3 The Implications of Method Placement on API Learnability

3.1 The main point of the paper as the authors sees it: Sohan

Notes:

They claim that method placement in particular classes have significant impact on API learnability.

They mentioned trade-offs for making a decision where to put a method vs usability vs technical issues.

In one of the tasks called email task, they showed the obvious when names carry a huge impact.

The most important part is in the implications section where they talked about

API design - naming, order of methods/sorting, placement of methods inside classes

Documentation design - placeholder methods, referencing methods from other classes

Tool design - auto complete might look into methods from other classes

Main limitation

They talked about API scan instead of API search

3.2 How this paper applies to tabletop research and our research in particular: Jamie

Notes:

The big point in trying to advance this is, someone will be using our programs, and we will likely be the ones writing future APIs for users to read. Much thought has to go into where it makes sense to make these functions, and what they should be named. One thing that they mention is this problem of people making an assumption about where the correct stuff should be, and being wrong about it, then having to figure out where the right place is, and actually finding it, which in itself can be problematic. Based on that, they recommend things like, naming important classes by the more common names, and assigning these classes names earlier in the alphabet. I personally feel that this is a bad solution, because the only reason we want it alphabetically higher is because that is how we ordered them, and at
some point, changing the name just to have it ranked better obscures what the class actually does. They also recommend having entries for classes that the user might expect to be there, and have links to where they actually are, which I personally think could also be a problem, because many times, I don't even read the whole documentation, I check for the name of a method I can use, and then just use it. What I think would be better is having the auto-complete suggest where this functionality actually is. I feel that these APIs should also put more detail into implications of using specific methods over other similar methods, such as with the event handlers.

Specifically when it comes to Tabletops, it's more about actually thinking about what the real user knows, and what assumptions we can make about what they can be expected to understand. More specifically, it's in realizing that we are not the normal user, and what information they may need to make this work.

**Discussion:**

- Previous papers were about usability, this paper is about learnability
- An API is learnable if the methods are in an intuitive place (first places)
- Does putting more important methods earlier in a search list (alphabetical ranking)
- Ranking is a big problem
- So how does this relate to tabletops?
  - Contact Up vs. Contact Leave events in Surface SDK, behaviour between the two is different.
- Don’t have to rely on your gut feeling when making design decisions for APIs.
- We can’t evaluate everything though.
- There are pitfalls in how we do studies. Studies can’t be the answer for everything.
- Example of this: Waterfall development in Software Engineering.
- Replicated studies are rarely done in Computer Science; the classic argument is what is new?
- You can create a wedge by adding different things to the study already done.