Reflections on Computing Education Research

Steve Draper
steve.draper@glasgow.ac.uk

ABSTRACT

In this paper and the talk built from it my general aim is to offer some reflections on the state of Computing Education Research (CER) right now (August 2020) – not like a review of the literature that looks only backwards, but to develop perspectives partly from outside Computing, that may cast various different lights on it, and thence make some suggestions about where it could perhaps go in future.

CCS CONCEPTS

• **Social and professional topics → Computing education**.

KEYWORDS

Computing education

ACM Reference Format:
Steve Draper. 2020. Reflections on Computing Education Research. In *United Kingdom & Ireland Computing Education Research conference. (UKICER '20), September 3–4, 2020, Glasgow, United Kingdom.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3416465.3416466

1 MULTI-DISCIPLINARITY

Think about both CER (Computing Education Research) and Computing as inherently multidisciplinary subjects. This links a number of different points. Firstly: CER researchers would seem to need a grasp of education theory and practice, as well as of computing theory and practice. A large majority of CER, as shown by authors' departmental affiliations, is based within computing departments at both UK-ICER20 (≈ 80%) and ICER20 (≈ 80% of authorship, however only ≈ 67% of papers had no input from Education based authors). Being based in computing departments is good for having strong connections with computing, and also good for access to computing classes to use them as participants. It is less obvious how they secure nourishment about education. Furthermore much of the published CER research and all the ten papers of this conference use quantitative research, which implies some access to methods usually located in yet other disciplines such as statistics and psychology.

2 THREE DIFFERENT ROLES OF TEACHERS

Whether their job title is "graduate teaching assistant" or "professor", the people organising learning for students have to cover three separate functions, and all of them are worth researching: a) "Delivering" a learning design i.e. managing and/or performing as part of it; b) Selecting, or inventing, a learning design; c) Deciding on the curriculum i.e. what it is that is to be learned. All of these are important subjects of research in education. An example of research on learning designs is McGregor & Maarek's paper at this conference "Software testing as medium for peer feedback" which applies the general idea of reciprocal peer critiquing[1] and develops a way for this to work on introductory programming courses. There is less focus on delivery, although many experience reports touch on important aspects of it: perhaps CER could develop this line of research more directly, starting by "mining" published experience reports. The third role, deciding on the curriculum, definitely merits more attention. CER clearly includes studying learners and teachers, and so includes studying the curricula produced and taught: both knowledge and skills, both intentionally and unintentionally, both explicitly and implicitly.

3 IMPLICIT AND EXPLICIT KNOWLEDGE

Implicit and explicit knowledge is a pervasive issue. In considering the broad subfields of curriculum design, these three should be noted on the spectrum of explicit/implicit knowledge:

(1) Knowledge that is explicitly taught, and directly assessed.
(2) Knowledge known by teachers to be important, but not directly assessed and perhaps not explicitly taught: basically cases between (1) above and (3) below. E.g. learning how to teach yourself another programming language without tuition, which might need an exam a month long to assess directly.
(3) Implicit knowledge: that is not stated, not explicitly taught, and at best accidentally assessed; but which is of professional importance in the careers the discipline feeds into.
    For instance:
    (a) Learning how to work in teams, including how to rescue a dysfunctional team when you are assigned to one.
    (b) How to write reports clearly, as measured by the reader's comprehension.
    (c) Commenting code, measured by the speed for someone to understand the code and modify it.

The implicit/explicit distinction applies equally to skills. Teachers have variable implicit skill at, say, chairing discussions in a group of students; or at "contingent tutoring" i.e. explaining concepts to one student not by a monologue that guesses (nearly always wrongly) the level needed, but by a dialogue in which they find interactively the level of partial understanding and intervene at that level (not too low level, not too abstract for that student at that time). Learners acquire enormous amounts of learning skills in their time at university, mostly implicit. Mazur's "peer instruction"[2] not only deepens students' understanding of the concepts but as a "side-effect" gets them into the habit of asking themselves similar

UKICER '20, September 3–4, 2020, Glasgow, United Kingdom

ACM ISBN 978-1-4503-8849-8/20/09.
https://doi.org/10.1145/3416465.3416466

[1] Reciprocal Peer Critiquing (clickable)
[2] Peer Instruction (clickable)

questions about new material in future, without this being prompted by teachers or texts.

With all that in mind, we can see that a fruitful and complex area for CER concerns the issue of what should be taught in computing. Some methods for doing this include:

- Look at cases where different curricula have been taught, and consult employers and mid-life professionals about whether some curricula have had better effects than others.
- By studying the implicit (and thus hidden) curriculum, enlarge the range of questions you ask of these sources. I.e. it is not just the documented curricula under enquiry, but the implicit aspects of it too.
- Consult subject matter experts e.g. HE computing academics. This is the obvious source to most people outside computing, and indeed to the government at various times.

