Developing Authentic Problem Solving Skills in Introductory Computing Classes

Katrina Falkner and Edward Palmer
SIGCSE’09, March 3-7, 2009
Presented by Xiannong Meng for ENGR 139
Spring 2010

Highlights
• Introductory computer science courses often teach students syntax and semantics along simple problems designed to develop logical, structured thinking.
• Authentic problem-solving skills are rarely developed at these early stages.
• The authors designed and tested with success a three-stage process to teach problem-solving:
  – Observing applications of programming techniques
  – Observing problem solving techniques
  – Applying cooperative problem-solving exercises in classroom

Outline Of Presentation
• Problem introduction
• Related work
• Authentic problem solving
• Analysis of experimental results
• Conclusion

Introduction
• Software development is a creative process, requiring skills in
  – Design
  – Innovative thinking
  – Communication
• Real world software development commonly
  – Start with partial defined problems
  – Communicate with clients and colleagues to determine the full problem specifications
  – Build prototypes to experiment with various solutions
  – Choose and develop the most feasible solution

Classroom Instructions Mismatch with Real World
• Classroom teaching concentrates on syntax and semantics, using problems to demonstrate the features of a language.
• Rarely do introductory computer science texts elaborate examples that illustrate that solving problems most commonly starts with failures, with alternatives.

Contribution of the Authors
• An approach to teaching introductory computer science that uses authentic problem solving to facilitate experiential learning
  – Using cooperative learning techniques to increase student engagement and enhance the authenticity of the learning process.
  – Using cooperative problem solving approaches throughout every aspect of the curriculum, including course content and delivery, and support services.
  – Choosing problems that enable both practice and exploration of the problem space.
Outline Of Presentation

• Problem introduction
• Related work
• Authentic problem solving
• Analysis of experimental results
• Conclusion

Polya’s Seminal Work (1945)

• Understand the problem
• Devise a plan
• Carry out the plan
• Look back

Computer Science Version of It

• Problem Solving Cycle by Barnes, Fincher, and Thompson
  – Understanding
  – Designing
  – Writing
  – Reviewing
• Top-down, step-wise refinement approach by Knuth and Riley

More Recent Approaches

• Students lead the problem-solving process with a partially specified problem (Flowers and Gossett, 2002).
• Discuss problem solving strategies first before teaching any programming elements (Allen and Kolesar 1997).

Other Issues

• First-year students transition: “part of an anonymous mass”, feeling of isolation and loneliness.
• First-year students engagement (studies show a strong correlation between attendance and student performance)
  – Missing classes
  – Lack of confidence in their own abilities plagues many first-year students

Cooperative Learning

• Glasser draws a distinction between “learning by observing” and “learning by doing” and “being involved”. Students retain about
  – 10% of what they read
  – 20% of what they hear
  – 70% of what is discussed with others
  – 95% of what they teach to someone else
• Cooperative learning is a teaching strategy where students work together to improve their understanding of a give topic.
Outline Of Presentation

• Problem introduction
• Related work
• Authentic problem solving
• Analysis of experimental results
• Conclusion

Authentic Problem Solving

• Authentic problem solving contains two components
  – Application of cooperative and active learning
  – Incorporate support systems into learning
• Cooperative and active learning
  – Observing application of programming concepts
  – Observing problem solving
  – Cooperative problem solving
• Support system
  – Computer Science Learning Center

Cooperative and Active Learning (1)

• Observing applications of programming concepts
  – Using code examples
  – Demonstrating running of the example code
  – Developing code live in classroom, explaining various choices along the way
• Observing problem solving
  – Demonstrating live problem solving: start with a high level and incomplete problem specification
  – Elaborating the problem
  – Identifying key elements of the solution
  – Exploring and comparing alternate solutions
  – Concluding by identifying key classes and necessary algorithms

Cooperative and Active Learning (2)

• Cooperative problem solving
• Weekly Worked Examples lectures
  – Students are given a small set of problems to solve
  – The problems are designed to utilize the concepts learned in the week before
  – Groups of 3-4 students first discuss the problems within the group to make sure everyone understand the problem
  – Each group then design a solution and start to implement the design
  – The instructor collects common problems and make a summary at the end of the lecture
• The weekly tutorial sessions work in a similar way.

Result and Analysis

• The new approach has a very positive impact clearly. Some sample course evaluation questions:
  – This course stimulates my enthusiasm for further learning increased from 57% to 74%.
  – I am motivated to learn in this course increased from 60% to 71%.
  – This course helps me develop my thinking skills (e.g., problem solving, analysis) increased from 79% to 88%.
  – I understand the concepts presented in this course increased from 64% to 80%.

Positive Effect on Attendance

Practical, lecture, attrition
Effect on Practical Examination

<table>
<thead>
<tr>
<th></th>
<th>2004</th>
<th>2005</th>
<th>2006</th>
<th>2007</th>
<th>2008</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE1</td>
<td>80</td>
<td>75</td>
<td>91</td>
<td>93</td>
<td>91</td>
</tr>
<tr>
<td>PE2</td>
<td>43</td>
<td>41</td>
<td>51</td>
<td>61</td>
<td>75</td>
</tr>
</tbody>
</table>

Average Exam Scores Over the Years
1. Data collected reflects the 4-year experiment period 2005-2008
2. Weekly Worked Examples lectures started 2006
3. Changed weekly Worked Examples format so students led all discussions

Conclusions

- Authentic problem solving
  - Observing applications of programming concepts
  - Observing authentic problem solving
  - Cooperative problem solving
- The approach increases students’ confidence and improves their performance
- Students-led problem solving sessions do better than passive learning, or instructor-led sessions

References