Comp 388/441 - Human-Computer Interface Design

Week 13 - 14th April 2016

Dr Nick Hayward

Designing a common interaction framework - I

Considerations

- identify core sets of features, tasks, actions, operations, and processes
- consider series of use cases that follow and share similar patterns of interaction
  - *editing application may allow user interaction with many disparate tools and actions*
    - common menu structure, tools...variance is the selected tool itself
    - interaction will be able to follow a similar pattern
  - *we can also see this type of example with games*
    - many different levels, challenges, opponents
    - similar interaction concepts from level to level
- create an initial list or breakdown of these similar tasks or features
  - *then start to design an interaction framework to describe perceived commonalities*
    - such as the presentation and behaviour of the user interface
  - *this list allows us to*
    - understand how the application will fundamentally behave
    - ensure consistency across such similar tasks
      - allowing users to develop correct mental models
  - *by simply documenting the commonalities between such tasks*
    - saves us from re-documenting the same aspects for individual tasks for our overall specs
- framework also useful for the development of the overall design and its technical underpinnings

Designing a common interaction framework - 2

Issues

- how tasks are started or triggered
  - *eg: user selecting an item on a menu...*
- required authorisations
- when and how tasks can be activated and any given cases where tasks may be disabled
- how and when the task is considered complete
- does the start or end of a task signal a change in any status, mode etc...
- what are the effects of the task on the system's data
  - *eg: is data saved automatically, does it persist or is it temporary*
  - *what happens if the task is abandoned*
  - *what happens if an error breaks the task...*

Designing a common interaction framework - 3

Data and persistency - part 1

- need to consider data transactions and persistency in an application
  - *eg: what, if any, of the application's data needs to saved or stored...*
- for the interface and interaction concepts
  - *consider how the actual saving of data works in the application*
  - *is the data generated by user interactions saved in a persistent store?*
  - *is the data saved in a temporary memory cache?*
  - *consider how such data saving and persistency is relayed to the user*
  - *are they aware that the data is being saved?*
  - *is it an explicit act in the interface design?*
  - *is it part of an auto-save option running as a background process?*

Designing a common interaction framework - 3

Data and persistency - part 2

- consider standard data design patterns that include validations of the data
  - *also consider accompanying error and notification messages*
- for the interface and interaction designs
  - *carefully plan how error messages are presented*
  - *whether the validation occurs on the client or server side*
- consider whether partial data for incomplete tasks is saved
- in the interface design, clearly identify potential *save points*
  - *helps correct notification to the user*
  - *we can also offer suggestions, reminders, completion estimates...*
  - *save points allow us to track current data*
  - *has it been saved recently?*
  - *is it a version or a re-write of saved data...*
    - is it a persistent save or cached?

Design and specification - I

- consider various techniques for designing and specifying user aspects of application's design
- how can we communicate options, choices, design concepts...
- prototypes and mockups act as an important part of this process
    - *may include highly detailed visual representations or low fidelity examples*
    - *choice often reflects development priorities and application complexity*
    - *iterative nature of prototypes may also be useful for more complex development*
- designs and specifications relative to clear distinction between
    - *application's appearance*
    - *application's intended or expected behaviour*

