CSC1016S Human Computer Interaction

* **Human Computer Interaction** HCI is a field of Computer Science that deals with making computer systems discoverable, learnable, effective and pleasant to use
* Involves making them more accessible, not just to specialists
* Number of skills from different disciplines are required:
  + Computer science
  + Cognitive science/psychology
  + Ergonomics
  + Business processes (must attract customers from competitors)
* Parts of HCI:
  + User experience (UX)
  + Human-centred design
  + Computer-supported cooperative work (work together with other programmers)
  + Visualization (e.g. of data)
  + HCI for development (to help with social issues)
* Programs should be designed in a way to reduce user errors – they are also to blame

# User Experience Design

* **User experience design** is the process design teams use to create products that provide meaningful and relevant experiences to users
* Deals with all aspects of the end-users’ interaction with the company, its services and its products
  + Involves the way the user will interact with the system, navigate their way around it following cues that guide them to their goal, with the right things visible to them, making them feel good about the experience
* **User interface design** is about how a product looks and its visual elements
  + Colours, buttons, icons, animations, layout
* Part of the overall product experience (UX), but is a subset

# User-Centred Design

* People that are good at programming tend to have bad instincts for what users actually want
* UCD is an iterative process in which designers focus on the users and their needs in each phase of the design process
* Design teams involve users throughout the design process, asking them questions and gaining an idea of their needs
* With an **expert** mindset, users are seen only as informants reacting to the designs of the experts
* With a **participatory** mindset, users co-design
* **Interaction design**: the design of spaces for human communication and interaction
  + About understanding dynamics of how we interact with computing devices
* High-level goals of UCD – make programs:
  + Easy to learn
  + Effective to use
  + Enjoyable
* Key performance indicators
  + User retention
  + More clicks/screen time
  + More ad views
  + Making more money!
* Overall approach
  + Observation
  + Inquiry
  + Analysis
  + Prototype
  + Repeat

## How to Find Users to Study

* Understanding users
  + Who will be the users you study? – sample must be **representative**
  + How will you recruit them?
  + Where will you do the study? – should do it in the environment where people will likely use the app (e.g. loud vs quite, office vs home)
  + What will you do with them?
* Different sampling approaches
  + **Random sampling** – using a random function to select participants from the entire population
    - NOTE: to call it random sampling, it has to be statistically random (e.g. random number generator)
  + **Convenience sample** – whoever is the most accessible
  + **Snowball sample** – participants are asked to help identify other participants and so on in a chain
  + **Stratified sampling** – dividing subjects into subgroups then sampling each of these separately using another method
  + **Purposive sampling –** intentionally choosing participants based on their specific characteristics
  + **Referral sampling** – choosing participants through an **intermediatory** who refers researchers to these participants
* Key informants of who can tell you what people to involve
  + **Field guides** are people who know the individuals, groups & organization well
    - Well informed enough to tell you who else to involve
  + **Liminals** are authorities accepted by most people
  + **Corporate informants** are the people who want the app to be developed
  + **Exceptional informants** are the gurus on the topic

## Empathy in UCD

* Empathy is important when users may be different to you
* People have various biases:
  + **Availability bias**
  + **Confirmation bias**
  + **Status quo bias** – reluctant to change if lots of work has already been done
  + **Ikea effect** – become attached to something because you made it

## How to Gain Insights

* Observation – do users achieve goal that app is meant to achieve?
  + Start by seeing how people do things without your app
  + Once you have the initial design, get them to do the same task
  + Often what we say isn’t what we would do
* User interviews – find out feelings/unspoken thoughts
  + Why did something take long?
* Contextual interviews – talk to people in the environment they will use it
* Usability – give target users tasks to do

## Getting the Best Insights

* Do not ask very high-level questions – GIGO (garbage in, garbage out)
* People will not put time into giving good answers
* Ask about specific events
* Ask open ended questions – learn more about the person
  + May have to prompt the person if they are only giving 1-word answers
* Ask about the emotional impact
* How many interviews are enough?
  + Must ask at least 5 people, unless there are different subgroups (then you need more)

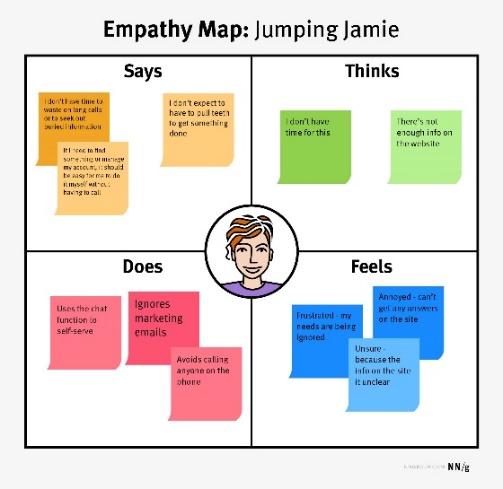
## Make Sense of the Findings

### Affinity Sorting

* Write sticky notes for each insight from each user
* Put on a wall
* Combine similar ideas
* Include which user you got the info from in case you need to ask them about it

