Commentary on Angus's CCSE seminar

This document is available on line at: http://www.psy.gla.ac.uk/~steve/educ/AngusCommentary.pdf

Contents:

BODY of the commentary

Angus McIntosh, a year 5 CS student, gave a presentation on his research project to CCSE on 4 May 2020 on "Merging code comprehension and writing tasks".

This document is a commentary on it: not a review nor a summary of his ideas, but a companion piece about some connections I see between his work and wider education concepts; and which might be of some interest to at least some of you (and it even has references at the end if you want to look further).
I explore three conceptual connections:

- "ROG" (the rules of the game), about how students mis/understand what is really required by a set assessment task, or indeed a whole course.
- "PWS" (progressive withdrawal of scaffolding)
- "PI" (Mazur's "peer instruction" learning activity, using clickers, brain-teaser MCQs, and discussion between students about the right answer to each question)

Links to JOD: what is the learning aim of an introductory programming course?

Naturally enough, Angus's talk immediately made me think again about the question of what the learning aims of an introductory programming course ought to be, but this has been discussed in CCSE, will continue to be discussed, and there's no need for my relatively amateur opinions to be exposed here.

However one clear, extreme, and logical position could be that learning to understand code in your head is a priority aim; that courses should proceed by introducing constructs one at a time, and understanding them exactly is the aim.

[And that you wouldn't choose Python as the language, but a simpler and clearer one selected to maximise learning, not pander to all the other learning objectives that various parties are attached to. I.e. this position is of the type that if you want to get there (i.e. understanding code exactly) "if it was meself that was going to Letterfrack, faith, I wouldn't start from here"; or closer to home, from a forum on programming:
"We are currently using a mix of UML, Shlaer-Mellor, Java, MFC, 20 analysts and 80 new programmers from 3 contract houses. How would we use all this to deliver our large project in 14 months?"
Answer: I Wouldn't Start From Here.]

Angus's own starting point, however, (and also Fionnuala's I believe) is that considerable numbers of first year students here do not accept, or even recognise, understanding code as the learning aim. How are we to comprehend this non-acceptance (and then find a way round or through or under it)?

ROG: the "rules of the game" phenomenon

There are other cases of it in fact than the one in CS1; and the first literature I came across which discussed it explicitly, called it "The rules of the game" issue (e.g. Bloxham, 2004; Read, Francis & Robson, 2001). I don't think these papers contain enough general insight to be worth your tracking them down, but in my view it IS important to realise that it is a general phenomenon that can and does crop up in many places in education. And sometimes even with the very best students. Essentially, there are many cases where students think an activity, especially a summative assessment task, has hidden rules of the game which for some reason you won't tell them, that you can't mean what you say because it doesn't make sense. In fact the ROG issue relates to Angus's talk in three separate ways: firstly when students don't believe or otherwise reject what staff say the purpose of a test, assessment, or piece of learning is; secondly, I think we the audence in Angus's talk experienced it when he asked us to find the answer to some calculations and most of us utterly refused to do it; thirdly, it relates in a hidden way to the effectiveness of Mazur's PI. I begin with my take on what the nature of the ROG issue is.

a) Although learners are often amazingly accepting of arbitrary academic tasks, sometimes they are not. Why should this non-acceptance happen? There are a series of reasons, any of which can cause this.

   i. The first response by academics (don't get caught saying this out loud) is that what is required is self-evident and students should already know it, because the typical HE teacher doesn't remember learning it so to them it feels self-evident.
   The obvious solution to this is to be explicit. We probably *should* be as explicit as possible about what is required; publish the explicit statements, and when students still express confusion lead them back to the statements and be as consistent as possible. This has often not been done, but GU at least has gotten progressively better at it in recent years.

   ii. Yet in fact, and perhaps to you counter-intuitively, being explicit does not necessarily work (or even, is not necessarily possible). Our culture in general, and academics even more so, and those who've learned some computer programming perhaps most of all, believe this profoundly and unthinkingly. But it isn't so. Consider basic programming. Often someone has thought deeply about what constructs to have in the programming language; then they design a syntax to express them as clearly as possible; then they build a compiler to operationalise it; then you write a program in that language: and then it really does behave as designed, and any errors you make in understanding or your errors, not the language's. But note that this is a one-way causal chain; and it all depends on a human hand-building each of the links. As has gradually emerged over the last 100 years, this is not how human language works; and nor could it. Our students are human, and they have learned human communication and language skills from an early age. When they try to make sense of what you say, they are NOT a compiler, they are applying their skills in how language and meaning work.

