


# 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



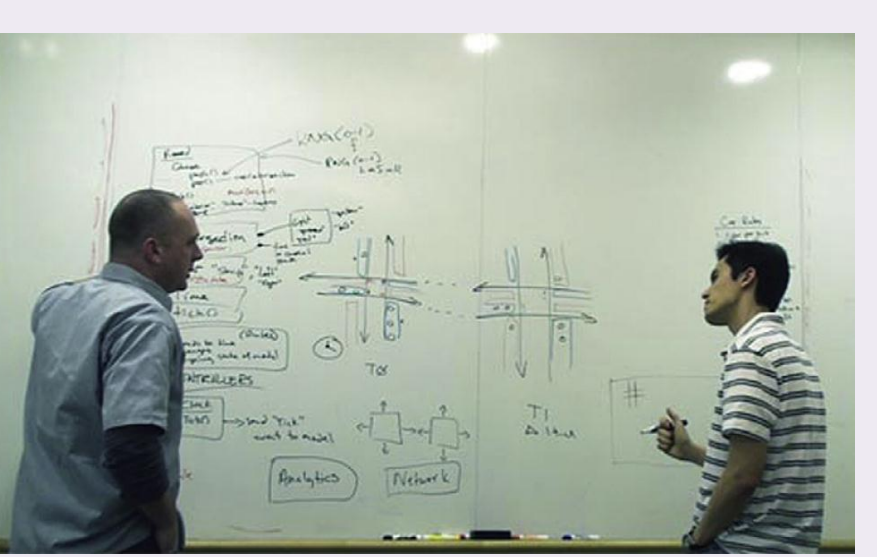
**Josh Tenenberg**  
 Computing and Software Systems  
 Univ. of Washington, Tacoma  
 Tacoma, WA, USA  
 jtenenbg@uw.edu



## Nature of Constraints

### Early conceptual design

(Petre et al. 2010; Baker et al. 2012)



Inform tool design  
 Inform pedagogy  
 (Via workshop publications)

**Design Prompt: Traffic Signal Simulator**

**Problem Description:**  
 For the next two hours, you will be tasked with designing a traffic flow simulation program.

**Requirements:**  
 The following level requirements should be followed when designing this system:

1. Students must be able to create a visual map of an area, being not unlike a system of their choosing. The resulting map need not be complex, but should allow for the possibility of varying traffic flow directions and different arrangements of intersections to be created. Your approach should readily accommodate at least six intersections, if possible.
2. Students must be able to describe the behavior of the traffic lights at each of the six intersections. It is up to you to determine what the exact interaction will be, but a variety of responses and timing schemes should be allowed. Your approach should also be able to accommodate left-hand turns produced by left-hand green arrow lights in a similar manner.
3. A variety of vehicles should be used to represent the traffic. These can be any type of sign, or compass, or other vehicles. All intersections will be 4-way. There are no "T" intersections to be designed.
4. Students must be able to design each intersection with or without the option to have a roundabout. Roundabouts are not to be designed in this project. The use of roundabouts is optional, but the intersection must be designed to accommodate them. These can be designed through the roundabout behavior of the lights in up to two directions.
5. Based on the map created, and the intersection types chosen, the students must be able to simulate traffic flow on the map. The traffic lights should be controlled in a way that allows for the simulation to be run in a loop. The lights should be controlled in a way that allows for the simulation to be run in a loop. The lights should be controlled in a way that allows for the simulation to be run in a loop.
6. Students must be able to design the traffic lights in a way that allows for the simulation to be run in a loop. The lights should be controlled in a way that allows for the simulation to be run in a loop.

**Finally, the tool should be easy to use, and should encourage students to explore multiple alternative approaches. Students should be able to observe any problems with their map's timing scheme, alter it, and see the results of their changes on the traffic program.**

**This program is not meant to be an exact, scientific simulation, but aims to simply illustrate the basic effect that traffic signal timing has on traffic. If you wish, you may use any means that you wish to create an existing software package that provides relevant mathematical functionality such as statistical distributions, random number generation, and sorting theory.**

**Your map and additional features and details to the simulation, if you think that they would support these goals.**

**Design Objectives:**  
 Your design will primarily be evaluated based on its elegance and clarity - both in its overall solution and conceptual representation.


**Desired Outcomes:**  
 Your work on this design should focus on two main items:

1. You must design the interaction that the students will have with the system. You should design the basic structure of the code that will be used to implement this system. You should focus on the higher-level design decisions that form the overall structure of the system, and not on the details of the implementation.
2. You must design the basic structure of the code that will be used to implement this system. You should focus on the higher-level design decisions that form the overall structure of the system, and not on the details of the implementation.


