Student Conceptions of Quality

Arnold Pears
Uppsala University
Sweden

This presentation

- builds on research that provides insight into student conceptions of software quality,
- draws conclusions about how we should teach.

Presentation Outline

- The issue at hand
- Relevant research
- Some key educational aspects in
  - Programming
  - Holistic appreciation of software
  - Role of process
- Recommendations
- Practical applications
- Conclusions

UPPSALA UNIVERSITET

Informationsteknologi

Definitions?

- Many definitions of software/program quality exist. Often with a strong technical focus (Denning 1992)
- Computing disciplines do not agree on what it means to learn to program, or program well (Pears et al. 2007)

Epistemological committment

- Students conceptions are powerful, and are often established early

[Entwistle 2007]

Primitive conceptions of software quality, which students form early in their education, have a powerful and lasting influence on their values and professional practice

UPPSALA UNIVERSITET

Informationsteknologi

Code quality

The programming skill of the average student after an introductory course is very low. (McCracken et al. 2001 and Lister et al. 2004)

WHY?

UPPSALA UNIVERSITET

Informationsteknologi

What is programming about?

Eckerdal's investigation of student undertanding of OO concepts demonstrates that students focus on unimportant details. [Eckerdal 2005]

Research of Kaczmarczyk et al. and Sorva implies that poor understanding of the notional machine is a contributing factor. [Kaczmarczyk 2010, Sorva 2008]

Holistic view

Writing “quality” code requires an holistic view.

- Soloway identifies goal/plan analysis as useful
- DeRaadt builds on this and extends it to a systematic learning approach
- Bennedsen and Caspersen note the need for students to observe and understand expert process.
- T.R.G Green discusses the cognitive complexity of IDE's and programming languages and iterative vs explorative programming

UPPSALA UNIVERSITET

Informationsteknologi

Correctness

Kolikant and Mussai found that students see code as individual lines and operations each of which could be correct or partially correct.

Stamouli and Huggard concluded that students are strongly preocupied with lines of code and syntax feedback from the compiler.

Can't see the wood for the trees

"Liam3: Its about getting a couple of books, really... Well it is the language that you are interested in anyway and the syntax of the language is what you learn from the book. Then you have to mess around with the syntax and the features of the language and having fun you learn more, really []. So what you can do depends really on the language and this is, really, what you learn from a book, the syntax of the language, thats all you need anyway."
[Kolikant and Mussai]

Conceptions of correctness

"We found that students conceptualized program correctness as the sum of the correctness of its constituent operations and, therefore, they rarely considered programs as incorrect. Instead, as long as they had any operations written correctly students considered the program partially correct."

[Kolikant and Mussai, p.1]

Conceptions of correctness

"We suggest that this conception is a faulty extension of the concept of a program's grade, which is usually calculated as the sum of points awarded for separate aspects of a program. Thus school [University] (unintentionally) nurtures students' misconception of correctness."

[Kolikant and Mussai, p.1]

UPPSALA UNIVERSITET

Informationsteknologi

The role of process

Hilburn and Townhidnejad propose the use of quality models in teaching and grading of code to make the role of quality process explicit.
[Hilburn and Townhidnejad, 2000]

Patton and McGill advocate longitudinal portfolios of programming work and artifacts in combination with the use of automated software quality metrics as a way to evaluate student progress

[Patton and McGill 2006]