   The symptoms of the failure of the "obvious" model of understanding through explicit statements are numerous when you look for them. The simplest is to think of an Engish dictionary for those whose first language is English. The dictionary explains one English word in terms of other English words you have to know already. That works well for new, unusual words; but at bottom, dictionaries are circular in their definitions. You cannot ground out meanings in it, because there is no base vocabulary built into humans from the start.

   The next, and related, argument is Quine's (1969). As he says, it is easy to imagine how you might learn a second language: you find someone who speaks both languages, and they explain the new one to you in terms of the old one which you already speak. But how could anyone, a young child for instance, possibly learn their first language? A completely different explanation is needed. (This point is also expressed in Draper (1994), where it is used to argue that/why constructivism in the educational sense must be right.)

It turns out that whereas the programming model assumes that communication and understanding are founded and built on top of language, in real life the reverse is true and human language is built on top of pre-existing communication. For instance Beebe (2005) (which is just one paper from a whole field of research) shows how infants begin communicating (typically with their mother) within about 20 min.s of birth -- a very long time before they know any words let alone syntax. This involves turn-taking and other sophisticated features, including how these two beings manage to establish both that the other understands them, and vice versa. So communication is not founded on language; nor (in other jargon) is it built on top of a pre-existing common ground, at least not in any simple way.

iii. Given that, then the problems get worse for clinging to any faith in explicit statements being all that's needed. From a pretty young age, children learn to recognise and understand lies. A huge amount of humour and jokes are based on lies of various kinds. Many parents are notorious for systematically lying to their children in various ways (Father Christmas, the Tooth Fairy, "It will be all right" as a standard soothing response regardless of reality, and so on). Some games are based on deception: whether poker for adults or hide and seek for young children. Children understand the difference between fiction and fact while quite young, and plenty of language use is communicating fiction without deception. (So one look at ROG, is that sometimes students think you are offering an amusing fiction, amusing to you anyway.) So having even a primary school level of competence in language comes with dealing with lies, insincerity, and lots of fiction. Furthermore, children are exposed as much as adults to a blizzard of advertising; and later to politicians, etc. Just because something is said, doesn't mean it has to be true; nor even that it is likely to be true. In fact much of what teachers say to learners in school is insincere -- they ask students questions to which the teacher already knows the answer -- the whole speech act is "fake".

The main issue for ROG in education is probably not malicious deception, but the prevalence of high levels of insincerity of various kinds; or simply not telling the whole truth and leaving out something of real importance. Going back to assessment tasks and ROG, students quite often think that you say you want X, but really there is a hidden goal Y that you really want (the real rules of the game), and marks are given in relation to Y. In my case, in an honours option examined by one-hour unseen essay exams, I didn't want them to conclude whatever my own opinion on the question was, but to give the best reasoned and critical argument they could make to some conclusion they chose. The whole department's standardised marking criteria too were explicit about "critical reasoning" being a big criterion. But it wasn't until I did a long revision session, where I put up a past question, gave them 5 mins to produce an essay plan, and then got volunteers to describe their plan to the class; and repeated many cycles of this, that they really believed I had meant what I'd said: because they had heard my equally enthusiastic comments on completely different answers and seen examples. (And it was only at the end of the session, that the student who was probably best in the class, said it hadn't been until that session that she had really believed me. In other words, I had been completely blind to their being a ROG problem for my course.) Being explicit is not necessarily enough, because any student has many (reasonable, and often justified) ways for interpreting what was really meant.

iv. Another source of "context" that can lead students to disbelieve what you seem to say, and so to understand it differently, is when the school version of the subject they are studying in HE is significantly different in its "take" on the subject. So here there is apparently a narrow and specialised context (the discipline), which they correctly try to apply, but what you say doesn't seem to fit. There is a lot of variation amongst subject disciplines in whether this is a problem: computing science often has this problem, physics not, but Maths apparently does since Stanford has put on a whole first year module on how university maths is quite different from school maths.

v. Comparing HE with hobby versions of a discipline can be another source of communication breakdowns because hobbies are frequently strongly grounded in a community (social context with shared meanings) and in an activity (creating your own programs) which gives not just a strong meaning, but also personal verification of those meanings.

