learning *Computing in the Modern World* (*CiMW*, Level 2 on the ACM Model Curriculum, and in the Georgia state high school curriculum) and in the AP CS:P pilots. An instrument to measure understanding of conceptual knowledge isn't as challenging as measuring understanding of the programming knowledge. We plan to develop a new test based on an existing test for that goal.

Allison Elliot Tew just completed her dissertation this last year working with our group, developing the first validated language-independent test of introductory computer science knowledge, the FCS1 (Tew 2010; Tew and Guzdial 2010; Tew and Guzdial 2011). The FCS1 is available for us to use as a benchmark for the teachers using our materials.

Elliott Tew found that pseudocode works as a reasonable notation for students to use in her instrument. Performance on her pseudocode test was highly correlated with student performance on an isomorphic test in the student's CS1 language (Java, MATLAB, or Python), and with performance on the student's CS1 final exam (Tew 2010; Tew and Guzdial 2011).

The FCS1 may not be sufficient to cover the computer science content that we want our in-service teachers to learn. While FCS1's language independence is desirable, it has only been tested on Java, MATLAB, and Python, not the languages commonly used in *Computing in the Modern World* and in the AP CS:P pilots, such as Scratch and Alice. Our plan is to develop assessments using Elliott Tew's pseudocode (as an understandable notation), but focus on CiMW and CS:P content. ***A contribution of the proposed project will be to develop language independent assessments for CS concepts in Computing in the Modern World and Computer Science: Principles.***

B. Do students become more efficient at programming tasks after using examples+practice?

We do not envision any medium for computing education to completely replace programming a computer in a development environment. We wouldn't want to. Programming is a central activity in the computer science *community of practice* (Lave and Wenger 1991). To teach computer science without having students program is *inauthentic* which can reduce engagement (Shaffer and Resnick 1999).

We hope to make the time that the in-service teachers spend learning through programming *efficient*. We want the programming activity to be productive (in terms of learning what is best learned by while programming) without a significant loss of time to extraneous detail. *We hypothesize that the **examples+practice** approach will lead students to learn structure, behavior, and function such that time spent programming in the development environment is productive.*

Our measure of productivity will be time spent and student self-report of quality of time spent. A student could spend two hours programming, and say that it was very productive because the time was spent exploring a problem and developing a solution. A different student may spend the same two hours but complain that only half of it was productive because he had difficulties (for example) getting the spelling right on library calls.

We plan to take baseline data during the first year of our project, in face-to-face teacher professional development workshops. We will ask teachers in these workshops to keep on-line diaries of time spent, what they spent the time doing (e.g., getting the program to compile without errors, getting the program to run without errors, debugging the running program), and the quality of that time.

As we develop instructional materials to offer the same professional development workshops on-line, we will ask participants to keep the same diaries. While the data is self-reported and between different cohorts, the data gives us a point of comparison to feed into the design-based research iterative. Our design goal is for our participants to find the programming time more productive than those in our distance CS education study (Benda, Bruckman et al. 2011).

C. How do we apply and adapt instructional design practices for computer science?

There will be dozens of design decisions that we will make while developing our instructional materials. The mapping from the educational psychology findings and instructional design practices to applications in designing computer science education is not always obvious.

- The multimedia principle (Mayer 2009) says that narration is better than static text for explaining complex material. Is that narration best offered as a video of static or slowly scrolling program code, or as audio narration connected to a user-controlled scrolling text frame?
- The SBF model of design knowledge suggests that we need students to practice their understanding of behavior. Would it be better to do that by guiding them through step-by-step tracing as in UUhistle (Sorva and Sirki 2010), or would it be better to show students a partial execution as an animation and then have them predict the next step in the execution (Byrne, Catrambone et al. 1999)? Both have been used successfully, and it may be that each is useful for different kinds of learning (e.g., step-by-step when learning data manipulations, prediction when dealing with learning control flow).

