Comp 341/441 - Human-Computer Interface Design

Spring Semester 2017 - Week 14

Dr Nick Hayward

Final Presentation & Report

- team presentation on Thursday 27th April @ 7pm
- team report due on Thursday 4th May by 7pm

Final Assessment Outline

- final demo
- presentation, online demo, video overview..it's your choice
- final report
- clearly detail design and development process
- outline testing, prototypes etc
- explain pros and cons of interface
- contrast old and new interface (where applicable)
  - *group report must clearly define each student's work and contributions, where applicable*
  - *no attribution, no mark*

n.b. ~ 10 minutes per group

Final Assessment Report

- report outline - demo and report

Positive user experience

- we need to be able to identify traits of a positive user experience
  - *conversely, understanding a negative experience is also helpful*
- application allows a user to feel they are in control
- helps develop a sense of confidence and competence with the application
- helps encourage high productivity and efficiency
  - *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...

Negative user experience

- application leaves a user with a sense of feeling a lack of control
- overwhelming the user, creating a sense of incompetence and inadequate ability
- hinders the user from improving productivity and general efficiency
  - *prevents 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 disorganised, cluttered, unappealing...
- slow in performing tasks, and exhibits unnecessary delays and lags in performance
- unstable, buggy, and prone to crashing...
  - *user loses data due to poor performance*
- **excessive complexity** and difficulty in general functionality
- **too much work** involved to use the application in general
- design that conflicts with a user's perception of previous applications, iterations of a design, and

competing products

Violating Design Principles

- issues that arise in usability
  - *consequence of poor interpretation, implementation, or misunderstanding general design principles*
- reconsider Norman's design principles
  - ***lack of consistency***
  - ***poor visibility***
  - ***poor affordance***
  - ***poor mapping***
  - ***insufficient feedback***
  - ***lack of constraints***

Designing an interaction concept

intro

- app's **interaction concept**
  - *basic summary of our base, fundamental idea of how the user interface will actually work*
  - *describes presentation of the UI to the user*
  - *general interaction concepts that allow a user to complete tasks*
- inherent benefit is that it will often highlight initial usability issues
  - *including navigation, workflow, and other carefully considered and planned interactions*
- every aspect cannot be defined and outlined at the initial design stage
- follow a more agile approach instead of formal specification documents
- prototyping a particularly effective method for
  - *testing different design ideas*
  - *receiving feedback through peer reviews and associated usability testing*
  - *representing and communicating intended design to a client etc*
- lightweight written records as supplemental and supporting material

Designing an interaction concept

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

Designing an interaction style

- app's **interaction style**
  - *fundamental way it presents itself to a user to allow interaction with available functionality*
  - *many different concepts for interaction styles and overlap*
  - *many will employ a variety or combination of these interaction styles*
- an application might present the following styles to its users
  - ***menu driven options*** *- user is able to select options from menus, sub-menus*
  - ***forms*** *- user able to enter data, respond to queries by completing forms*
  - ***control panel options*** *- may show data visualisations, summaries, quick access options*
  - ***command line*** *- allows expert, power users to control the app using commands and queries*
  - ***conversational input*** *- user may interact in a back-and-forth dialogue or conversational style*
    - a sense of question asked and reply returned
  - ***direct manipulation*** *- direct user manipulation of objects within the app on the screen*
  - ***consumption of content*** *- app is simply a way to consume content*
    - eg: e-Book readers, music and video players...
- an app will normally use a combination of the above interaction styles

Image - iPhone

considerations of mobile application interaction styles

Apple iPhone

- Source - Apple iPhone

Designing the information architecture

intro

- concerned with the organisation of information into a perceived coherent structure
- structure is considered comprehensive, navigable, and in many situations searchable
  - *eg: concepts, entities, relationships, functionality, events, content...*
- designing such information architecture requires the following considerations and implementation
  - *data model*
  - *naming scheme or glossary*
  - *names and titles for identification of places*
  - *navigation and location awareness*
  - *navigation map and associated mechanisms*
  - *breadcrumbs and navigation notifications*
  - *presentation of such places*
  - *searching*

Image - Designing the information architecture

visualisations

Apple iOS Health

- Source - Apple Health

Designing the information architecture

data model, naming scheme, naming places...

- identification and recording of the entities, attributes, and operations for each entity
- also includes identification of the relationships between the entities
- often argued that the data model is, in fact, part of the app's interaction concept
  - *perceived to help define the nature of the product*
- coherent and consistent naming scheme is important to aid user's mental model
- definition of official names for an app's key elements and processes
  - *can be formalised and recorded in the defined interaction concept*
- apps with specialised domains may require a glossary of names and labels
  - *helps define the official, preferred terminology*
  - *interaction concept may then link or reference this glossary*
- places within an app should be clearly named and labelled
  - *helps users determine what they are viewing and where in the app*
  - *helps users differentiate places and concepts within an app*
  - *clear naming of places helps define them in menus, instructions, help text...*

- user-defined place names are OK as well
  - *eg: a title of a document in an editing app*

