Three Years in the Startup Trenches: Reflections on People, Product, and Software Evolution

Andy J. Ko, Ph.D.
Associate Professor, The Information School
Co-Founder & Chief Scientist, AnswerDash
About me

- I’ve been a professor for 9 years

- I’ve been doing research about programming and software engineering for 18 years, inventing new developer tools and studying teams.

- Up until 2012, I’d never worked as a developer, I’d only ever studied developers.

- This is a talk about what I learned after finally engaging in practice, but with a research lens.
Background

• In 2013, I co-founded a startup based on my lab’s research.

• I had three goals:

  1. Disseminate my lab’s research

  2. Learn why research rarely makes it into practice

  3. Reflect on software evolution by embedding myself in a software startup
We know *how* developers work

- Hundreds of research papers *describe what developers do*:
  - Questions developers ask
  - Information they need
  - Coordination they require
  - Factors that affect productivity
Why do we work this way?

- Very hard to observe decisions longitudinally
  - Interviews/surveys rely on human memory
  - Field observations are time-consuming and can’t observe individual decision making
  - Repository mining only sees a partial trace of a subset of decisions
Method

- **Participation observation** (from anthropology)
  - Founded a software company, worked as CTO full time, 60 hours per week, for three years
  - I wrote daily in diary, reflecting on “*What structured today’s decisions and why?*”
  - Captured 15,000 emails with employees and customers
  - Interviewed co-founders and other executives for triangulation
• Raised ~$2.5 million in venture capital with co-founders Jake Wobbrock & Parmit Chilana

• 20+ full-time employees, 100,000+ lines of code, customers big and small

• I managed 8 engineers, managed product, conducted sales, shaped marketing, architected system, wrote a *lot* of code
AnswerDash

Contextual FAQ

Selection-based search

Contextual Browsing
Analysis

• Synthesized diary, emails, and interviews into **claims** about engineering decisions

• Distilled into **claims** and a **theory** of software evolution decision-making
Results

- Decisions were ultimately structured by a collective effort to **define product value**

- **Product value proposition**: an explanation for *why* software is valuable to stakeholders such as end users or customers

- Multiple factors structured the evolution of this proposition, which then structured the evolution of software
6 (of 9) factors…

Read the paper for the other three…

http://faculty.uw.edu/ajko/publications/answerdash
Diffusion of power constrained collective agreement of value

value\textsubscript{answerdash}

value\textsubscript{customer}
Consistent engineering of value required required consistent understanding of value

cognitive debt

value a

value b

value c

value d
Prosocial communication streamlined communication of value

value = value = value
Cross-discipline implementation of value required trust in expertise.
Social decision fatigue under uncertainty
limited value consensus

value?
value...
happy hour...
Expression of value in code was lossy
A theory of software evolution decisions...

Software evolution is structured by the frequent, collective communication and agreement of a coherent proposition of a product’s value across design, engineering, product, marketing, sales, support, users, and customers.

Decisions in software evolution are primarily about shaping and communicating product value, not about code.
Implications

- Product management and process were far more important than languages, tools, infrastructure in shaping how, why, and when software evolves.

- Developers needed to excel at understanding, communicating, and translating value into code.

- Tools might help developers enforce, check, communicate, prioritize, and filter representations of value in code.
Open questions

- Does this theory hold in other settings?
- What is the effect of inconsistent understanding of product value propositions on defects?
- When a value proposition changes, how can teams disseminate the change and reflect it in code?
- How does value proposition ownership affect software evolution?
- How should engineering balance product value-driven and engineering-driven decisions?
Questions?

Andy J. Ko, Ph.D.
Associate Professor, The Information School
Co-Founder & Chief Scientist, AnswerDash

Research takeaway: Software evolution is structured by the frequent, collective communication and agreement of a coherent proposition of a product’s value across design, engineering, product, marketing, sales, support, users, and customers.

Practice takeaway: engineering isn’t just about great code, it’s about translating and expressing a value negotiated across multiple disciplines.