Programming

MICHAEL BERNSTEIN
CS 376
Final Presentations

- Friday of finals week, 3:30pm
- Here!
- 5 minutes to present
  - Next group plugs in laptop during questions
- Make sure to test your slides in the room before presentation day
- Final paper due...well, let's talk?
A Small Matter of Programming

- Software engineering is a highly complex task, a microcosm of many challenges in HCI
- Making software engineering more accessible could empower millions to customize applications and write programs
Research agenda

- Understand the challenges in programming
- Design more effective software engineering interfaces
- Aid novices in learning to program or writing programs
- Abstract best practices into toolkits
Understanding programmers
Information Needs in Programming
[Ko, DeLine and Venolia, ICSE ’07]

- Observed 17 developers in 90-minute sessions and transcribed all activities
- Thematic coding of information needs
  - Writing code e.g., how do I use this method?
  - Submitting a change e.g., which files are included?
  - Triaging bugs e.g., is the problem worth fixing?
  - Reproducing failure e.g., what are failure conditions?
  - Understanding execution e.g., what caused this behavior?
  - Design e.g., why is the code implemented this way?
  - Awareness e.g., what are my collaborators working on?
- Most common need: collaborator awareness
Obstacles to learning APIs

[Robillard and DeLine, Empir. Software Engineering ’11]

- Survey and in-person interviews, combined reaching 440 professional software engineers
- Biggest challenge: inadequate documentation
- API intent: how it was intended to be used
  - “Nowhere in there does it say, and we intended to be used for a few graphics of small size because the memory footprint is going to be this.”
- Code examples: snippets, tutorials, working apps
- Penetrability: how much detail and implementation to expose?
Web foraging and programming
[Brandt et al., CHI ’09]

- Laboratory study: ask programmers to implement a chat room in PHP
- This paper articulated how programmers make heavy use of the web
  - JIT learning of new skills
  - Clarifying existing skills
  - Reminding themselves of details
- Average participant spent 19% of their programming time on the web
Software engineering interfaces
Goals of software engineering interface research

- Design a better toolbench, produce a better programmer
- This research typically assumes that the programming language is static, but the interface of the IDE can be molded
Example-centric programming

[Brandt et al., CHI '10]

- Close the loop between the development environment and web search
- Autocomplete code via web examples
Asking ‘why’ questions of code

[Ko and Myers CHI '04, ICSE '09]

- Debugging problems often reduce to “why” questions
- Analyze program traces to answer them
Missing user-facing feedback

[Ko and Zhang, CHI ’11]

- Usability heuristic: all user inputs should produce some form of feedback
- Statically analyze code to identify user inputs that produce no feedback

Feedback!

project Calculator

Feedback found 54 places in your code that appear to be missing feedback:

nd() at overlib.js 927 may not produce feedback

script() at Calculator.html 29 may not produce feedback

func() at newcalc.js 919 may not produce feedback

digit(n) at newcalc.js 820 may not produce feedback

When the user performs a

- mouseout (Calculator.html 603),
- mouseout (Calculator.html 947),
- mouseout (Calculator.html 1026),
- mouseout (Calculator.html 600)
Keyword programming

[Little and Miller, UIST ’06, ASE ’09]

- Macro programming is difficult to learn
- Allow keyword search over an API:
  e.g., “click search button” or “left margin 2 inches”
Visual layout of code snippets
[Bragdon et al., CHI ’10]

- Most engineering time is spent navigating across multiple related code snippets
- So, design for many small windows into files
Emergent programming practice

[Fast et al., CHI 2014]

```ruby
1 # Creating a nested Hash
2 my_hash = Hash.new { |h,k|
3     h[k] = {}
4 }
5
6 my_hash[:CHI][:Toronto] = true
7
8 # Naive way:
9 Hash.new({}) # This is a bug!
```

Creating a Nested Hash

- Total count: 66
- Project count: 10

 Creates a Hash with a new empty Hash object as a default key value
Languages that learn from crowds
[Fast and Bernstein, UIST ’16]

- If your functions sent back information to a central community server, could they...
  - Recover from crashes?
  - Auto-optimize?
  - Test themselves?

Count the vowels in a string

```
import re

@meta(parent="5700375c2f6a2f000330436a")
def count_vowels(s):
    return len(re.findall('[aeiou]', s, flags=re.I))
```

Warning: Meta has found a possible alternative that is 1.3 times faster

Example inputs:
- `count_vowels("UIST")` #=> 2
- `count_vowels("CHI")` #=> 1

Known errors:
- `count_vowels(["CHI", "UIST"])` #=> expected string or bytes-like object

You can load this snippet with:

```
count_vowels = meta.load("http://www.meta-lang.org/snippets/5700375c2f6a2f000330436a")
```
Learning programming
Goals of programming education

- Make programming accessible to new populations: children, scripters, interested amateurs
- Tools and innovations depend on the population
Logo: programming for children
[Papert ’93]

- Constructionist learning: learning happens most effectively when people are making tangible objects
- Lego Mindstorms followed this mold and was named after it
Scratch: kids remix and create

[Resnick et al., CACM '09]

- Social: upload and remix others’ programs
- All programming has been done online. This data has led to many papers on understanding notions of authorship and creative remixing.
Online python tutor

[Guo, SIGCSE '13]

- Embeddable Python data structure visualization
- Over 200,000 users and a dozen universities using it

```python
def listSum(numbers):
    if not numbers:
        return 0
    else:
        (f, rest) = numbers
        return f + listSum(rest)

myList = (1, (2, (3, None)))
total = listSum(myList)
```
Watch many learners code and debug in real time

[Guo, UIST '15]
Overcode: clustering solutions

[Glassman et al., TOCHI ’15]
Programming by demonstration
Goals of PBD

• Teach a computer to program simply by demonstrating what should be done

• Challenges
  • There is an infinite, and hugely branching, space of programs that might be inferred
  • Inferred macros can be extremely brittle
Recall: EAGER

[Cypher, CHI ’91]

- Infer a macro by watching the user’s behavior
Simultaneous structured editing

[Miller and Myers, USENIX '01]

- Utilize lightweight structure in text
- Today, versions of this exist in Sublime Text
Toolkits
Threshold/Ceiling Tradeoff

[Myers, Hudson and Pausch, TOCHI 2000]

Difficulty of use

Sophistication of what can be created

C++
Web
Server-side
Client-side
Balsamiq
Research agenda: toolkits

- Crystallize and formalize a perspective on a difficult engineering problem
- If successful, shift the entire programming practice for the area
Sikuli: programming with screenshots
[Yeh, Chang, and Miller, UIST '09]

- Visual template search in desktop scripting
Recall: Chickenfoot
[Bolin et al., UIST 2008]

- Lower the threshold to writing programs
- Allow users with little programming skill to author behaviors
  - e.g., Chickenfoot

```javascript
isbn = find('number just after isbn')
with (fetch('libraries.mit.edu')) {
  pick('Keywords');
  enter(isbn)
  click('Search')
  link=find('link just after Location')
}
// back to Amazon
if (link.hasMatch) {
  insert(before('first rule after "Buying"'),
    link.html)
}
```
Research agenda: HCI and programming

• Understand the challenges in programming
• Design more effective software engineering interfaces
• Aid novices in learning to program or writing programs
• Abstract best practices into toolkits