Controlled lab experiments could be conducted to evaluate each of these design decisions, but there will likely be interactions between them. Design experiments is a methodology originally proposed by Ann Brown (Brown 1992) which suggests a different approach. Since it is so hard to be successful at achieving desired learning outcomes, we first aim to get it working right (by some measure), then develop hypotheses of why it worked and conduct more careful experiments to test

those hypotheses. Today's design-based research methods focus on iterative development of the learning setting, keeping careful attention to what got changed and how learning was impacted (Cobb, Confrey et al. 2003; Lamberg and Middleton 2003).

In our design-based research model, we will focus on the five key design areas of our approach: worked examples, practice, SBF, use of mixed modality, and awareness via skillmeters. We plan to use CS content and productive programming time measures (from research questions one and two) as two of our outcomes. During each design iteration, we will conduct at least two think-aloud protocols with two teachers using our materials to get detailed insight into how our materials are being used. We will conduct a sample of five phone interviews to get self-reports about satisfaction, quality of materials, and learning effectiveness. Finally, we will conduct surveys with *all* participants in each design iteration focusing on the five areas and the three subjective measures of satisfaction, perceptions of quality, and perceptions of learning effectiveness.

D. Logic Model

Research Question	Instruments	Sampling	Analysis Questions
RQ1. Do students learn computer science using the examples+practice model?	i. FCS1 ii. New measures focused on *Computing in the Modern World* and *CS:Principles* outcomes, using Elliot Tew's pseudocode as language.	i. Baseline measures at face-to-face workshops in Year 1. ii. Measures of learning for on-line workshop participants in every-other design iteration	Comparison of current iteration CS learning performance against baseline measures and earlier design iterations. Is performance increasing, and in what areas?
RQ1. Do students become more efficient at programming tasks after using examples+practice?	Self-report via diaries of time spent on programming assignments, activities during that time (compile-time focus, run-time focus, debugging), and satisfaction with learning.	i. Baseline measures at face-to-face workshops in Year 1. ii. A sample of five participants (selected randomly) during each design iteration to keep diaries.	Does time spent on programming task decrease? Does focus shift to higher-level cognitive activities? Do subjects perceive learning value?
RQ3. How do we apply and adapt instructional design practices for computer science?	i. Learning and programming task diaries from RQ1+2. ii. Think-aloud protocols. iii. Phone interviews focusing on five key design areas and perceptions of satisfaction, quality, and learning effectiveness. iv. Surveys on same areas and perception issues.	ii. Think-alouds from two participants in each design iteration iii. Phone interviews with five participants each design iteration. iv. Surveys for all participants	Is each design iteration resulting in more learning, greater sense of time effectiveness, and greater sense of satisfaction and quality? What key design areas are not improving? What design alternatives should be explored?
Issues of Underserved Populations	Counts of number of students in CS classes. Performance on AP CS exam.	All teachers who take our workshops will be polled annually for counts in their classes and for performance of their students.	Are teachers drawing more students into their classes? Are their students doing better on measures of CS learning?

III. Implementation Plan

A. Timeline for Implementation and Evaluation

Our timeline calls for two sequences of a six month development effort, followed by three three-month design iterations, with learning measures in the middle iteration, letting us develop infrastructure and learning measures, and then gives us three design-based research iterative cycles. Our professional development workshops usually have about 20 teachers per session. We predict higher numbers of teachers when the workshop is on-line at a distance. *We conservatively predict n=25 for each iteration, so impact of at least n=150 teachers, and their students indirectly. Our focus is on building the instructional materials, which can then be used with many more teachers later*.

