What do teachers teach in introductory programming?

Carsten Schulte, Jens Bennedsen
Introduction: Some statistics...

• ... on the statistics in the paper

  • 28 items in three dimension on four different groups, plus some other issues ~400 numbers, each with 3 digits = 1200 digits

• ~20 min presentation time

  • 20min*60 sec=1200 -> 1 sec per digit

• Let’s go ;-)
Focus of presentation

- topics -

• Teachers’ opinions about what should be taught in CS1 (as an introductory programming course)

• What to teach besides topics

- Milne and Rowe (2002, p. 55): “inability [of students] to comprehend what is happening to their program in memory”

- Lathinen et al (2005, p. 15): “Students also have problems with understanding that each instruction is executed in the state that has been created by the previous instructions”

- Ragonis and Ben-Ari (2005, p. 214): “[high school] students find it hard to create a general picture of the execution”
Dimensions of learning programming

Notional machine
Notation
Structures
General orientation
Pragmatics
Du Boulay
Notional machine for OO?

1. **Interaction with objects**
   The student can understand simple forms of interactions between a couple of objects, such as method calls and creation of objects. The student is aware that the results of method calls depend on the identity and state of the object(s) involved.

2. **Interaction on object structures**
   The student is able to comprehend interaction on more than a couple of objects, including iteration through object structures and nested method calls. The structure is created and changed explicitly via creations, additions and deletions.

3. **Interaction on dynamic object structures**
   The student knows the dynamic nature of object structures, understands the overall state of the structure and that interaction on the structure or elements from it can lead to side-effects (e.g. implicit changes in the structure).

4. **Interaction on dynamic polymorphic object structures**
   The student takes into account polymorphism in dynamic object structures and is able to understand the effects of inheritance and late binding on dynamic changes in the object structure. The student takes into account side-effects of late binding (different method-implementations, different actual objects referred to by the same variable).
Research Theme

• What do teachers teach in introductory programming?

• Topics
  • Imp. and/or OO

• Didactical perspective
  • Research from the eighties
  • Role of mental model for oo -> object interaction

• Descriptive (not Explanatory)
Population

- **Participants**: teachers at university, college and high school, worldwide
- are attending educational workshops or conferences
- are Experts for Teaching
## Responders

<table>
<thead>
<tr>
<th></th>
<th>University</th>
<th>College</th>
<th>High School</th>
<th>Other</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denmark</td>
<td>5</td>
<td>49</td>
<td>1</td>
<td>3</td>
<td>16.7%</td>
</tr>
<tr>
<td>Germany</td>
<td>16</td>
<td>4</td>
<td>40</td>
<td>6</td>
<td>19.0%</td>
</tr>
<tr>
<td>USA</td>
<td>79</td>
<td>34</td>
<td>4</td>
<td>1</td>
<td>33.9%</td>
</tr>
<tr>
<td>Other</td>
<td>98</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>30.5%</td>
</tr>
<tr>
<td>Overall</td>
<td>56.9%</td>
<td>26.1%</td>
<td>14.1%</td>
<td>2.9%</td>
<td>100% 348</td>
</tr>
</tbody>
</table>
OO included

[Bar chart showing OO concepts covered and OO-first course across University, College, High school, and overall.]

- University: 50% OO-concepts covered, 60% OO-first course
- College: 90% OO-concepts covered, 70% OO-first course
- High school: 80% OO-concepts covered, 60% OO-first course
- Overall: 70% OO-concepts covered, 50% OO-first course
Important teaching topics

What topics to ask for...

- Result: list of 28 topics
- Difficulty
- Relevance
- Level
Difficulty of 28 topics, all teachers

Scale: 1 – 5, left side: most; right side: least

Milne and Rowe: polymorphism, recursion, pointers
CC2001: Recursion, AlgEfficiency, Generics, Adv-Data-Str, Poly are part of CS2
We find that today’s teaching faces the same problems as noticed by [CC2001] and therefore students still associate programming with coding and not with more abstract, design-oriented and intellectual challenging activities.
Level of 28 topics, all teachers

As before /Relevance): Focus on coding
Correlations between Difficulty, Relevance and Level

Object communication

Interpretation: Role of ‘time spend’ to teach a topic? Typical for OO-topics, not as typical for Non-OO (Selection...)