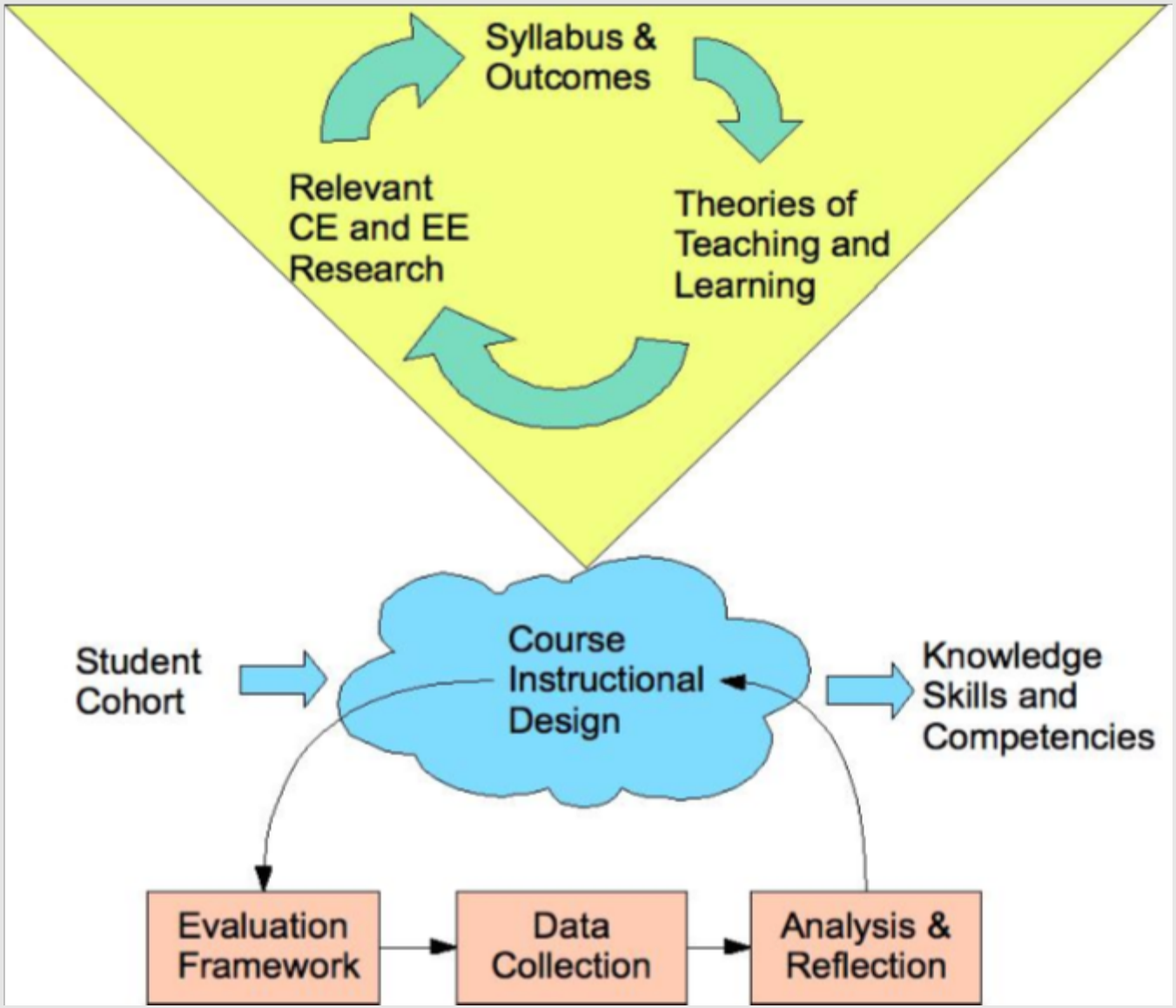
UPPSALA UNIVERSITET

Informationsteknologi

Recommendations

- Adopt an explicit holistic goal/plan based approach to code development even in introductory courses.
- Design formative assessment strategies to align key aspects of quality with high grade achievement, and discourage beliefs about ”partial correctness”.
- Include testing and debugging as explicit parts of the curriculum

From theory to practice



UPPSALA UNIVERSITET

Informationsteknologi

Holistic approach

- Make the social constructivist epistemology of programming teaching explicit in instructional design.
- Establish student motivation by increasing the degree of self determination in programming tasks.
- Enhance student self efficacy using pair programming with strict partner rotation and speak aloud protocols.

Instructional design

Cohort: 75 students
Staff: Lecturer and teaching assistant
12 Weeks of
- 90 minutes of lecture
- 90 minutes of supervised pair programming

10 weeks of
- 90 minutes of supervised project programming
- 5 hours independent and group work
- Individual and group consultation as required

Aligning assessment

Final grade: Fail, Pass, Pass with credit, Pass with distinction

Grade based on:

- 10 satisfactory supervised programming sessions,
- project assessment,
  - 30 minute group oral exam
  - 15 minute individual oral exam

UPPSALA UNIVERSITET

Informationsteknologi

Conclusions

Primitive conceptions of software quality, which students form early in their education, have a powerful and lasting influence on their values and professional practice

- Better approaches to educating software engineering professionals are needed, which align professional and community values with curricula.
- Key aspects of student conceptions identified in this paper need to be addressed through constructive alignment of assessment practices.

Conclusions

- One approach to developing more sophisticated student perceptions of code quality was presented here.
- More research is clearly needed to identify alternative assessment approaches that align grading outcomes with engagement in practices that promote the development of quality software.
- WARNING: Assessment approaches are highly sensitive to national and institutional culture, one size does NOT fit all.