Comp 422 - Software Development for Wireless and Mobile Devices

- Semester: Fall 2016
- Dr Nick Hayward

User Experience and Interaction

A brief consideration of useful hints and tips for designing an application.

Contents

Positive User Experience

As we start designing our products, it would be useful to be able to identify traits that help create a positive experience for our users in applications and designs.

It would also be useful to understand what might or might not create a negative experience or impression for our users.

So, if we consider a few initial characteristics we see that the following often help foster a positive user experience.

- our application allows a user to feel they are in control
- likewise, it helps develop a sense of confidence and competence with the application
- the application helps encourage high productivity and efficiency
  - it effectively enables and encourages our user to develop a sense of **flow**
- allows simple, routine tasks to be completed as quickly and easily as possible
- produces valid, useful output for the user
- user feels confident with the validity of produced results, calculations...
- considered aesthetically pleasing
- exhibits acceptable, sufficient performance to avoid unnecessary delays and waiting
- stable and reliable for the user...no *blue screen of death*
- makes it easy for a user to correct or modify any errors, mistakes...
- inspires trust and confidence in the user with logical, well-ordered design, navigation...

Hopefully, it's obvious that not all of these traits and characteristics will apply to all types of application, games, and so on. Indeed, many of these characteristics are purely subjective. Oncemore, thorough user testing and evaluation will help develop many aspects of these characteristics in your application.

Negative User Experience

For many of the perceived positive experiences, we can also consider their converse and how they might adversely affect our users. For example, consider how the following may adversely affect a user.

- the application leaves a user with a sense of feeling a lack of control
- it may be overwhelming the user, creating a sense of incompetence and inadequate ability
- the app hinders the user from improving productivity and general efficiency, thereby preventing a sense of **flow**

- simple tasks and routine patterns prove overly complicated for the user
- output from the application is flawed, incorrect, poorly formatted...
- the app may produce unreliable results and calculations
- the UI design is aesthetically dis-organised, cluttered, unappealing...
- it is slow in performing tasks, and exhibits unnecessary delays and lags in performance
- the app is unstable, buggy, and prone to crashing...a user loses data due to poor performance
- finally, the app produces a sense of annoyance, frustration, and thereby a general lack of satisfaction in its overall usage and performance...

Such issues can also be further complicated by a general sense of the following within an application,

- **excessive complexity** and difficulty in general functionality. Most Adobe products would fit this category and perception...
- **too much work** involved to use the application in general. For example, there might be too many repetitive steps to perform a given task, or too many clicks, links, points in the navigation, and so on...
- a general design that conflicts with a user's perception of previous applications, iterations of a design, and competing products. Again, consider the recent backlashes against Microsoft products, and their general designs and functionality...

Violating design principles

Issues that arise in usability can often be seen as a consequence of poor interpretation, implementation, or misunderstanding general design principles.

For example, if we re-consider some of Don Norman's principles we can see how they may impact user experience and create issues within our designs and applications.

- **lack of consistency** - perceived inconsistencies in layout, labelling, behaviour will make it harder for users to detect and learn patterns and relationships. This, of course, will make it harder for our users to form correct mental models. It will also increase cognitive overload, thereby forcing our users to remember more than is strictly necessary.
- **poor visibility** - if there is poor visibility of controls and information, we make it harder for our users to locate specific information, navigation, and general controls. Our users lose time whilst searching for such elements and concepts.
- **poor affordance** - if controls within our applications do not present appropriate visual clues to their operation, some may not even be perceived or understood correctly by our users. Therefore, functionality may become hidden and limited for our users.
- **poor mapping** - default behaviour for controls needs to match a user's expectation. If this does not happen, then confusion may be the result for our users.
- **insufficient feedback** - insufficient, poorly timed, and unclear feedback can often leave a user with a sense of confusion and doubt. A control, action etc should clearly indicate its state to the user.
- **lack of constraints** - a lack of constraints, proper rules, limits and so on may allow a user to perform invalid operations and tasks. It may enable a user to enter incorrect, or even corrupted, data, thereby leading to errors and unexpected results for both the user and the application.

Interaction concept

We're now going to start considering how to plan and design an application's **interaction concept**.

As designer's, we can consider an application's **interaction concept** as a basic summary of our base, fundamental idea of how the user interface will actually work. It describes how the interface will be presented to the user, and the general interaction concepts that allow a user to complete tasks.

One of the inherent benefits of such a concept outline is that it will often highlight initial usability problems, including navigation, workflow, and other interactions that have been carefully considered and planned. However, it is not always possible to define and outline each and every aspect of the interface and its interactions at the initial design stage. Naturally, there will be some assumptions and guesswork until further testing and evaluation has been completed.

Whilst it's possible to describe the interaction concepts for an application by writing a formal specification document, we are going to follow a more agile approach. This includes a combination of prototyping and minimal, lightweight documentation.

Prototyping can be a particularly effective method for testing and trying different design ideas, receiving user feedback through peer reviews and associated usability testing, and then representing and communicating the intended design to a client or other test group.

As prototyping cannot capture and record all aspects of the design decisions and rationale, we can also use lightweight written records as supplemental and supporting material.

So, we will now consider the following as part of an analysis of interaction concepts,

- interaction styles
- information architecture basics, which often include the following
    - a data model
    - a naming scheme, or defined glossary of preferred names and labels
    - a navigation scheme
    - a search and indexing scheme
- an outline of a framework for interactions and workflow
- an outlined concept for transactions and any necessary persistency
- AND, a framework for the general visual design of the application

It's also important to realise that not all of these will necessarily be applicable or advisable for all types of applications.

Interaction style

When we mention an interaction style, we are effectively referring to the fundamental way an application presents itself to a user to allow interaction with available functionality. There are many different concepts for interaction styles, and there is quite a bit of overlap between some of these defined classifications. Many applications will employ a variety or combination of these interaction styles, more out of necessity to allow a user to fully interact with the application.

For example, an application might present the following interaction styles to its users:

- menu driven options - obviously, a user is able to select options from menus, sub-menus etc
- forms - a user is able to enter data, respond to queries etc by completing on-screen forms
- control panel options - might show data visualisations, summaries, quick access buttons and links, and so on within a summarised application display (often considered equivalent to more traditional hardware examples such as a car dashboard etc)
- command line - normally allows a more expert or power user to control the application using commands and queries entered at a command line. However, it may also be a slight modification of this concept through the use of scripts and add-ons for the given application.
- conversational input - a user is able to interact with the application with a back-and-forth dialogue or conversation. This does not, necessarily, need to replicate a traditional human conversation, but there may be a sense of a question is asked, and a reply is expected. A good example of this may be with the application's help or feedback options, but it could also include a query language for the application, selecting options from the menu system and contextual filtering, or even a direct remote feedback and assistance tool such as remote desktop connections.
- direct manipulation - this allows a user to directly manipulate objects within the application on the screen. For example, move elements, shapes etc to create a drawing, modify a photo, type words into a wordprocessor, and so on. The user is directly interacting with the given application.
- consumption of content - the application could simply be a way to consume content. For example, the primary goal of the application might be watching a video, listening to music, or reading a book. With such interaction, a

user is predominantly occupied with navigating and selecting content. A good example of such interaction is in e-book readers and digital video players.

Don't forget, your application will normally use a combination of the above. It may also use novel and alternative interaction styles that will present the user with interesting options within the application. Again, gaming is a good example of such novel interaction concepts.

References

- Norman, D. *The Design of Everyday Things.* Basic Books. 2013.