Programming 3 - Reviewing ourselves considering BRACElet

Christo Potgieter
WINTEC
Tristram Street
Hamilton, New Zealand
christo.potgieter@wintec.ac.nz

Lesley Gousmett
WINTEC
Tristram Street
Hamilton, New Zealand
lesley.gousmett@wintec.ac.nz

This quality assured paper appeared at the 2nd annual conference of Computing and Information Technology Research and Education New Zealand (CITRENZ2011) incorporating the *24th Annual Conference of the National Advisory Committee on Computing Qualifications*, Rotorua, New Zealand, July 6-8. Samuel Mann and Michael Verhaart (Eds).

Abstract

This poster reviews our own teaching of introductory programming. We observed increases to pass rates, listing possible causes (see poster "Programming 1...") and then explored "problem-solving thinking" into BRACElet and psychology of programming (see poster "Programming 2…."). Now we review the teaching using the Body of Knowledge.

Keywords

Novice programmers, programming curricula, comprehension, schema, patterns, problem solving

Reference Body of Knowledge (BoK)

Principles from BRACElet:

- Understanding SOLO hierarchy/ies in program code
- Assessing student "thinking" from their articulation
- Doing exercises, developing skills re problem solving
- Keeping it simple rather than comprehensive-complex
- Checks the complications re ethics for publication

Concepts regarding "Thinking" and "Problem Solving"

- "The psychology of computer programming", with cognitive psychology being prominent.
- Also: Problem-Solving Heuristics, Critical Reasoning, Philosophy Logic, Cognition & Learning, Educational Psychology and Patterns.

Tutor practices regarding above:

- Iterative development (full programming experience)
- Use logic puzzles at start of class to help focus
- Use Moodle with quizzes – like a game for them

- Repetition - several exercises, several times

Exploring teaching some more: Case

The following practice of stages of development of students contains most of the other detail mentioned separately:

> *My teaching style goes across 4 stages:*
>
> *a) 1st Stage: **Direction** - Here I basically control the flow of the class and run a very structured class. It is 80% hands-on with 20% covering the theory behind what is being taught. Use a mixture of powerpoints, live demonstrations i.e. I type in the code directly into the code window and the whiteboard.*
>
> *b) 2nd Stage: **Coach** - In this stage, the students are given scenarios and using the knowledge they have accumulated at this stage, using pen and paper, they work out "what" the programme will do and once they have a thorough understanding of what the programme will do, they then figure "out" how it will work and from there they write the programme. Use whiteboard extensively in this stage showing how to build up an application using good old "pen and paper " logic.*
>
> *c) 3rd Stage: **Mentoring** - In this stage, we play a game on the whiteboard. It is an old game that only requires pen and paper and based on the players guessing a digit number which only the main player knows. Players have 10 attempts and the only feedback given is the number of correct numbers in the correct position and number of correct numbers in the wrong position. We play this game 2-3 times on the whiteboard. The students then figure out what the game does and how it works and once they have mastered this, then they write the programme. At this stage of the student's learning, they have learnt the basic fundamentals of programming*
>
> *d) 4th Stage: **Delegation** - In this stage, the students have applied their knowledge by thinking through a problem, working out a solution and implementing it. This stage is also used as preparation for their theory and practical tests. Students are given a series of problems and are expected to come up with a good, working solution. They can brainstorm amongst themselves if they wish and can also work in groups, although a majority of the students find this very challenging and WANT to work on their own.*

Assessment of above Case

Re BRACElet principles:

- Understanding SOLO hierarchy/ies in program code – During the stages, complexity of the exercises increase, albeit not explicitly stated to be along SOLO levels (could be investigated)

- Assessing student "thinking" from their articulation – Using student questions and observation of their work, make informal assessment of their location, but not explicitly stated to be along SOLO levels (could be investigated)

- Doing exercises, developing skills re problem solving – Clearly happening in the above case description

- Keeping it simple rather than comprehensive-complex – Also happening in the above case description

Re "Thinking" and "Problem Solving":

- "The psychology of computer programming", with cognitive psychology being prominent – Did not research student views and not clear from above description how feedback was obtained and used.
- Also: Problem-Solving Heuristics, Critical Reasoning, Philosophy Logic, Cognition & Learning, Educational Psychology and Patterns – Several appears to be relevant and might be justified of further investigation.

Conclusion and going forward

The tutor let students practice at the lower levels of SOLO, typical for entry level programming courses, but not doing it explicitly and we did not gather observations for that. To confirm a causal link with pass rates requires in-depth study of situations that include comparisons to peers. Important to study theory of programming and learning of programming.

Acknowledgement:

Fernanda Potgieter shared theories and ideas regarding "Thinking, Problem Solving, Logic" and other aspects from cognitive psychology.

References

Gousmet, L. (2011). Analysis of pass rates for introductory programming courses 2006-2010. Wintec internal report.

Lister, R. & Edwards, J. (2010). Teaching Novice Computer Programmers: bringing the scholarly approach to Australia. A report on the BRACElet project, *Australian Learning & Teaching Council*.

Posters. (2011). "Programming 1…" and "Programming 2…". *Annual conference of CITRENZ 2011, Rotorua, New Zealand.*