ITiCSE 2025 sponsors included Gold Supporter Jetbrains and Silver Supporters Codegrade and Oracle, as well as SIGCSE, ACM Europe, and Informatics Europe. We wish to thank everyone who made this conference the great success that it was, including the organizing committee, steering committee, student volunteers, and all the presenters and other attendees. We hope to see you at the 31st ITiCSE, at the Universidad Rey Juan Carlos in Madrid, Spain, 13-15 July 2026.

Member Spotlight: Juho Leinonen

By Juho Leinonen, Julie M. Smith and Charles Wallace

Juho Leinonen is an Assistant Professor of Computer Science and Academy Research Fellow at Aalto University.

How did you first get involved with the computer science education community?

I'm somewhat of a rarity in the computing education research (CER) community in that I started my research career directly in CER, although that's becoming more common nowadays. I started as an undergraduate research assistant in 2015. I was mostly interested in applied machine learning, but I was also interested in teaching, so I thought that computing education research (specifically ML applied to computing education) would be a great match for me. Luckily, there was a lecturer, Arto Hellas, at the University of Helsinki actively doing research in this area. After my first conference, Koli Calling 2015*, where I presented my first paper ever, I was hooked and decided I want to pursue a PhD in CER. Koli Calling as my first conference experience was very fun. I got to discuss research with professors while drinking sparkling wine in a jacuzzi. The downside was that it gave me an unrealistic expectation of what typical conferences might be like – I was disappointed to learn that not all of them include a spa and sauna session! A couple of years later, in 2017, I officially started my PhD, although some of my PhD work was already done as an undergrad and as a master's student.

* Serendipitously, I'm now the senior PC chair for Koli Calling ten years later.

Photo credit: Irma Savolainen

Can you describe some of the ways you've contributed to the development of computer science education?

In my PhD, I worked on keystroke dynamics for authenticating students in online courses. Keystroke dynamics is a fancy way of saying "identifying people based on how they type on the keyboard". The University of Helsinki had a very popular massive open online introductory programming course, and it was possible to get admitted into the university if you did well enough in the course. However, it was hard to know if the student who came to the final, proctored exam on campus actually completed all of the online, unproctored exercises themselves. Thus, in my PhD, I studied whether we could match the typing patterns of students taking the proctored exam to their typing patterns in the unproctored exercises, and we found that it was possible to verify that students had completed all the exercises themselves this way. We also found that you can infer other interesting information about the students based on how they type, such as whether they have programmed before. Those who had were much faster at typing special characters, for example.

I finished my PhD at the end of 2019, and a couple of years later, I was lucky to get exposed to generative AI relatively early. I quickly realized it will have a huge impact on what programming will be like in the future, and also on how we will teach students (especially in introductory programming). In the past couple of years, I helped organize two

ITiCSE working groups where we studied the impacts that generative AI is having on computing education, which hopefully has helped the community better understand the impacts it already has and will have in the future.

Another project that I'm proud of is developing Prompt Problems together with my colleagues Paul Denny from the University of Auckland and James Prather from Abilene Christian University. During my postdoc in 2023 at the University of Auckland, in the beginning of the year when ChatGPT was still relatively new, we ran a session where we asked students to solve a relatively simple programming problem purely by prompting ChatGPT. We were shocked at how poor some of the prompts that students wrote were, and we thought about how we could teach them to prompt AI models better. This led to the creation of Prompt Problems, where students are presented with a visual problem description*, and their task is to then write a prompt to generate code that solves the problem. We've since found that students enjoy these types of exercises, finding them relevant to the real world, and that students' prompts get better with practice. It's been rewarding seeing Prompt Problems being adopted by multiple universities around the world as well as a couple of commercial programming education platforms.

Beyond research, I've been quite involved in service, e.g. chairing the Koli Calling conference, being a hybrid chair and a submissions chair for ICER, a WG chair for SIGCSE virtual, organizing the CSEDM workshop that aims to bridge the computing education community with the educational data mining and learning analytics communities, as well as reviewing and being a senior PC member for many conferences in our field.

* Back then, we thought this was a clever way of the students not being able to copy-paste the problem description into ChatGPT, but then half a year later GPT-4 was updated to be able to take images as input.

Where is computer science education headed in the next 5–10 years?

I think that generative AI is probably the biggest change to computing education since the advent of the internet. We will have to figure out how to teach students to operate in this new era where generative AI is ubiquitous. I think in 5-10 years, we'll have a lot more programmers operating at a new higher abstraction level of programming, where their work will largely be managing AI agents, prompting them with tasks, and reviewing, editing and debugging code produced by AI.

Traditionally, syntax and the high rigidness of programming have been a barrier to enter computer science. I think that with generative AI, the barrier is now lower. Although there has been discussion about enrollments seeming to be down, I think we'll have a lot more people doing some form of programming than before – and I'm counting educating people outside of universities under computing education, so I think this is relevant to think about for our field.

