Understanding Property-Based Testing

by *Talking to People*

Benjamin C. Pierce

WG 2.8
May 2022

Partners in Crime

Joseph W. Cutler
1st Year PhD

Harrison Goldstein
3rd Year PhD

Adam Stein
1st Year PhD

Andrew Head
Asst. Prof

Property-Based Testing For All

Everyone^loves PBT!
in this room

We want the other ~7.753 billion people to love it too.

How do we find out what would help them love it better?

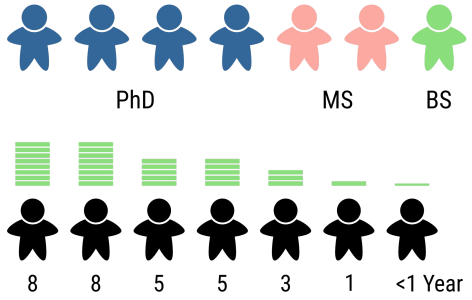
A Preliminary User Study

(in preparation for a bigger one to come...)

Preliminary User Study

Focused on “interviews for need finding.”

Recruited 7 industry users of *Hypothesis* (in Python).



Interview Questions

1. Tell us about your most memorable time doing PBT.
   (To get subjects thinking about a specific experience.)

2. How did you come up with the properties that you tested?

3. Did you need custom generators? If so, what did they generate?

Analyzing Qualitative Data

1. Recruit Informants
    a. We recruited from Twitter (@alpha_convert and @hgoldstein95 🙂)
2. Informants Sign Consent Form
3. Conduct Interviews
    a. 30 minute interviews over Zoom
    b. Calls were recorded, transcribed using Otter.ai
4. "Code" Data
    a. Tag interesting paragraphs of the transcript with labels
    b. Codes depend on the study goals
    c. May need to compute "agreement"
5. Analyze Coded Data

More detail on coding (if requested)

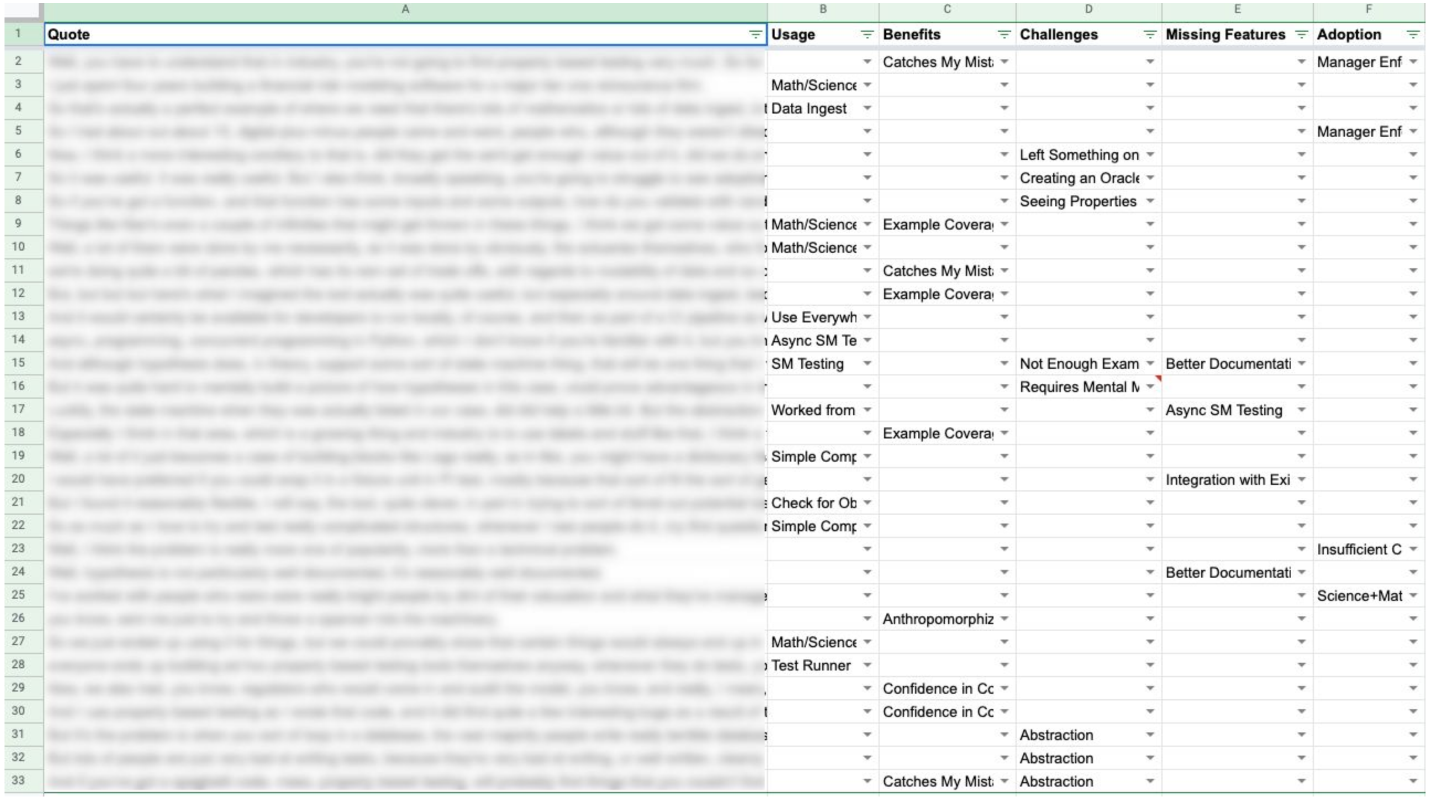
Code categories:

- “Usage”: Specific details about how informants had used PBT
- “Benefits”: Reasons the informants liked using Hypothesis
- “Challenges”: Issues the informants faced during their use of Hypothesis.
- “Missing features”: Features that the informants either explicitly or implicitly asked for
- “Adoption”: Grab bag of codes related to the informants adoption and advocacy for Hypothesis in their workplace

Example codes:

- “Round-Trip property” (Usage)
- “Data Ingest Testing” (Usage)
- “Confidence in correctness” (Benefits)
- “Time/Effort” (Challenges)
- “Unclear Mental Model” (Challenges)
- “Cache Failed Tests” (Missing Features)
- “Worries about Maintenance” (Adoption)

Analyzing Qualitative Data



	A	B	C	D	E	F
1	Quote	Usage	Benefits	Challenges	Missing Features	Adoption
2	[redacted]		Catches My Mist			Manager Enf
3	[redacted]	Math/Science				
4	[redacted]	Data Ingest				
5	[redacted]					Manager Enf
6	[redacted]			Left Something on		
7	[redacted]			Creating an Oracle		
8	[redacted]			Seeing Properties		
9	[redacted]	Math/Science	Example Covera			
10	[redacted]	Math/Science				
11	[redacted]		Catches My Mist			
12	[redacted]		Example Covera			
13	[redacted]	Use Everywh				
14	[redacted]	Async SM Te				
15	[redacted]	SM Testing		Not Enough Exam	Better Documentati	
16	[redacted]			Requires Mental M		
17	[redacted]	Worked from			Async SM Testing	
18	[redacted]		Example Covera			
19	[redacted]	Simple Comp				
20	[redacted]				Integration with Exi	
21	[redacted]	Check for Ob				
22	[redacted]	Simple Comp				
23	[redacted]					Insufficient C
24	[redacted]				Better Documentati	
25	[redacted]					Science+Mat
26	[redacted]		Anthropomorphiz			
27	[redacted]	Math/Science				
28	[redacted]	Test Runner				
29	[redacted]		Confidence in Co			
30	[redacted]		Confidence in Co			
31	[redacted]			Abstraction		
32	[redacted]			Abstraction		
33	[redacted]		Catches My Mist	Abstraction		

What Have We Learned (So Far)?

What have we learned?

1. People who use PBT **really** like it!

   “I've found probably half a dozen major corner case bugs”

   “[PBT] was 1000 times better than the alternative”

   “QuickCheck appealed to me as someone who makes a lot of mistakes and wanted the computer to find them for me”

What have we learned?

2. There are two surprisingly distinct classes of users...

Power Users

- Fully “bought in”
- Tend to have strong mathematical backgrounds (often PhD in Math/CS)

- Need better generators
- Care about speed and efficiency

Everyday Users

- Use PBT occasionally
- More traditional SE background

- Need help “seeing” properties
- Tend to test simple, “extremal” properties:
  - “Program doesn’t crash”
  - “Program behaves exactly like oracle”

These groups can teach us different things!

What have we learned?

3. PBT is hard when code is poorly abstracted
    - Some reported that “carving out” an interface was much of their testing effort
    - Others reported resorting to “end-to-end” properties
    - “I can’t see any properties to test” was a common refrain

What have we learned?

4. We need to do a better job of *teaching* people PBT!
    - Several subjects cited lack of examples / experience as a problem
    - PBT documentation often uses terminology unfamiliar to engineers
    - *We should all teach PBT in our courses!*

Shriram Krishnamurthi has written a bunch about how to do this!

What’s Next?

This is just the beginning!

We are planning a full-scale user study this summer.

Big Questions to Answer:

Where are the points of highest leverage for getting PBT out into the world?

Concretely: What is the best way to move the needle with a PhD dissertation or two over the next 2-3 years?

Discussion…

What would <u>you</u> want to learn from such a study?

Who should we interview? How do we contact them?

How do we find places in industry where PBT can help?

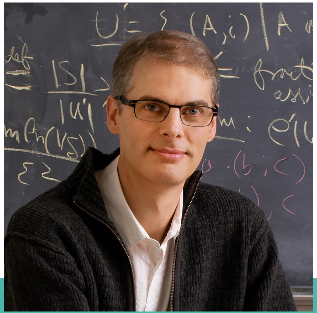
How can we help developers “see the properties”?

How do we help make PBT an ongoing part of a company’s culture?

Are today’s “property DSLs” sufficiently expressive? Aimed at the right set of users?

Besides interviews, where else can we look for insight into these issues? (e.g. GitHub?)

Watch This Space!



PENN PL

PENN hci