Navigating Constraints: The Design Work of Professional Software Developers

Research summary

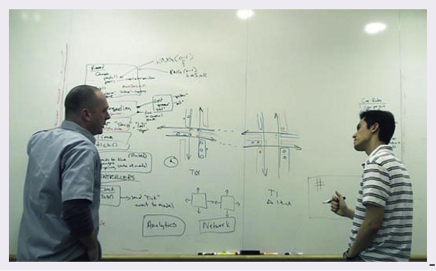
Our research uses **video ethnography** to study **professional software developers doing their authentic work in their place of work**.

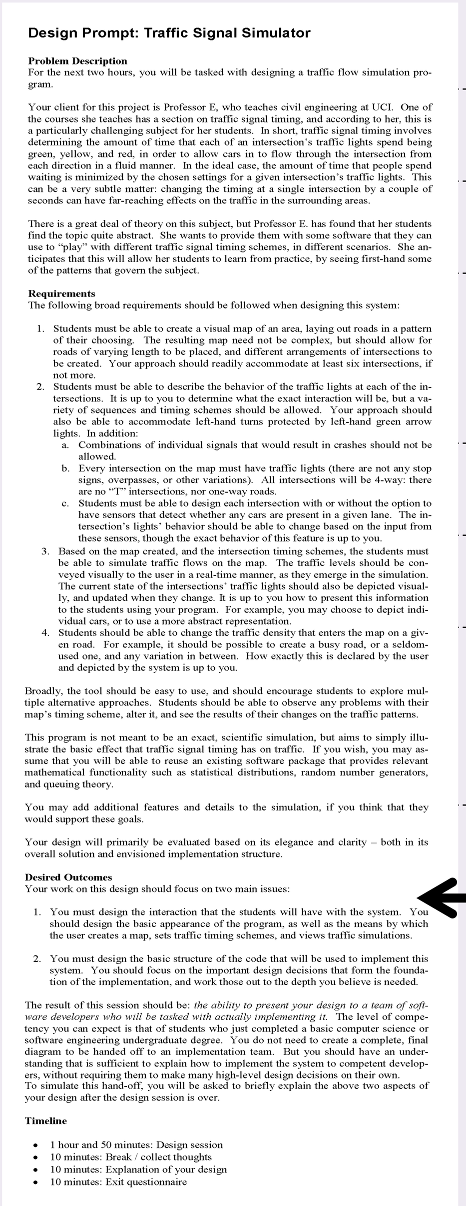
We see **rapid cycles of hypothesis-probe-interpret (HPI)** as the software developers **navigate the highly complex and largely invisible set of constraints** of their existing software system.

David Socha
Computing and Software Systems
Univ. of Washington, Bothell
Bothell, WA, USA
dsocha@uwb.edu
UNIVERSITY of WASHINGTON BOTHELL

Josh Tenenberg
Computing and Software Systems
Univ. of Washington, Tacoma
Tacoma, WA, USA
jtenenbg@uw.edu
UNIVERSITY of WASHINGTON TACOMA

Nature of Constraints

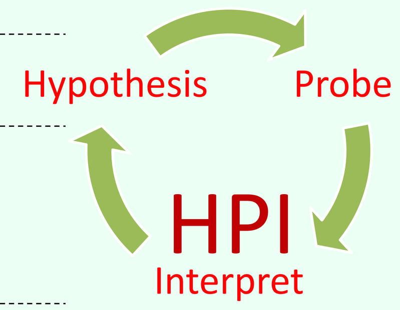


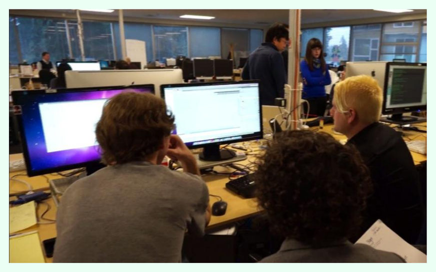

Design Prompt: Traffic Signal Simulator

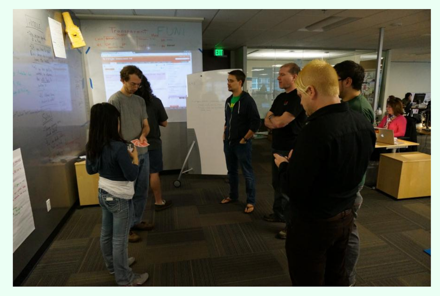
Early conceptual design (Petre et al. 2010; Baker et al. 2012)		**Daily work on successful product** (Socha & Tenenberg)
Inform tool design	***Macro goals***	Increase Customer Lifetime Value (CLV)
Inform pedagogy		Increase Recency, Frequency, Monetary Value (RFM)
(Via workshop publications)		(Via product changes)
Researcher & Educator	***Task derivation***	Business
Research lab / workshop	***Where design happens***	Business workplace
Model possible system	***Focus / Final product***	Deliver value to customers
2 (in design session)	***Design team size***	2 (when pairing), 4 (when probing requirements),
54 (in workshop)		50 (in organization), millions (customers)
None	***Product domain history***	Years
None - Some	***Team history***	Years
2.5 hours (design session)	***Commitment***	Livelihood; years
2.5+ days (workshop)		
{ Highly visible (in prompt)	***Nature of constraints***	Hidden / Bumped into
22 (in prompt)		Uncountable
Simplistic context		Highly multidimensional context
None •	*Existing code*	• 750K LOC
None •	*Existing data*	• Lots
None •	*Organizational*	• 8-year-old company with 50 employees
None •	*Contractual*	• 10+
None •	*Users*	• 13+ million
About what COULD BE		Mostly about what IS
None (in design session)	***Length of feedback cycle (probing)***	Seconds – minutes (when pair programming)
		Minutes – hours (when probing requirements)
		Days – months (in organization)
		Minutes – months (for users)

