Lightweight Methods for Developing Pedagogical Content Knowledge for HCI

Eliane S. Wiese
eliane.wiese@utah.edu
University of Utah
Salt Lake City, Utah, USA

Jason Wiese
wiese@cs.utah.edu
University of Utah
Salt Lake City, Utah, USA

Marina Kogan
kogan@cs.utah.edu
University of Utah
Salt Lake City, Utah, USA

Joshua Dawson
joshua.dawson@milsci.utah.edu
University of Utah
Salt Lake City, Utah, USA

ABSTRACT

An Unsolved Challenge in HCI education is developing pedagogical content knowledge (PCK) for teaching HCI and other courses on human-centered computing. As a complement to rigorous and intensive research methods for finding and validating PCK, we propose quick, lightweight methods focused on self-reflection and instructional design. We present our findings both as PCK that might benefit other instructors and as a demonstration of the usefulness of lightweight methods. We hope this work encourages others to reflect on and share PCK from their own teaching, which more rigorous methods can then validate.

ACM Reference Format:
Eliane S. Wiese, Jason Wiese, Marina Kogan, and Joshua Dawson. 2022. Lightweight Methods for Developing Pedagogical Content Knowledge for HCI. In *EduCHI'22: 4th Annual Symposium on HCI Education, April 30-May 1 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 7 pages.

1 INTRODUCTION: WHY PCK FOR HCI?

Pedagogical Content Knowledge (PCK) is understanding how to teach a specific subject to specific learners [11]. PCK is distinct from expertise in the content and from general pedagogical practices. PCK is specific to the discipline and includes anticipating student misconceptions and knowing what teaching strategies are likely to be effective (or not) [11]. All disciplines need PCK, and Oleson et al. called for developing more of it for HCI [9]. As people who are in HCI because we love it, it can be hard to get into the mindset of a student who doesn't. PCK helps us bridge that gap. As a community, we can benefit from lightweight methods that help us identify PCK and communication strategies that help us share our findings. Difficulty in communicating PCK is that students' struggles may not make sense to an instructor who hasn't encountered them first hand. For example, one instructor, such as J. Wiese (second author), might say, "students don't understand the difference between a problem and a solution." A new instructor, such as E. Wiese (first author), may have no conception of how students could confuse these two things. While E. Wiese was warned about it, she could not create effective instruction to counteract it because the misconception was so baffling. After a semester of teaching, E. Wiese agreed that this was an excellent characterization of students' difficulties. However, as a community, it would be useful to have ways to communicate these kinds of ideas in precise and actionable ways that new instructors can understand without having to live through them. This paper presents lightweight methods for HCI instructors to develop and share PCK, with examples of the PCK that we have generated through these methods.

EduCHI'22, April 30-May 1 2022, New Orleans, LA, USA

2 METHOD: YOU CAN TRY THIS AT HOME

As a community, we can and should study PCK systematically and rigorously by analyzing students' work, surveying and interviewing a broad range of instructors, and closing the loop by testing the effectiveness of instruction specifically designed around that PCK. And as a complement to those rigorous, in-depth, and time-intensive methods, we should also explore ways to develop PCK casually, self-reflectively, and with our colleagues.

E. Wiese, J. Wiese, and Kogan (the first three authors) have collectively taught 6 semesters of HCI at the same institution over 6 years, within a CS department housed in a college of engineering. Dawson (the fourth author) was a TA for one of those semesters. Our HCI curriculum is modeled after *CSE 440 Introduction to HCI* from the University of Washington [3]. The centerpiece of the course is a semester-long group project, in which students work in teams of 3-5 to: identify a topic, conduct contextual inquiries, define a human-centered problem, conduct a task analysis, ideate, sketch, and storyboard potential solutions, create an initial paper prototype, and refine the prototype through rounds of user testing and inspection-based methods. The final prototype is a digital mock-up. The course does not involve programming. Our students are all CS majors, and course enrollments for the past 6 offerings were 49, 14, 41, 51, 66, and 68, with approximately one TA per 25 students.

