Joyner Document Format v2.2: For Use in CS6460, CS6750, and CS7637

Author Name
sra0374@gatech.edu

1 QUESTION#1

1.1 Q1: Interface Selection

I would like to discuss OSCAR, Georgia Tech's course registration tool in terms of the different models of HCI principles. For the context of this discussion, we shall limit using this interface for *Registrations* feature only.

1.2 Q1: Processor Model Discussion

The Processor Model would necessitate OSCAR to help students register for desired courses *quickly* and *effectively*. By "quickly", the expectation is that hundreds of students are accessing this portal at the onset of their time-tickets and are vying for courses that have limited slots. And by "effectively", the expectation is that OSCAR should not cease registration progresses or render the process futile having students to restart all over again potentially resulting in losing their spots.

Effective Processor Model design would need *OSCAR* to take the least time and effort for a user to find the *REGISTRATION* tab and look up courses. This would involve incorporating *usable physical attributes* like Visual Text, Clickable options and presenting Typeable fields that would get the user to see, touch/click the options available and type to look up desired courses.

In other words, the model design would entail quantifying the time and effort it takes between *opening* OSCAR to *registering* for desired courses. The interface presents a banner *Registration* which the user can visually locate on the screen, allows a Clickable interaction right there, which would then bring up a window where the user can visually find another clickable banner *Add or Drop Classes*; once clicked the interface allows the user to look up classes and *key-in* text to search for courses.

1.3 Q1: Predictor Model Discussion

The Predictor Model would focus on allowing users to navigate through a series of workflows based on user actions to effectively have them register for courses. Here the model intends to delve into the intention of the user rather than plainly presenting a visual or sensory interface with no logical linking of actions.

In the context of OSCAR, the predictor model would present *Registration* hyperlink option under *Student Services* tab that would enable students to logically find a place to register. The hyperlink would then lead a student to another window where all registration related information is grouped together. An expert user may directly register for the desired course by clicking on *Add or Drop classes* option and use the course code. A novice user may click on *Look up Classes* to look up course code and then register using the *Add or Drop*.

This model would anticipate what logical step a student would take in the process of registering for a particular course with least confusion. This would need appropriate grouping of related capabilities together and making these options available for the user to physical click and choose.

1.4 Q1: Insights and Suggested Changes

Effective processor model for OSCAR would dictate better *physical* availability (visibility and arrangement) of the *Registration* functionality. Its focal question would be this - How fast or easily can the user locate the registration option and achieve the user's goal of finding the desired course to register successfully.

The predictor model, takes it a step further, analyzes the series of actions a student would perform from the point of start to end up registering. This requires *logical* availability of resources that make it easy for a student to find the *Register* option, *Look up Classes* and *Add or Drop Classes* – all in a single workflow that trails the user's thought process.

The sorts of improvement a *processor* model would suggest in this case are better *physical* availability of resources on the screen, that could catch the user's visible attention for registration easily, without cluttering too much of information on the screen to allow for confusion-less progression. For ex, it could be presenting banners like *Widgets* that are easily legible, concise, and self-explanatory.

The sorts of improvement a *predictor* model would suggest in this case are better *logical* availability of resources, presented as a series of workflows that trails the user's thought process in the effort of finding courses and registering for them successfully. For example, all registration related information could be grouped under one workflow. *Registration* widget on the first screen, leading to another screen where the user can find *Add or Drop Class*, *Look up Courses, Status* etc.

2 QUESTION#2

2.1 Q2: Interface Appropriateness

YouTube on iPhone is a good contender for an interface that is used in many contexts – listening to content while taking a stroll, watching a recipe video while cooking, listening to music while driving.

2.2 Q2: First Context

Watching a recipe video and trying to follow the instructions while cooking – This is something I personally do when I cook each time whilst timebound. With an intention to prepare a dish, I would first look up and select a recipe video that matches my expected outcome, then play that to learn.

2.3 Q2: First Context Constraints/Challenges

Having mentioned that this context applies especially when I am time bound (most workdays), in-order to not lose time, I start following the instructions as the video progresses, having to physically pause and play each time the video moves ahead faster than my actions. This takes my focus off cooking having to rewind and play it again; especially with soiled hands while cooking, I would be soiling the phone's screen as well posing another risk that of damaging the phone itself.