Image - Designing the information architecture

personal naming schemes

Apple iOS Photos

- Source - Apple Photos

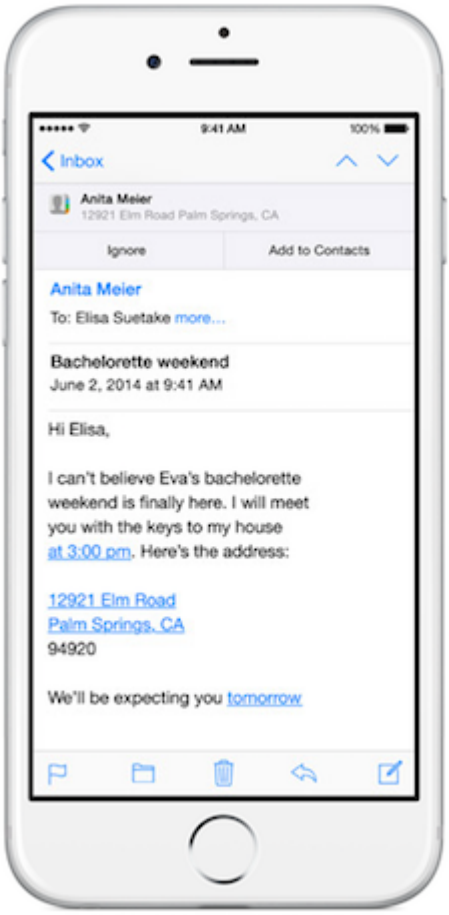
Designing the information architecture

navigation and places

- app design often references navigation relative to defined places
  - *eg: in a web app places may be defined as pages or screens*
- not all places need to be user accessible
- places may also refer to sub-divisions such as panels, tabs, sub-sections...
  - *sub-sections may also include dialogs, image presentations etc*
- for apps with many places, a design should help users determine and differentiate
  - *where they are currently located within the app*
  - *where they can go next*
  - *how to easily get where they want to go*
- in addition to naming places, we need to consider their actual presentation as well
  - *how do we present different places to our users*
  - *view multiple places at once, or page/navigate through single places*
  - *can these places be resized, moved and rearranged, opened, closed, hidden, removed entirely...*
  - *can we relate content from one place to another*

Image - Designing the information architecture

determining places



Apple iOS Mail

- Source - Apple Mail

Designing the information architecture

navigation map

- allow us to consider and define the places that may exist within our application
  - *the movements allowed from one to the other*
- beneficial if represented in a graphical manner within quick reference diagrams
- designing a complete navigation map at the design stage may be impractical and counter-productive
  - *initial map can always be expanded and modified as we develop the application.*
- some instances where a navigation map is simply impractical
  - *eg: dynamic applications, such as catalogues, wikis, some games...*
  - *many different links, pathways, and related material a user may generate*

Designing the information architecture

navigation mechanisms

- many different ways for a user to switch places and content. A few defined examples include
  - ***bookmarks***
  - ***buttons***
  - ***events*** *- triggered by a user action or application process can show a notification or message window*
  - ***flow diagrams*** *- visualise steps and outcomes relative to the current complex process or workflow*
  - ***hierarchical structures*** *- eg: trees used to display hierarchical depth of data...*
  - ***history***
  - ***links***
  - ***maps*** *- data points represented geographically, or conceptual map of data, app domain...*
  - ***menus***
  - ***searching*** *- simple act of searching by keyword, selecting from a faceted list of terms...*
  - ***switching*** *- move between multiple places currently available within the UI*

Designing the information architecture

user location

- clearly identify a user's current location
- acts as a quick reminder to the user
  - *also creates a familiar contextual placeholder within the app*
- indicate the user's current location in a number of different ways
  - *clearly display the title or name of the current place with any associated contextual name*
  - *highlight the current place name or title on a visual map or flow diagram*
  - *include a representation of location on a visual flow diagram for a process of series of tasks*
  - *locate a current place within a defined hierarchical structure*
    - such as a tree representation of the current document or data...
- breadcrumb trail useful for hierarchical data representations
  - *benefit of acting as both location indicator and simple form of navigation*

Image - Designing the information architecture

user location example

Apple Keynote

- Source - Apple Keynote

Designing a common interaction framework

a few 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

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

data and persistency

- 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

data and persistency - cont'd

- 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?

