Reflections on Computing Education Research

Steve Draper, Psychology, Glasgow University
http://www.psy.gla.ac.uk/~steve/

The paper in the proceedings has footnotes leading to a companion web page with references for this talk,
As does this short-form link:

http://tiny.cc/qqcosz

Introduction

Me (and this talk)

I have an MSc in computing, but only a conversion course one. My knowledge of it is thin – though enough to have embraced many of its attitudes / values. I have always liked working with computer scientists: their love of truth (because they created the things they are certain about) and their open-mindedness (because it's all relatively and fast-changing).

Of course, I've gradually realised that what is enjoyable or agreeable isn't necessarily what is good for me (or you).

I have worked in related (interdisciplinary) fields: Artificial Intelligence (1970 style), HCI, Psychology of Programming, Educational technology, Education. So I have naturally got both experience in interdisciplinary work, and an interest in how it works.

(Me and) This talk

CER = Computing Education Research

I'm going to use my position as a part outsider to comment on CER; and to convince you (if you are a CEResearcher) that you too are partly an outsider to Computing Science. And that that is probably a good thing.

I plan to have 2 "motifs": things which are mentioned repeatedly in many sections of the talk:

a) New research topics (or at least neglected ones) in CER
b) Multi-disciplinarity: that what you / we do, usually involves more than one discipline but that this fact is under-attended to.

This talk

(besides the motifs) Here are the contents – the main sections of the talk. It is unlikely I will get to the end ....

- The 10 papers at this conference, and their topics.
- Multidisciplinarity
- Learning to program: the perspectives of Hobby, Reasoning, Engineering.
- School education related to computing.
- Selected educational concepts e.g. the 3 separate roles of a teacher;
- ... and how these might be a further source of topics for CER.

The topics in the 10 papers at this conference

The topics in the 10 papers at this conference

(with two cases of double counting):

6	Introductory programming courses
2	Applying the learning design of Reciprocal Peer Critiquing (RPC)
2	Creativity
1	Learners feeling of belonging (or lack of it): A social factor with serious impact on learner motivation.
1	T-practice changes from Covid.

Comments

The list of topics represented by the conference papers is one way of getting a snapshot of what the current foci of CER are.

Seeing so many based on studies of introductory programming is also interesting because it reinforces the impression that of all the new and old topics in compSci, and the different views taken on what is most important, it does seem that the majority of academics see the subject as revolving around actual programming.

RPC connects to the "third factor" in learning (besides Ls and Ts): *peer interaction*.

To my mind, it becomes ever more peculiar how we humans seem unable to think by ourselves ("How can I know what I think, till I hear what I say?" as one wag said.)

Comments (2)

Creativity is a topic not new, yet not well studied in CS. It might bear fruit to do a project on how it works in different disciplines. In particular: in some disciplines the feeling is that you want to learn the one right answer and that brings competitive feelings about getting that answer. But in Art schools students are under pressure from the start to be "original": to produce something as different as possible from each other person in their class, and often feel how oppressive that is. Understanding this better might bear dividends.

Learners' feelings of belonging, which in this paper are about gender and race, are part of the general issue of how feelings of social connectedness have an impact on a student's academic success as Tinto addressed in his concept of "Social intergration" (a companion to "academic integration").

Comments (3)

Finally the paper on Covid's impact on HE teachers shows that CER should indeed be researching Teachers as well as Learners; and is also the most likely topic for immediate research on the impact of Covid on education in HE.

My suggestion would be that impacts on social connectedness (for both teachers and learners) might be a particularly good measure to focus on, both in itself, and for its impact on peer interaction as a key part of learning. Furthermore, the issue (mentioned later in the talk) of implicit vs. explicit skills and knowledge may well be a way of understanding why Zoom type meetings leave people so exhausted currently because much of our social interactions are implicit skills and having to focus on them consciously is both a strain and is often done less well, reducing the success of such meetings.

Multi-disciplinarity

Multidisciplinarity

a) The six papers on introductory programming are notable because many of them bring in an external discipline to tackle an aspect of it e.g. Ethel Tshukudu's paper applies concepts from Linguistics, Jack Parkinson's applies the concept of spatial skills and Albina Zavgorodniaia's that of cognitive load from Psychology — disciplines that are neither computing nor education.

b) CER already involves CE i.e. both the Computing and Education disciplines.

c) And CER furthermore frequently involves statistics and experimental design from (for instance) Statistics or Psychology

Multidisciplinarity (2)

d) The major modern applications of computing use computing but also intrinsically involve other disciplines in the heart of their design. This was illustrated by Hannah Fry in her RI Christmas lectures, where she showed this, and also succinctly stated what each discipline was good for. This might perhaps be something that should be explicitly taught in CS courses, instead of leaving students with the impression that computers are the centre of interest in giving value to the economy and changing people's lives.

Hannah Fry's perspective

- **Mathematics**: its great value is the degree of certainty. [sky Dive with no parachute]
- **Computers**: can do exact calculations much faster than humans.
- **Statistics**: some important cases do not allow deterministic calculation BUT are highly regular when many trials are combined e.g. traffic flow, weather prediction.
- **Probability**: correctly combining multiple bits of partial information. e.g. multiple sensors, imperfect tests.
- **Machine learning**: programming is so difficult in some cases e.g. face recognition that handing it over to machine learning is the only way.

Perspectives on "programming":

As a hobby, as reasoning, as engineering

The 3 viewpoints

Programming can be seen as a **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. [creativity was one theme in the 10 papers]

As **reasoning**:
produces the deepest understanding, but is only as good as its axioms – and that doesn't necessarily include understanding how those assumptions relate to the actual world.

As **engineering**:
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.