### Empathy Map

* Says – what people said in the interview
* Thinks – what people thing through your observations
* Does – what they have done (mistakes, took a long time)
* Feels – how people feel (confused, stupid cause stuck)



# Standards and Guidelines

* **Guideline**: vague, need to know theoretical underpinning, should consider
  + Lower authority
  + Conflicts/overlap/trade-off with standards
  + Less focused
  + Interpretation required
* **Standard**: more specific, must do if you want your thing to work with others
  + Higher authority
  + Little overlap
  + Limited application
  + Minimal interpretation
  + Example: file name extensions
  + Large communal following leads to better technology

## Golden Rules of Design – Ben Sneiderman

NOTE: these are guidelines

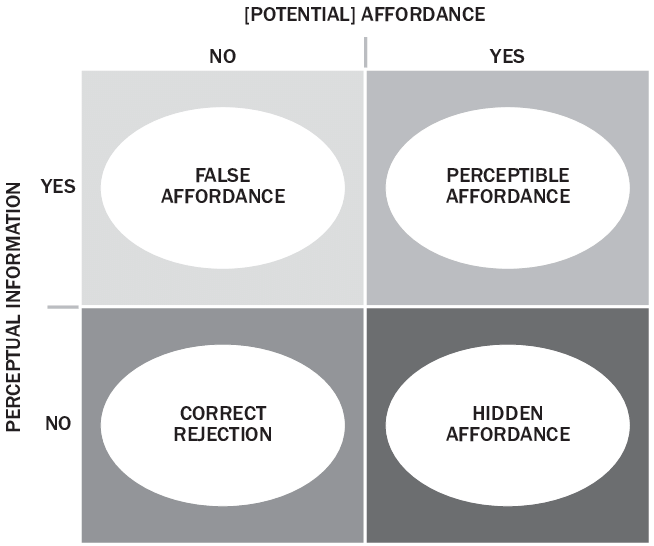
1. Strive for consistency
2. Cater to universal usability
   * Includes disabilities, languages, differences in skill levels (e.g. including explanations, shortcuts for advanced users)
3. Offer informative feedback
4. Design dialogs to yield closure
   * Sequences of actions should be organised into group, with feedback upon the completion of each group
5. Prevent errors
   * Includes data validation, grey out menu items,
   * If users make an error, there should be simple instructions for recovery
   * Erroneous actions should leave the interface state unchanged, or the interface should give instructions about restoring the state
6. Permit easy reversal of actions
   * Relieves anxiety and encourages the exploration of unfamiliar options
7. Support internal locus of control
   * Experienced users strongly desire the sense that they are in charge of the interface and that the interface responds to their actions
   * Do not want changes in familiar behaviour or inability to produce their desired result
8. Reduce short-term memory
   * Avoid interfaces in which users must remember information from one display and then use that information on another display

## Everyday Design Guidelines

### Affordances

**Affordances** are the potential actions and interactions that the environment offers

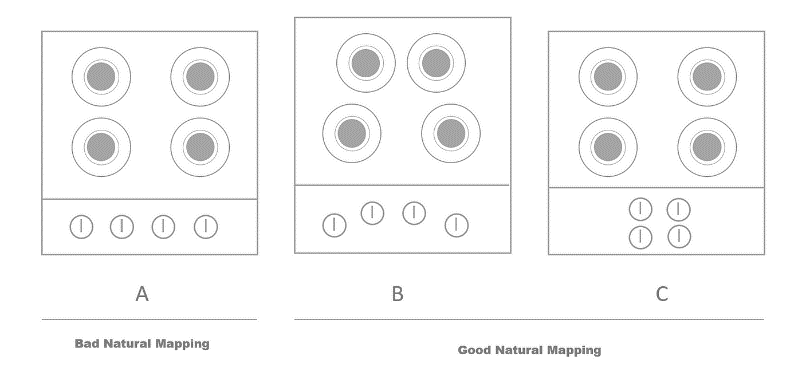
**Perceived affordances** of an object are those properties of the object which give users clues as to how the device is used



* False affordance – looks like you can do something, but you can’t
* Hidden affordance – there is something you can do, but it is not shown to you
* What you see as possible should be what you can do with the object/interface

