Introduction

The component of the Wisconsin bus app that we redesigned was the search function. Based on feedback from the participants of our cultural probe, we concluded that the current search function is confusing because it lacks real world application. One of Nielson's heuristics for usability is "match between the system and the real world," and unfortunately the search function of the bus app fails to satisfy this heuristic. An ideal search function should allow users to input a query in whatever format they desire, and consequently be provided with results that are relevant to their search. With this goal in mind, search functions should not restrict the inputs of users, yet, the bus app only permits users to enter bus stop numbers. In the real world, users know their geographic location by proximity to physical, structural landmarks or human-readable street names. Thus, it is naive for the app to demand a "bus stop number" in which a user would only know if 1) they had previously looked up said number (violating the "recognition rather than recall" heuristic) or 2) if they were physically next to a bus sign (which may not be the correct bus stop since the user couldn't look it up beforehand).

Currently, tapping the "search" button provides a pop-up text box that asks the user to type in the 4 digit number of the bus stop for which they are looking. At this point, if the user doesn't know the bus stop number, which is likely, their search reaches a dead end. If they do know the 4 digit bus stop number, the result does not produce a bus route, but simply adjusts the map to display the bus stop that the user indicated. From there, the user must tap the "more details" button to see which bus routes use that bus stop, and then check if the bus stop they are currently at is on the same bus route. Overall, the current search feature is fairly useless. It is confusing, frustrating, and does not provide the results that many users expect.

In order to make the search function apply to the real world heuristic, we approached its redesign in two different ways. The first redesign (design 1) maintains the current functionality, but instead of calling and labeling it as "search" we replaced it with a destination toggle (pin icon) often seen on apps like Yelp or GoogleMaps which represents precise locations. For the second redesign (design 2) we kept the function as an arbitrary search, but completely overhauled the functionality to make the search process more applicable to search functions in the real world.

Redesign #1

In our first redesign, we replaced the search icon and functionality with a "destination icon" that ultimately zooms in and displays the 3 nearest bus stops. By doing this, we eliminate the confusion of the current app; rather than offering an unintuitive search feature that limits users to querying bus stops by their stop numbers, now with just a single tap they can see the 3 nearest points of interest and can tap the ones for which they'd like to see like more information. Ultimately, we are constricting users' ability to interact with the map even more than the original app, but we offer a faster and more intuitive solution. However, by offering less opportunity for

users to query specific requests, there is a chance that the user may have to inspect several bus stops nearby to determine the one that will suit their needs.

In order to test this design, we used CogTools to evaluate the overall efficiency and time required to find a bus route to the desired location. This design requires users to analyze at least one bus route from a list of multiple bus routes, and in the best case scenario a bus stops near their desired location. In the worst case, the user would have to analyze all 3 bus stops and every bus routes that are shown on each stop (presuming one of those 3 is valid), significantly extending the total time. In order to test this on CogTools, we tested the best case scenario, in which the user is lucky and finds a bus route that works on their first try. However, we set the think time for the user to analyze the list of bus routes to be 5 seconds. We photoshopped 7 different screens to complete the task, and then completed the tasks in 16.4 seconds (best case scenario). This solution eliminates the real world heuristic violation, but does not fix the problem of inefficient and ineffective search. The user is still unable to accurately find a bus route to a specific destination.

Redesign #2

In design 2, we focused on implementing a true search. We kept the search icon, and changed the functionality completely to make the search process more applicable to search functions in the real world. In this design, the user clicks the search icon and is provided with a pop-up text box in which they can type a street address or a specific place such as a restaurant or store. The app then provides the user with the closest and overall shortest bus route to that destination. This solution fixes the real world heuristic violation because it now allows for open-ended search and provides a specific answer/solution to the users problem. We used CogTools to evaluate the efficiency and time required to find a bus route to the desired location. Overall, design 2 is more efficient because it allows the user to type their exact desired destination, and provides specific results as to what bus route will get them there and where the nearest bus stop is. The only reason that this design would be slower or less efficient is because of typing, which requires the user to take more time to think about what they are typing and physically type in their destination. This is a tradeoff that we are willing to accept: more time required to search, but substantially greater flexibility for search

Time comparison

Based on the visualization report, the time for processing the first three steps is the same for both designs. It takes user 3.3 seconds to get from the home page navigation to hitting the search/toggle button (a total think time of 2.2 seconds and a total wait time of 1.1 seconds).

