Innovative Pedagogical Practices in the Craft of Computing

James H. Davenport, Tom Crick, Alan Hayes, Rachid Hourizi

University of Bath (initials.surname@bath.ac.uk)
Crick is at Cardiff Metropolitan University (tcrick@cardiffmet.ac.uk)

2 April 2016

Theses

- Programming as such is fundamentally a craft: it cannot be taught in lectures, but rather learned by doing,

  (but a summative piece of coursework is not enough: one needs doing throughout the course)
- and by having that doing coached, supported and (constructively) criticised,
- hence the tutor, delivering, either interactively or deferred, feedback, is key to the experience
- "large" $\equiv$ the lecturer is not the tutor, at which point tutor management becomes important

There's also an odd paradox: we teach children to read before we teach them to write, but we (generally) teach students to write programs before (if ever) we teach them to read programs!

JHD's setting

Programming and Discrete Mathematics [1]

Situation All year first year module: 20% of First Year Mathematics programmes

Content 50% programming, 50% discrete mathematics (DM)

330 students in 2015–6

Per week 1 programming lecture, 1 DM lecture, one example class, to all 330

And one DM tutorial (approx 20 students), one programming lab (on Friday, so after cohort classes)

initially no separate DM tutorials, then additional DM tutors in the labs, but these didn't work

It took a two-year fight to get the labs all scheduled after the whole-cohort events, but it significantly simplified the teaching

Lab physically holds 75, seated at 5 tables with 15 computers each (also master console, beamer, whiteboards)

Conceptual model for programming

(It's a good question how much further this extends: quite possibly other practical aspects of computing, possibly other subjects like Chemistry)

Students are the equivalent of medieval apprentices: there to learn a craft

Lecturer is the master who teaches, and ultimately assesses, that craft

Tutors are the journeymen [literally *journée-men*, in that they are paid by the day!] who assist the master and show the apprentices directly

Note That I do not use the word "demonstrator" (despite university pay scales!) — it gives the wrong impression

All playing their part

(a fair amount of orchestration is required; it doesn't always go according to plan, and each year is slightly different)

Lecturer	demonstrates (preferably live) the writing, testing and debugging of a suitable program
Program	and ideally the diary of the session, and the video of the session, are provided to the students afterwards
Lab exercise	is then a development of that program
Tutors	assist with the exercise, and give feedback

How does one make the students do the exercises?

A good question. Two models at Bath.

Bath CS (120 students) The exercises are marked (increasingly automatically) and worth 17% of the total

Bath Maths (330 students) Marked manually on the spot, or automatically before the lab, as pass/fail.

Tutors are told that a student who does the work, or asks sensibly for help and acts on it, should get the pass.

But failure to get 80% of the ticks results in pro rata deduction from the summative coursework.

in 7 years JHD has never had an appeal against the justice of a pass/fail (problems with the recording, certainly, as tutors are only human)

80% rule explicitly deals with minor medical circumstances, family absences etc.: late starters/transfers in are the majority of "special cases"

Example: Week 1

Lecturer	defines "n factorial" formally, then writes the MatLab, and tests it
Then	Notes that `factorial(-1)` loops, so fixes this, and adds an error message
Then	Notes that `factorial(1.5)` gives this message inappropriately, fixes this, and adds another error message
Students	Are then asked to write Fibonacci as that week's exercise
Tutors	are told that `Fibonacci(19)==6765` is an adequate check for the tick, **but** they should push students towards error messages, comments etc.

Marking

Weekly exercises 100% correctness: tutors give informal feedback on style etc.

- Initially, a simple question e.g. `Fibonacci(19)`
- Later on students download a test harness, and Moodle gives a random (3 out of 13) quiz on the answers: students get a bonus pass for this

Summ/Formative Coursework 80% automatic correctness, 20% style assigned by tutors

10+ tutors implies a need for moderation

Automatic marking has a tendancy to over-penalise: "I make one mistake and therefore get many penalties".

Students/Tutors can request re-consideration (the marking scripts also evolve)

One collective complaint about justness in six years, and correctly so (new coursework)

Management of Tutors

and the tutor/student relationship

N.B. In a week, JHD delivers $1\frac{1}{2}$ hours, and the tutors 25

Assign students and tutors to tables in the lab (1:14 not 5:70)

Brief tutors well at the start of term, and set up a tutors mailing list, which they can use

JHD attends all of the first week labs (and ideally all the second week).

Mix Experienced/new tutors in the same lab

Appoint senior tutor(s) (and have a staffing strategy)

- For junior tutors to ask questions
- to report problems back to the lecturer (avoid "everyone's problem is no-one's problem") NEW! this has just happened
- To bounce ideas/ new briefing materials off
- To write papers with: [1] has three tutor authors

Conclusions

1. Automation pays as student numbers grow: JHD started at 220 students in 2009, and now has 330, **but** it's not a free ride, and you have to accept problems
2. As CS has grown (from 60 to 120) RH has adopted more JHD techniques: imitation is the sincerest form of flattery!

* Automated marking adopted, and now accounts for 60% of marks of weekly exercises

3. Both technological and managerial/pedagogical

* More writing live of programs in lectures

* Senior Tutor rôle started in 2015 in CS

JHD? Room for more "learning analytics" — currently limited to warning students who don't complete weekly work

References

J.H. Davenport, D. Wilson, I. Graham, G. Sankaran, A. Spence, J. Blake, and S. Kynaston.
(4 professors and 3 tutors)
Interdisciplinary Teaching of Computing to Mathematics Students: Programming and Discrete Mathematics.
http://opus.bath.ac.uk/37841/.

To appear in MSOR Connections, 2014.
http://journals.heacademy.ac.uk/doi/abs/10.11120/msor.2014.00021.