We three instructors began looking for PCK by meeting together and asking, "Why is it so hard to teach HCI?" Some healthy venting ensued. It was validating to see that our difficulties were not unique. After feeling better that we were not alone, we started asking each other for more details describing the issues and what evidence we had that these things were problems for students. We looked for groupings for the problems and tried to identify what missing

principles or perspectives might account for these problems. We present two broad categories below. *Attitudes and Perspectives* refer to students' general approach to the topic, including what they believe to be knowable and how to determine that something is true. Students may not be aware that they hold a particular attitude or perspective, let alone recognize how it can affect their learning. As instructors, we see evidence of this across all topics within the course. The second category, *Knowledge and Skills*, refers to gaps and misconceptions that pertain to specific methods or topics. As the most recent instructor, E. Wiese describes the teaching strategies used to address specific misconceptions.

3 ATTITUDES AND PERSPECTIVES

As instructors and HCI researchers, we see HCI primarily as a perspective. While HCI methods, theories, and findings are important, the main value we identify is in HCI as a worldview. For us, HCI is about understanding problems before jumping to solutions; appreciating both the strengths and the limitations of human beings; humbly looking to our users for insight into their experiences; and valuing the design process as a way to improve our thinking. When we see HCI primarily as a perspective, it is not surprising that the attitudes that students bring to class have a huge impact on their learning. Unhelpful attitudes include: (1) HCI is easy, (2) HCI is all busywork, (3) only objective data is valid, (4) writing is not part of CS, (5) HCI is about implementing interfaces.

The most salient student misconceptions are in their attitudes about HCI work. On the one hand, students anticipate that the course should be easy because it does not involve programming. This portrays an attitude that the most difficult (and most important) part of design is implementation. While instructors can emphasize the workload expectations of the course, both in hours per week and in required deliverables throughout the semester, students still believe that the class *should* be easy. The incongruence between thinking that the class should be easy but finding that it is time-consuming creates a second misconception: HCI work is "busywork". That is, assignments are interpreted not as authentic opportunities to practice HCI methods but as a way for the instructor to keep students occupied.

We believe both of these attitudes — HCI should be easy, and that it's all busywork — come from an epistemological mismatch. Whether they realize it or not, our engineering students are steeped in positivist epistemology, which posits that only objective, precisely measurable, and quantifiable data are a valid basis for knowledge. HCI requires an interpretivist perspective, acknowledging that the designer actively creates knowledge through subjective interpretation rather than simply recording facts as an objective observer. In many CS classes, students can track their progress and evaluate their work with objective measures, such as test cases or correctness of calculations. The expertise required for CS classes is rightly seen as specialized knowledge. However, students don't always see HCI as requiring specialized knowledge but rather "common sense." These perspectives create a vicious confluence: details of the methods are seen as unimportant because HCI only requires "common sense"; when students do not apply the methods correctly, they cannot tell because they lack objective benchmarks; students reject the instructor's critiques because, again, they are not objective; and in the end, the students conclude that the entire enterprise is meaningless busywork. The subjective, qualitative, open-ended, and iterative aspects of the design-centered HCI process are all incongruent with positivist conceptions of knowledge production and therefore are undervalued and resisted.

Moreover, many of the deliverables in our HCI course are in the form of writing or oral presentations. Many of our students strongly believe that communication, especially writing, is not part of computing. While engineering students may see the necessity of communicating about their work (through, e.g., code comments and documentation) the writing is often seen as a burdensome additional task that is done after the "real" work is completed. In HCI, writing is not just used for communication but also as a way to generate knowledge. The acts of drafting, editing, and revising serve as supports for reflection on the design process and create foundations for interpretivist insight. However, this form of knowledge production is not valued under positivism.