Testing and Evaluating Usability

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*

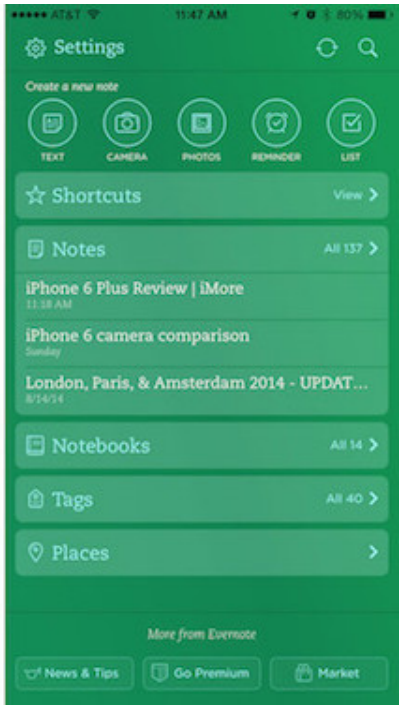
Testing and Evaluating Usability

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*

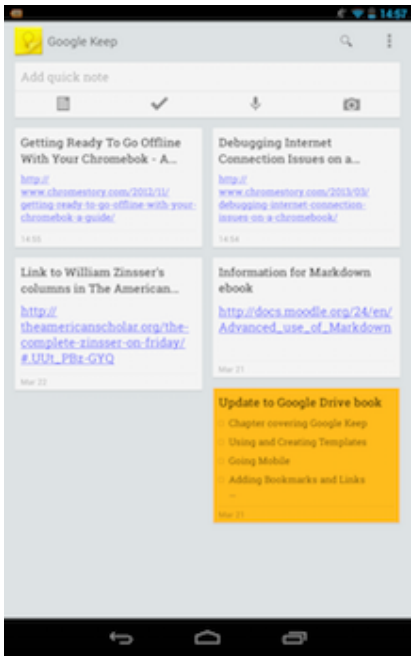
Image - Testing and Evaluating Usability

comparisons



Evernote

iOS Notes



Google Keep

Testing and Evaluating Usability

comparisons

Windows

- Source - Windows Comparison

Testing and Evaluating Usability

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

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*

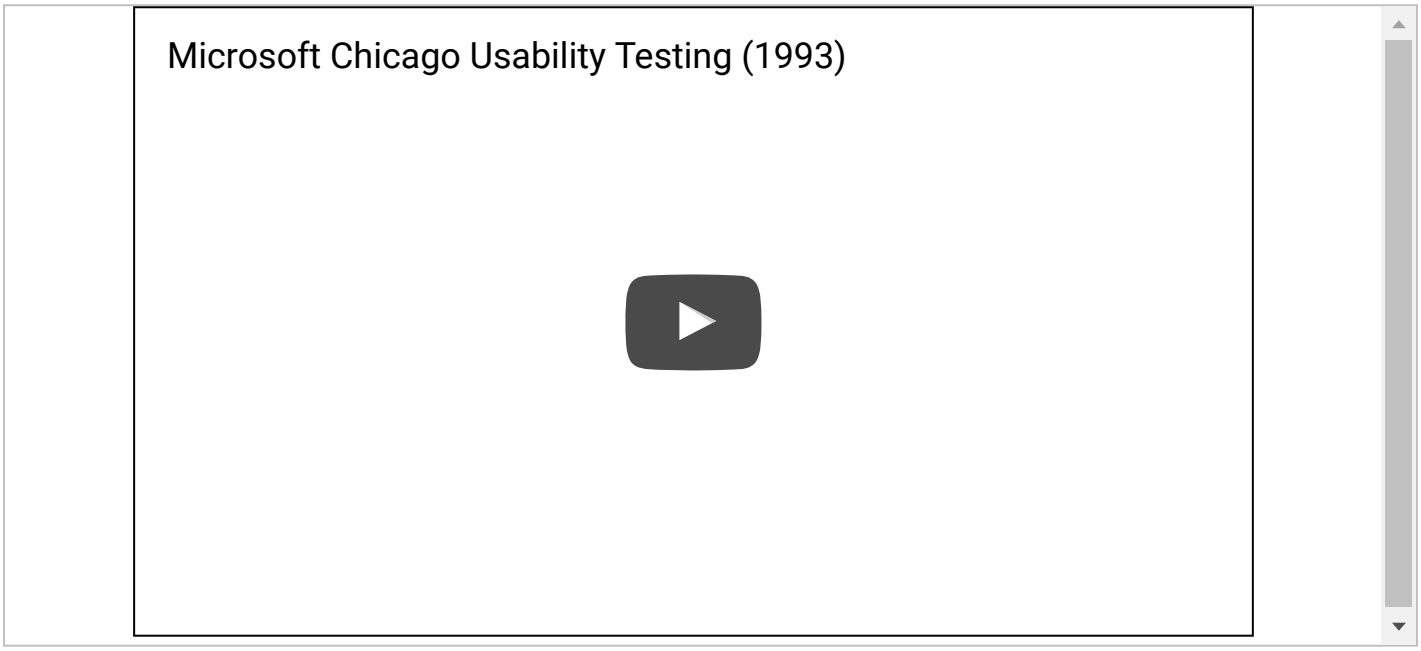
Testing and Evaluating Usability

user observation

- involves testing sessions to observe users operation and reaction to an application
- often considered the most effective way to evaluate a design
  - *whether it is actually usable and learnable*
- may quickly reveal where your users are encountering problems
  - *show if results are outliers or common to most users*
- considerations for the testing session may include
  - *where to host the testing environment?*
  - *how to observe each session and its users?*
  - *how to effectively record notes of your users?*
- more formal testing lab or less formal local environment
  - *try to avoid 'Big Brother' type scenario, create a familiar environment*
- possibly test your app in situ
  - *tourguides whilst conducting a test tour...*
- recording users actions and thought processes whilst performing tasks
  - ***think aloud protocol***

