Computing education and Programming education research

Challenges in the field, research practices, methods and outcomes

Lauri Malmi, Department of Computer Science and Engineering

Contents

- Background
- Overview of Computing Education Research
- Challenges
- Approaches and methods
- Research at COMPSER/SVG

Contents

- Background
- Overview of Computing Education Research
- Challenges
- Approaches and methods
- Research at COMPSER/SVG

Background

- The roots of my own research originates in the intensive work for improving education in huge programming, data structures and algorithms courses at TKK, started in early 1990's.
  - How to activate and give feedback for students on courses with 500+ students?
  - Practical tools development project for automatic assessment
- Computer Science Education Research group (COMPSER) was founded in 2000.
  - You can see some results of improving education immediately
  - But can you measure them in some way?
  - Could it be generalizable?
  - And do you understand what is actually happening?

Background cont.

- Software visualization group (SVG) was formed in 2002 to emphasize that technical research is a central part of the work.
  - Especially algorithm & program visualization and automatic assessment
- National Center of Excellence in Education (2001-2003, 2004-2006, 2010-2012 (the whole CSE department))
- Research training in Computing Education Research (CER) and Engineering Education Research (EER) fields

Contents

- Background
- Overview of Computing Education Research
- Challenges
- Approaches and methods
- Research at COMPSER/SVG

Computing Education Research

- Interdisciplinary field of science researching
  - How students learn computing concepts, processes and practices in Computer Science (CS), Computer Engineering, Software Engineering, Information Systems and Information Technology
  - How can the learning process be supported in terms of tasks, learning resources, teaching / learning methods / general or specialized learning environments / structure of curriculum / building motivation ...
  - Most emphasis in the field is related to CS
- CER is not a pure subfield of Education, nor CS but combines theories, methods and technologies from several fields: Education, Social sciences, Computer Science or other fields of Computing

Subfields of CER

Fincher, Petre (2004)

1. Student understanding
2. Animation, visualization and simulation
3. Teaching methods
4. Assessment
5. Educational technology
6. Transferring professional practice into the classroom
7. Incorporating new developments and new technologies
8. Transferring from campus-based teaching to distance education
9. Recruitment and retention
10. Construction of the discipline

Subfields of CER cont.

Pears et al. (2005)

1. Studies in Teaching, Learning, and Assessment
2. Institutions and Educational Settings
3. Problems and solutions
4. CER as a discipline

Programming education research

- Concerns
  - How do students learn programming concepts, processes, practices?
  - How to support learning programming in terms of methods, tasks and tools?
  - How to evaluate student learning?
  - What should students learn?
  - Very much concerns CS1/CS2 level

Research on CS1

- There are hundreds of papers written since 1960's that adress teaching / learning programming
- Many, many different tools have been developed to support the learning and teaching processes
  - Improved compilers/interpreters
  - Debuggers, visual debuggers
  - Program/algorithm visualization tools
  - Automatic assessment tools
  - ...
- And even more different teaching methods, learning tasks, assessment forms have been tried

Subfields of programming education

Valentine (2004)

- Classification of 444 papers concerning CS1/CS2 papers in 1984-2003 (from SIGCSE symposium)
  - Marco Polo
  - Tools
  - Experimental
  - Nifty
  - Philosophy
  - John Henry

But...

- Have we made any significant progress?
- After CS1,
  - Students cannot write code (McCracken et al., 2001)
  - Students cannot read code (Lister et al., 2004)
  - Students cannot design code (Eckerdal et al., 2006)
- And many, many students drop out from CS1
- *So where is the problem?*

What is learning programming? And why it is so difficult?

- How could we improve teaching, if we do not understand well enough, how students learn programming, and which topics they find difficult?
  - Most of CS1 research has considered the topic from teacher's point of view: where is the problem, which new method could I apply to improve my education, what kind of tool could improve my students' learning
  - Much less research has been carried out in investigating students' point of view: How they experience the learning process and what is difficult from their perspective?

Contents

- Background
- Overview of Computing Education Research
- Challenges
- Approaches and methods
- Research at COMPSER/SVG

Learning programming is difficult

- duBoulay (1986) listed 5 domains of learning that are necessary
  - General orientation, what programs are for?
  - Notional machine, model of computer
  - Notation, syntax/semantics of languages
  - Structures, schemas, plans
  - Pragmatics, coding, testing, debugging
- You need to all of these to some extent to be able to do anything
- Everything is abstract and students have little or no previous everyday experience of these topics
- How would learning mathematics compare to this?

... is difficult ...