   This aspect is probably best seen as a special case of what Dewey (1910) calls "realism" or "the realistic principle" (and more recent education literature tends to call "authentic tasks"). The essential thing is that people in general, and students from an early age onwards, not always but frequently judge the value of a piece of learning in terms of how it relates to what the external adult world values. It can be seen in a pure form, to take an old-fashioned example, when children play (or used to play) doctors and nurses. Typically they would have almost no idea about what that really involves, but focus on wearing a white coat, and using an imaginary stethoscope. Children are interested in the real world, even when they have little idea about the meaning of what they see; and still more interested when they do.

   The importance of Dewey's realistic principle doesn't only affect the ROG issue of student beliefs about the point of an exercise or assessment or course, but also students' understanding of the utility of what they are learning i.e. future real-world connections to the content. The Graduate Apprentices team is concerned with this (perhaps we will hear something about this in the CCSE seminar on 18 May); and Waqar's recently accepted paper (Nabi et al., 2020) is about communicating this to the apprenticeship students.

b) Summary of why Dunning's work is NOT relevant here, to Angus's work
Matt Barr put a ref. to Dunning (2011) on to Zoom's "chat" during Angus's talk. I had a long think about this, and read all 50 pages of it. But actually, I concluded that despite appearances, it isn't worth your reading it for the current purpose here (of why some students of programming reject the teacher's view of it being about code comprehension).

It is a nice piece of work as psychology, and over some years they have refined their experiments to a relatively unusual level of quality, and worked out more of what lies behind the effect. The Dunning-Kruger effect is part of a slightly wider field in all of which the main effect is that under a fairly wide range of conditions people over-estimate their knowledge, leading to such amusing results as that a large majority of men believe they are above-average drivers (lovers/ programmers, ...); which of course you know cannot be true without even bothering to find out which individuals are mistaken. And the worse someone is in reality, the more their estimate of themselves tends to be inflated.

However Dunning (and Kruger) are psychologists, not educationalists; and their work does not connect usefully to education. Their experiments are of the form: set a (usually academic) assessment; do not tell the participants their scores let alone which questions they got wrong; ask them to estimate their own score both as an absolute scale (e.g. 15 out of 20) AND as relative to their peers (others in their class, taking the same test). This doesn't tell us whether each learner will go on to remedy their imperfect learning, given the chance.

Dunning frames his chapter in terms of the four combinations of "known unknowns" etc.; and particularly in terms of "unknown unknowns", arguing that people are vulnerable to making huge mistakes in life through "meta-ignorance" i.e. not even knowing what they don't know. This framework is interesting to think about for a few minutes (though Dunning doesn't to my mind discuss it well). But this does not connect with his experiments. In the experiments almost everyone over-estimates their score and rank, but no-one thought they were perfect. So they all did know something important about what they didn't know, even without any feedback. However the educational question relevant to education, and to Angus's work, is: *would* the learners remedy this? And that depends on things Dunning doesn't even mention, let alone research e.g. is a 50% mark good enough? Is 100% attainable, or (as in many Humanities subjects at least until very recently) is 72% the most any student can get by convention? Have I missed the chance to learn, or can I go and remedy it? And also: was the test measuring anything I would value knowing at all?, which is the issue Angus's work relates to.

It IS an interesting educational topic to study how learners in fact discover gaps in their knowledge (Mazur's PI is a good example); but there are also cases where learners don't know that they DO in fact know something, as well as the reverse case that Dunning glooms about (but doesn't really study) when they don't know they are missing something. My own way of framing this area,in contrast to Dunning's, is sketched in relation to the psychology of meta-memory (Draper, 2015). We will say some more about this again below. But in my view, Dunning doesn't help us here.

c) The ROG issue relates to Angus's talk and work in not one but three ways:

1. The argument above that there is no complete solution to it, and we must always be prepared for it to occur. (Also, I believe there is no research on how common it is in practice in HE.)