Video - Testing and Evaluating Usability

usability testing Windows 95 in 1993



- Source - Microsoft Usability Testing - YouTube

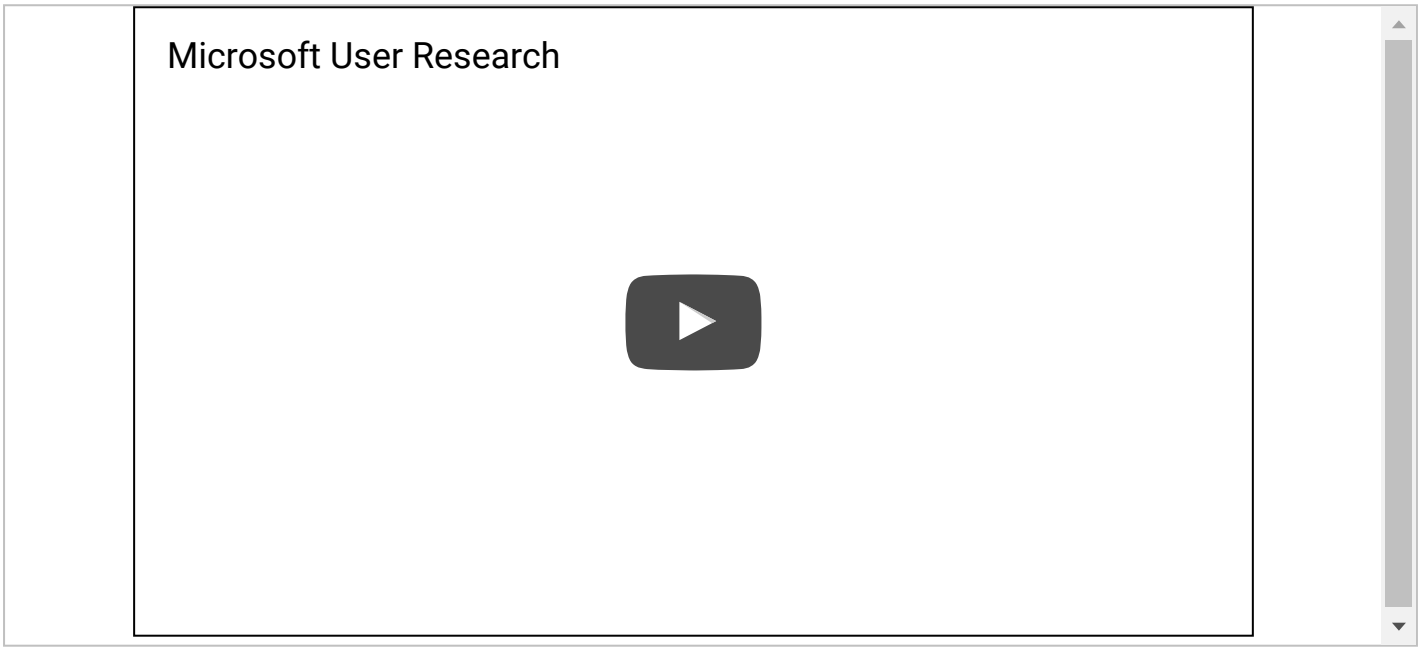
Testing and Evaluating Usability

user observation

- be clear with users what you are trying to achieve in the test session
- ensure user consent and agreement for recorded sessions
- pattern and format of testing session influenced by type of collected data
- standard pattern often emerges for test sessions
  - *we ask test users to accomplish one or more goals*
  - *then observe how they interact with and explore the app to achieve those goals*
- how much help and assistance to offer to users?
  - *avoid trap of **leading** users to complete goals*
- carefully consider test results
  - *not all recommendations need be incorporated in final design*

Video - Testing and Evaluating Usability

a quick history of usability testing



- Source - Microsoft User Research - YouTube

Testing and Evaluating Usability

user observation

- ask test users to complete a quick survey or questionnaire on the testing session
  - *helps inform future test sessions*
- collate our notes and recordings from the test session
  - *review where applicable*
- review test results as well
- calculate any defined test metrics
  - *compare statistics, if available, with any previous testing sessions*
- such analysis allows us to identify problem areas
  - *helps to recommend possible solutions for an updated application design*
- produce a brief report of the test session
  - *summary of test results etc*
  - *set of recommendations for the application's design*

Video - Testing and Evaluating Usability

touring a Usability Lab

- Source - Touring SOE's Usability Lab - YouTube

Testing and Evaluating Usability

cognitive walkthrough

- technique defined by **Wharton et al** in 1994.
- effective way of recognising and detecting various types of usability defects
- technique developed as a less involved option compared to user observation sessions and testing
- may be equally conducted by a single evaluator or a within a group setting
- to conduct a **cognitive walkthrough**
  - *select a task scenario, eg: a typical goal that a user may have in the application*
    - carefully outline actions required to complete tasks necessary for the defined goal
    - actions typically optimal sequence for an average, intermediate user
    - alternative sequences may be worth evaluating in separate test scenarios
  - *select a user profile for the test*
    - begin role-playing as a member of this user group
    - test the application scenario as a user for the first time
  - *step through the defined sequence of actions*
    - carefully inspect the application or prototype with questions and checks
    - consider each question in the role of the defined user profile
    - questions based upon concept that users learn by trial and error...
    - questions also test how well user can interpret and learn each step
    - answers to questions may reveal weaknesses or opportunities to improve application