Finally, many students believe that HCI is interface implementation. Thus, students often come to class with a specific idea for a technology they want to design and assume that the course will equip them with strategies to ensure usability. Again, there is an epistemological mismatch: because HCI examines technology in context and takes into account differences among users and their goals, there cannot exist a simple set of interface design rules that will guarantee usability. However, a more vital mismatch is in *what kind of knowledge is viewed as important.* As a field, HCI values our understanding of problems over designing a slick interface. In contrast, students often don't appreciate the importance of understanding their users and their contexts, in many cases because the student is content to design based on their own preferences. Students see themselves as experienced users of technology and therefore as valid representative sources of data for design. Simply explaining to students that they are not their users is not sufficient to dislodge this attitude. It is not just that the interpretivist stance — of trying to understand the perspective of their users by getting into "their shoes" — is unfamiliar and difficult. Additionally, students cannot grasp the value of exploring a problem context if they believe they already understand it based on their own experiences.

3.1 We can help students adopt new perspectives

Instructors can change student attitudes [2]. How to do it in HCI is an open question. We have some ideas to start.

3.1.1 Share broad views of HCI. E. Wiese began one semester by assigning students to watch two keynotes: Amy Ko at Koli Calling 2019 [7] and Ruha Benjamin at CHI 2021 [1]. Both keynotes examine the values that technology enacts, the ways that our values and biases affect how we define problems, and ways that technology can cause harm when the designer's framing of a problem doesn't match their users' contexts. E. Wiese hoped that these keynotes would expand students' views of what HCI is but did not have a way to assess if this occurred.

3.1.2 Explicitly contrast quantitative and qualitative data. An idea we haven't tried yet is to start with a topic that students are familiar with, but is difficult to characterize quantitatively, such as how warm and welcoming a classroom environment is. Students could consider survey results that present likert-scale ratings on different aspects of course climate, from different hypothetical classes. Students will see that the survey data alone could plausibly support several contradictory interpretations. In contrast, notes from hypothetical observations or interviews will be much more informative. This activity may help students see the limits of numerical data and may encourage them to try a new epistemology.

3.1.3 Provide examples of solving the wrong problem. Students might change their perspectives slowly by seeing the same principles across multiple contexts. E. Wiese used ProctorU [10] as such an example in one offering. ProctorU is one of many systems that claim to prevent and detect cheating on remote exams. To prevent a test-taker from cheating, these systems lock the browser, record video, and even analyze gaze patterns to infer usage of off-screen resources. These systems invade student privacy and cause undue stress for students [5]. They are an answer to the question, *"how can we be sure students are not cheating on remote exams?"* If the problem were framed differently, such as *"how can we assess student's learning in a valid way?"*, one could imagine technology that not only looks different but solves a fundamentally more important problem. Students appreciated that example and wanted more.

As a community, we could gather and share examples that illustrate different aspects of solving the wrong problem. Another example, which E. Wiese plans to try next semester, is the story of an 8-year-old who wanted to skip remote school during the pandemic [8]. She would lock herself out of her Zoom account by repeatedly entering the wrong password, triggering a timeout period. When an adult would try to log her back in with the right password, it wouldn't work until the timeout was over (Zoom's inaccurate error messages didn't help). Zoom's security features were designed to protect against a malicious attacker, not for users who want to get out of meetings. By illustrating the difference between problems that technology solves and problems that are important to a community of users, examples like these might help students value an HCI perspective on problem-framing.

3.2 Perspectives interact with knowledge and skills

Attitudes matter for HCI methods. It is difficult to successfully ideate or create a meaningful affinity diagram without trusting the method and taking it seriously. In addition to holding attitudes that conflict with HCI perspectives, students are generally unaware of how their attitudes affect their design process. Students who view the design process as busywork are likely to have self-fulfilling prophecies. Unhelpful attitudes may make it harder for students to use the methods effectively and may reduce their motivation for learning them well. And, students who struggle in learning the methods may dig in more strongly to unhelpful attitudes. Therefore, PCK that helps us teach individual content areas more effectively is also crucial.

4 KNOWLEDGE AND SKILLS

This section explores individual content areas. Our goal is to contribute to our EduCHI community by starting to identify our own challenges in teaching specific HCI concepts and methods, and by sharing ideas for addressing them.

