where *user experience* and *software engineering* meet

Andrew J. Ko
in **method**, I’m an **HCI** researcher

I study

problems

I **evaluate** solutions

I **design** solutions
in method, I’m an HCI researcher

I study problems
I evaluate solutions
I design solutions
in topic, I’m an SE researcher

I want to make it easy to create useful, usable software that empowers and enriches users’ lives
human-computer interaction

how do we get the right design?
(the lives of users)

software engineering

how do get the design right?
(the lives of developers)
user-centered software evolution

how do we get the right design right?
why is software evolution difficult?
a study of information needs at Microsoft

how can tools help software evolution?
debugging with the Whyline

how can users help software evolution?
recent work in leveraging the crowd
talk outline

why is software evolution **difficult**?

*a study of information needs at Microsoft*

how can **tools** help software evolution?

*debugging with the Whyline*

how can **users** help software evolution?

*recent work in leveraging the crowd*
information needs at Microsoft
with the Human Interactions in Programming group at Microsoft Research

observed 25 hours of coding and bug fixing, in the role of “new hires”

357 pages of handwritten notes

4,231 events in an spreadsheet

Monday, November 30, 2009
17 developers hard at work across 25 hours

Visual Studio
Windows Vista
service packs
mobile devices
discussion boards
educational tools
SQL server
MS Office
Encryption
DRM

...
<table>
<thead>
<tr>
<th>8 activities</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>writing code</td>
<td></td>
</tr>
<tr>
<td>submitting code</td>
<td></td>
</tr>
<tr>
<td>triaging bugs</td>
<td></td>
</tr>
<tr>
<td>reproducing a failure</td>
<td></td>
</tr>
<tr>
<td>understanding behavior</td>
<td></td>
</tr>
<tr>
<td>reasoning about design</td>
<td></td>
</tr>
<tr>
<td>maintaining awareness</td>
<td></td>
</tr>
<tr>
<td>non-work activity</td>
<td></td>
</tr>
</tbody>
</table>

Monday, November 30, 2009
8 activities

- writing code
- submitting code
- triaging bugs
- reproducing a failure
- understanding behavior
- reasoning about design
- maintaining awareness
- non-work activity
9 reasons for switching tasks

face to face conversation
phone call
instant message
e-mail alerts
bug report change alerts
task avoidance
getting blocked
meetings
task completion

Monday, November 30, 2009
**9 reasons for switching tasks**

- Face to face conversation
- Phone call
- Instant message
- E-mail alerts
- Bug report
- Change alerts
- Task avoidance
- Getting blocked
- Meetings
- Task completion

*Monday, November 30, 2009*
software development work is highly fragmented

*interrupted* every ~5-10 minutes

*blocked* every ~10 minutes

Monday, November 30, 2009
next, we looked for information that developers sought to get their work done...
next, we looked for information that developers sought to get their work done...
21 information needs observed (by frequency)

what have my coworkers been doing?
what code could have caused this behavior?
have resources I depend on changed?
what code caused this program state?
how do I use this data structure or function?
did I make any mistakes?
what is the program supposed to do?
in what situations does this failure occur?
is this problem worth fixing?
why was this code implemented this way?
what’s statically related to this code?
what are the implications of this change?
what changes are part of this submission?
how difficult will this problem be to fix?
what information was relevant to my task?
how can I coordinate this with the other code?
did I follow my team’s conventions?
### 5 information needs least often satisfied

<table>
<thead>
<tr>
<th>Question</th>
<th>% Unsatisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td>What have my coworkers been doing?</td>
<td></td>
</tr>
<tr>
<td>What code could have caused this behavior?</td>
<td><strong>36%</strong></td>
</tr>
<tr>
<td>Have resources I depend on changed?</td>
<td></td>
</tr>
<tr>
<td>What code caused this program state?</td>
<td><strong>61%</strong></td>
</tr>
<tr>
<td>How do I use this data structure or function?</td>
<td></td>
</tr>
<tr>
<td>Did I make any mistakes?</td>
<td></td>
</tr>
<tr>
<td>What is the program supposed to do?</td>
<td><strong>15%</strong></td>
</tr>
<tr>
<td>In what situations does this failure occur?</td>
<td><strong>41%</strong></td>
</tr>
<tr>
<td>Is this problem worth fixing?</td>
<td></td>
</tr>
<tr>
<td>Why was this code implemented this way?</td>
<td><strong>44%</strong></td>
</tr>
<tr>
<td>What's statically related to this code?</td>
<td></td>
</tr>
<tr>
<td>What are the implications of this change?</td>
<td></td>
</tr>
</tbody>
</table>

Monday, November 30, 2009
were debugging related ...

what have my coworkers been doing?
what code could have caused this behavior?
have resources I depend on changed?
what code caused this program state?
how do I use this data structure or function?
did I make any mistakes?
what is the program supposed to do?
in what situations does this failure occur?

reproducing, diagnosing, and scoping failures were the most time-consuming activities

is this problem worth fixing?
why was this code implemented this way?
what’s statically related to this code?
what are the implications of this change?