Testing and Evaluating Usability

cognitive walkthrough

- Wharton et al originally recommended four primary questions for the **cognitive walkthrough**
    1. ***Will the users try to achieve the right effect?***
    2. ***Will the user notice that the correct action is available?***
    3. ***Will the user associate the correct action with the effect to be achieved?***
    4. ***If the correct action is performed, will the user see that progress is being made toward solution of the task?***
- some evaluators prefer to focus solely upon questions 2 and 4
- perceived limitations include
    - *does not test the interface with real users - may lead to false assumptions by evaluators compared to users*
    - *evaluators may find an unusually high number of defects and issues*
    - *may be disproportionate to actual issues perceived by a real user*
    - *technique often favours ease of learning for beginners over options and efficiency for experienced users*

Testing and Evaluating Usability

analytics

- monitor application's performance in real day-to-day use
- **analytics** allows developers to monitor data on usage statistics
  - *analyse data to detect and predict potential patterns, trends, preferences...*
  - *eg: validation of design decisions, assumptions, choices...*
  - *help determine usage for app functions*
  - *identify problem areas, interaction issues, bugs, slow working methodologies...*
- example collected data can include
  - *time spent per usage session - averages, longest, shortest, frequency of visits...*
  - *recurring errors and bugs within the app*
  - *regularly used functionality, common interaction elements, menu items, popular shortcut combinations, general viewing habits*
  - *popular places visited, including pages, tabs, screen sections, including time spent*
- analytics can be applied for many different application types
  - *desktop, web, mobile, server...*
- other features can include
  - *contextual and geographic data, frequency of visits, visit repetitions, search terms...*