4.1 Problem vs. Solution

HCI problem-framing centers on users' goals, resources, and workflows. A common student mistake is to frame a problem around a technical solution, of the form "users need my technical solution, and the problem is that they don't have it yet." For example, in one offering, E. Wiese assigned half of the student groups to design something that would support undergraduate research. One common idea was to create a website where faculty could post research opportunities and interested students could apply. Students would explain the problem by saying "there is no centralized website where students and faculty can connect for research." And while this statement is true, it does not describe users' needs, constraints, and conflicts. Shallow characterizations of the problem framed it as a simple lack of communication, concealing the nuances that make this problem complex, including, e.g., the differences between student and faculty views of research and the challenges of predicting a good fit between a student and a project. Students struggle to understand the difference between a human-centered problem framing and one centered on technology. Students may also start with a solution in mind, and then attempt to reverse engineer a problem to match. The next year, E. Wiese tried to explain the difference between a problem and a solution with an excerpt from *Braiding Sweetgrass* [6, p. 181]:

> I once met an engineering student visiting from Europe who told me excitedly about going ricing in Minnesota with his friend's Ojibwe family. He was eager to experience a bit of Native American culture. They were on the lake by dawn and all day long they poled through the rice beds, knocking the ripe seed into the canoe. "It didn't take long to collect quite a bit," he reported, "but it's not very efficient. At least half of the rice just falls in the water and they didn't seem to care. It's wasted." As a gesture of thanks to his hosts, a traditional ricing family, he offered to design a grain capture system that could be attached to the gunwales of their canoes. He sketched it out for them, showing how his technique could get 85 percent more rice.

After reading this story, E. Wiese asked the class what problem was being solved. Students noted that the problem was waste, or a lack of efficiency. In response, another student pointed out that there was no evidence in the story that these issues were perceived as problems by the target users. E. Wiese then continued with the hosts' response [6, p. 181-182]:

> "Yes, we could get more that way. But it's got to seed itself for next year. And what we leave behind is not wasted. You know, we're not the only ones who like rice. Do you think the ducks would stop here if we took it all?" Our teachings tell us to never take more than half.

While this example was helpful, students may benefit more from multiple examples. Instruction and practice might take the form of analyzing a set of contrasting cases, where the same problem is framed in different ways. For one kind of practice, students might simply identify which framing is human-centered. For another kind of practice, students could critique a system-centered problem framing by making explicit what problem the system is solving and proposing lines of inquiry that could help determine if that problem framing is accurate.

4.2 Task Analysis

A shallow understanding of task analysis can quickly turn it into busywork: creating a list of obvious actions a user needs to accomplish. "Obvious" meaning that the list could be generated with a superficial understanding of the problem. Common methods for task analysis may not help students distinguish between shallow thinking and insight: breaking a task into hierarchical components or asking 11 questions about the task (from "Who is going to use the system" to "What happens when things go wrong") could easily be done without considering contextual user data. Therefore, students can go through the motions of a task analysis without gaining deeper insight into their problem.

Another student difficulty is telling apart tasks and system features. This confusion likely stems from not understanding the difference between a user-centered and a system-centered view. For example, a student might propose a task like "When students sign up to do research, they will create a profile with their skills and interests." This phrasing describes an action that a user will take with the system, but the focus is on the features of a specific system instead of the steps required for the user to achieve their goal. While the instructor can emphasize that tasks should be phrased in a system-independent manner, students may still be confused about what that means. In the example above, a student could imagine many different ways of implementing a profile and populating it with skills and interests. As such, the phrasing may seem system-independent to the student. Another difficulty may come when students consider actions associated with technology, such as email, setting alarms, or registering for classes. Students struggle to identify the tasks "from a user's perspective" when, for the students, the task is intertwined with the system that supports it.