The nature of programming

While both programmers and academic computer scientists know this implicitly, there is little explicit acceptance of the three viewpoints on programming.

It is important firstly because new students arrive frequently with only one of these and don't connect with it as staff would wish; and because staff themselves regard one of these as "right" or official, and tend to deny the others. Yet surely a rational science AND a rational teaching approach would address all three, and perhaps directly teach aspects of all three e.g. debugging, how to play, ...

Choosing which language to teach

One of the places where the different perspectives appear in Computing departments is when a programming language is chosen for students to learn.
As mentioned, learning to program is the most-agreed on part of an HE degree in computing; so this is an important issue.

- Will it make it easy to create visually appealing outputs? [hobby]
- Is it full of features that are useful (only) in managing the development of large programs? [Software engineering]
- Is it a language that is easy to reason about? [reasoning, proofs]
- Is it a language that is currently appearing in job ad.s for programmers? [immediate market appeal]

Choosing which language to teach (2)

Making the reasoning about the choice of programming language explicit to the students could only be good, and help them understand the interplay of different properties.

Having exam questions on the pros and cons of each language would be good to carry through on this.

There is also a theory (Papert's "Constructionism") that could prove to be a basis for teaching students to reason about these perspectives.

School Education related to Computing

The relationship of school – HE – jobs

There is a general educational issue of how learning should be related to a) the school version of a subject, b) to the HE version of a subject, c) to the knowledge and skills used in jobs.

Many subjects are not taught in schools – there is no automatic reason why an HE subject should have a counterpart in school.

There is also the issue of how a school version of a subject mis/matches the HE version, leading to disappointment in learners who chose an HE subject on the basis of liking it in school.

There is also the issue of how a hobby has led to HE subjects assuming prior learning from the hobby – notably identified in CMU's study of why so few girls did computing in HE.

(These are all good research topics for CER.)

Advice from HE experts

In recent years there have been a lot of ideas from HE computing about what might be taught in schools e.g. computational thinking.

However there are many reasons why this should not be taken for granted as a reliable source of advice.

The simplest arguments against naive advice from HE experts

A) Constructivism
(the most basic and widely held theory in education)

1. Learning & teaching is NOT plugging a pendrive into a L.
2. Learning is more than mere information; it involves finding the L's prior knowledge and weaving the new into that.
3. Journey planning is not a function of the destination only; but of both the start and the end places.
   Learning is the same: defined by BOTH the ILO of the teacher AND the learner's prior knowledge, experiences, and perhaps misconceptions.

=> Experts in computing at best know only about the destination.

The simplest arguments against naive advice from HE experts (cont.)

B) Natural Egocentrism

1. Remembering how you learned, especially in a new discipline, is not a model because CS academics are not typical students. Especially true in a new discipline.
2. Getting a job based on your research ability has no predictive value for how well you will teach: they are two independent qualities, and all 4 combinations of talent and ineptitude are observed in practice.

The simplest arguments against naive advice from HE experts (cont.)

C) School children especially before age 7 are just not like adults in how they think and learn.

Primary teachers, and still more "Early years" teachers are specialists in children of this age and how they think.
CS HE academics do not have this expertise.
They do not even know there is something they are ignorant of: meta-ignorance as well.

There is nothing surprising or disgraceful about this: How many disciplines can *you* name? There are probably 60-100 in a large university. But this is a good reason for not adopting educational advice in vacuo from a CS academic.

Disciplinarity again

This is another issue related to the need for multidisciplinarity.

Furthermore, even if a discipline is taken as the sole authority of its nature and importance, this changes because disciplines are periodically restructured (partly because changes in fundamental concepts eventually change the self-concepts a discipline has). Disciplines evolve, split, fuse.
Computing is about 50 years old (1965 Manchester first undergraduate intake to CS; 1971 at Cambridge). Biology has changed massively in the last 50 years – Botany and Zoology are not the main divisions any more.

It may well be that the essential nature of computing will be reconceived shortly, making previous arguments about its essence outdated.

Selected educational concepts

The 3 (not 1) roles of a teacher

3) Delivery of a learning design: managing and/or performing it.

2) Selecting or inventing a learning design: an activity that has the effect of learning.

1) Deciding on the curriculum – on what is to be learned.

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.

Intrinsic and extrinsic motivation

These are logically distinct and can be measured separately: it is a 2-dimensional construct. Quite a bit of CER asks learners how interested they are etc.

The tricky bit is that we, and learners, are mostly aware only of the dominant one, as if there was a single scale with intrinsic at one end, extrinsic the other. Like sweet and sour.

A related, but important and neglected, concept is of "appetite". We are familiar with how much we like to eat; but also know that the desire disappears when we reach satiety; but reappears next day.

A person might easily be strongly intrinsically motivated to read or to program as a hobby or at school; but not enough to want to study it for 40 hours a week instead of 10 or 20.

Explicit vs. Implicit knowledge and skill

In most subjects, but definitely in programming and professions, there is not only explicit knowledge that is learned and is usually present in course descriptions;
but also implicit knowledge.

It matters. It is important. And designing both courses and instruction would be more rational if this were made more explicit at least to the teachers. It is a great topic for CER; and has some history already in things like "patterns" which tend to be acquired as implicit knowledge.

There can also be a huge payoff when a bit of implicit knowledge is successfully converted into explicit.
(A reduction in learning time of about 32,000 times was made for the visual skill of chick-sexing.)

That's all, folks.

I hope I've got you thinking in one way or another.

Over to you....

A place to stop

For the notes, pointers to the literature etc. see the companion web page to this talk:

http://tiny.cc/qqcosz

or:

http://www.psy.gla.ac.uk/~steve/educ/keynote.html