where HCI and software engineering meet*

Andy Ko
Information School

* why you should work with Andy
HCI

software engineering
what should be  what must be

software engineering

HCI
developers are users too
end-user programming
software development is CSCW
HCI ≈ requirements engineering
software models the world
developers are users too
end-user programming
software development is CSCW
HCI ≈ requirements engineering
software models the world

my research
the structure of this talk

developers are users too
end-user programming
software development is CSCW
HCI ≈ requirements engineering
software models the world
HCI for software developers
developers are users too!

the first kind of HCI, since all computer use was programming tools haven’t changed fundamentally in decades

- Eclipse and VS = Lisp for imperative languages

lots of HCI challenges in software development tools
better programming languages

programming languages were the first user interface

- the *first* user interface

allow users to manipulate the future instead of present

- inherently descriptive

a more usable language reduces the distance between computation in head and computation in code
left <- ((this previousView).right + 5.0)
better editors

code editors are like editing any other document, but

- code has more structure
- code is *executed*, giving it a temporal dimension

few opportunities for meta data or multimedia in code, unlike static documents
rich metadata in Barista

bringing markup to code

```java
public class ImageTransformer {

    Rotates an `Image` by an angle given in degrees.

    public void rotate(Image img, double degrees) {
    }

}
```
better navigation

Software is a massive web of dependencies

Navigating these dependencies is slow and time-consuming.

better debugging tools

what, of millions of possible causes, could have cause my program to misbehave?

studies of why debugging is hard

- novices in the lab  Ko et al. 2003
- novices in the field  Ko et al. 2004
- professionals in the lab  Ko et al. 2005
- professionals in the field  Ko et al. 2007
why is debugging hard?

people...

ask **why** questions about output

guess **why**

check their guess with low-level tools like **breakpoints** and **call stacks**

find out their guess was **wrong**

repeat
the Whyline for Java
end-user programming = customization

not a programmer

not a programmer

not a programmer
the long tail of user needs

# of needy

- word processing
- online banking
- photo management

100's of apps

Twittering a random synonym of the music genre playing on my iPod

millions of apps

100's of users

millions of users

need
end-user programming

programming to achieve the result of a program, rather than the program itself

coding for result $\rightarrow$ opportunism

opportunism $\rightarrow$ bugs, fragility, failure
end-user programming

opportunism is ✓ if

- code isn’t reused
- requirements don’t drift
- and output isn’t critical

these are rarely the case

in 2003, TransAlta bought $24 mil in worthless hedging contracts because of a bidding calculation error (cut and paste)
key challenges

designing languages that are expressive enough for broad range of unique needs

choosing the right primitives

how can we bring the benefits of software engineering practices, without having to learn them?
HCI is a kind of computer-supported cooperative work (CSCW)
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
information needs at Microsoft
information needs at Microsoft

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

<table>
<thead>
<tr>
<th>Question</th>
<th>% Unsatisfied</th>
<th>Asked by</th>
</tr>
</thead>
<tbody>
<tr>
<td>What code caused this program state?</td>
<td>61%</td>
<td>Coworkers</td>
</tr>
<tr>
<td>Why was this code implemented this way?</td>
<td>44%</td>
<td>Coworkers</td>
</tr>
<tr>
<td>What code could have caused this behavior?</td>
<td>36%</td>
<td>Coworkers</td>
</tr>
</tbody>
</table>
software development is social

difficult to model people and practices formally

↓ specifications live in developers’ heads

↓ developers have to talk to each other to stay in sync
other examples

version control = coordination mechanism
requirements = boundary objects
code = shared documents (like wikis)
HCI and requirements engineering
getting the right design or getting the design right

HCI

Requirements Engineering
tasks that go into determining the needs or conditions to meet for a new or altered product

requirements must be actionable, measurable, testable, related to identified business needs or opportunities
what is requirements engineering?

eliciting requirements
- communicating with customers and users

analyzing requirements
- accounting for and resolving other constraints

recording requirements
- use cases
- scenarios
- processes
what is requirements engineering?

In 80’s, **prototyping** was the proposed solution to requirements engineering, but was abandoned...

- Managers mistake prototype for final product
- Developers use real code from prototype
- Prototypes do not **explicitly** state requirements
- Designers focus “too much” on UI design

Pretty silly from our perspective...
what HCI misses

there’s more than just user needs

- business constraints (scheduling, marketing, intellectual property)
- politics (perceptions of features)
- capitalism (planned obsolescence)
what RE misses

- stakeholders often don’t know what they need
  - RE uses some pretty bad methods

getting the right design can save time and money

- still don’t have strong evidence of this

prototyping is more useful than RE suggests
the whole world is in a line of code
the world in a line of code

software is a tapestry, woven from threads of human concern

- business concerns
- implementation concerns
- user needs
- collaboration requirements
- language constraints
every level constrained by reality

classroom dynamics ➔ the architecture

a student ➔ a data structure

student rankings ➔ an algorithm

submitting grades ➔ a statement

# kids in a class ➔ a parameter
bugs = mismatch with the world’s needs
what’s a bug?

simply an **undesirable** program behavior

- therefore, “bugs” are inherently defined by **people** and their **desires**, not by technology

this is why “bug report” is obsolete

- now called “issues” or “modification requests”
not all bugs are created equal

a fatal crash, if easy to work around, is a minor issue
  - bad for program $\neq$ bad for person

a seemingly innocuous redundant field could lead to losses in time and money
  - minor detail in program $\neq$ minor detail for person
lastly...

none of these points would be a surprise to HCI practitioners in industry

it’s the artificial boundaries between “HCI” and “Software Engineering” that hide these issues
HCI

software engineering
what I’m doing next
work with me if you like...
reinventing software development tools

Whyline for Java
work with me if you like...
making programming useful to people’s work and hobbies

what makes some people hate code and others love it?
work with me if you like…
understanding software development as a collaborative work

how do software developers win/lose design arguments?
work with me if you like...

studying the impact of software on society

“The mediating role of abstractions in HCl”
reinventing software development tools

making programming useful to people’s work and hobbies

understanding software development as a collaborative work

studying the impact of software on society