The Psychology of Developer Tool Usability

Sara Ford
Sr. Product Manager
GitHub

- **This is not a course on "how to design"**
  - I've sat through enough courses, talks, books on designing UIs, etc. I'm going to try my best not to put you all through another.
  - My undergrad degrees are in CS and Math. I've been in industry 15 years, primarily working on developer tools, e.g. Visual Studio. I led the UX testing efforts for Visual Studio 2005, which got me interested in HCI / UX.
  - In May 2015, I received my Masters degree in Human Factors / Ergonomics w emphasis in HCI, where I wrote a Kinect app for a Kinesiology professor to study how people learn new motor skills.
- **What you will discover from this talk is** all the things that go wrong from a psychological / cognitive point of view when we developers try to design our own developer tools.

Hello Reader from the Internet!

Thanks for checking out my slides! While preparing this talk, I cut about 50% of the possible material I had hoped to cover, even for a 75 minute talk. For my full notes from these slides and for this deleted material, please follow me on medium.com/@saraford where I'll put all the content.

Yes deleting code is hard,
but remember
“simplicity implies genius”

-my graduate advisor

- I went to grad school because I want to make developer tools usable
- Just because something is complex doesn't mean it can't be usable.
- The importance of being concise – intelligence doesn’t imply complexity. It’s just the opposite. Simplicity implies intelligence.
- To prove this point, my graduate advisor for her Motor Learning class made us read 30 research papers but then do only a 5 minute talk. It was horrible. How do you prove you did the reading when all you get is 5 minutes? but this was what she was driving at. The more concise, the more hard-work is implied.
- **This is why it is so hard to delete code.**
    - We feel it doesn’t show how hard we worked to get to this UI.
    - Personally, I’ll even comment out my old code before I’m able to delete it, despite having it committed to git / github.
    - **But simplicity implies hard work.**
    - We just have to allow ourselves to believe this to be true.

Thesis: Our tools are not usable because we design for our own testing scenarios

-@saraford

- Even in my own experiences building my Masters project, for a Masters degree in UX as a developer who wants to make dev tools usable, **I still made major usability errors!**
- Doing root cause analysis, I decided this happened for 2 reasons:
    - 1. in grad school, trying to get a prof's time is impossible much less trying to get a prof's time to do usability testing or finding another grad student who has the time to do usability testing at the end of a semester.
    - 2. We think "this is the idea way to use the software" but it **becomes the ideal way for us to test it**
    - Dev tools suck because we will eventually convince ourselves at a subconscious level we are our own user.
    - what I noticed was that my UI and Default Settings were optimized for the minimum amount of work for me to do the testing.
        - e.g. The UI was designed for the least amount of mouse-movements from me to test the software, since I was constantly testing every 5-10 minutes for accuracy. You can't really automate a Kinect application end-to-end.
        - e.g. I also noticed my default settings were optimized for the quickest ways to verify any changes I made to the code.
    - Yet I couldn't help but feel this design felt correct or in the ballpark.

- Although I said this would ***not*** be a talk on UX concepts, I want to include a few things that blew my mind studying psychology..
- If you only take away one thing, let it be this…
- You know how you see the moon bigger on the horizon than you do looking straight up? It’s an optical illusion. Yep, I swear. I had no idea. And unlike other illusions, you can’t convince your eyes what they are seeing is wrong.
- No one knows why this is. Feel free to google “Optical Illusion” or read this article. https://www.lhup.edu/~dsimanek/3d/moonillu.htm

Sensation vs Perception

Sensation is deterministic
Perception is not

Topic #1

- Sensation - the process of converting physical energy from environment into neural signals
- Perception - the selection, organization, and interpretation of sensory information
- Sensation is the same across the board for all people, unless someone has some sort of injury. Sensation is an engineering process. E.g. I put my hand on the table, and the nerves in my fingers send chemical signals up to my brain.
- Perception is what I do with those signals. To me, the table feels “polished” and a bit cold. To you, the table might feel a bit warm. Perception is non-deterministic.
- Same with color. The sensation is the wavelengths hitting my eye and doing that chemical chain reaction thingy the human eye does. However, how I perceive color is individual. More on this in a sec…