In design 1, we set a mental processing time of 5 seconds when a list of bus routes (all the available buses for one bus stop) is given to the user. After reviewing the possible routes, the user needs to tap the “route” button to select the bus route to see if that bus will go to his/her desired destination. We assume that the user will successfully select the right bus route after a

duration of 5 seconds mental processing. At the end, it takes the user 16.4 seconds to complete the task in the best case scenario.

In design 2, it takes the user 4 seconds to type the destination “van hise” with a think time of 0.4 seconds and a system response time of 0.1 second for each letter input. The process of typing the address takes the most time (almost one third of the total time). However, it eliminates the thinking process of having the user to analyze which bus route he/she should take by providing the best bus route for the user. This is not fulfilled in design two. At the end, the time for completing the task with design one is 12.7 seconds.

The completion time difference between the two designs:
Design 1: 16.4 seconds (result is not verified, unreliable, best-case scenario)
Design 2: 12.7 seconds (result is optimal route, shortest)

We can see that design 2 requires less time (3.7 seconds) to complete the task than design two. The longest time for a single process in design one is 4 seconds for the destination input. On the other hand, the longest time for a single process in design two is 5.1 seconds for selecting one bus route from a list of bus schedules.

Besides the overall time required to complete the task, another difference is the result produced by the search. Design 2 returns the closest and shortest route for user while design 1 (more similar to the original app) provides several bus routes to choose from that the user needs to analyze themselves, and does not give guidance to help the user pick the best route. What is the point of showing a list of bus routes if the user does not have previous experience with bus routes or the app? We assume that the user will pick the correct bus route with the first attempt in our second design. However, what if it is the worst case scenario, and they pick the correct one on their last attempt? This requires a large memory load for users and it is not user-friendly for user without previous experience.

Conclusion

As a whole, user performance modeling gave us a completely new perspective on building & testing better interfaces. For the first time, we were forced to think more quantitatively about the designs we produce— whether it was the number of taps needed to complete a task or the total time to complete a task (we also learned that these quantifiers [taps & total time] don’t necessarily correlate!).

We learned that quantifiable results can provide more indisputable, objective evidence to aid the decision-making process when choosing between multiple design proposals. Unfortunately, this process doesn’t guarantee one design to be better than another, as inherently subjective user-preferences are also of importance. Since quantitative approaches like keystroke-level modeling don’t account for user preferences (or even business goals, which could potentially

demand a task take *longer* than necessary), we see this methodology to be beneficial only under particular circumstances. For tasks that involve substantial navigation, searching for custom-tailored/request-specific information, or anything that requires pre-determined knowledge, keystroke-level modeling can be applied to compare designs in a numeric, objective manner. But for open-ended tasks, like scrolling through Instagram indefinitely or aimlessly hopping around news articles, keystroke-level modeling fails to provide substantially meaningful information.

Additionally, we learned that it may have been better to consider the best-case, average-case, and worst-case scenarios when running our scripts. For example, in our second design the user has complete freedom to search whatever they want. In the worst case, their search may be longer than 15 characters. But in the best case, they may only need to type 2 or 3 characters. In our first design, we present the user with the 3 nearest bus stops, and leave it to them to investigate which one will suit their needs. In this instance, the best case scenario would be if the user selected the correct choice on their first try… but the worst case may need 3 (or possibly even more) attempts. Beyond this, at no point in the script-writing process did we account for differences in users (i.e. it can be hard to tell if we are setting the correct think-time, and there may also be unaccounted correlations like elderly users who take longer to think could also be using legacy machines with substantially longer system response times). The lesson here is that thinking quantitatively can be advantageous for a plethora of reasons, but the results should be interpreted with caution.

Ultimately, we found this experience to be worthwhile. For designers and engineers alike, we can add another perspective to our arsenal of decision-making tools. Using this software made us think very critically about every step, and allowed us to eliminate unnecessary steps before we finalized the design. As we created each new screen for navigation, we changed the design several times to make the process follow Nielson's heuristics more closely. We sought to eliminate redundancy, confusion, mismatch, and overall thought time. Not only did CogTools provide us with meaningful conclusions about efficiency, it taught us to use a critical thought process that can be used when creating/improving systems ***or*** interfaces. Though Cogtools may have its imperfections, it successfully conveyed a new approach that makes it easier and more straightforward to evaluate designs with less subjectivity.