<table>
<tr><th></th><th>1st Quarter</th><th>2nd Quarter</th><th>3rd Quarter</th><th>4th Quarter</th></tr>
<tr><td>Year One</td><td colspan="2">Take baseline measures on RQ2 (programming time and efficiency)
Development of cognitive model for Computing in the Modern World (CiMW) goals.
Development of web infrastructure for authoring and delivery of first iteration examples+practice instruction.
Creation of first professional development workshop materials
Development of content evaluation for CiMW cognitive model.</td><td>Development of lessons
Deployment with participants
Gather RQ3 design-based research (DBR) measures
Gather RQ2 (programming activity self-report diaries) measures</td><td>Development iteration.
Deployment with participants
Gather RQ3 design-based research measures
Gather RQ1 (learning) and RQ2 (programming activity self-report diaries) measures</td></tr>
<tr><td>Year Two</td><td>Development of lessons
Deployment with participants
Gather RQ3 DBR measures
Gather RQ2 measures</td><td colspan="2">Develop cognitive model for Computer Science: Principles content. Both Tiffany Barnes (U. North Carolina, Charlotte) and Beth Simon (UCSD) will work with us to develop content for their courses.
Creation of CS:P lessons
Extend infrastructure for CS:P lessons. Since our goal is to meet in-service teachers' time-constrained schedules, we will also explore other delivery mechanisms (such as mobile devices) during this development cycle.
Development of content evaluation for CS:P cognitive model.</td><td>Development of lessons
Deployment with participants
Gather RQ3 DBR measures
Gather RQ2 measures</td></tr>
<tr><td>Year Three</td><td>Development and Deployment.
Gather RQ3 DBR measures
Gather RQ1 and RQ2 measures</td><td>Development and Deployment
Gather RQ3 DBR measures
Gather RQ2 measures</td><td colspan="2">During this last cycle, we will aim to develop materials and deploy them for one of our partner groups, such as the Columbus State University endorsement program or a CS:P pilot site.</td></tr>
</table>

B. Reaching Underserved Populations

We plan to target the in-service teachers currently taking professional development workshops through "Georgia Computes!" offered by the Institute for Computing Education at Georgia Tech (ICE@GT). ICE@GT has been offering workshops even before "Georgia Computes!" was

established, and we expect to continue, with an increased use of distance education through the proposed work.

The teachers currently taking our workshops are 39% White, 45% Black, 4% Asian, 8% Hispanic, and 3% multiracial. 56% of those teachers teach in minority-majority schools. 44% of those teachers teach in economically disadvantaged schools. Thus, we are reaching an audience of mostly-minority teachers who teach in mostly-minority schools.

However, we are not having much success in reaching these minority students. In 2010, we had the largest group ever take the AP CS exam: 692. The increase in AP CS A takers is not uniform across all races or genders. The percentage of test takers in 2004 that were Black was 17% at 66 students. The percentage of test takers in 2010 that were Black was 9.8% at 68 students. The peak for Black students was in 1999 when 163 Black students took the exam. While the number of women taking the exam has increased from 70 in 2004 to 118 in 2010 the peak was also in 1999 with 155 women.

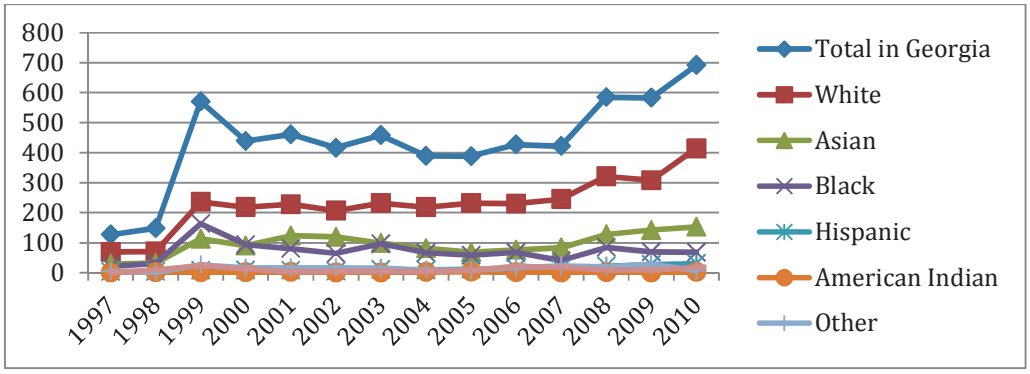

Figure 1: Number of AP CS test-takers in Georgia, by Race

Even more concerning is how few Black students pass the CS AP A exam. Students who earn a 3 or above pass the exam. The highest number ever passing in Georgia was 18 in 2003, but 96 Black students took the exam. The highest percentage of Black students passing was in 2010 with 16 passing of 68 students who took the exam. In 1999 when 163 Black students took the exam only 9 passed.