4.2.1 Teaching Strategies Attempted. One way E. Wiese has tried to counteract this is by presenting task analysis as something that helps the designer understand and communicate about the problem. An assigned reading, a blog post about Proflowers, illustrates the utility of task analysis [4]. By observing and interacting with users in physical flower shops, the designers realized that the key task for a person buying flowers was usually not to construct a specific bouquet but rather to find a bouquet appropriate for a particular occasion. People who buy flowers are often trying to fulfill a goal of getting forgiveness, celebrating, or showing care for someone. However, users may struggle to choose a specific flower that communicates their message. Therefore, the designers realized it would be easier for users to select bouquets from pre-defined categories based on different occasions (among other design choices). Presenting the task in the language of the user, in a way that matches how the user thinks about their task, is also emphasized in this example. While this example seemed understandable to the students, they had difficulty generalizing from just one story.

To help students distinguish between tasks, goals, and system features, E. Wiese created online matching questions with feedback. Students were given a statement, such as the one in above about creating a profile, and determined if it was a task, goal, or system feature. The question presented three contexts (buying flowers, applying for a research opportunity, and selling crafts), with three statements per context (one each for goal, task, and system feature). The phrasing for each statement was based on task analyses conducted by previous students. E. Wiese thought the cohort who did the matching activity conducted better task analyses than their predecessors. However, some groups still struggled in distinguishing tasks and system features in their projects — even after multiple rounds of personalized instructor feedback.

4.2.2 Ideas to Try. While the Proflowers story was helpful, it would be better to have multiple accounts that show the importance of task analysis for specific designs — effectively, worked examples of task analysis. A student could analyze the worked examples to identify: (1) what are the users' goals (which could be accomplished through a variety of tasks); (2) what insight about the users' goals and tasks was gained through the task analysis; and (3) what features of the system supported the identified tasks? This kind of analysis could also help students distinguish between tasks and system features. As a community, we could gather and share these kinds of worked examples. One idea for preempting shallow task analyses in student projects is to have students conduct an initial task analysis before conducting their contextual inquiries. This exercise could help students solidify the inquiry's focus. When analyzing the data, students could note where their initial ideas were wrong, incomplete, or oversimplified. Then, students could refine their task analyses, showing how the tasks changed as a result of user data.

4.3 Contextual Inquiry

A core challenge for students learning contextual inquiry (CI) is understanding why it is necessary and the value of the method's output. Across our experiences, we found that despite explicit instruction on these topics, many students did not grasp a basic understanding of the value of conducting a CI. Even worse, they do not recognize this shortcoming in their knowledge — they don't know what they don't know — which makes this challenging to catch and correct. Additionally, since they do not understand the CI method well, students often allow their own biases into the session; this can affect the questions they ask during the interview and the understanding they ultimately gain from the inquiry. If there are gaps of understanding from the interview, students tend to fill them based on their intuition, leading to results that miss the point of the method.

For example, some students use CIs as a way to get a log of user interactions with an interface. Since they do not yet understand how they will use this data, they don't realize that this data is typically too low level to be helpful. Students also struggle with the line between inserting themselves into the CIs (e.g., treating them like interviews) and trying to stay out of the way too much (e.g., not sharing or checking their interpretation with participants).

Even if students understand the ideas in the abstract, they struggle to implement them in context.

Some students also misunderstood that they should not be asking participants to do something other than what they would usually be doing when completing the task. For example, students sometimes ask participants to try a new app, service, or process, which is more akin to a usability test. Other times, students get tripped up by thinking that if there isn't software or a tool for the participant to use, it is not an acceptable setting for conducting a CI.

We propose a specific exercise to combat learning how to conduct a CI properly. Present a pre-recorded video of a CI. It does not have to be a perfect demonstration of the method; if it contains some common missteps, it can help the student critique the good and bad elements. Have the students write a description of what happened; this is helpful for the instructor to see misconceptions of what context or interpretation is. When it is time for students to execute a practice CI, have them record it. Then they can do the same exercise again, identifying good and bad CI techniques from their recording. For our undergraduate HCI course, we had three assignments surrounding a CI: (1) CI Plan, (2) CI Findings, and (3) CI Final Report. Having the students communicate their ideas through the planning, execution, and reflection phases of conducting a CI made it easier for the course staff to catch any negative trends, misunderstandings, or misconceptions early in the process and help the students overcome them.