### Mapping

* Ensure a natural correlation between objects and the interface controlling them
* **Mapping** associates a potential action with a particular reaction - should be able to predict the reaction of your action
* Should not have to give instructions – should be clear



#### Control Simplicity

* Should aim to minimize complexity and cognitive load for users
* Should use symbols that people are comfortable with (e.g. from when stuff was done manually – pause button)
  + **Skeuomorph**: when a derivative object retains ornamental design cues that were necessary or essential to the original
  + Allow mappings and affordances to match things people are used to from outside of the IT world
  + Many people are against this – designers should not be stuck in the past
  + Should not be taken to the extreme – can lead to bad UI design

#### Mapping and Modality

* Good buttons will sometimes show the current state – mute button

### Constraints

* **Constraints:** control a design so that it can only be used in the right way
* Involves using components like drop down lists, greying out options which cannot be used

## Cultural Dependencies

* Affordances and mapping both suggest how to use the object
* Can depend on:
  + Experience
  + Knowledge
  + Culture

## Interface Design Guidelines

### Usability Heuristics - Nielsen

Design principles or “rules of thumb” to evaluate a design

Often get a HCI expert to evaluate your interface with regard to these usability heuristics

* System status should always be valuable
  + Example: progress bar
* User control and freedom
  + Support undo, redo and exit points to help users leave an unwanted state
* Aesthetic and minimalist design
  + Don’t show irrelevant or rarely needed information since these extra elements diminish the relevance of others
* Flexibility and efficiency of use
  + Make the system efficient for different experience levels through shortcuts and advanced tools
* Help and documentation
  + Make necessary help easy to find and search, without technical terms
* Match between the system and the real world
  + Use real world words, concepts and conventions familiar to users
* Error prevention
  + Check before users commit to actions
* Consistency and standards
  + Follow platform conventions
* Recognition rather than recall
  + Make objects and actions visible at the appropriate time to minimize users’ memory load
  + **Breadcrumbs** – show the path of where you have navigated
* Help users recognise, diagnose and recover from errors
  + Express error messages in plain language to indicate the problem and suggest solutions

### More Design Guidelines

Do not need to memorise these, but rather be able to identify them

* Knowledge and chucking
  + To improve memory we chunk actions
    - Group them into a chunk
    - Seek meaningful relationships
    - Example: 082 856 9582
* 80/20 Law – Pareto’s Principle
  + 20% of functionality will account for 80% of usage in a menu
* Principle of least effort
  + Make frequent things easy, unlikely things harder
  + Should be implemented in menus and with ‘dangerous’ options
* Principle of least astonishment
  + The design should match the user’s experience and expectations
* Aesthetic usability effect
  + Things that look better perform better in usability tests
  + Apparent usability is different to inherent usability
  + Often has nothing to do with usability, but rather attractiveness
* Modes
  + Modes allow different behaviours from the same interface options (e.g. caps lock)
  + Current state should be obvious
* Feedback
  + Every user action needs to show evidence that it has been complete
  + Causes issues regarding complex processing – should we show quickly feedback before or slowly after processing?
* Equal opportunity
  + Should be no difference between input and output values
  + Examples: sliders and dials – shows current value and allows you to change it
* Fitts’s law
  + Time taken to reach a target is a function of the distance to and size of the target
  + Bigger and more central things are easier to use
* Hick’s law
  + The time taken to reach a decision goes up as the number of choices increases
  + Apps and websites should be customisable so that you can choose what items are easily available to you
* Single to noise ratio
  + Should not make things more complicated that they need to be as it can distract the user
  + Too much density leads to degradation of viewer interest
  + Example: 3D chart

# The Design Process

Design & Analysis -> Prototype -> Evaluate -> REPEAT

#### Define the problem / goal

* Choose the appropriate framing:
  + Instead of “bicycle cup holders”
  + Should be “helping cyclists to drink coffee safely”

#### Understand your users

* Create specific personas as there is no one general type of user
* Every persona has different goals
* Example: designing a car
  + Alex wants to go fast and have fun
  + Abby wants to be safe and be comfortable
  + John wants to haul big loads and be reliable

#### Brainstorm

* Can be between team members, or with potential users
* Aim for quantity, hope for quality
* Guidelines:
  + One conversation at a time
  + Stay focused
  + Encourage wild ideas
  + Defer judgement
  + Build upon ideas from others

#### Task and User Flow

* **Task flow**: series of steps to get a job done
* **User flow**: a user story/path that a specific user (persona) follows, including their decision points
* Helps you to understand:
  + Who the user is
  + What they are carrying in
  + What is the best outcome

#### Sketch the User Interface

* Focus on the idea, not the detail
* Don’t think, just draw
* Crazy 8s sketch – draw 8 different designs, with a 1-minute timer for each