4 DISCIPLINES ARE PERIODICALLY RECONSTRUCTED

However there can be drawbacks to simply consulting computing academics about what needs to be taught and learned. The first is that periodically every discipline undergoes significant change in what is thought to be the most important knowledge in its field.

To take the example of Biology: my university campus, built in the 19th century, has one building with "Zoology" engraved deep in the stone, and another with "Botany". However over the same period that computing has existed as a discipline, it has emerged that those are just two of five equally important "kingdoms". Furthermore, looking at biology from the angle of key theories, most people outside biology think there is only one important theory: Darwin's. But Sir Paul Nurse in a Jan 2018 public lecture[3] suggested that there have been four fundamental theoretical advances in Biology and that a fifth is imminent. So the intellectual structure of biology has been revolutionised since 1950. It might be time for Computing to be restructured around what is most important things to teach.

Another approach is ask what and how a discipline contributes to notable achievements, especially ones that contribute to the economy. In December 2019 Hannah Fry presented the Royal Institution's Christmas lectures[4]. She used current examples of innovations, all involving computation. For each example she identified the discipline central to its operation, and its essential quality. She said that the great thing about maths is the certainty of its reasoning (example: the man who sky dived from 30,000 feet into a big catch net without a parachute). The great thing about computers is they can do exact calculations much faster than humans – and given time limits in some problems, this can be a critical benefit e.g. matching kidney donors to recipients. The key quality about statistics is that some important situations do not allow you to calculate individual events deterministically BUT are highly regular when many trials are combined. This is true of a lot of human behaviour e.g. traffic flows, pedestrian flows, predicting how infectious disease does and doesn't spread, with or without vaccinations. A key quality about probability is correctly combining multiple bits of partial information e.g. combining many sensors so that an alert from any one shuts a vehicle down; or using Bayes theorem applied to imperfect medical diagnostic tests to give the true chance of this test result meaning you are ill or clear of an illness.

The second reason why relying on academic Computing researchers for what should be taught may be a poor idea, is that disciplines are typically formed by people who taught themselves something new without help. They naturally infer it is therefore easy for everyone to teach themselves. Maths, in contrast, after hundreds of years is taught to all children and many undergraduates, but the style of learning and teaching at those levels is completely unrelated to the training needed for a future contributor to original maths research. Computing hasn't learned this lesson, though CER people have to some extent. The study of why CMU had so few females on the Computing major brought out the causes of this.[5] In brief, the course tacitly assumed that all computing students had explored and used and programmed computers as a hobby all their life (10,000 hours?), so it didn't have to teach most of this; and the students typically spent 16 hours a day doing even more of it because it was the love of their life on top of being their education.

Even worse, today in Scotland using advice from Computing in HE for what to teach in schools is proving even more magnificently wide of the mark. What is needed is the knowledge of Education people, of school teachers, about what can be learned and what kind of teaching does or doesn't work on each age group.

5 PROGRAMMING AS HOBBY, REASONING, AND ENGINEERING

The issue that may simultaneously show the biggest tensions within computing as a discipline, and be the single biggest topic for computing education is how the "same" subject of learning to program can be seen in three different ways. The sad thing is that for decades this has mostly been treated as alternative ideas only one of which can be true. The educational viewpoint should probably be that these alternative conceptions are alive, strong and already in our students: and if they are implicit that just makes them more not less influential. So perhaps educators should actively show how those alternatives apply to each situation, as physicists do in teaching about wave and particle "views" of electrons and photons. One view is of the hobby, the creative angle, the joy of making stuff just for yourself regardless of whether it will work in all situations or as a temporary toy. The reasoning angle can produce the deepest understanding, but is only as good as its axioms – and as was eventually learned from Euclidean geometry, that doesn't necessarily include understanding how those assumptions relate to the actual world. The engineering aspect is about dealing not just with what we want, but with all the things that have to work in practice whether we understand them or not: and how safety factors and so on allow engineers to design successfully without understanding everything. If you prefer to relate your CER research to a theory, then Papert's theory of "constructionism" seems to be a serious attempt to relate these three views both to computers, to programming, to learning, and to concrete activities.[6]

[3] Paul Nurse on landmark Biology theories (clickable)
[4] Hannah Fry's Christmas Lectures (clickable)
[5] CMU's project on getting women into computing majors (clickable)
[6] Papert's constructionism (clickable)