4.4 Usability Testing

Some common problems students had with usability testing include: (1) choosing and phrasing tasks; (2) providing the right level of help during a test; and (3) presenting realistic data through a low-fidelity prototype.

4.4.1 Choosing and phrasing tasks. A specific, detailed task in a task analysis demonstrates insight about the problem. In a usability test, that same task phrasing is often over-specified, artificially ordered, or phrased in the language of the system. To prevent this, E. Wiese demonstrates usability testing by starting with a very broad prompt, such as "How would you use this website to help you have a worthwhile research experience?" Essentially, the prompt asks users how they would reach their goal without specifying a particular task. Even so, in one semester, some groups used tasks that were not relevant to their participants' goals. To address this, in the next offering, E. Wiese required that each task in a usability test be followed by a "sanity check" to ensure that the task was relevant to the participant and that the participant found the sample data in the interface to be realistic. While this requirement seemed to help many groups check the validity of their tasks, other groups phrased their sanity checks around the user's experience with the prototype (e.g., "was it easy to do that task with our system?") rather than on the validity of the selected task.

4.4.2 Providing the right level of help during a test. Students would step in and give pointers even after readings and lectures that emphasized not helping participants during a usability test. One confusion point was identifying if the participant was completely stuck. While the emphasis of the instruction was on not providing help, there was a caveat that if the participant was so stuck that the test could not proceed, the team could provide assistance and note it in their critical incidents as a failure of the system. Students had difficulty judging when a participant was truly stuck and would often provide help whenever the participant asked. One idea to address this could be to show students a video or transcript of a usability test. At points where the participant seems stuck, students could answer a multiple-choice question for what the design team should do next (e.g., provide help, encourage the participant to try as if the team wasn't there, or end the test). Only after making a choice would students be able to see how the test proceeded. This could be done with an exemplar usability test (to learn from) or a poorly done test (to critique).

4.4.3 Presenting realistic data through a low-fidelity prototype. Systems are not just made of interface elements. They also contain data that users rely on to make their choices. In the course offering where half of the student teams were assigned to design a system to support undergraduate research, E. Wiese found that several teams used data that was meaningless (e.g., lorem ipsum or squiggly lines) or unrealistic (e.g., a posting about doing undergraduate research with a fictional character). One reason may have been that students were unsure where to find realistic data to present in their prototype. Another reason, which some students stated explicitly, was that the data was out of scope for their design because it was intended to be user-generated (e.g., professors would post opportunities and students would create applications). This perspective is system-centered, focusing on interface components and layouts, rather than user-centered, focusing on the information that users will need to reach their goals. In the next offering, E. Wiese explicitly required that realistic data be presented in the prototypes.

4.5 From Ideating Through Prototyping

From our experience, especially in a classroom environment, students all too often have the end goal of class in mind. Knowing that they would have to produce a final digital prototype, many groups tried to leap to a specific digital prototype too early in the design process, skipping critical ideation and paper-prototyping steps. What they felt was efficiency actually detracted from the course's learning objectives and they did not get the full understanding of the advantages and differences between a sketch, low-fidelity prototype, and high-fidelity prototype. However, it is hard to take a group that went to digital early and tell them they must go back to paper. Next time we will establish clear guidelines in the rubric to ensure the proper level of granularity is enforced so the students can learn the value of each technique.

A student's bias toward a specific design can also limit ideation. Since they have a particular implementation in their mind, they feel that sketching and paper prototypes are just busywork they need to complete before moving forward. Since they already have a plan, they are not using the tools taught in the classroom to brainstorm, communicate, and collaborate over multiple ideas. We tried to prevent this by forcing students to develop multiple ideas, identify radically different ones, and explain them. It's hard to make students explore — or convince them there are more options — aside from doing the thinking for them.

In a previous semester, students didn't iterate on their prototypes enough because they scheduled usability tests back to back. As a course staff, we can prevent this by requiring separate deadlines for each test and revisions to the prototype in response. Students also don't know what they are missing with high-fidelity prototypes. They feel it doesn't affect the testing because they don't see the difference, and there is no way to prove it. We found the only way to combat this is with targeted course staff feedback based on the student's reflection of their usability test.