#### Prototypes

* Once you have your ideas, build a prototype
* Invokes building something tangible to get feedback
* Experiment with alternative designs (cheaply)
* Fix problems with the concept before code is written
* **Low-fi prototype** is a low effort, low detail version that shows the idea
* **High-fi prototype** is more polished and gives a better representation, but may not have been coded yet

#### Constructing a Prototype

* Common method is drawing out interfaces and using a cardboard cutout of a phone to show what is visible
* Important to keep the same aspect ratio
* There will be different pieces of paper for every time something changes on the screen
* Can create an interactive digital prototype in software like Figma or PowerPoint

#### Why Not Just Hi-Fi?

* Formal representations indicate the finished nature
* People comment on colours, fonts, alignment
* It discourages major changes
  + Testers don’t want a finished design
  + Designers don’t want to lose effort put into the design

# Qualitative Approaches

* Identify values, goals and actions
* Understand the context
* Perform:
  + Interviews
  + Contextual inquiry
  + Cultural probes
  + Focus groups workshops

## Inquiries

* Contextual inquiry
  + The designer watches the participant begin an activity
  + The designer interrupts to question the reason behind the activity
  + The participant provides commentary, explaining while doing
* Diary studies
  + Have participants record their activities at a regular interval
  + Done when you can’t speak to the person face to face (e.g. if they are in hospital, geographically far)
  + Can be automated using software on phones or interactive
* Interviews
  + Structured / semi-structured (guiding questions) / unstructured (no preparation)
  + Prepare interview questions

## Cultural Probes

* Useful when you can’t get hold of users face to face
* Can ask them to take photos/postcards of the problem/process/solution

## Focus Groups & Workshops

* Choose your participants
  + 3-7 for focus group
  + More for workshop
* Prepare a guide
  + Outline questions and activities
* Choose your location
  + Consider where they will feel comfortable
* Should be 30m – 4hr and lead by a trained moderator
  + Workshop may be up to several days
* Follow up with analysis and written report

### Focus Groups Moderator Tips

* Observe/watch other moderators
* Facilitate natural conversation & make people feel comfortable
* Dig deeper, ask open ended questions
* Direct participation -> provide opportunities for everyone to have a voice
  + Some people are shy, others dominate the conversation
* Moderate, don’t participate
* Don’t push your own point of view

## Analysis of Observations

* Transcribe audio to text
* Categorise and organise (coding, such as affinity maps, annotations)
* Reflecting and interpreting
* Making consumable (i.e. implications for design)

All time consuming!

One person cannot find all the problems

# Conducting Evaluations

## Genres of Assessment

* **Automated** – usability measures computed by software
* **Empirical** – usability assessed by testing with real users
* **Formal** – Models and formulas used to calculate measures
* **Inspection** – consult experts, based on heuristics, skills & experience

Traditional qualitative measures are expensive and time consuming

## Discount Usability Techniques

Look at most commonly used functions of the app (80/20 rule)

#### Cognitive walkthroughs

* Detailed task with a clear goal
* Take a handful of people
* People must talk how it goes and what they’re thinking
* **Wizard of Oz approach** is when there isn’t actually an app, but someone with pieces of paper simulating the application
  + Need 1 person as computer and 1 as note takers
  + Can still see what users are thinking and where they are going wrong

#### Heuristic Evaluation

* Use Neilson’s heuristics and give the app to experts
* 3-5 expert evaluators examine the UI
* Different evaluators find different problems
* Can communicate afterwards to aggregate findings

1. Pre-evaluation training

* Give the experts context and a list of heuristics for reference

2. Evaluation

* Work in 2 passes – first get a feel for the flow, then focus on specific elements
* Each evaluator produces a list of problems

3. Severity rating

* Rate problems individually, then in a group

0 – violates heuristics, but not a usability problem

1 – cosmetic, fix if you have time

2 – minor, low priority

3 – major

4 - catastrophic

4. Debriefing

* Evaluators discuss outcomes with the design team
* Minimize criticism
* Suggest potential solutions
* Assess difficulty of fixing problems

#### Crowdsourcing

* Online, remote experiments using something like Mechanical Turk
* People give use the app then feedback

#### System Usability Scale (SUS)

* Ten question assessment
* Not diagnostic
* Recognise as the 10 questions to ask, but it is geared towards English speakers (not good for SA context)

### Pros and Cons

|  |  |
| --- | --- |
| **Heuristic Evaluation** | **User Testing** |
| 1 – 2 hours per evaluator | Days or weeks of working with users |
| Findings are straightforward | Requires interpretation of actions |
| May miss problems or find false positives | Far more accurate as it incorporates real users |