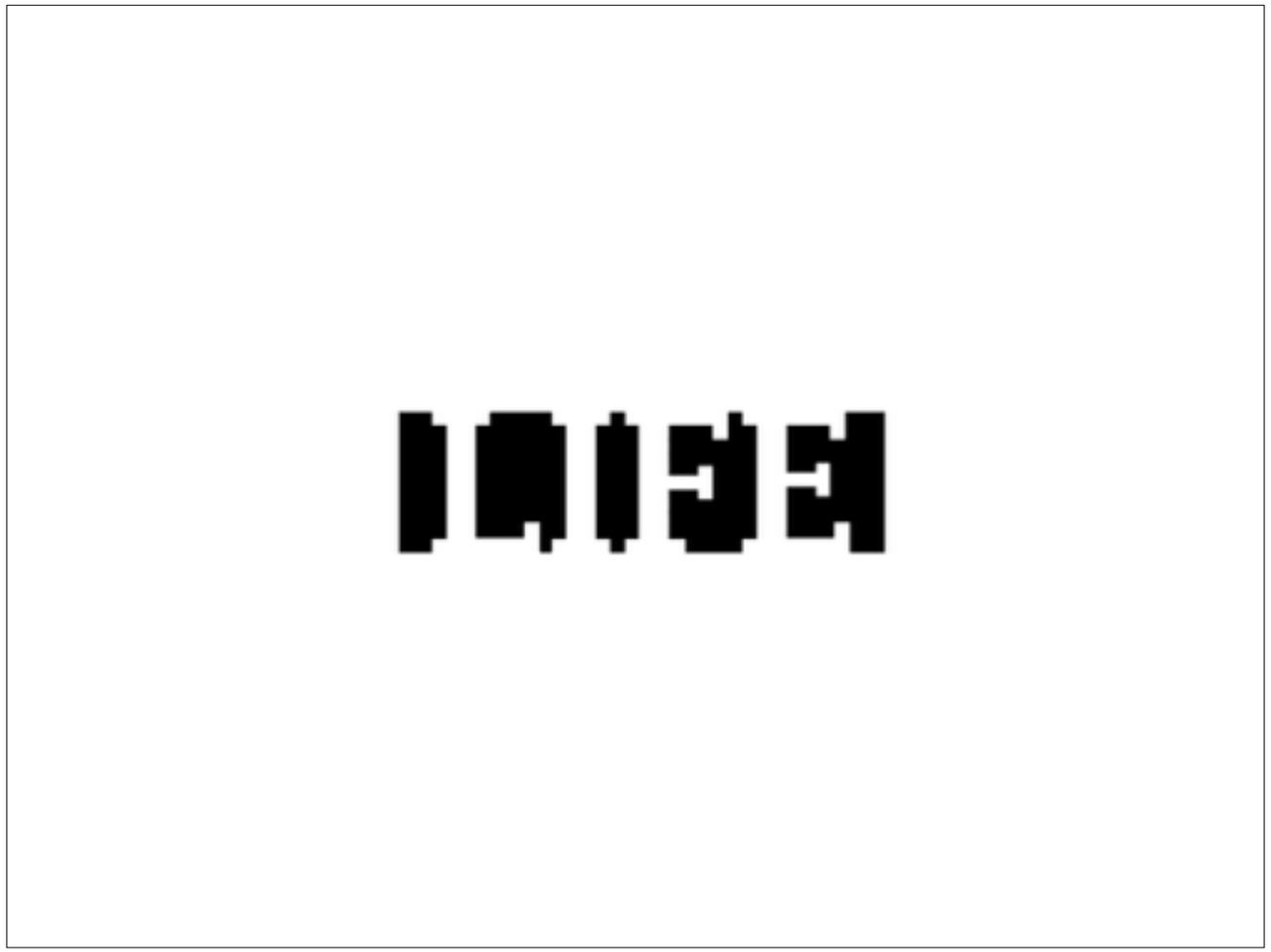
IMO, if a tree falls in the forest… no it doesn’t make a sound because no one is there to perceive it. It’s just wavelengths otherwise.

"What is perceive is based on our experience"

-Designing with the Mind in Mind
page 1

As I told the audience during my talk, “Stop what you are doing right now. Leave my talk, go to amazon, and buy Designing with the Mind in Mind. It is everything I learned in my Masters program without the pain.”

For this next exercise, I have to read a quote directly from the book,
“Imagine that you own a large insurance company. You are meeting with a real estate manager, discussing plans for a new campus of company buildings. The campus consists of a row of five buildings, the last two with T-shaped courtyards providing light for the cafeteria and fitness center. Now the real estate manager shows you the map…”



Everyone see the five black shapes representing the buildings?

- everyone nodded

how many are reading between the lines?