2. During the Zoom talk itself I believe it occurred for us, the audience. Angus asked us to get the answer to some large arithmetic problem, and no-one except Elizabeth did this. If others felt the same as I did, we froze because we were uncertain what Angus "really" meant by the request. Would it be cheating to use a calculator (this is in the context of the talk being about the problem of CS students cheating at getting the code)? The sqrt() clue seemed too obvious: was this a trap, like the questions in the TV show "QI"? Even if I did, what would I learn from the attempt? I didn't feel I knew what was required. I think this is a classic example of ROG in action: not believing that I'd been told what was allowed and asked for.

3. Below in the third and last main section I argue that Mazur achieved a large systematic change in students' understanding of the ROG eventually, although that is not how he talks about it.

d) What to do, then, given the ROG problem in general, and in the particular and substantial problem it poses here and now in CS1?

PWS: The progressive withdrawing of scaffolding

Angus seems to understand the progressive withdrawal of scaffolding. Most people grasp the general idea of educational scaffolding very quickly, but few seem to realise that scaffolding for learning (as opposed to propping up a building during construction) requires it to be progressively and almost continuously withdrawn. Angus alludes to this with his comment on ski poles being useful for beginners and experts but NOT for learners in the middle while developing the skill, when they are learning, and learning deeply. Here, "fail early, fail often" is more productive so padding and high performance equipment, may be bad. The same is said to be true for ice skates: very sharp blades are best at the start and when expert, but not during skill development.

If you want to get a job done fast, then keep scaffolding permanently in place (e.g. in a modern programming environment) as an accelerator for experts. If you want to get learning done fast, then you want heavy scaffolding at first, then progressive withdrawal of it so that learners can and do learn to do larger and larger proportions of the overall task. A further point is that at first, the human expert or other scaffolding also holds the whole process together, so the last stages of withdrawal move from learning the part-skills to learning how to coordinate and combine the doing of the separate parts. (A further point on how PWS can interact with learner proactiveness is in Draper (2007).)

Angus showed this appreciation in the principles he articulated to guide his development of his learning design e.g. "only do partial traces" to reduce unproductive tedium; but contrariwise, definitely ask the student questions when they recently answered wrongly on the same topic.

What I call PWS is also related to the work of Dave Wood. (Wood, and his PhD supervisor Bruner coined the term "scaffolding" as expressing an aspect of Vygotsky's theory of human development, and how people learn from experts.) Wood's work (Wood et al.; 1975, 1976, 1978) tested "contingent tutoring" and his hypothesis that this was the optimal possible method of tutoring, with the experiments supporting his theory.

The tactic or procedure for contingent tutoring is that each time the learner gets some step wrong, the tutor reduces the level of their comment to a lower, more concrete level; every time they get something right, the tutor raises their next comment to a higher, more abstract and general level. If I were helping a student debug their code I might scan it then point to the line with a missing semicolon and say nothing and see if with their attention focussed they can spot the problem themselves, but if that doesn't work I might say "semicolon?", or if that doesn't work explain how semicolons work in that language. Or I might say something like "what will happen when the input value is negative?" and see if they can work out the implications, but if not then say "the test needs changing to deal with negative values" .. and so on downwards or upwards. The procedure is also highly relevant to worksheets for science lab classes. A widespread problem with STEM subject labs has been, at least in the past, giving a very detailed worksheet / procedure, which results in "success" in getting to the end of the procedure and the right result, but very little learning because low level actions/instructions do not require or allow you to think about why each step matters. Conversely in psychology, offering help with stats to project students is done on condition that the student first fills in a pro forma describing their experimental design and the stats they propose to use. Of course hardly any student can do this completely, or they wouldn't need help. But doing it is very useful in both getting the student out of a helpless attitude, getting them to come to a consultation with their problem warmed up and in the front of their mind; and allowing the tutor with one short look at the pro forma to make a good estimate at where in the hierarchy of conceptual understanding to begin the conversation (exactly what contingent tutoring is continually trying to do).