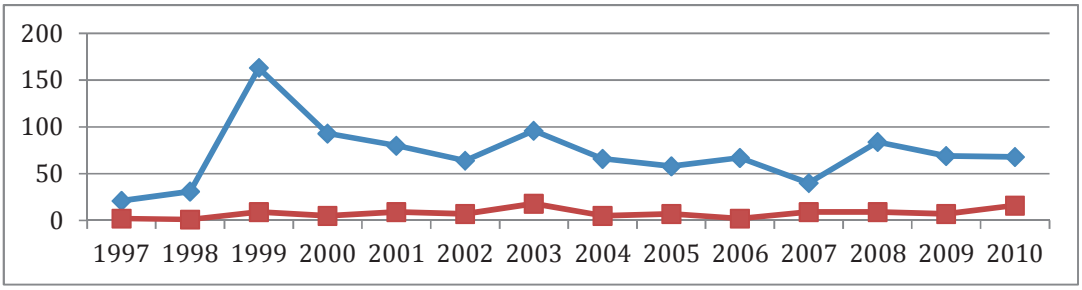

Figure 2: Number of Georgia Black AP CS test-takers (blue), and number passing (red)

We have a long way to go yet. We have touched 38% of the high schools in Georgia with some professional development, but not yet enough. Of the other 62% of high schools in Georgia, 53% are minority-majority schools. The CS teachers in the non-ICE high schools are 40% Black, 45% White, and 10% Hispanic.

We predict that a majority of our teacher participants will themselves be members of under-represented groups and will teach in minority-serving schools. Given our track record and our plan to develop in Georgia, we are confident to reach that goal. Will our approach have an influence on the *student* numbers? We don't *know* but we *suspect* that a significant problem in

these minority-majority high schools is a lack of content knowledge by the CS teachers. We have qualitative, sampled studies suggesting that there are teachers in these minority-majority high schools who lack CS content knowledge and who feel less confident in their teaching (Bruckman, Biggers et al. 2009; Ni, McKlin et al. 2010; Ni, Tew et al. 2011). We have some evidence that teachers who know the content better are stronger in their sense of identity as computer science teachers. Teachers with a strong sense of CS identity are better at recruiting students into CS, because they have greater personal value for the content (Ni and Guzdial 2011; Ni, Guzdial et al. 2011; Ni, Tew et al. 2011) – quite dramatically, with several instances in our studies of a doubling of CS class sizes in a single year. *Thus, a confident CS teacher who knows content is more likely to boost recruitment throughout the CS program in a school*.

We will measure impact of our materials on underrepresented groups in two ways. (1) We will have a direct measure of impact in terms of number of teachers participating and their ethnicity. (2) More significantly, we hope for an indirect impact. ***By improving content knowledge of in-service teachers at minority-majority high schools, we aim to influence performance of those teachers against two metrics: number of students taking CS and performance on the AP CS test.*** This measure becomes more direct when AP CS:P is created.

We plan to extend our reach across Georgia through our partnership with the *Disciplinary Commons in Computing Education* (DCCE) effort (now moving to Southern Polytechnic State University) and with the Georgia chapter of the Computer Science Teachers Association (CSTA). We have letters of support from Briana Morrison who heads the DCCE effort and Ria Galanos who is the Georgia CSTA Chapter president. They will team with us to help us find teachers who could use our distance education in-service materials, and thus help us reach teachers not yet involved with ICE@GT.