5 INSTRUCTIONAL PRACTICES FOR DEVELOPING PCK

Our everyday teaching practices can help us develop PCK. When we examine student work, we look for evidence of understanding and misconceptions. When we change a lesson plan in response to a common incorrect answer, we create instruction that is targeted to our newfound PCK. To make this process more efficient for ourselves and more useful for our community, we just need to document our steps along the way.

Our PCK comes from examining students' ideas about HCI in light of our instruction. We gather those student ideas through our assessments. Therefore, the design of our learning goals, instruction, and assessments affect the kinds of PCK we will gain. Our attitudes will also affect how we develop PCK. When looking at student work and asking ourselves, "how did they not understand that?" it can be tempting to answer by blaming the students (e.g., *they didn't study enough*) or ourselves (e.g., *I'm just a bad teacher*). A blame-focused answer will never yield useful PCK. Instead, our follow-up questions should focus on identifying the knowledge, skills, and attitudes that are missing or misaligned. The depth with which we can ask these questions will depend on the format and quality of our assessments. In particular, E. Wiese found two types of self-reflection activities to be especially useful for PCK:

5.0.1 Design Process: Initial Thoughts and Reflection. In the first week of class, students propose what steps they would take to design a specific new technology or improve and existing one (e.g., to improve a university website). Students are prompted to consider specific issues, including what information they need about their users, how they will get that information, and what steps they would take between getting an initial idea and implementing the final design. At the end of the semester, students critique their initial response, explaining which parts of their thinking they still agree with and what they have changed their minds about. The pre-test shows that most students gravitate to surveys to get information about their users, do not consider brainstorming or exploring multiple ideas, and do not plan for user testing until a functional prototype is ready. The pre-test can reveal misconceptions about the design process before the semester starts. The reflection afterward can show what high-level ideas the student finds important at the end.

5.0.2 Reflection on Specific Methods. For the key HCI methods in the course, students practice the method and then write a reflection. In these reflections, students identify instances when they used the methods well and when their usage could have been improved. For example, for CI, students choose from different categories (e.g., context, interpretation, focus, etc.) and note when their inquiry correctly demonstrated that aspect of the method, and when it did not. Scoring is based on the accuracy of their self-critique, allowing students to make mistakes in their initial use of a method but earn full points on their final understanding of it. These assessments show us what students are learning about the methods (in contrast to what they are learning about their users).

Open-ended questions where students generate an answer from scratch are often thought to be the gold standard of assessments. For example, to assess if students can introduce a usability test to a participant in a ethical manner and yield valid data, we could ask the students to write a script. However, we may still not get a complete picture of students' understanding (e.g., a student may write something correctly, but believe that alternative options are equally good). An alternative is to use critique questions, where students are given a script and asked what was done well and what should be improved. That way, the instructor can target specific concepts and ensure that all students engage with them. However, in both cases, students are still generating text (either a script or a reasoned critique of a script), which is time-consuming to grade.

Fortunately, research on HCI pedagogy can guide us in creating informative multiple-choice questions [12]. Multiple choice questions are often thought to be easier than open-response and to be ineffective for assessing critical thinking. Wang et al. designed multiple-choice questions with options based on past students' correct answers and misconceptions [12]. In a controlled *in-vivo* experiment, the multiple-choice questions were just as difficult as the matched open-response questions, and answers for the open-response questions used ideas and phrasings similar to the multiple-choice options. Luckily for us, the path from an initial set of open-response answers to an easy-to-grade multiple choice question is the same path we walk to develop PCK: explore student thinking, identify types of misconceptions, and determine which ones are common.

We can develop PCK by conducting action research in our own classrooms. For assessment of new topics, we can use open-response questions to elicit a range of student ideas. For topics we have taught before, we can generate multiple choice questions from the previous year's open responses. If our assessments reveal extensive misunderstandings, we can deploy similar multiple-choice questions after the targeted instructional activities (to tell us how effective the instruction was). Keeping and sharing notes on our process can help us anticipate student ideas, point us to new teaching strategies to try, and save us from approaches that others have found to be less effective.

