Big ideas
behind the Whyline

Andy J. Ko, Ph.D.  Associate Professor

UNIVERSITY of WASHINGTON

ICSE 2018
Gothenburg, Sweden
Thanks Peggy

- My undergrad research mentor Margaret Burnett introduced me to HCI and software engineering
- She taught me how think, how to read, and to develop scientific arguments
- She helped me navigate to graduate school, to connect with other mentors
- I wouldn’t be here if she hadn’t mentored me for the past 20 years
Thanks Brad

- My Ph.D. advisor, **Brad Myers**, taught me how to choose great projects, how to convey the essence of their insights
- He seeded me with the intriguing idea of asking systems to *explain* themselves
- His relentless constructive critique but unbounded availability helped me learn fast
Thank you

• This community taught me technical rigor, tested the limits of my humanism

• You provided a (then) 30-year history of powerful ideas about dependencies, analysis, architecture, program comprehension, and encapsulation
Thank you academia

- I’ve been fortunate to have dozens of outstanding teachers across my life, spanning math, physics, sociology, psychology, neuroscience, business, English, philosophy, design, art, learning, and chemical engineering.

- My ideas are mere compositions of those I’ve learned from my teachers.
Big ideas
in the Whyline
Debugging reinvented: asking and answering why and why not questions about program behavior.

*International Conference on Software Engineering*


Key theoretical insight

• Debugging is **slow** because developers iteratively test brittle hypotheses about what caused a failure by manually collecting runtime data.

• Debugging would be **faster** if developers worked backwards from well-understood failure to cause, relying on dynamic dependencies precisely gathered by a tool.
The tool

• Record an execution trace, reproducing an interactive timeline of program output

• Allow developers to select questions about properties of output they know to be wrong
The tool

- Answer questions with **precise backwards dynamic slicing** on output properties
- Present slice **interactively**, allowing developers to navigate causes to isolate the defect, using their knowledge of architecture and requirements to identify defects
Key results

• Novices with the Whyline debug $8x$ faster than novices without it  Ko & Myers 2004

• *Novices* with the Whyline $2x$ faster than *experts* without it  Ko & Myers 2008

• Experts with the Whyline were $3x$ more successful and $2x$ faster than experts without it  Ko & Myers 2009
Academic impact

• Across 4 papers and many citations:
  • Influenced the design of dozens of other interactive developer tools in SE and HCI
  • Inspired dozens of empirical studies about other hard questions to answer about software behavior in SE
  • Replicated and extended on dozens of other platforms and languages in SE, HCI, CSEd, Databases
  • Helped trigger a resurgence of research on trace-based debugging tools in SE, HCI, PL
Industry impact

• Caused **Adobe** to investigate debugging tools for Flash and other design tools

• Influenced **Microsoft**’s efforts at building .NET execution tracing infrastructure, Debugger Canvas, ChakraCore

• Influencing **Apple**’s Safari developer tools

• Influencing **code.org**'s K-12 tools for learning to code
Big ideas
about scientific practice
Reading accelerates innovation

• “The way to get good ideas is to get lots of ideas and throw the bad ones away” – Linus Pauling, Nobel laureate, Chemistry

• One way I took this was to never forget that there are hundreds of thousands of papers full of powerful ideas, and we should use them

• I spent 3 months reading 900+ papers about debugging, diagnostics, human error, root cause analysis, well beyond the boundaries of CS
Reading accelerates innovation

• The work that most influenced me was a paper that Mark Weiser cited in his *Program Slicing* paper:
  • It showed that
    • Debugging required analyzing data flow
    • Developers satisficed their data flow analysis
    • Developers analyzed many more irrelevant than relevant statements
Observation develops insight

• “A few observations and much reasoning lead to error; many observations and a little reasoning lead to truth” – Alexis Carrel, Nobel laureate, Physiology

• As an HCI researcher, I took this to mean that if I didn’t deeply understand the experience of debugging, I could not simplify it, no matter how much I reasoned about it.
# Observation develops insight

- I spent another **3 months** after reading *observing* people debug: hundreds of novices, experts, and myself.

- Led to a rich *intuition* about debugging that helped me predict the utility of design choices I made in the Whyline

- I still use this intuition today to judge the utility of my research ideas and the ideas published in this community
3. Explain why, not just how

- “He who loves practice without theory is like the sailor who boards ship without a rudder and compass and never knows where he may cast” – Leonardo da Vinci

- I took this to mean that the true value of inventions is not in explaining how they work, but why they work.

- These explanations are the generalizable knowledge that stands the test of time, that transfer from tool to tool
3 Explain why, not just how

• The key thing that made the Whyline work was that I synthesized my intuition about debugging into an theory of how people debug and how tools mediate their strategies.

• It was this theory, and not the tool itself, that was the core of the Whyline’s innovation.

• The tool was merely an embodiment of that theory, helping me test and refine the theory.
Big ideas
about automation
Automation is insufficient

• “...in practice slicing is fairly fast, and can often eliminate large numbers of unnecessary statements from slices of programs” – Mark Weiser, “Program slicing.” ICSE.

• He did not claim that it was useful.

• And yet, of 4,500 papers that have investigated slicing, only 3 evaluated developers’ use of slicing tools, all finding that slices are too large, hard to navigate, and incomprehensible at scale.
Automation is insufficient

• Our field’s key mistake was assuming that
  1. *Useful slices are trivial for developers to express*
  2. *The size of a slice determines its comprehensibility*

• The Whyline showed neither are true. Making slicing useful required:
  • A new paradigm for **expressing** a slice (output interrogation)
  • A new paradigm for **navigating** a slice (one dependency at a time)
  • **Re-architecting** of slicing algorithms themselves to align with these new paradigms
Automation is insufficient

- Since the Whyline, others have shown that automation is also insufficient for other technologies to be useful:
  - Refactoring (e.g., Murphy-Hill)
  - Static analysis (e.g., Pugh; Ernst)
  - Machine learning (e.g., Burnett; Fogarty)
- Probably also true for formal verification, program synthesis, testing tools, bug patching, etc.
Augmentation > automation

- We like to believe that with enough data and the right algorithms, our tools can outperform humans.

- The Whyline showed that this overlooks the power of developers’ knowledge and intuition.
  - In the evaluations of the Whyline, participants interacted with slices with 50,000+ LOC.
  - By leveraging their knowledge, expertise, and intuition, developers only ever looked at a few dozen LOC, and still found the defects.
Augmentation > automation

• Two consequences of ignoring developer knowledge:

  1. Our innovations often aren’t useful at all, because they don’t account for what developers know

  2. We miss opportunities to combine human and machine insights to achieve even greater power

• We must invent for the entire system of tools+developers+teams+organizations
Wisdom old and new

1. Accelerate progress by reading
2. Develop a personal intuition for SE practice
3. Explain *why* your tools work
4. Automation is insufficient
5. Augmentation > automation
Thank you.

1. Accelerate progress by reading
2. Develop a personal intuition for SE practice
3. Explain *why* your tools work
4. Automation is insufficient
5. Augmentation > automation