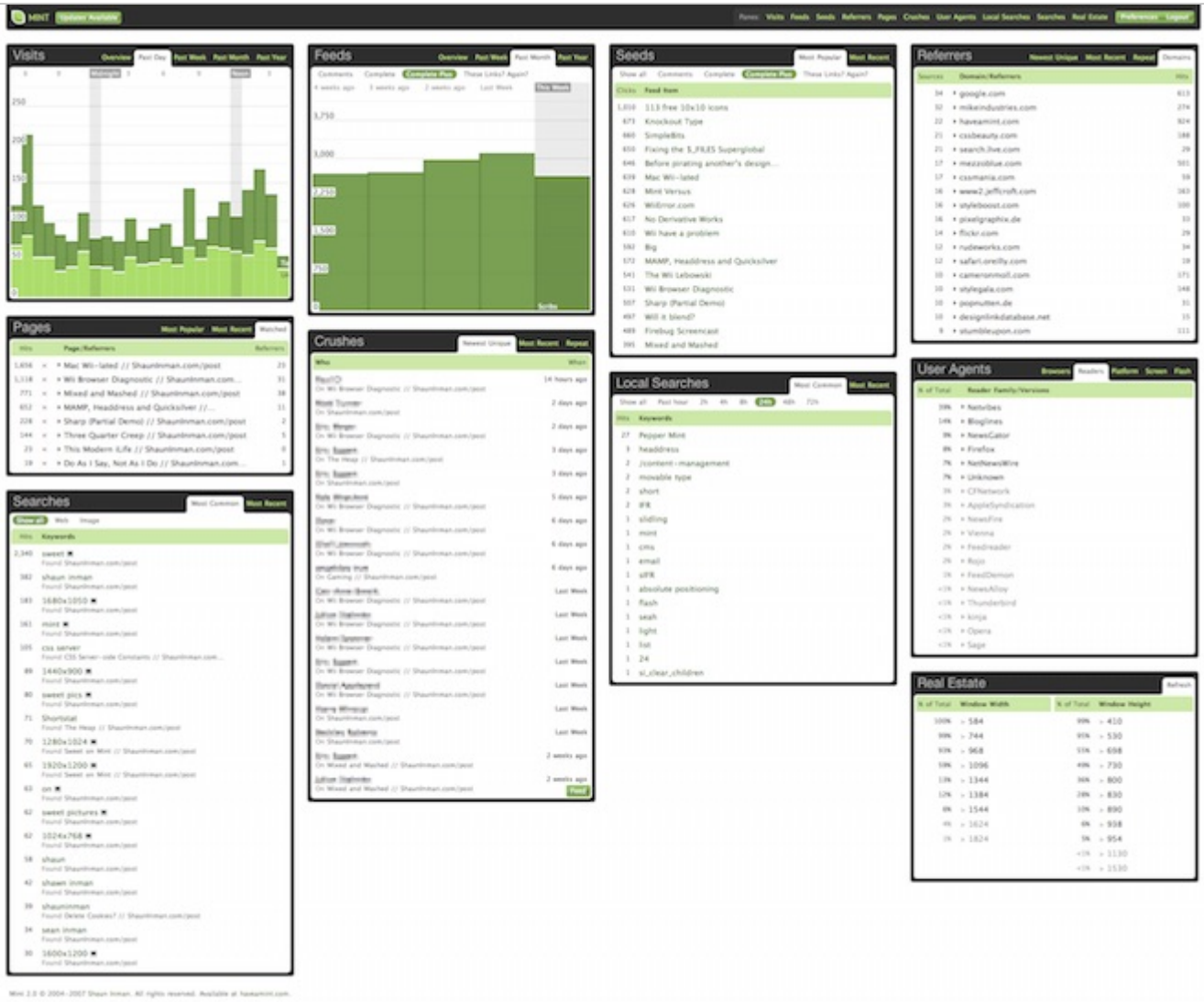
Testing and Evaluating Usability

analytics examples - web analytics

- different forms including self-hosted server-side solutions to online services
  - *eg: Mint and Google Analytics*
- Mint is a self-hosted application
  - *monitors and records site activity, including overall visits*
  - *referrers to your website, common searches*
  - *most popular and recently accessed pages, user agents, and much more...*
- Google Analytics offers a hosted solution for web and mobile applications
  - *monitor and check advertising performance*
  - *check site content, audience data...*
  - *browser and OS statistics*
  - *flow through a site or app*
  - *location specific data, sources of traffic, social reports...*
- useful feature of Google Analytics is option for **content experiments**
  - *compare performance of different web pages or application screens*
  - *use random sampling, define percentage of user to test*
  - *choose required objective for testing*
  - *get regular updates on the performance of the experiment*