- slowly people started smiling

Perception is based on prior experience. “This sometimes confounds software designers, **who expect users to see what is on the screen”**

-Designing with the Mind in Mind
page 4

Doesn’t that quote just sum it up perfectly? Here I am as a developer giving you a building map, and you're seeing the word LIFE?!
It’s fun to mention that I’ll re-read this book every 1-2 years and every time I forget what the word is between the lines.

Perception Takeaway

We are bias towards our own testing scenarios

We are bias towards our own testing scenarios

- we will streamline for our own scenarios
- why having others PMs, Users, your neighbor, involved is so important to give you a different perspective.

Color does not exist

#2 - Color

- I hate colors. I think I'm missing some cognitive abilities or something because I can never match my clothes. I love jeans, neutral color t-shirts with bright accents, but I cannot figure out styles, color palettes, etc. I'm not color-blind. I just suck at fashion.
    - The thing that drives me crazy about color is that **it doesn't exit.**
    - Remember, the wavelength is the engineering process, but the actual color does not exist. it's just a wavelength.
    - And this scares the bejesus out of me. If I'm holding this purple flyer, WHAT COLOR IS IT REALLY???

btw, if you're intrigued by this stuff, go read "The Man Who Mistook His Wife for a Hat" - OMFG, I couldn't make it past Chapter 4. it was too intense for me. freaks me out.

Color Takeaway

Seek diverse set of feedback

- if there's no way we can agree we're all seeing green, is there anything we can say we know for certain?!
- My call to action is for your defaults when using colors, make sure you are setting up your end users for success, especially if they are like me, who can’t do fashion.
- also, if you’re going to give people choices w color, try to help guide with pre-defined color palettes. Why Themes (e.g. PowerPoint) are so useful.

Don’t forget cleanup steps

Topic #3

"After we achieve a task’s primary goal, we often forget cleanup steps"

"Attention is a very scarce resource. Our brain does not waste it by keeping it focused on anything that is no longer important.”

- “Turn car headlights OFF after arrival, to prevent draining the battery”

all quotes from - Designing with the Mind in Mind p106

Cleanup Takeaway

Test defaults in clean state

Dev tools so bad…

- When we test our developer tools, how often do we remember to revert to an absolutely clean state. Why it is so hard to start from a clean image. Our brain is screaming, "Yay! We're done with the task!"
- We overlook things like our default settings in our dev tools
    - Don't forget to set default out of the box that are usable
    - It's bad enough we ignore our own tools, but even worse when we ignore our own defaults, and hence right out of the box the apps being built are not user friendly.

    - pound of cure vs ounce of prevention

Recognition easy
Recall hard

"See and choose" is easier than "recall and type" - every Psychology book out there

- In the talk I gave an example of being part of a research experiment in class testing recall vs recognition. Although these concepts seem like no-brainers, when you're dealing with non-determinate things like perception, you have to use the scientific method and conduct research experiments to produce reproducible p-values, significance, etc., otherwise you're just doing pseudo-science.
- this is why GUIs took off vs CLI.
- this is also why intellisense is so huge. I don't have to remember all the possible properties and methods off an object. VS just tells me.

A couple of takeaways:

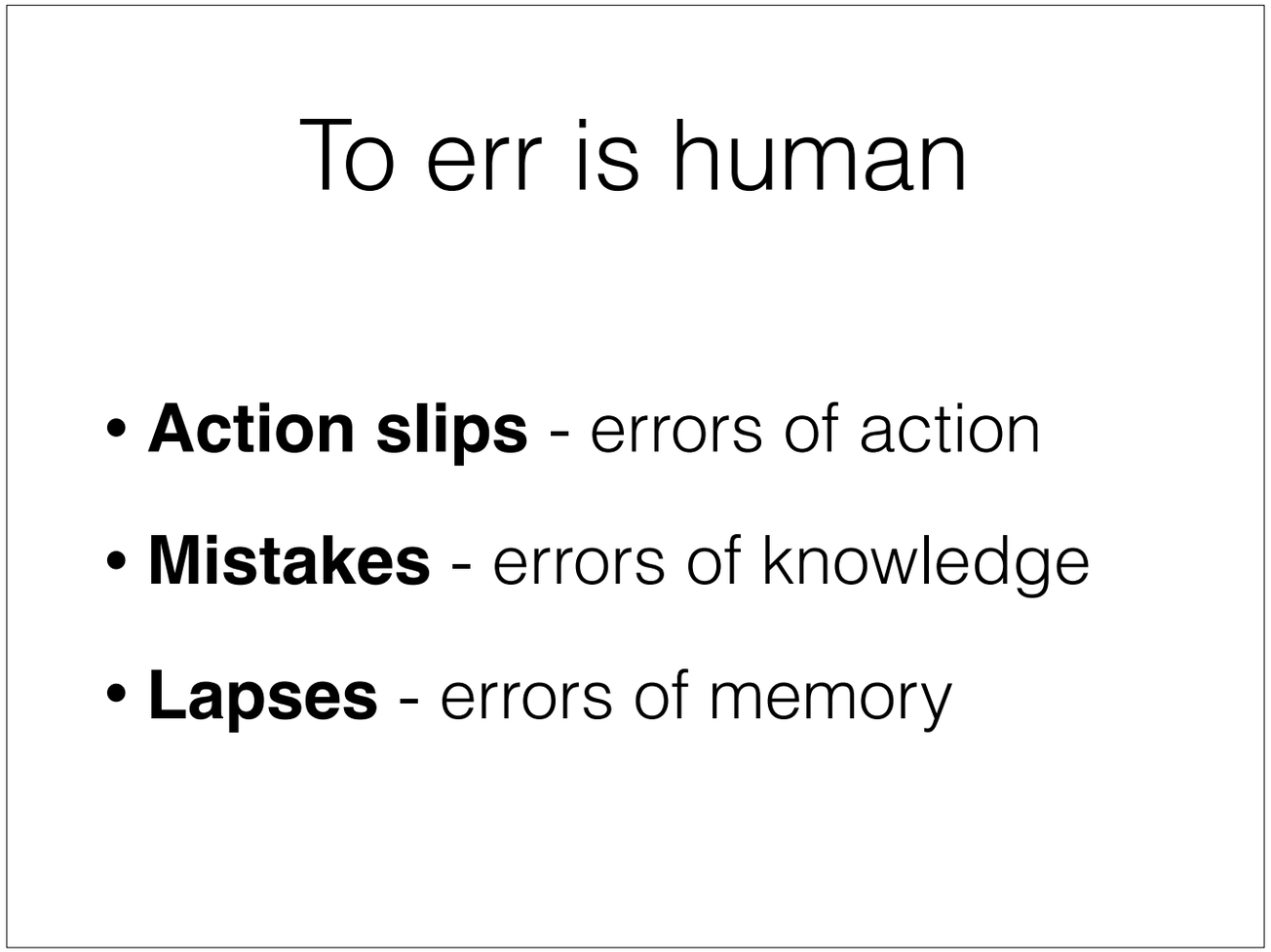
- Don't make users type in if they can choose.
- Error messages - have information persist after debugging ends

“We learn faster when the risk is low”

-Designing with the Mind in Mind
page 149

Topic #5
Do your dev tools allow users to explore risk-free? E.g. how easy can they undo or revert?
Practical application:

- Isn’t it pure evil when you drag a button onto a form and then hit delete and the code-behind or other stuff created behind-the-scenes isn't deleted?



Topic #6

There were numerous topics I wanted to talk about that I studied at length: contextual interference, how to acquire expertise, and human error. Notice how I said “error” and not “mistake”. mistakes are a type of error, here’s why:

1. Action slips - you are driving to grocery store early one Saturday morning but start going to work. You are thinking grocery store, but “auto-pilot” takes over trying to get you to go to work. You had the right intent, but the wrong motor program was used.

- takeaways for action slips:

- try using your framework under extreme stress. Go run around the block. Go say something embarrassing to your boss, then explain why. :) Whatever it takes to get you a bit flushed / heart pumping. Then try to do your talk or use your UI. You’ll find mistakes you’ve never noticed before.

- Action slips are why I tell people to practice, practice, practice your coding demos - it’s not about when things go well. it’s all about the recovery, when things go wrong b/c of action slips. You won’t catch them until your demo becomes “auto-pilot” and you start thinking about the size of the audience, if they caught you messed up on slide 2, etc.

2. Mistakes

- This is where in a multiple-choice exam, I circle ‘C’ thinking it is the right answer, but it turns out it was ‘B’. I had the right motor program, but the knowledge was wrong.

- E.g. Payton Manning said he could play as long as he did because he always knew what the defense was going to do b/c he’d seen it all before. Now compare that to rookies in the NFL. There are 6 rookie quarterbacks. They don’t have that knowledge so they’ll make mistakes.

- this is where new users / usability studies come in, because we know the right actions for the desired outcomes, they don't. We can’t pretend not to have this knowledge. More on this in the next slide.