6 CONCLUSION: PCK FOR ALL

Any of us can try this approach for developing PCK. Reflect on your teaching. What was the most complex concept to get across? Where did students have the lowest scores on their projects or exams? When you gave partial credit, what elements were students most likely to miss? Look at your student course feedback if you have it. What did students complain about? All of these data sources can be foundations for finding PCK. The key to looking for PCK in this kind of data is to look for knowledge gaps, misconceptions, and unhelpful perspectives. While it is tempting (and sometimes

even accurate) to explain students' difficulties with lack of effort on their part, that line of reasoning doesn't lead to developing our own PCK or to actionable improvements to instruction. You can do all of this alone, but it's more fun with friends. Then report back and tell us what you found.

We can develop PCK as a community by (1) reflecting on our own experiences, (2) trying out new ideas that we develop responding specifically to our growing PCK, and most importantly, (3) sharing our techniques across the community. We also need to make our assessments informative for PCK and not be too burdensome for students to complete or for the course staff to grade. Developing PCK as a community should be a rewarding and useful activity. There is no risk; you can try this at home! PCK should be developed for all conceptual levels, whether to influence overall attitudes about HCI as a field to the fine details of conducting a usability test. Developing PCK goes hand in hand with individual improvement to our teaching methods and materials. Sharing what we know with others does take additional effort; it requires reflection, refinement, and communication. But through collaboration, we can significantly improve the quality of instruction and our student's understanding. Let's do it together!

REFERENCES

[1] Ruha Benjamin. 2021. Which Humans? Innovation, Equity, and Imagination in Human-Centered Design. https://www.youtube.com/watch?v=kDcz44ifdQw

[2] David Blazar. 2018. Validating Teacher Effects on Students' Attitudes and Behaviors: Evidence from Random Assignment of Teachers to Students. *Education Finance and Policy* 13, 3 (07 2018), 281–309. https://doi.org/10.1162/edfp_a_00251 arXiv:https://direct.mit.edu/edfp/article-pdf/13/3/281/1692704/edfp_a_00251.pdf

[3] James Fogerty. 2015. CSE 440 Introduction to HCI. https://courses.cs.washington.edu/courses/cse440/15au/index.html

[4] Intuitive Design Group. 2021. proflowers.com A Case Study in UX. https://www.intuitivedesign.com/proflowers-story

[5] Jane C. Hu. 2020. Online Test Proctoring Claims to Prevent Cheating. But at What Cost? *Slate* (October 2020). https://slate.com/technology/2020/10/online-proctoring-proctoru-proctorio-cheating-research.html

[6] Robin Kimmerer. 2013. *Braiding Sweetgrass: Indigenous Wisdom, Scientific Knowledge and the Teachings of Plants.* Milkweed Editions, Minneapolis.

[7] Amy Ko. 2019. 21st Century Grand Challenges in Computing Education. https://www.youtube.com/watch?v=mjX3yLPKjvE

[8] Robertas Lisickis. 2021. Modern Problems Require Modern Solutions: 8 Y.O. Skips Online Classes For 3 Weeks Using A Zoom 'Hack'. *Bored Panda* (February 2021). https://www.boredpanda.com/8-year-old-zoom-school-exploit-story-twitter

[9] Alannah Oleson, Meron Solomon, and Amy J Ko. 2020. Computing Students' Learning Difficulties in HCI Education. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–14.

[10] ProctorU. 2022. https://www.proctoru.com/

[11] Lee S. Shulman. 1986. Those Who Understand: Knowledge Growth in Teaching. *Educational Researcher* 15, 2 (1986), 4–14. https://doi.org/10.3102/0013189X015002004 arXiv:https://doi.org/10.3102/0013189X015002004

[12] Xu Wang, Carolyn Rose, and Ken Koedinger. 2021. Seeing Beyond Expert Blind Spots: Online Learning Design for Scale and Quality. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 51, 14 pages. https://doi.org/10.1145/3411764.3445045