Image - Testing and Evaluating Usability

Mint analytics

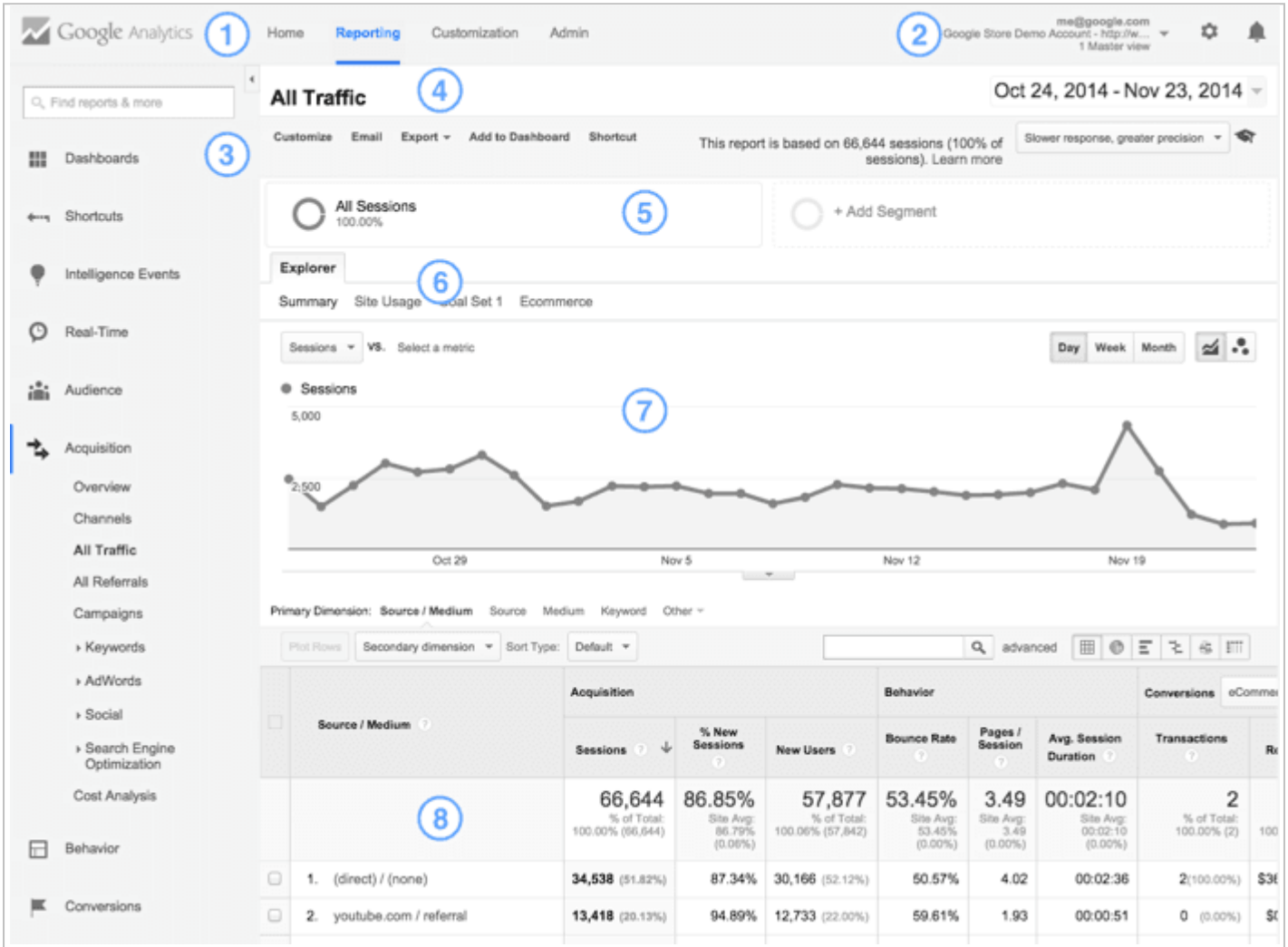


Mint Analytics

- Source - Mint | Live Demo

Image - Testing and Evaluating Usability

Google analytics

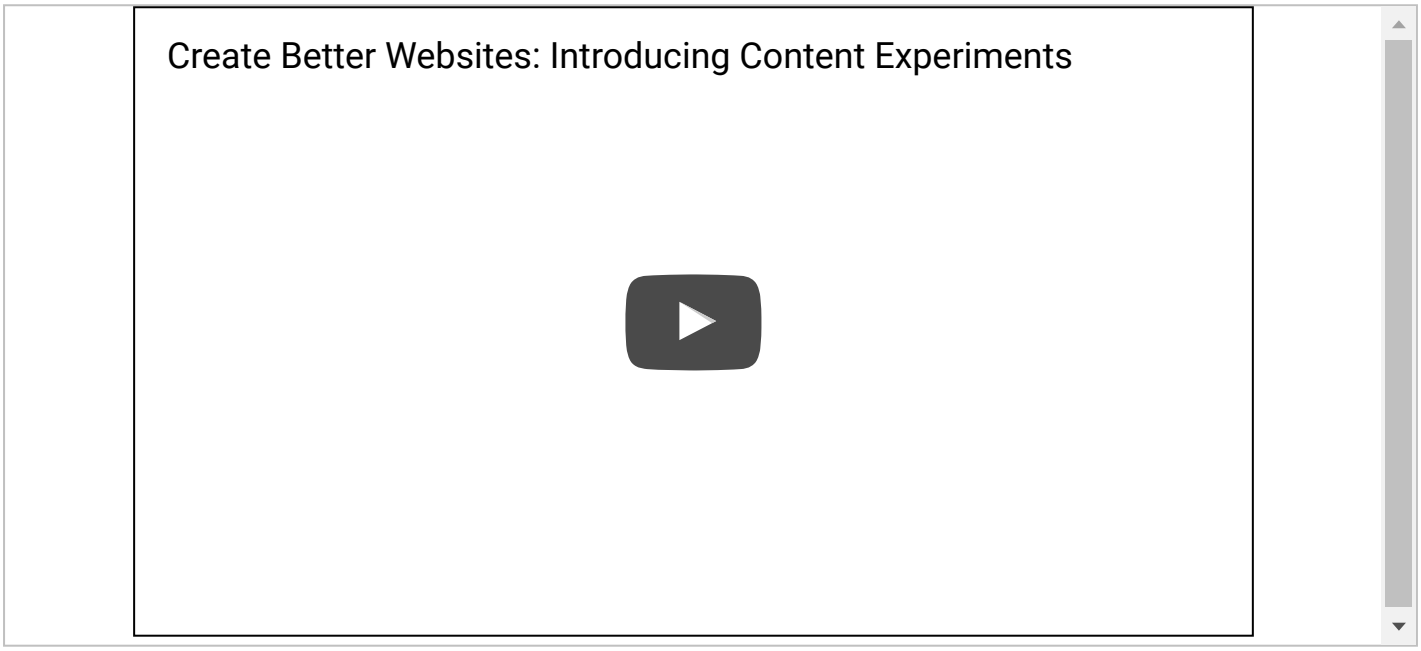


Google Analytics

- Source - Google Analytics

Video - Testing and Evaluating Usability

content experiments in Google analytics



- Source - Create Better Website: Introducing Content Experiments - YouTube

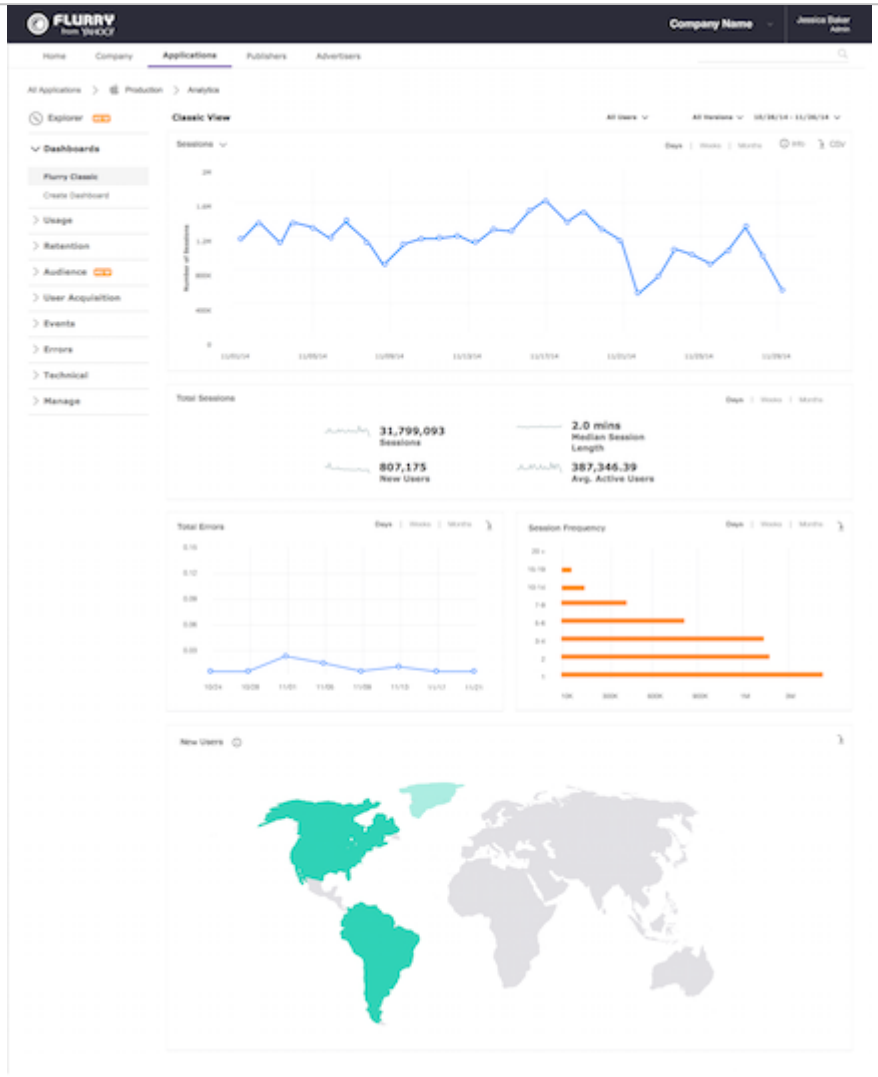
Testing and Evaluating Usability

mobile analytics

- Google Analytics also provides mobile statistics and analysis
  - *developers can learn*
  - *who is using their apps, on which devices, geographical locations...*
  - *includes integration with Google Play*
  - *learn how a user discovered an app, the path that led to a developer's app*
  - *includes real-time analytics to show how users actually use an app*
  - *event tracking, application flow*
  - *visualisation show places and user interaction, commonly used features...*
  - *developers can also learn about application crashes, bugs*
  - *help determine isolated and recurrent errors and bugs*
  - *set goals for analysis of an application*
  - *track purchases, user clicks, click rates and conversions*
  - *e-commerce tools allow tracking of real or virtual goods*
- Apple's **App Analytics** for iTunes Connect
  - *use iOS SDK for Google Analytics*
  - *Flurry by Yahoo*

Image - Testing and Evaluating Usability

Flurry mobile analytics



Flurry Mobile Analytics

- Source - Flurry Analytics

Testing and Evaluating Usability

questionnaires

- questionnaires still useful under the right circumstances
- allows us to guide test users through a series of questions and survey points
- primary benefit can be control over test parameters and required responses
  - *inherent option to open questions and feedback to broad responses*
- use feedback questions to calculate limited quantitative data
  - *collect responses to boolean questions*
  - *ask participants for a numerically based satisfaction score*
    - standard **Likert** scale - 1 to 10
  - *then calculate the average of the returned results*
- numerical responses useful when considered over multiple product iterations
  - *compare and contrast each iteration's results*
  - *determine if bugs, issues, design flaws continue per iteration*
  - *track satisfaction patterns as well*
  - *changes per iteration may not always be perceived as positive by users*