Design and specification - 2

Application appearance

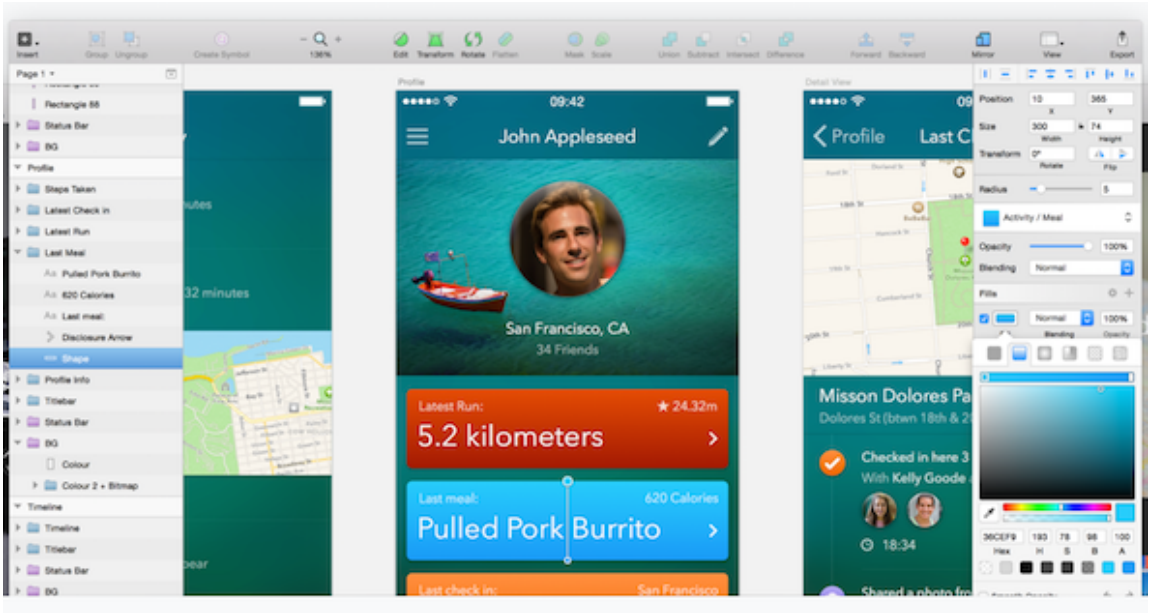
- prototype or mockup helps us plan and visualise an application's appearance and interface
  - *could be high fidelity or low fidelity*
  - *choice often reflects state of the application and intended purpose of the mockup or prototype*
  - *eg: sales/funding demo vs design for development*
  - *perceptual difference between mockup and prototype*
  - *static mockups do not specify behaviour*
    - rely upon additional interaction and behavioural specifications
  - *prototype designed to demonstrate an application's intended behaviour*
  - *prototype perceived as an interactive piece of software in its own right*
  - *not considered fully functional, finished product*
  - *may only represent small components of the application*
  - *intended to show sample scenarios, interactions...*

Design and specification - 3

Hi-Fi mockups

- intended to act as a realistic approximation of an application's design
- allows us to represent and visualise the appearance of the user interface
    - *often used for demonstration purposes, such as attracting funding, sales contracts...*
- allows us to test colour schemes, design layouts, patterns...
- hi-fi mockups normally designed as static images with no actual interaction
- Adobe's Photoshop, Illustrator, In-Design...often popular tools for creating such mockups
    - *offer detailed, relatively quick mockups to help visualise an application*
- HTML, CSS...also popular options for creating quick, hi-fi mockups
    - *can be used for a variety of application mockups*

Design and specification - Hi-Fi mockup



Source - Sketch

Design and specification - 4

Hi-Fi prototypes

- prototype intended to act as an interactive application
  - *not intended as fully functional application*
  - *a concise working simulation*
- prototype intended to create a rapid, working example of functional components of an app
- code often sufficient to simulate and replicate results for a given action and scenario
  - *often will not include a database or persistent data storage*
  - *may simply simulate and demonstrate action of saving the data*
- important to create a prototype of the interface and user interaction
  - *not backend logic and implementation*
- prototypes normally limited in their breadth and depth of functionality
  - *should not be shallow in its implementation*
  - *demonstrate and evaluate an app's specified details in depth*
  - *shows careful, well-planned concept and design for each aspect of your app*
- **NB:** high fidelity prototypes can be time consuming to produce correctly

Design and specification - Hi-Fi prototype

Framer

- many examples available at the Framer website
  - *OK Google*
  - *Android Lollipop*
  - *Carousel*

Design and specification - 5

Low-Fi mockups and prototypes

- low-fi mockups often seen as a **rough sketch** or outline
- often referred to simply as **wireframes**
- their simplicity can offer an inherent utility and speed of creation
- not trying to recreate the exact look and feel of an app
- often more interested in layout of visual components and elements
    - *offers a quick reference point for further development*
- easily sketched on paper, or use formal tools such as
    - *Adobe's Photoshop, Illustrator...*
    - *The Gimp - an interesting open source alternative*
    - *could even use a simple tool like Google Drawings*
    - *many mobile drawing apps as well*
- inherent benefit of low-fi mockups is quick creation
    - *quick to modify and update*
- low-fi prototypes often seen as a series of linked low-fi mockups
    - *simple interaction leads to mockup sketches*
    - *again, not aiming for pixel accurate representations of app*

Design and specification - Low-Fi mockup

Source - Flinto

Design and specification - 6

Rapid prototyping

- provides quick examples of an application's design
  - *helps promote and encourage development and iterative design*
- iterative design helps encourage feedback early in the design process
  - *continues throughout the design process as well*
- we might consider the following as we develop our prototypes
  - *consider what needs to be prototyped early and often*
  - *how much do we actually need to prototype at each stage?*
    - consider the most common design elements and interaction
    - checking how something will work and not prototyping a full application
  - *work out how different places in the app are connected*
    - connection between interactions, places...
    - consider the patterns that exist within the app
    - example pathways for a user through the app to achieve a given goal
  - *choose your iterations for prototypes*
    - helps us avoid the temptation to prototype the whole application at once
  - *different fidelity for different iterative stages*
    - low-fi mockups for initial design layout and elements
    - low-fi prototypes for many initial interactions
    - hi-fi prototypes as we approach the final product