Contingent tutoring is a specific recipe for dynamically adjusted scaffolding. Lowering the level of help from the tutor towards being more concrete and specific reduces the amount of immediate learning, but keeps the process going forward so that the outcome always succeeds in concrete terms. Obviously, then, doing many examples of the activity will be necessary for full and complete learning, just as you wouldn't expect a student to learn programming by writing one program.

An interesting extra observation from this work was that the method seems to be unnatural to humans: not only did Wood never observe parents using it with their children, but even the assistants trained and paid to apply it in the lab were poor at following it. On the one hand perhaps this suggests that humans are naturally poor teachers (surprising, given what an enormous amount, compared to other species, humans explicitly teach their children, as opposed to providing a playground with opportunities for offspring to teach themselves). But on the other hand, perhaps it suggests that even if contingent tutoring is indeed the most efficient way of teaching and learning, it is not too sensitive to departures from it i.e. unlike rocket science, a small error is not disastrous but just a bit less effective. This is obviously good news for designing and then iteratively improving learning designs such as Angus's. This optimistic interpretation is broadly consistent with the results reported in Crouch & Mazur (2001) which shows Mazur increasing the gains from his PI method over 10 years by a large amount, presumably because he got better at it; and also in Smith et al. (2009) which recorded individual student's changes in response to each question, showing that many students got worse on some questions, yet the overall pattern was strong improvement but with "noise" in both directions.

Mazur's unconscious strategy

I now want to lead up to a possible hidden similarity between Angus's strategy for solving the problem in introductory programming courses, and Mazur's PI ("Peer Instruction") method for improving learning both qualitatively and quantitatively in first year huge STEM lectures (Mazur, 1997; Crouch & Mazur, 2001; Smith et al., 2009). As you probably know, the method is to spend large amounts of "lecture" time in putting up brain-teaser MCQ questions related to the material; getting students to vote individually (e.g. with clickers); not telling them the right answer; getting them to discuss the question and what the right answer should be with their neighbours; then re-voting.

What is in fact unsaid, implicit, and deeply cunning is that most of the students believe that in these STEM subjects there is a right answer (black and white, no debate), as do most staff for these questions. They implicitly believe that they aren't there to understand the material but to learn/know the right answers.

But in discussing the question and answers with their neighbours, especially when the neighbour believes in a different answer from themselves, it is socially utterly natural to give reasons and argue about them. Learning reasons is a much better learning objective for the course, and for understanding physics (or genetics or programming) than learning facts. What is cunning is not to argue with the students about this, or waste time being explicit about this, but to manoeuvre them into a situation where they deal with reasons, even though the reasons are not mentioned in the question. So successful is this that a) Smith et al. showed that they could immediately transfer what they had learned to "isomorphic" questions i.e. they didn't just learn the right answer but the principles that let them transfer the knowledge to a different question; b) Jim Boyle (first UK person to introduce clickers and PI in his classes) said that because PI takes a lot of time he couldn't cover the whole syllabus, but after two semesters of it, they could teach themselves the rest out of the book before the exams to the same level. In other words, they gradually changed their understanding of what "learning" was for the subject in a highly desirable direction. Which may be one reason that PI is used much more often in first year courses than later -- not because it isn't good in other contexts, but because it isn't so essential in its tacit meta-level role of changing students' understanding of what this learning is really about.

Angus's proposed solution is shaping up to be like that, at least judging by some of the interview data he collected. By being nudged into seeing tracing of their code execution, and being asked questions about what they predict happens, they not only see things that surprise them but in a context which makes them feel they should know about this. Part of this is seeing that their self-estimate that they do understand roughly what it does, is far off the mark *(Actually, this inflated self-estimate of a learning aim they don't care about IS what Dunning studied and showed.)* Part is that instead of feeling that the exercise of understanding code is "fake", they tend to feel that it is their code and they should be able to understand it. (And also, that understanding it isn't too hard, given that the tracing mechanisms allow lots of inspection of values; just as discussing privately with peers what the right answer to an MCQ is, isn't hard.)

References

[If a reference seems to be missing, or incomplete, it is also worth looking in Reading List of educational references which is my compilation of references which I've found myself giving to students, or colleagues, over some years.]

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