Designing User Studies for Programming Tools

Björn Hartmann, UC Berkeley
ExCAPE PI Meeting
June 11, 2013
Purpose of Evaluation

• **Formative:** to inform the design

• **Summative:** to characterize a design, once it is complete
  (e.g., to convince reviewers)
Genres of Evaluation

**Automated** Usability measures computed by software

**Inspection** Based on skills, and experience of evaluators

**Formal** Models and formulas to calculate measures

**Empirical** Usability assessed by testing with real users
Empirical: Quantitative Studies

• **Goal**
  - Use to reliably measure *some* aspect of interface
  - Compare two or more designs on a measurable aspect

• **Approaches**
  - Collect and analyze user events that occur in natural use
  - Controlled experiments

• **Examples of measures**
  - Time to complete a task, Average number of errors on a task, Users’ ratings of an interface
Empirical: Qualitative Data

- Example: observe users, use talk-aloud protocol, surveys

- Qualitative methods help:
  - Understand what is going on
  - Look for problems
  - Roughly evaluate usability of the interface

- Can be very valuable to convey experience!
Practical Challenges

• Order-of-magnitude differences in programming productivity

• Difficult comparisons:
  • Lopsided comparison of research code to tools with decades of development
  • What if your tool enables entirely new tasks?
Practical Challenges

- Most valuable results (e.g., learning, productivity) are hard to operationalize
- Participating in studies not a “community value” for many programmers
Addressing the Challenges

- Triangulation of different methods to get over limitations of any one evaluation:
  - Sharpen the focus of quantification: ask more specific questions
  - Combine with qualitative observations on open-ended tasks
  - Collect self-report data (weak support)
• Great lessons:
  • Instrument your tool!
    (But: have hypotheses ahead of time so you don’t blindly sift afterwards)
  • Importance of task choice & tutorials.
  • Run pilot studies.
• General limitations of single-use lab studies apply.
  Think about longitudinal use (“living with the tool”)
• p<0.2?
Statistics: p<0.05?

Think effect sizes first, significance second!
Picking tasks

• Make sure you’re not just measuring the **learning curve** of your tool (tutorials, warmup tasks, discard early data)

• Some **narrow tasks** that exercise the features of the tool

• Some **open-ended tasks** that don’t prescribe how to use the tool (external validity)
Quicksilver

- Methodology:
  Tutorial; within-subjects with randomization

- Clear win for the study tasks. Users can perform tasks that they could not before.
Quicksilver

• How representative are tasks?
  What are the boundaries of what users can accomplish with the tool?
  What queries are easy/hard to express?
  What kind of errors would users commit?

• Measures: Preference, Efficiency
  Hard to eliminate bias. Better: also ask to compare and contrast for within-subjects.
Initial evaluation done offline - good idea for “fragile” code to get started - but no feedback from users.

Beyond aggregate analysis, look at boundary conditions.

Online tool has to robust enough to run without hand-holding. Benefit: scale!

Classes large enough to allow for clean A/B testing (between-subjects: no learning or fatigue)

Online deployments can scale - but we also know less about users’ experience. What if the operationalization of “learning” doesn’t show effects?

Beyond on-campus classes to MOOCs?
Parting Thoughts

All studies have limitations.

Despite limitations - all of these examples give us much more information than not running any studies!