Testing and Evaluating Usability

heuristic analysis

- heuristics are a set of **rule of thumb** principles or guidelines
  - *may help guide or influence our decision making for design and development*
- inherently broad in scope and terms, may be perceived as difficult to specify precisely
  - *assessing heuristics is inherently a subject decision*
- conducting a **heuristic evaluation**
  - *Jakob Nielsen, 1994*
- benefit is quick, inexpensive, and often remarkably effective testing
- useful initial check of an application
  - *helps identify problems, issues, potential defects, oversights...*
- predicated on the assumption of underlying expertise in usability, interaction, design...
  - *may be helpful to co-opt a group of testers and compare results*
- define heuristic rules, then define series of potential user scenarios
  - *work our way through each scenario checking defined rules...*

Testing and Evaluating Usability

heuristic analysis

- Jakob Nielsen introduced the concept of **heuristic evaluations** in 1994
  - *defined ten general rules to consider in such evaluations*

    1. visibility of system status
    2. match between system and the real world
    3. user control and freedom
    4. consistency and standards
    5. error prevention
    6. recognition rather than recall
    7. flexibility and efficiency of use
    8. aesthetic and minimalist design
    9. help users recognise, diagnose, and recover from errors
    10. help and documentation

Further details

Testing and Evaluating Usability

heuristic analysis

- heuristic evaluation creates a list of potential usability issues, problems, and potential oversights
- inherent weakness is the use of usability experts and not real users
  - *becomes difficult to abstract from domain knowledge*
  - *responses to evaluation tempered by pre-existing knowledge*
- consider such heuristic evaluations as potentially biased, skewed, or based upon incorrect user assumptions
- heuristic evaluation is still a very useful initial testing method
  - *combine with other testing options and tools*

References

- Nielsen, J. *Heuristic evaluation.* Usability inspection methods. New York. John Wiley and Sons. P. 30. 1994.
- Shackel, B. *Usability - context, framework, design, and evolution.* Human factors for informatics usability. Cambridge University Press. PP. 21-38. 1991.
- Wharton, C. et al. *The cognitive walkthrough method: A practitioner's guide.* Usability inspection methods. New York. John Wiley and Sons. PP. 105-140. 1994.