Due to the lower barrier to start programming, we should think about how to teach students who might only be interested in building an app or a website for fun (e.g., those taking a single programming course or minoring in CS). This might now be more realistic for students to be able to do after just a single programming course. I think Leo Porter's and Daniel Zingaro's book "Learn AI-Assisted Python Programming" is a great example of how this might be operationalized in practice. One common criticism of this approach is that "but students won't then learn the fundamentals!" However, I don't think it matters that much if they only want to build that one app or website for fun. They're not going to be building critical systems. Thus, I think the learning objectives need to change, with more emphasis on testing code (that the student might not fully understand), reviewing code produced by AI, and being able to decompose larger problems into smaller chunks that AI can write the code for.

For our CS majors, I think the main question will be how we teach them to use generative AI productively without developing an overreliance on it. While we might not care if the non-majors building a single app learn the fundamentals, we definitely care about it for those majoring in CS. It seems that AI is a productivity booster for experts as they can quickly judge whether the produced code is useful, but the results aren't as clear for novices. I hope that in 5-10 years, we will have figured out how to ensure students learn to use AI in useful ways.

What are the biggest challenges facing the CS education community?

I kind of mentioned this already in the previous answer, but one huge challenge is how to teach students to use AI productively. For our majors, they'll definitely need to be able to use it. If they don't use it, they'll fall behind compared to their peers who are using it. If they over-rely on it and are not better than the AI in isolation, why would anybody hire them? We need to figure out how to help students to be ready to enter the workforce one level higher than before, as managers of AI agents rather than as the junior developers of the past whose main tasks were completing small code writing or refactoring tickets, which can now be handled by AI. Another challenge also caused by generative AI is how to motivate students to learn programming when it looks like there's this tool widely available that can do it. This is somewhat analogous to motivating kids to learn basic math when they know a calculator can do it. For kids, we just force them to not use calculators before they learn the basics, but this seems unfeasible to do at the tertiary level where most of our students are adults who are there out of their own free volition.

I realize many of my responses to these questions are about generative AI, partially because it has been my main research interest for the past couple of years, and partially because it is having a huge impact on our field. However, there's one challenge not directly related to generative AI*, which is that we're seeing more and more paper submissions in computing education, while our venues have not been able to keep up with the pace in recruiting reviewers. Part of it might be due to the field getting more popular – which is great! – but part of it is caused by the same submissions being resubmitted again and again, causing a lot of review work. As an example, if a paper is submitted five times before eventually being accepted, it will have probably caused the need for 20 reviews (15 regular and 5 meta-reviews). For our conferences, each of these reviews might be written by different people, even though it would be a lot more efficient if we could match reviewers across conferences and get the same reviewers to re-review the submission. I think we should consider alternatives, such as the Reciprocal Reviews effort led by Amy Ko. Another alternative would be to go with something similar to the ACL Rolling Review system, where multiple conferences would do peer review through a single system and authors would need to show they've considered reviewer comments when they resubmit work. Ideally, the same reviewers would review the submission again when it's resubmitted, similarly to journals.

* It could be indirectly related to genAI though – maybe authors are using genAI to accelerate how quickly they can do research, and there are now a huge number of papers studying different impacts of genAI on computing education, which might be partially responsible for the increase in the number of submissions.

What are the biggest challenges for diversity, equity, and inclusion in CS education today? And what can CS educators do to help encourage diversity?

I want to start by noting that I'm speaking from my quite privileged, European/Finnish perspective where the DEI challenges are partially different compared to other countries and their contexts. From my point of view, a recent challenge is that generative AI might be widening the gap between experts and novices, and not just in CS. In all fields, those who know more are more able to know when to use AI, e.g. what tasks can be delegated to it. They're also more capable of evaluating AI outputs quickly. One of the challenges that we're already seeing is that fewer and fewer junior positions are available in the industry, potentially because senior programmers can use AI for tasks they would've delegated to junior programmers in the past. Even though diversity in CS is still in a pretty bad shape, it has gotten better over the years, but there's now a risk that our students won't be able to get junior positions in the industry and never enter the workforce, which might leave the current composition of people in the industry in the same unideal composition that it is in currently. Even if we can attract more people from diverse backgrounds into universities, it might be in vain if they never enter the workforce.

Despite many important downsides like potential biases in outputs, I think there are potential ways that generative AI could also help attract more diverse students into CS. For example, generative AI can be used to create highly personalized exercises that could be tailored for students' interests and cultural backgrounds, potentially making CS studies feel more relevant to them. I think there's potential that whole courses could be personalized to guide

students on how to use programming in their lives and specific contexts for the exact tasks that attracted them to CS in the first place.

What do you enjoy doing when you're not working?

In my past life, I used to spend way too much time on video games. My main vice was World of Warcraft. In high school, I'd play around eight hours a day – pretty much all of my free time. Once I started my PhD, I had to drop it, although I've occasionally picked it up again for short periods of time. Nowadays, I mostly play analog games. For example, Magic the Gathering (mostly the commander format), Dungeons and Dragons, and board games (recent favorites include Brass: Birmingham, Wingspan, and Dune: Imperium). In addition to games, I've tried to get a little more physically active and now play padel and badminton semi-regularly. I also enjoy traveling with my partner – our next trip will be to South Korea.