What are empirical studies?

- "[Empirical studies] attempt to learn something useful by comparing theory to reality, and to improve our theories as a result." Perry, Porter & Votta, 2000
- Components of an empirical study
  - Formulate a hypothesis or question to test
  - Observe a situation
  - Abstract your observations into data
  - Analyze the data
  - Draw conclusions with respect to the tested hypothesis
- Some types of empirical studies
  - Controlled experiments
  - Observational studies
  - Case studies
  - Surveys
  - Prototyping studies

Components of an empirical study

- Formulate a hypothesis or question to test
- Observe a situation
- Abstract your observations into data
- Analyze the data
- Draw conclusions with respect to the tested hypothesis

Some types of empirical studies

- Controlled experiments
- Observational studies
- Case studies
- Surveys
- Prototyping studies

Research questions

- We have an intuition that experimental evaluation is rare in computer science research. Is that intuitive impression verifiable?
- How much of CS research includes experimental evaluation?
- How does the proportion of experimental evaluation in CS research compare with that in other scientific and engineering disciplines?

Study Design

- Define classification scheme for CS and SE papers into 5 categories:
  - 0%, 0-10%, 10-20%, 20-50%, >50%
- Design and modeling
  - 5 subclasses, subclassed by percentage of space within article devoted to description of experimental setups, presenting observations, and interpreting results
- Empirical work
- Hypothesis testing
- Other
- Independent classification of articles by 4 authors
  - Each article classified by 1, 2 or 4 authors

Tichy et al., 1995

- Tichy, W. F., Lukowicz, P., Prechelt, L., and Heinz, E. A.

Study Design

- Define classification scheme for CS and SE papers into 5 categories:
  - 0%, 0-10%, 10-20%, 20-50%, >50%
- Design and modeling
  - 5 subclasses, subclassed by percentage of space within article devoted to description of experimental setups, presenting observations, and interpreting results
  - 0%, 0-10%, 10-20%, 20-50%, >50%
- Empirical work
- Hypothesis testing
- Other
- Independent classification of articles by 4 authors
  - Each article classified by 1, 2 or 4 authors
Results

- Controlling for error
  - Inter-rater discrepancies
  - Classification errors & ambiguities
  - Results in major categories
    - Majority of articles are design & modeling work
    - Most CS samples have a lower percentage of empirical work than the non-CS/SE samples
    - Hypothesis testing in 1% of all samples
  - Results within design & modeling
    - CS samples have a much lower percentage of empirical work (43%) than the non-CS/SE samples (14%)
    - Non-CS/SE samples devote more space per article on average to experimental evaluation
    - SE-related samples are worse than CS-related samples.

Conclusions & Suggestions

- Large parts of CS may not meet established standards of natural and engineering sciences
- What we should do about it
  - Set higher standards for acceptable design papers, incl. reasonable evaluation
  - Recognize empirical work as first-class science
  - Establish shared sets of benchmark problems
  - Facilitate discovery of rules for conducting repeatable experiments in CS
  - Support validated results with tenure & funding
  - Computer scientists must lead the way!

Empirical studies of programmers

- Research questions
  - “What are they thinking?”
    - how programmers, software engineers & architects actually use and reason about tools, techniques & methods
  - “Will that really work for people?”
    - validating new tools, techniques & methods through demonstrating use by humans
- Appropriate research methods
  - CS/SE considerations
    - AND
  - Human subjects considerations

Working with human subjects

- Human subjects add complexity
  - Generalizing from sample to population
    - Selecting a sample that will generalize to the population you are trying to understand
  - Controlling for confounds
    - Sample size
    - Randomization
    - Counterbalancing
    - Blinding
  - Effects of extraneous variables
    - e.g. age, domain-specific knowledge, education, experience, gender
Soloway & Ehrlich, 1984

- Research question
  - Do expert programmers have and use specific knowledge that novice programmers do not, in the areas of programming plans, and rules of programming discourse?
- Programming plans
  - Program fragments that represent stereotypic action sequences in programming, e.g. running total loop, item search loop.
- Rules of programming discourse
  - Rules that specify conventions in programming, e.g. the name of a variable should agree with its function. Authors expect these set up expectations in the minds of programmers about what the program should contain, analogous to discourse rules in conversation.

Study Design

<table>
<thead>
<tr>
<th>Study 1 – Fill In The Blank</th>
<th>Study 2 – Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Goal</strong></td>
<td>Compare ability of experts vs. novices to fill in a “missing” line of code in plan-like and un-plan-like programs, to fulfill the intent of the rest of the program</td>
</tr>
<tr>
<td><strong>Number</strong></td>
<td>139</td>
</tr>
<tr>
<td><strong>Sample</strong></td>
<td>Student programmers, 34 novice and 45 advanced</td>
</tr>
<tr>
<td><strong>Experience</strong></td>
<td>Novice &lt; 1 first course in Pascal programming, Advanced ≥ 3 programming courses, + “extensive” Pascal experience</td>
</tr>
<tr>
<td><strong>Independent Variables</strong></td>
<td>Program version (2 versions)</td>
</tr>
<tr>
<td></td>
<td>Program type (4 types)</td>
</tr>
<tr>
<td></td>
<td>Level of expertise (2 levels)</td>
</tr>
<tr>
<td><strong>Dependent Variables</strong></td>
<td>Accuracy of task response</td>
</tr>
</tbody>
</table>

Study Materials

- Short programs (≤ 15 LOC)
- 4 Program types
  - MAX
  - SORT (Study 1 only)
  - AVERAGE
  - IF/WHILE
- 2 versions of program for each type
  - Alpha: plan-like
  - Beta: un-plan-like

Results & Conclusions

<table>
<thead>
<tr>
<th>Study 1 – Fill In The Blank</th>
<th>Study 2 – Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Expert</strong></td>
<td>Experts outperformed novices</td>
</tr>
<tr>
<td><strong>Plan-like vs. un-plan-like versions</strong></td>
<td>Blank line filled correctly with plan-like versions more often than un-plan-like versions. Un-plan-like versions took longer than plan-like versions to answer.</td>
</tr>
<tr>
<td><strong>Interactions</strong></td>
<td>Significant interaction between expertise and version: experts and novices performed more alike with un-plan-like versions than with plan-like versions</td>
</tr>
<tr>
<td><strong>Conclusion</strong></td>
<td>Programming plans and rules of discourse affect experts more than novices. When plans and rules are followed, expert performance is significantly better than novice performance. Otherwise, not.</td>
</tr>
</tbody>
</table>

- Critical line recall in plan-like versions better than in un-plan-like versions
- Significant interaction between expertise and version: experts and novices performed more alike with un-plan-like versions than with plan-like versions
- Difference between recall of plan-like and un-plan-like versions decreased in third trial
- Expert programmers recall programs significantly better which follow programming plans and rules of discourse. The present of extant cognitive classifications is indicated.
Some further reading