2.4 Q2: Second Context

Listening to music on YouTube while driving – This happens more often than desired especially with YouTube providing free services and due to lack of availability of regional music, that I sometimes listen to, that is not on other audio platforms like Spotify or Apple music.

2.5 Q2: Second Context Constraints/Challenges

Listening to music on YouTube's free version, is ridden with ads. It brings up ads at almost all times, at the beginning of the content and even in between contents sometimes. *Skip-ads* is a physical press option that YouTube prompts, which is not integrated with any voice command or in-car button press accessibility. This ends up for the user having to take the focus off the road, locate the Skip-ads prompt on the screen and press with far less physical precision. This presents serious perils in terms of road safety with lost focus on the road. Operating screen with less physical precision may even bring up some another un-desired content inadvertently with the finger missing the skip-ads prompt.

2.6 Q2: Third Context

I personally use YouTube to listen to talks or podcasts while taking a stroll with the device in my pocket and audio delivered via airpods. YouTube's free services will not let the phone's screen to be locked and leae the app running in the background.

2.7 Q2: Third Context Constraints/Challenges

When the app is streaming content and is out of the user's physical monitoring, in this case, pockets, while walking specifically, there is an inadvertent screen press potentially lurking every second. That would need the user to grab the phone back and reset the content.

2.8 Q2: Redesign First Context

The interface could present an Eye-ball monitoring option on every video, which when selected will pause playing whenever the user takes the eyes off the screen. This feature could also be extended to rewind or forward at simple eye gestures. This way users need not touch the screen every time for simple video control. This will also prevent damage to the device in my use case, that of touching the screen with soiled hands each time I need to pause/play/rewind etc. Please note that this feature is contingent with the device's camera/hardware capabilities.

Q2: Redesign Second Context

The YouTube app could simply present a prompt like *Say "Hey Google Skip-ads" to skip-ads*. Or integrate the in-video commands to be operated by the car's play/pause buttons. This will avoid losing focus off the road in this context by relaying an audio instruction to the phone to skip-ads when the user realizes ads are being played. This will also avoid listening to ads that the user is not interested in, some of which run beyond a minute or two if un-skipped.

3 QUESTION#3

3.1 Q3: Gulf of Execution Stages

Identify Intentions: Post a question as a *Question* thread using Ed and get answers to the question as responses from Staff/Peers.

Identify Actions: Iterate the list of actions the user would need to undertake as a workflow to perform this task and achieve the user's intentions.

Perform Actions: Physically/logically perform the list of actions from above.

3.2 Q3: Identify Intentions:

In context of the system here, the intention is to post a question to the forum using Ed Discussion and get answers in the form of responses from peers/staff. To expand the intricacies associated with this stage in this context, the intention is to post a thread in the *Class Discussion* category in Ed, as a Question and get answered in the form of responses from peers/staff with their text in the same thread reply.

3.3 Q3: Identify Actions:

Assuming the user is novice, they need to find a clickable button labelled as *New Thread* (highlighted in Blue in ED in the top-right corner), select *Question* to post as a question, enter required title, select appropriate category (*Class Discussion* in our case), enter description for the question, mark *Private/Anonymous* radio buttons as needed and click *Post.*

3.4 Q3: Perform Action:

This stage is where the user, physically performs the above processed actions by the user himself.

3.5 Q3: Gulf of Evaluation Stages

Interface Output: Ed would notify the user as a notification through a visual *Bell Icon* highlighted in the top-left corner of the interface screen indicating response for the action performed by the user.

Interpretation: The user notices the *Bell Icon* highlighted with a numeric indicator and interprets this as a response to their action.

Evaluation: When the user accesses this notification, based on the response received they evaluate if there were any responses, physically visible that they can read as text in the thread their question was posted. This would enable the user to then evaluate this outcome as satisfactory or unsatisfactory.