- More aspects
  - Programming is inherently problem solving but do we teach this aspect explicitly?
    - And how should we teach it?
  - One of the biggest challenges is finding the match between problem concepts and programming language concepts
    - ”...misconceptions on language constructs do not seem to be so widespread or as troublesome as is generally believed. Rather many bugs arise as a results of *plan composition problems* - difficulties in putting the pieces of programs together” (Spohrer & Soloway, 1989)
    - Note. Soloway studied imperative programming

... is difficult ...

- More aspects
  - There is little correspondence between the ability to read programs and the ability to write programs. Both need to be taught. (Winslow, 1996)
  - OO paradigm is conceptually more challenging than imperative paradigm (Sajaniemi, 2007)
  - ”The distributed nature of control flow and function in an OO program may make it more difficult for novices to for a mental representation of the function and control flow...” (Wiedenbeck, 1999)

... is difficult ...

- More aspects
  - Humans experience variation only if they see it
  - In programming education there is often too much variation in examples and assignments: problem domain, language construct, reading/writing/designing
  - Variation should be carefully controlled (Eckerdal, 2009)
  - Students are different. Their motivation, skills, self-confidence, studying skills, external load vary case by case. (Kinnunen, 2009)

Contents

- Background
- Overview of Computing Education Research
- Challenges
- Approaches and methods
- Research at COMPSER/SVG

Research vs. something else

- There are many papers which cannot clearly be characterized as research. Typical examples:
  - Experience reports presenting a novel technique, method, ... and reporting teacher's observations, some final results and possibly feedback from students.
  - Discussion papers raising issues for debate or presenting ideas
  - Tools papers describing a novel tool or feature, with little else (like theory-based arguments, rigorous evaluation)
- Research papers have a more rigorous approach:
  - Research questions or hypothesis set up
  - Clear methodology
  - Connection to some theory or model behind the work
- But: these are not strict categories, and open to debate

Some research approaches

- Qualitative studies
- Quantitative research
- Multi-institutional studies
- Literature surveys
- Tools research
- Community building activities

Qualitative studies

- Example questions
  - How do students understand some concepts or processes?
  - How do teachers understand something?
  - What the reasons behind high dropout rates?
- Data collection: interviews, open questions in questionnaires, essays, discussion logs, ...
- Analysis methods: phenomenography, grounded theory, content analysis, ...
- Typically based on small samples
- Increase our understanding of some topic or issue
- Do not aim at wide generalizations

Quantitative studies

- Typical examples
  - Does method X have a positive effect on student's learning results?
  - What factors have an influence on students' success in CS1?
  - Follow-up studies of applying some method for several years.
  - How do different student groups perform on a course or respond to a novel practice (e.g., male / female)
- Data collection: questionnaires, exam or exercise results, background information
- Analysis methods/practices: pre- and posttests, randomized groups, voluntary groups, statistical tests

Multi-institutional studies

- A group of teachers/researchers from many institutes carry out the same research in their own institute and the results are combined to get a wider perspective of the problem
- Examples:
  - McCracken group study on students writing code (2001)
  - Lister group study on students reading code (2004)
  - Fincher et al. study on students designing code (2005)
- Collect rich data
- Provide strong evidence that the problems are real

Literature surveys

- Collect, analyze and summarize information about some specific topics
- Examples:
  - Robins, Rountree, Rountree (2003): Learning and teaching programming: A review and discussion
  - Kelleher, Pausch (2005): Lowering the barriers to programming
  - Pears, Seidman, et al. (2005): Constructing a core literature for computing education research
  - Pears, Seidman, et al. (2007): A survey of literature on the teaching of introductory programming
  - Ala-Mutka (2005): A survey of automated assessment approaches for programming assignments
- Excellent sources for getting started in some topic

Tools research

- Aims at improving learning or teaching by constructing specialized software.
  - Automatic assessment
  - Algorithm visualization
  - Program visualization
  - Intelligent tutoring systems
  - Programming environments
  - ...
- Often originates from practical needs, not from theoretical bases.
- May or may not include rigorous evaluation
- Success stories:
  - BlueJ, Alice, JFLAP
- Often dissemination problems

Community building activities

- Research activities that support building competences
  - BootStrapping
  - Scaffolding
  - BRACE
  - BRACElet
  - PhiCER workshops
  - DCER workshops
  - ICER Doctoral consortium, Koli Calling DC
  - Research methods courses
  - ITICSE working groups