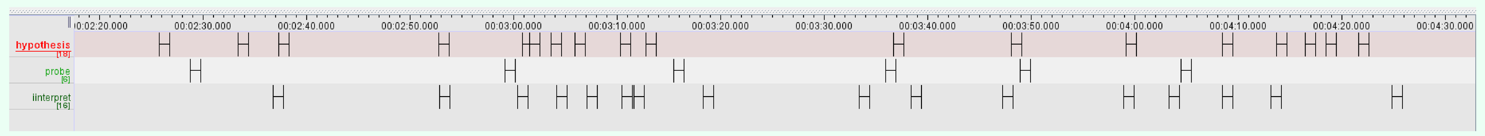
Insights

1. Nature and complexity of constraints is very different from early conceptual design
   a) Huge number and complexity of constraints in the day-to-day work
   b) These constraints are usually invisible
   c) Developers continually probing actual system to understand & discover constraints
   d) The *hypothesis - probe – interpret (HPI)* cycles are extremely fast (seconds - minutes)

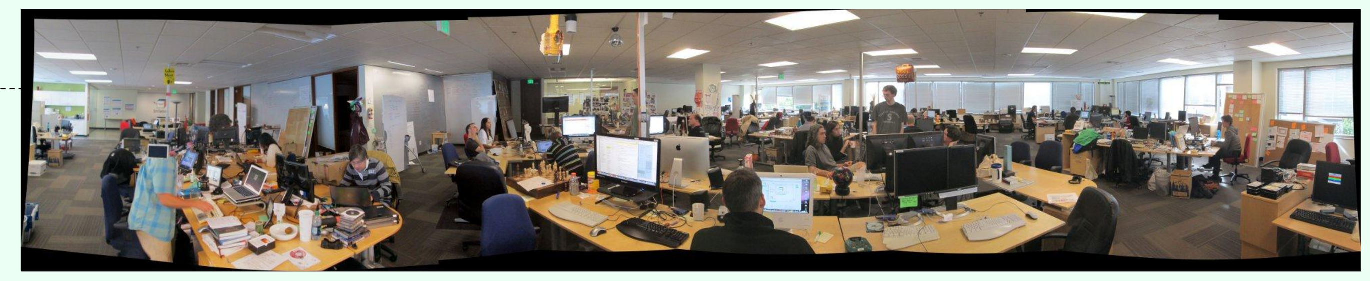






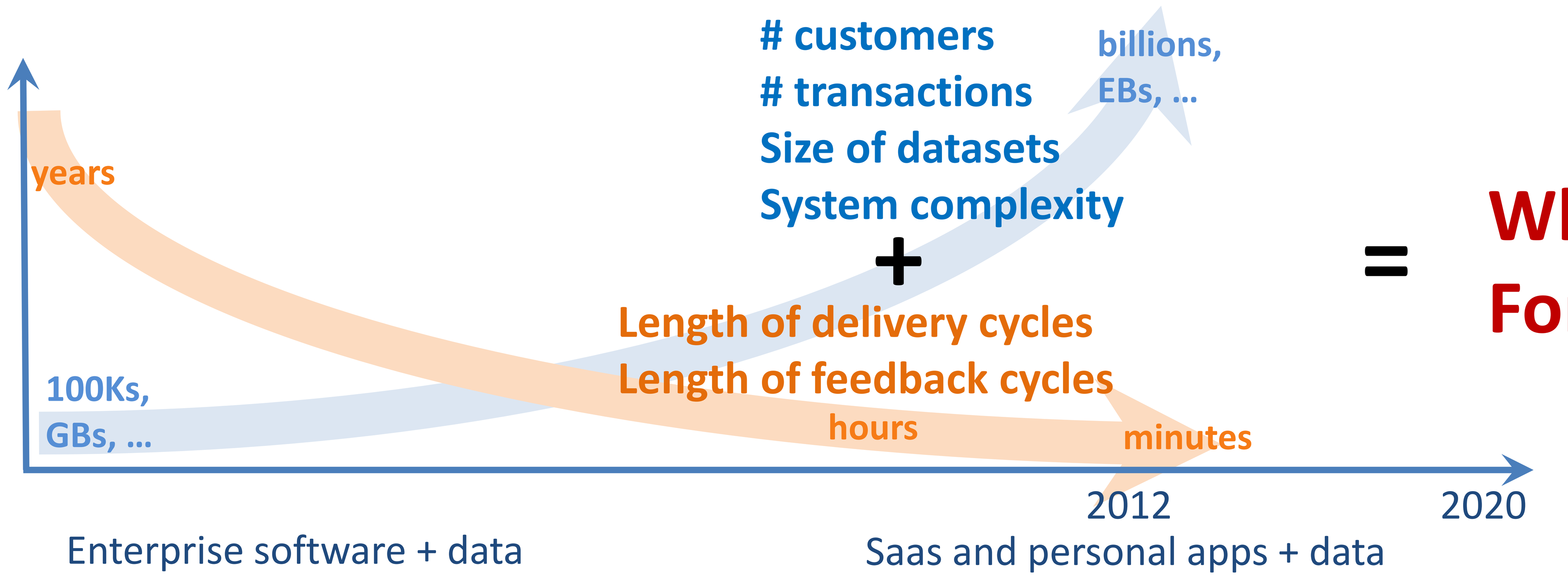


   e) Developers spend huge proportion of time navigating constraints
2. Centrality
   a) Developers physically and metaphorically in center of organization's work
   b) Disco ball and music as visual & auditory metronomes



3. Video ethnography rocks: it quickly uncovers interesting insights

So what?



=

What does this mean for software design? For its theory, practice, tools?