3.6 Q3: Interface Output:

Once the user hits *Post* with all the relevant content filled in, the interface switches the screen to display the question posted by user in the thread preview pane under *Class Discussions* category with the user name to confirm that the thread was indeed posted by the user in that category. Assuming peers/staff answer the question by responding to the thread, a visual notification would appear as a highlighted *Bell Icon* with a numeric indicator in the top left corner of the screen. The numeric indicator would refer to the number of responses to the user's post.

3.7 Q3: Interpretation:

When the user clicks on the *Bell Icon*, they would be able to see a notification list suggesting that someone replied to the user's comment in conversation *Class Discussion*. This would mean that the user can now interpret that his task of getting response for his question seems to have had some form of completion.

3.8 Q3: Evaluation:

When the user clicks on the actual notification from the list of notifications, the interface would scroll the screen to display the question posted by the user and the response from peers/staff underneath it. If the user feels his context was answered through this response, his evaluation of this task could be marked as completed.

4 QUESTION#4

4.1 Q4: Activity Selection

My old Digital yard sprinkler controller as an interface that I used in my day-to-day life to quote larger Gulf of execution and Gulf of evaluation.

4.2 Q4: Activity Description

The old Digital Yard Sprinkler controller designated up to 10 channels to control various sprinkler valves connected to different areas at my house, like backyard grass, front yard trees etc. The problem was that there was no way to name these channels to user's customization within the interface. It had to be labelled on a sticker or marked outside on the back of the panel which could wipe away over time. So, to make changes to watering schedules or durations associated with weather changes, I had to physically run to each of the locations to check if channel presets correspond to the area, I need watered.

Larger Gulf of Execution: To navigate to a particular channel and set run time, I would need to use one single knob that had press and rotate abilities. The first press would bring *channels* mode that would cycle through when the knob was rotated. The second press would bring *run time* mode that would change with knob rotation to a range of 60 mins (60 knob clicks). If I were to run channel 6 for 10 mins, I would need to press the knob 12 times to get to Channel 6 and then click and rotate for 10 clicks for 10 mins.

Larger Gulf of Evaluation: To know what channel was being fired, I would need to physically run to cover all areas of my yard and physically watch out for water sprays/trickles to interpret what channel corresponds to which area. If there was a water leakage in more than 1 channel, I would never be able to tell which channel corresponds to which area of the yard unless I let the water long enough to leave wet patches on the ground that I would need to hunt for. So there is significant water wastage also posed as a risk.

4.3 Q4: Second Activity Selection

My Smart Sprinkler Controller (Rainbird ST8) that I currently use.

4.4 Q4: Second Activity Description

The new Smart Sprinkler Controller provides a super improved interface that makes yard watering a breeze.

Improved Gulf of Execution: The new device connects to the internet and presents an app interface available on smartphones that lets users work with up to 12 sprinkler zones; each customized to read accurately what each area represents. It provides a direct way to click on desired channels and program run times without having to cycle through all the channels like in the older model. I can directly fire Channel 6 to water my backyard grass (for example) for 10 mins without having to run and check all areas of the yard to see what channel fired what area like previously with the old digital timer. These bridges the Gulf of Execution significantly multifold. This interface also automatically adjusts watering cycles according to local weather resulting in almost no Gulf of Execution for this use case.

Improved Gulf of Evaluation: The interface accurately indicates what area is being watered, duration and time elapsed to give a more real time information to the user. It also presents with a daily summary of watering duration for each of the channels, which is very helpful for me as a user to evaluate especially when the device automatically adjusts with weather changes. This also significantly reduces wastage of water.

4.5 Q4: Lesson Application

The former, being an old Digital timer, may not have the technological prowess to adopt improvements from the latter, a Smart IoT based timer. The Smart timer provides cutting-edge technology to the user in the form of direct channel control and programming, remote control and automatic weather adjustments that is not possible with the older device owing to its hardware design limits. But at the least, what the older timer could adopt is the customizability of naming channels that would help identify what channel corresponds to what sprinkler area *within* the interface itself. This would largely reduce the Gulf of Evaluation. Also, providing more designated buttons for example – one button for channel selection, one for run time and one rotating knob for cycling through the values would significantly reduce the Gulf of Execution.