**The result of this exercise should be the ability to present your design to a group of software developers who will be critical with initially experienced. The level of complexity and the amount of detail that you present should be such that you can explain the design to a group of software developers who will be critical with initially experienced. The level of complexity and the amount of detail that you present should be such that you can explain the design to a group of software developers who will be critical with initially experienced.**

**Handout:**

- 1 hour and 20 minutes: Design system
- 10 minutes: Break + collect thoughts
- 10 minutes: Explanation of your design
- 10 minutes: Exit questionnaire



About what COULD BE



None (in design session)

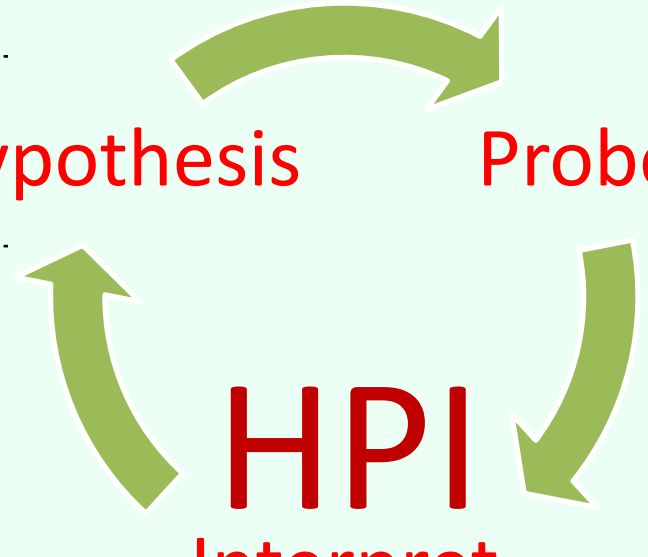
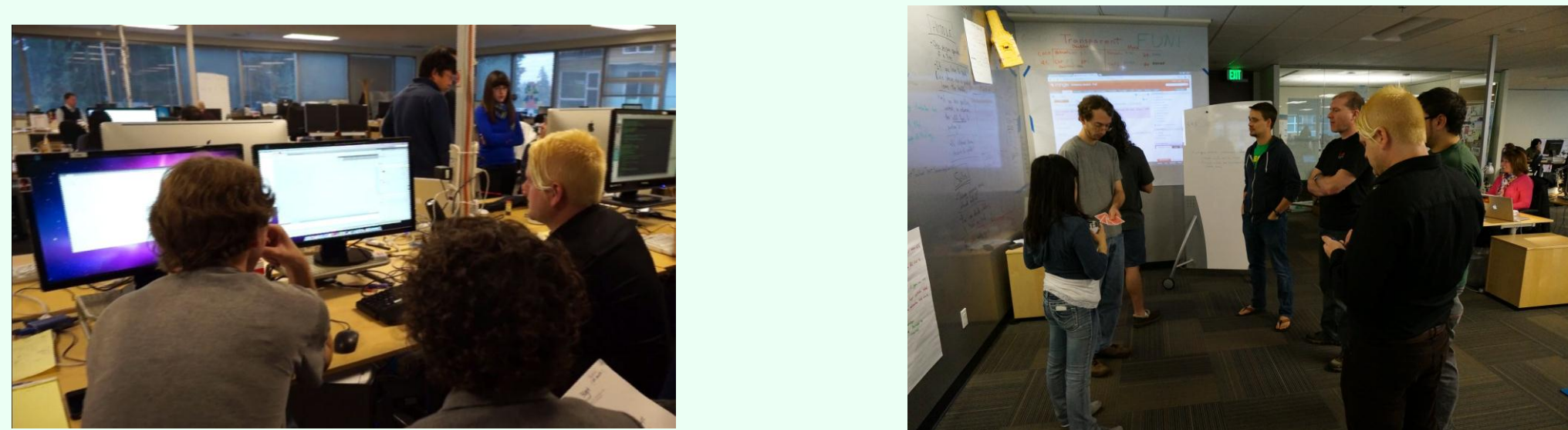
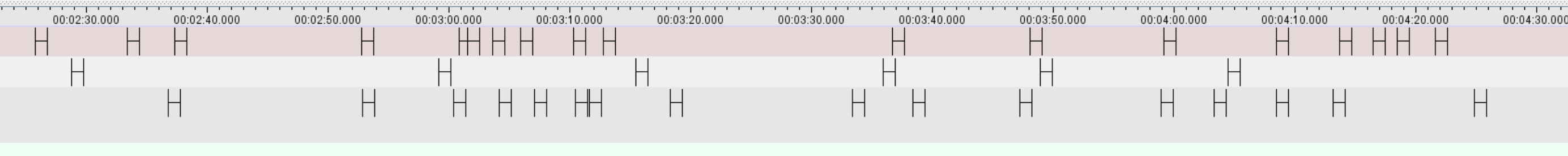
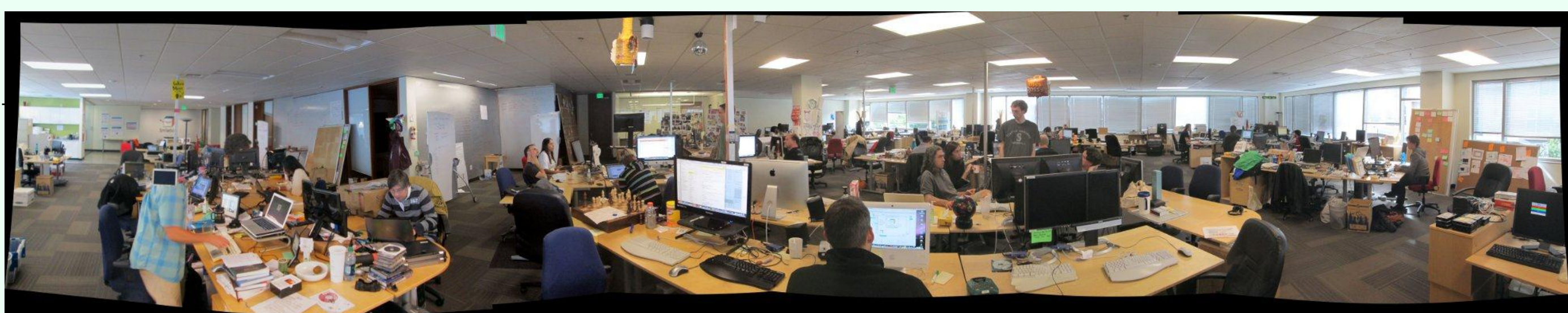
### Daily work on successful product

