CS1 Course Content: General/Breadth First

Wesley Griffin
Knowledge

- Yuen characterizes three levels of knowledge
- Automatic
  - memorization, no depth, no ownership
- Associated
  - explicit connections, difficult to break
- Conceptual
  - integrated
- Problems
  - need to code, no planning, no generalization
Taxonomy

- Design
  - classes, teaching materials, assessments

- Analyze
  - responses

- Several different taxonomies surveyed, Bloom (1956) most popular

- Can divide learning

<table>
<thead>
<tr>
<th>Objective</th>
<th>Develop Artifacts</th>
<th>Critique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Style</td>
<td>Producing</td>
<td>Interpreting</td>
</tr>
</tbody>
</table>
A New Taxonomy

• Bloom (1956): six categories, builds on previous categories

• Knowledge → Comprehension → **Application** → Analysis → Synthesis → Evaluation

• Revised: Remember → Understand → Apply → Analyze → Evaluate → Create

• Divide Revised Bloom by learning style

<table>
<thead>
<tr>
<th>Produce</th>
<th>Create</th>
<th>Apply</th>
<th>none</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interpret</td>
<td>Remember</td>
<td>Understand</td>
<td>Analyze</td>
</tr>
</tbody>
</table>

### Important and Difficult Concepts: Top 11

<table>
<thead>
<tr>
<th>ID</th>
<th>Topic</th>
<th>Importance</th>
<th>Difficulty</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Procedural Programming</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA3</td>
<td>3. Parameter Scope, use in design</td>
<td>9.1 (0.9)</td>
<td>7.5 (1.0)</td>
</tr>
<tr>
<td>PROC</td>
<td>4. Procedure design</td>
<td>9.8 (0.4)</td>
<td>9.1 (0.8)</td>
</tr>
<tr>
<td>SCO</td>
<td>12. Issues of scope, local vs. global</td>
<td>9.4 (0.7)</td>
<td>8.0 (0.0)</td>
</tr>
<tr>
<td></td>
<td><strong>Object Oriented Programming</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INH</td>
<td>15. Inheritance</td>
<td>7.6 (1.7)</td>
<td>9.5 (0.5)</td>
</tr>
<tr>
<td></td>
<td><strong>Algorithm Design</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APR</td>
<td>19. Abstraction/Pattern recognition and use</td>
<td>8.8 (0.4)</td>
<td>9.0 (0.4)</td>
</tr>
<tr>
<td>REC</td>
<td>22. Recursion, tracing and designing</td>
<td>7.8 (2.4)</td>
<td>9.2 (0.9)</td>
</tr>
<tr>
<td>MMR</td>
<td>26. Memory model, references, pointers</td>
<td>7.5 (1.7)</td>
<td>8.9 (0.7)</td>
</tr>
<tr>
<td></td>
<td><strong>Program Design</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DPS1</td>
<td>27. Functional decomposition, modularization</td>
<td>9.3 (0.6)</td>
<td>7.9 (0.8)</td>
</tr>
<tr>
<td>DPS2</td>
<td>28. Conceptualize problems, design solutions</td>
<td>9.5 (0.5)</td>
<td>8.5 (0.5)</td>
</tr>
<tr>
<td>DEH</td>
<td>29. Debugging, Exception handling</td>
<td>9.0 (0.0)</td>
<td>8.6 (0.5)</td>
</tr>
<tr>
<td>DT</td>
<td>32. Designing Tests</td>
<td>9.3 (0.8)</td>
<td>8.4 (0.8)</td>
</tr>
</tbody>
</table>

Pedagogy

- Depth-First (Leutenegger and Edington, Murtagh)
- Software Engineering (Rao and Mitra)
- Computer Science (Solomon)
- Fundamentals (Sanders and Mueller)
- Breadth-First (Dodds et al)
Depth-First (Luetenegger and Edington, Murtagh)

- Games (Luetenegger and Edgington)
- Networking (Murtagh)
- Still “breadth-first” for CS topics
- Attempt to unify material with cohesive theme to provide context
Software Engineering (Rao and Mitra)

- Solving a programming problem from scratch is hard
  - Data representation and algorithm design
  - Translate design into code
- Divide problems into cases
- Provide scaffolding for cases
Computer Science (Solomon)

- Introductory programming no longer about computer science
- Instead it is memorization and engineering concepts
- Problem solving, algorithms, abstraction
- No results
Fundamentals (Sanders and Mueller)

- Traditional CS learning outcomes
- Provides some breadth
Breadth-First (Dodds et al)

- Breadth introduces how CS applies to other disciplines
- Course page link sent to list
- Reviews from paper readings not good
  - Students still missing context
Thoughts

- Context is important
- Assignments/Projects need to be more interesting/engaging
- Labs/Weekly Assignments improved success
- Developing problem solutions from scratch is difficult
- Other disciplines have good base-line for expected learning in first year
- When measured, everybody found their method succeeded
- Dodds et al did not see increase in CS2 enrollment
References

• Rao and Mitra. Early software engineering approach to teaching CS1, CS2, and AI. In SIGCSE, page 143-147, 2008.

From: https://spaces.umbc.edu/display/pycs1/Bibliography