Design and specification - Tools

A few example tools for mockups and prototypes

- HTML, CSS, JavaScript...
- Adobe Photoshop, Illustrator
- Sketch3
- Proto.io
- Flinto
- framer
- mirror.js (useful for Android)
- Google Drawings
- XCode Interface Builder

Testing and evaluating usability - I

Initial considerations

- consider our design for a user interface
  - *may include prototype or a full product/application*
  - *how do we decide and ensure that it meets our users' needs?*
  - *how are we sure it is sufficiently usable?*
- many ways to test and evaluate the usability of a product/application's design
- before testing and evaluation itself
  - *ensure we have a clear idea of our target goals*
  - *type of information desired in our usability evaluations*

Testing and evaluating usability - 2

Selecting goals for our usability testing

- always a good idea to be sure of the data or learning goals desired from the testing
- helps us determine the best and most appropriate methods to employ
- by setting goals we are also more likely to stay focused on testing requirements...
- such goals may include
    - *find places where users become easily confused, hesitant, or unsure how to proceed*
    - *which places in your application are causing users to make the most errors*
        - any error hotspots within the application's design
    - *which places cause users to regularly consult documentation and application help*
    - *collect information, feedback, suggestions etc from your users*
        - what they think is working well, what needs improvement
    - *collect general judgements and feedback from your users on*
        - general aesthetics, usability, and value of the application and its available features
    - *collect feedback from users on similar, competing applications they may have used or tried*
        - how does your application compare to these alternative options
    - *for each given application task, determine percentage of users able to complete it successfully*

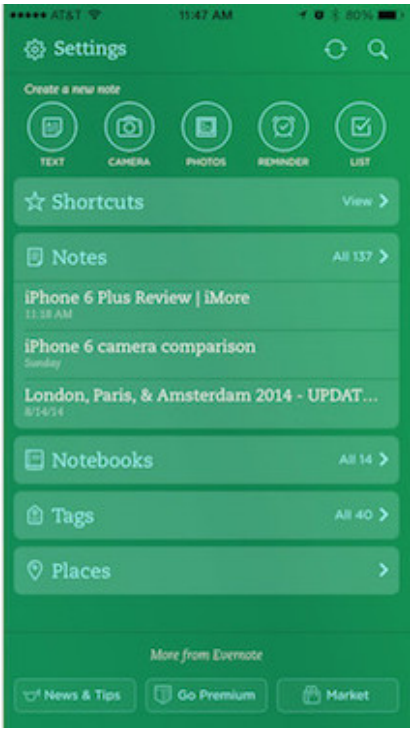
Testing and evaluating usability - comparisons

Offer users examples of different interfaces for comparison

Source - Mobile Comparison

Testing and evaluating usability - comparisons

Offer users examples of competing applications for comparison

		
Evernote	iOS Notes	Google Keep

Testing and evaluating usability - comparisons

Offer users examples of previous iterations for comparison

Source - Windows Comparison

Testing and evaluating usability - 3

Metrics

- usability testing enables us to collect metrics for general application usability
  - *in particular relative to prototypes and models*
- for example, this might include
  - *measure and record user error rates*
  - *average times for tasks*
- compare statistics across different iterations of designs, testing sessions
  - *determine whether application changes actually led to quantifiable improvements or not*
- Tyldesley in 1998 suggested a few considerations for testing and usability, including
  - *amount of time a user spends on errors*
  - *percentage or number of errors*
  - *number of commands used to complete a given action or task*
  - *amount of time a user spends using the help system or documentation*
  - *user frequency of help system or documentation*
  - *number of users who prefer your application to competing options*

and many more...

Testing and evaluating usability - 4

Evaluating our users

- need to consider options we might employ to help effective usability testing
- not all options suitable for all evaluation scenarios
  - *pick and choose most appropriate options for testing requirements*
- a few examples include
  - *user observation*
  - *cognitive walkthrough*
  - *analytics*
  - *focus groups*
  - *questionnaires and surveys*
  - *heuristic evaluation*

References

- Robinson, W.L. *Conscious competency - the mark of a competent instructor.* Personnel Journal, 53. PP. 538-9. 1974.
- Shackel, B. *Usability - context, framework, design, and evolution.* Human factors for informatics usability. Cambridge University Press. PP. 21-38. 1991.
- Tyldesley, D.A. *Employing usability engineering in the development of office products.* Computer Journal, Vol. 31. No. 5, PP. 431-436. 1988.