(Socha & Tenenberg)

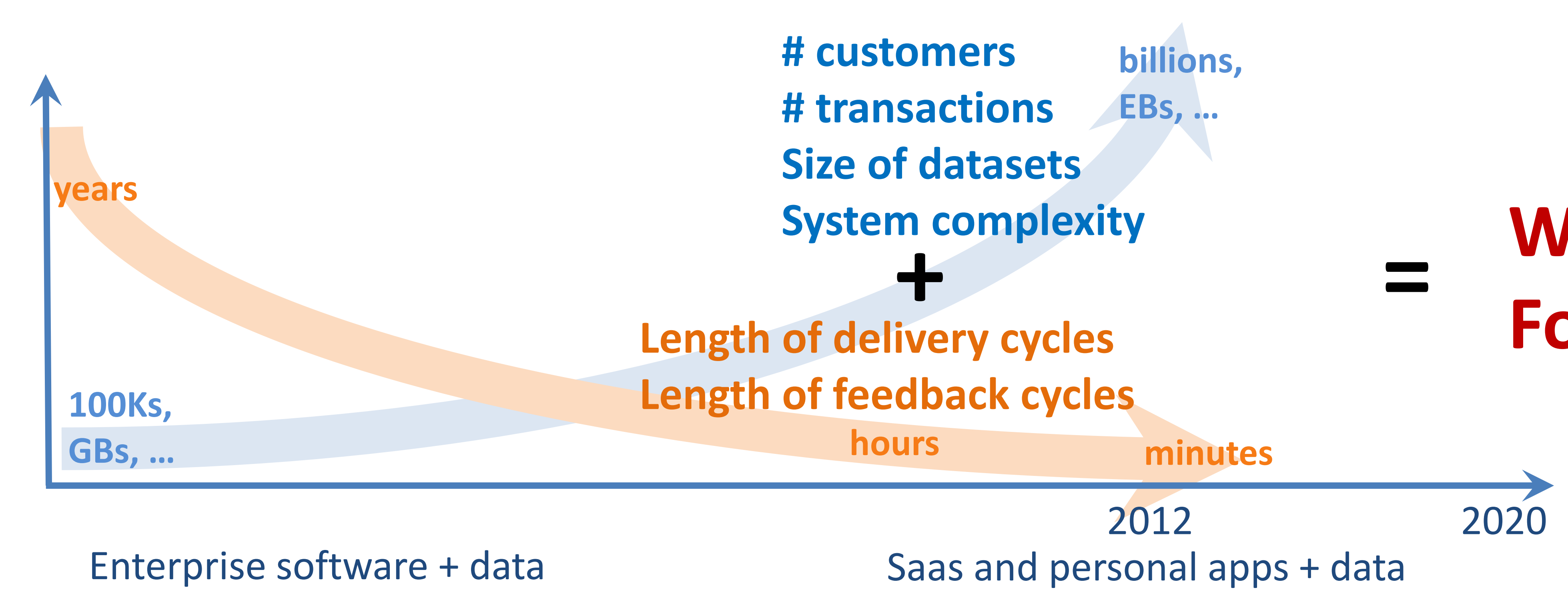
<b>Macro goals</b>	Increase Customer Lifetime Value (CLV) Increase Recency, Frequency, Monetary Value (RFM) (Via product changes)
<b>Task derivation</b>	Business
<b>Where design happens</b>	Business workplace
<b>Focus / Final product</b>	Deliver value to customers
<b>Design team size</b>	2 (when pairing), 4 (when probing requirements), 50 (in organization), millions (customers)
<b>Product domain history</b>	Years
<b>Team history</b>	Years
<b>Commitment</b>	Livelihood; years
<b>Nature of constraints</b>	Hidden / Bumped into Uncountable Highly multidimensional context
<b>Existing code</b>	• 750K LOC
<b>Existing data</b>	• Lots
<b>Organizational</b>	• 8-year-old company with 50 employees
<b>Contractual</b>	• 10+
<b>Users</b>	• 13+ million
	Mostly about what IS
<b>Length of feedback cycle (probing)</b>	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)
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?**