Schulte, Bennedsen ICER06
## Subgroups: “OO Yes” vs “OO No”

<table>
<thead>
<tr>
<th>OO concepts covered</th>
<th>Topics that are seen as statistically significant more <strong>difficult</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Poly&amp;Inheri, Obj&amp;Class, <strong>ptr&amp;refs</strong>, <strong>recursion</strong>, DesignClasses, <strong>parameters</strong> and Encapsulation</td>
</tr>
<tr>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OO concepts covered</th>
<th>Topics that are seen as statistically significant more <strong>relevant</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Sel&amp;Iter</td>
</tr>
<tr>
<td>Yes</td>
<td>StatVsNonStat, AdvDataStr, ObjComm, Poly&amp;Inheri, Generics, Obj&amp;Class, UMLClassDiag, VarTypes, DesignClasses, DesignSglClass, CRC-cards, MethodDesign and Encapsulation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OO concepts covered</th>
<th>Topics that are seen as statistically significant to be taught on a <strong>higher level</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>Yes</td>
<td>AdvDataStr, ObjComm, Poly&amp;Inheri, Obj&amp;Class, UMLClassDiag, DesignClasses, DesignSglClass, CRC-Cards and Encapsulation</td>
</tr>
</tbody>
</table>
Areas of learning programming

- Notion
g machine
- Notation
- Structures
- General orientation
- Pragmatics
Role of Areas

importance
taught
Hierarchy of Object Interaction

All Scales: 1-5
Underst. & Use: Lower=better
Level: Higher=better
Differences between paradigms

<table>
<thead>
<tr>
<th>Teaching OO</th>
<th>Teaching PROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>procedural topics</td>
<td>procedural topics</td>
</tr>
<tr>
<td>OO topics</td>
<td></td>
</tr>
<tr>
<td>Areas</td>
<td>Areas</td>
</tr>
<tr>
<td>Object Interaction</td>
<td>Object Interaction</td>
</tr>
</tbody>
</table>

Overall: very similar
## Topics assigned to Areas (by the authors)

<table>
<thead>
<tr>
<th>Topics</th>
<th>Du Boulay area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sel&amp;Iter, Parameters, Scope, Syntax, <em>UMLClassDiag</em>, Ptr&amp;Refs, Library</td>
<td>Notation</td>
</tr>
<tr>
<td>SimpDataStr, ProbSolStra, AdvDataStr, Recursion, Generics, <em>Poly&amp;Inher</em>, Encapsulation</td>
<td>Structures</td>
</tr>
<tr>
<td><em>ObjComm</em>, MentalModel, <em>StatNonStat</em>, <em>VarTypes</em>, <em>Obj&amp;Class</em></td>
<td>Notional machine</td>
</tr>
<tr>
<td>Ethics, AlgEfficiency</td>
<td>General Orientation</td>
</tr>
</tbody>
</table>
Areas Topics: relevance

- General Orientation
- Notional machine+
- Notation++
- Structures
- Pragmatics

Graph showing the relevance of areas with various notations and orientations.
Areas compared

topics

du Boulay

Schulte, Bennedsen ICER06
Summary / Conclusions

• ‘Classic’ topics /iteration syntax) similar relevance, level and difficulty regardless of “teaching-paradigm”
• Including OO in CS1 seems to be an addition of topics
• OO-topics are seen more difficult by those who do not teach them
  • OO Teachers have a tendency to rate ‘abstract concepts’ as less difficult (table)

• Notional machine
  • Least relevant (areas)
  • More relevant in OO (topics, but…)

• Hidden curriculum (structures vs. notation)
• Focus on coding in OO, too
  (notation vs. notional machine)
Differences between paradigms

Teaching OO

- procedural topics
- OO topics
- Areas
- Notional machine
- Object Interaction

Teaching PROC

- procedural topics
- OO topics
- Areas
- Notional machine
- Object Interaction
Thank you!