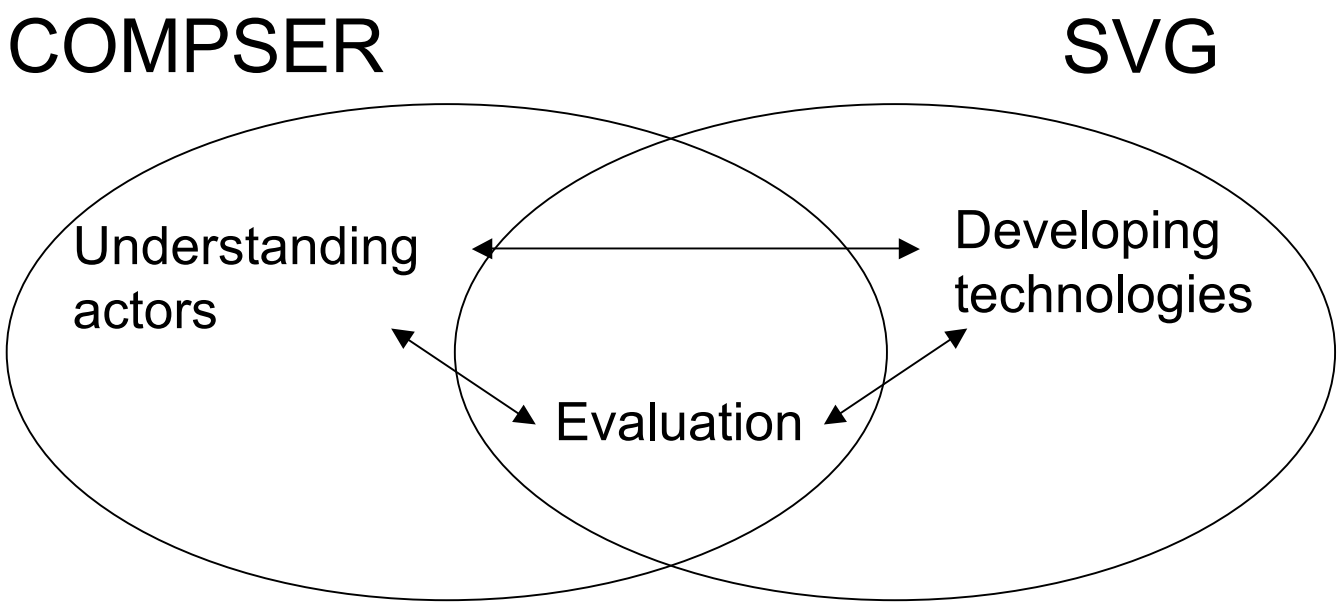
Contents

- Background
- Overview of Computing Education Research
- Challenges
- Approaches and methods
- Research at COMPSER/SVG

COMPSER / SVG

- There are actually two research groups with heavy overlap and close collaboration
- Software Visualization Group concentrates in developing tools and technical methods that are typically applied to support CS education
- Computer Science Research Education Group concentrates on evaluation of the tools / methods in CS education context and exploring data to better understand programming process and learning programming.

COMPSER / SVG groups



Overview

- Currently there are
  - one professor
  - 2 senior doctoral teacher / researchers
  - 5 full time doctoral students doing research
  - 2 full time teachers doing doctoral research
  - 1 full time MSc students doing MSc thesis
  - several part time MSc student
- PhD graduates: (Korhonen 2003, Surakka 2005, Karavirta 2009, Kinnunen 2009)
- Funding from several sources (Academy of Finland, Ministry of Education, doctoral school positions)

Current / recent PhD research topics

- Understanding actors
  - How students understand variables and storing objects into memory? (Sorva)
  - How students understand development of concurrent programs? (Lönnberg)
  - Students' difficulties on learning programming from students', teachers' and organization's point of views. (Kinnunen)

Current / recent research topics

- Tools research
  - Transferring and manipulating algorithm visualization data between AV systems (Karavirta)
  - Automatic recognition of algorithms from source code (Taherkhani)
  - Visual algorithm simulation exercises for spatial data structures (Nikander)
  - Jype - visual debugger of Python programs, with program backtracking and automatic assessment of programming exercises. (Helminen)
  - Visual program simulation (Sorva)
  - Visual debugger for debugging concurrent programs (Lönnberg)
  - Automatic analysis of program testability and tests quality (Ihantola)
  - Automatic recognition of students' misunderstanding of algorithms from algorithm simulation traces. (Seppälä)

MSc thesis projects

- PeerWise
  - Student generated MCQs on CS education
- RubyRic
  - Flexible rubrics assessment tool

Automatic assessment

- Several systems developed by self or in collaboration
  - TRAKLA (1991-2003)
  - TRAKLA2 (2003 -)
  - Ceilidh (TKK version) (1994-2006)
    - User interface testing (2004-2006)
  - Scheme-Robo (1999-2003)
  - Goblin (2004-) from dept. of Automation and system techiques
  - Ox - Java method level correctness evaluation (2004-2005)
  - JYPE - Python programs (2009)
  - Vislaamo - Visual program simulation (2010)

Thank you