Monday, November 30, 2009
what have my coworkers been doing?
what code could have caused this behavior?
have resources I depend on changed?
what code caused this program state?
how do I use this data structure or function?
did I make any mistakes?
what is the program supposed to do?
in what situations does this failure occur?
is this problem worth fixing?
why was this code implemented this way?
what’s statically related to this code?
what are the implications of this change?

2 were design related

knowing what software should do was rarely known
Software development is tacit.

Plans and specifications are *unwritten*.

Developers have to *communicate* to make progress.
software quality depends highly on the quality of human communication and cognition
these human activities are faulty and unreliable
talk outline

why is software evolution **difficult**?

*a study of information needs at Microsoft*

how can **tools** help software evolution?

*debugging with the Whyline*

how can **users** help software evolution?

*recent work in leveraging the crowd*
talk outline

why is software evolution difficult?
*a study of information needs at Microsoft*

how can **tools** help software evolution?
*debugging with the Whyline*

how can **users** help software evolution?
*recent work in leveraging the crowd*
why is **debugging** such a challenge?

- observe failure
- guess the cause
- check the guess

wrong!
solved!
Whyline
a Workspace for Helping You Link Instructions, Numbers, and Events

observe failure

click the faulty output

follow dependencies

solved!

Monday, November 30, 2009
an example ... 

why was the line black?
an example ...

why was the line black?
record the problem
record the problem
Resolving classes (856 remaining)

load the recording
why was the line color black?
why was the line color black?
why was the line color black?
why was the line color black?

Monday, November 30, 2009
why was the line color black?
why was the line color black?
why was the line color black?

followup questions about selected event

PencilPaint #25,299's field color was Color #19,941
(UPS) why did this execute?
(1) why did color = rgb(0,0,0)? (source)
(2) why did this = PencilPaint #25,299? (source)
why was the line color black?
why was the line color black?
because gSlider was used twice, ignoring bSlider

why was the line color black?
a comparison study

Whyline group vs control group

both groups had modern IDE features
show declaration, show callers, show references, etc.
subject program

ArgoUML, an open source software design tool

\~150,000 lines of code

22 external libraries

chose two bug reports from version 18.1

- one w/ simple fix
- one w/ complex fix
bug 1 results

**# successful**

- Yellow: whyline
- Gray: control

**minutes**

- Yellow: p<.05
- Gray: p<.05

more successful in half the time
bug 2 results

Whyline: 4 of 10 gave up

Control: Only 2 of 10 gave up

More successful in the same time

p < .05
unsolicited quotes from users

“This is great, when can I get this for C?”
“My god, this is so cool.”
“It's so nice and straight and simple...”
**talk outline**

why is software evolution **difficult**?
*a study of information needs at Microsoft*

how can **tools** help software evolution?
*debugging with the Whyline*

how can **users** help software evolution?
*recent work in leveraging the crowd*
talk outline

- why is software evolution **difficult**?
  *a study of information needs at Microsoft*

- how can **tools** help software evolution?
  *debugging with the Whyline*

- how can **users** help software evolution?
  *recent work in leveraging the crowd*
does open bug reporting work?
with my PhD student, Parmit Chilana

comprehensive analysis of ~500,000 reports from the Mozilla community

- quantitative analysis of to characterize bug report resolution trends
- qualitative analysis to explain bug report resolution trends
four types of contributors

1% **CORE devs** drivers, super reviewers, module owners, peers

1% **ACTIVE devs** assigned bug reports

80% **REPORTERS** reported and commented on bug reports

18% **USERS** only commented on bug reports
REPORTERs are the most active commenters
REPORTERs are the most active commenters
resolution by reporter type

most REPORTER reports are not FIXED
resolution by reporter type

most REPORTER reports are not FIXED
% reports FIXED by each type

REPORTERs have dropped in effectiveness
% reports FIXED by each type

REPORTERs have dropped in effectiveness
why are REPORTERs ineffective?

sampled and categorized 100 reports of each resolution type...

most REPORTER reports = technical support for power users’ tinkering and using old builds

rarely provided static and dynamic context adequate to reproduce problems

reported problems, resolved shortly after
is open bug reporting useful?

yes, but ...

- significant overhead to process bad reports
- only a **skewed subset** of users report bugs
- users who report bugs are bad at providing the **static** and **dynamic** context of problems
- **text is a terribly imprecise medium** for expressing this context
ongoing work

enabling every user to submit precise structured aggregatable bug reports with zero training?
three takeaway points

software quality depends highly on the quality of human communication and cognition

human communication and cognition are faulty and unreliable

carefully designed interactive tools can compensate for these limitations

questions?