We don't see our project as being Georgia specific. However, we do see Georgia as a model. We have enormous challenges here in terms of reaching minority groups. We have access through ICE@GT and our partners to teachers. We hope to show that we can broaden participation through improving these teachers' content knowledge. Since our intervention is distance-based, it can easily be transported elsewhere.

C. Roles, Team, and Collaboration

Roles and Team: PI *Mark Guzdial* holds a joint Ph.D. in both Education and Computer Science. He is recognized for his expertise in education, e.g., he serves on the editorial board of the *Journal of the Learning Sciences* and was one of the first NSF Career awardees from the EHR Directorate. He is recognized for his expertise in computer science education, e.g., he is one of the founders of the ACM *International Computing Education Research* workshop series, and he serves on the editorial board of *ACM Transactions on Computing Education.* Guzdial serves on the AP CS:P Commission. Guzdial will be the overall project manager and will be particularly responsible for the evaluation plan, for the instructional design, and for the interaction with the advisory board.

Together, PI Guzdial and Co-PI Ericson have written several textbooks used in high school computer science classes (Guzdial and Ericson 2006; Dann, Cooper et al. 2009; Guzdial and Ericson 2009; Guzdial and Ericson 2010). They are the 2010 co-winners of the ACM Karl V. Karlstrom Outstanding Educators award. They direct the NSF BPC Alliance "Georgia Computes!"

Co-PI *Barbara Ericson* is the Director of CS Outreach for the College of Computing. She is an expert on teacher professional development, e.g., she directs ICE@GT, she served on the CSTA Board, and she has co-chaired the NCWIT K-12 Alliance. She serves on the current AP CS Development Team. Ericson will direct the effort to create and deploy the distance education instructional materials.

Guzdial and Ericson form a team with expertise in education research, computing education research, high school teacher professional development, and broadening participation in computing.

We are budgeting for three other members of the team:

- A 50% Research Scientist will be hired to implement the Web infrastructure for our materials.
- One Graduate Student Research Assistant (GSRA) will focus on design-based research assessment. This graduate student (most likely in our Human-Centered Computing PhD program, though possibly in Psychology) will answer Research Question #3, on the design guidelines for applying educational psychology and instructional design approaches to computing education.
- A second GSRA will focus on Research Questions #1 and #2, and the evaluation of reaching underserved populations. This graduate student (in CS or Psychology) will look for the linkages between the distance education and the three outcomes of (a) learning in CS, (b) productivity on programming tasks, and (c) impact on student recruitment and performance.

Collaborators: We have invited a three member Advisory Board:

- *Richard Catrambone* is Psychology Professor who is an expert in educational psychology, especially on the design of examples. His subgoal learning theory will be used to define the cognitive models used in our instructional materials. He will advise us on the design and evaluation of our materials.
- *Ashok Goel* is a Cognitive Science and Computing Professor who developed the Structure-Behavior-Function model of design knowledge. He worked with Cindy Hmelo-Silver to develop her SBF-informed hypermedia for teaching ecological systems. He will advise us on the design of our materials.
- *Wayne Summers* is a Professor and Chair of Computer Science at Columbus State University. He has been a co-PI on the effort to extend "Georgia Computes!" to other campuses. He directs the endorsement program. He will advise us on the applicability of our materials to his and similar formal distance education for teachers efforts.

We have budgeted to have quarterly meetings with each of our advisors, which will mesh with our quarterly-based design iteration model.

We will work with Dr. Tiffany Barnes of U. North Carolina at Charlotte and Dr. Beth Simon of the University of California at San Diego to collect their curriculum on teaching *Computer Science: Principles*. While Guzdial and Ericson are active curriculum developers, the focus here is not on creating more media computation materials. Rather, we aim to create a computing education medium for many approaches. Thus, we value highly the cooperation for Drs. Barnes and Simon, to access their curricular materials in creating the CS:P distance education materials.

As mentioned, we will also work with Briana Morrison of Southern Polytechnic State University and Ria Galanos of the Georgia CSTA chapter, to find teachers for our professional development studies.