3. Lapses - e.g. I’m seeing a ton of people I haven’t seen in a while but their name escapes me. It’s a lapse in memory. I know their names, i just can’t remember.

Implicit versus explicit knowledge

Topic #7

- Explicit knowledge is the knowledge that you verbalize.
- Implicit knowledge is unconscious knowledge.
    - **Need a volunteer:** In the talk I had a volunteer rest his hands on the desk. I told him I'd ask 2 questions and he had to instantly answer without picking up his hands. He agreed. I asked him, "Yes or no, did you drive a car here to SVCC?" He immediately said yes. Then I asked, "When you tie your shoes, do you tie them right over left or left over right?" He had to think for a few seconds, and he was trying to pick up his hands, and then he guessed.
        - Like tying your shoes, you don't think which shoe string to pick up first. You tie your shoes in the morning as you think about your first meeting of the day. This is because these motor programs are being operated by parts of the brain that control autonomous functions, which frees you up to do other tasks, like have a conversation. That's why he wanted to use his hands to simulate the motion because he didn't know explicitly.

Knowledge Takeaway

We don’t know what we know!

- This is why dev tools become difficult to learn if you make it easier for experts.
- For example, have you ever tried to teach someone how to code and totally forgot to mention the variable on the left-size of the equal sign is the one being assigned, and not the one on the right? It isn’t really because you forgot. That information just got stored in your unconscious, implicit knowledge.

- We move info into implicit knowledge to free up cognitive resources needed to learn things like lamda expressions, anonymous methods, and Test Driven Development, until those because implicit, and the next generation of developers are like what’s a lamda?

Don’t over engineer

Topic #8

1. When designing an electric hot water heater for tea, remember that people just want to drink a nice cup of green tea. They do not want to be a master tea brewer. - quote from Don Norman’s “The Design of Everyday Things” - the definitive guide to UX design.
2. As developers, especially when working with dev tools, we want to give them a swiss army knife, when all they want is a can opener. We simply cannot help ourselves.

But don’t under engineer

#9

1. The UI needs to be transparent.

- Transparency: the degree to which an interface presents the relevant underlying operations that should effect user actions or the results of their actions.

2. E.g. I initially confused the actions: “Create a GitHub repo” vs “Publish a Repo” in team explorer, because

    1. I was new. As soon as a new user finds a command that might work, they’ll try it. and with the Create a GitHub repo being more prominent in the UI, I went with it.

    2. In my head I was thinking “I want to create a new github repo with the local git repository I have in team explorer” so there was a strong terminology match

“We’re wired for language, but not for reading”

-Designing with the Mind in Mind
page 33

“Reading is unnatural.” - All the piece of info below come from Designing with the Mind in Mind starting on page 33, and from every intro to psychology book out there :) last topic - #10

- Kids are born with an innate ability to learn spoken languages
    - but as you get older it is harder to learn new languages.
- However, writing and reading are only a few thousands of years old, long after brains have evolved.
    - E.g. there are very few people who never learned a spoken language, but many people never learned to read
- With big words, When readers—even skilled ones—encounter such a word, their automatic reading processes probably won’t recognize it.
    - top-down versus bottom up processing
    - Instead, their brain uses less automatic processes, such as sounding out the word’s parts and using them to figure out its meaning
    - This puts stress on the person. More "cognitive load"

Example of Cognitive load

- when you're driving down the street and you are trying to tell a story, you might have trouble paying attention to the road. Your visual areas of the brain are being overloaded.
- I am so visual I cannot listen to podcasts while driving. I can’t visualize the podcast and pay attention to the road.

Reading takeaways

Turn UI upside-down

- Break out of your comfort zone by turning the UI upside-down. Then you're forced to use what's called bottom-up processing (i.e. using the physical characteristics of each letter to make out what the letter is and what the word might be - read the book for more info). You'll be able to figure it out after a bit.
  - Many dev tools will make you read to figure out what you are supposed to do, because again, there's this misconception that because it is complex, it can't be useable, so they try to solve it with a label. Then the screen is overrun by text!

There was a good discussion about this. In the trivial case, where you have a handle for a door and a “push” label above, clearly someone failed at the design, because the handle gives the “affordance” of wanting to pull.
However, for more advanced usage, you’d want users to read the documentation. At least i would hope because I love writing documentation. :) But this is where I’d like to do a v2.0 of this talk digging in deeper to actual research studies of IDEs, etc. looking for patterns.