Goals of HCI

- Improve **Usability** -> make software easier to learn and use
- Improve the **Effectiveness**-> make software better at doing the work it does
- Improve task **Efficiency**-> improve efficiency of the user's achievement in their task
- Improve **Safety**-> A well designed interface reduces errors.
- Improve **Functionality**->make system better equipped with tools needed for tasks

Usable tools

- The software being used for a reason
- All software must be used in a way it can help the user
- Should always act as an aid, like a tool that can be used for a purpose.

5 Attributes

- **Learnability**
  - Ease of learning for novice user
    - Measured by Time to perform certain tasks
- **Efficiency**
  - Steady state performance of expert users
    - Measured by Time and performance of expert users to perform tasks
- **Error Rate**
  - Error rate for minor and catastrophic errors
    - Measured by Count minor and catastrophic errors made by users while performing tasks
- **Memorability**
  - Ease of using the system intermittently for casual users
    - Measured by time to perform tasks after coming back after a specific amount of time
- **Subjective Satisfaction**
  - How pleasant the software is to use?
    - Measured - Asking users opinion after trying the software and performing tasks

Usability Metrics

- **Satisfaction**
  - Looks at user's overall opinion of system
  - One of the most important elements of usability
  - Can influence the decision
    - Users opinion is given-they can dictate how it can be improved
- **Effectiveness**
  - Making the software better at what it does
  - Increases the likelihood that the user will get the task done
  - Can influence the decision
    - Asks question: does it do what it is meant to? How can it be improved?
- **Efficiency**
  - Performance of expert users.
  - Outlines the time taken to perform more complex tasks
  - Can influence the decision
    - Feedback can be provided based on time taken to perform certain tasks

Usability Lifecycle

Requirements Gathering

- Competitive analysis, Context of use

Design

- Task Analysis, Prototyping

Usability Evaluation

- Heuristic Evaluation, Cognitive Walkthrough, User Testing

Nielsen's Heuristics

- **Visibility of System Status**
  - The system should always keep the user informed about what is going on through feedback within reasonable time e.g. progress indicator etc.
- **User control and freedom**
  - Users often chose system functions by a mistake and need a clearly marked exit to leave unwanted state without having to go through an extended dialogue. It should support undo, redo and cancel operations.
- **Consistency and standards.**
  - Users build up expectations about where things are and how things are done from many sources. Consistency in software, across all software products, between versions etc.
- **Help and documentation**
  - Help should be obviously available in the right context, be easy to search, be focussed on the user's task and list concrete steps to be carried out.
- **Recognition rather than recall**
  - Make actions, objects, and options visible. The user should not have to remember information from one page to another. Instructions for system should be visible and easily readable.
  - Mind is in the world - Counting on fingers. Forms remembering data.
  - Command line interfaces are based on recall – GUI based on recognition.
- **Aesthetic and minimalist design**
  - Dialogues should not contain certain information which is irrelevant or rarely needed. Every extra bit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility. An example includes looking at form data that provides too much data to fill in or even looking at a website that is too cluttered.
- **Match Between System and Real World**
  - System should speak the user's language, with words/phrases/concepts that are familiar to the user. It should follow real world conventions, making information appear in a natural and logical order
- **Error Prevention**
  - Even better than good error messages are a careful design which prevents a problem from occurring in the first place.
- **Flexibility and efficiency of use**
  - Support redo, accellerators and shortcuts. Avoid forcing user to repeat themselves or take some long route to their goal.
- **Help users recognise, diagnose, and recover from errors**
  - When problems occur, users should be informed in a way they will notice and in a way that will allow them to do something about it

Dix – involving users

Just because the developer can use the software, doesn't mean that the use can.

They are different people with different backgrounds. Just because a developer has a technical background, it doesn't mean that the user will. This ensures that the software is built based on the basic user's skills and experiences.

- The user knows the tasks that need to be completed. A developer may not be familiar with the domain that they are developing for. The user can provide them with basic information that they need to fulfil the requirements for the software.

Give and explain two reasons why involving users in the evaluation may be problematic.

- Having the user involved in the evaluation may be problematic as time is needed. The user and the developer may eat into the implementation stage using their meeting times. The developer will have less time to make the software.
- Cost. If the time is running out or taking longer than expected due to the meetings between the user and the developer, extra costs may be used. Especially for developer's time, hosting etc. The user may have to pay more than expected for the application.

Cognitive load

Info and interactions involved in learning it. (Hollander). Working memory

- **Intrinsic**
  - Level of difficulty of the task. Effort involved in learning it.
- **Germane**
  - Learning it. Metal effort of creating connections. Practice by repetition. Learn CTRL + C and
  - Shit that's helping you learn it. Context menus, underline shortcuts.
- **Extraneous**
  - **Caused by poor ICL**
  - Distractions. Cluttered interface. Way info is presented. Describe triangle. Form info

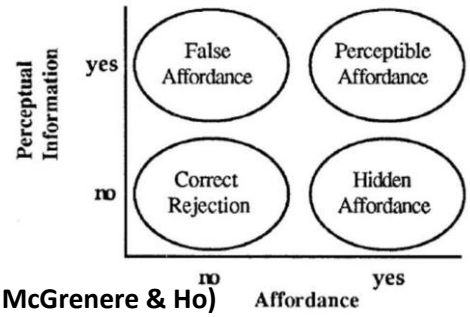
In relation to the concept of usability, ECL is used based on the user's attention. I.e. the user's attention is needed to learn how to use the software. GCL can be used when teaching the user how to perform different operations within the system.

Points of interest

- **Memory load**
  - Between 5 and 9 pieced of information (7 +-2 theory) Millers paper
  - Chunks – only store 5 and 9 in your head. ((Miller number thing)
  - The amount of information we process is limited.
  - The relationship between the limit to channel capacity and the limit to working memory is not linier.IE. working memory can deal with more information than expected given the limits of out "channel capacity"
  - Chunks limit span of memory.
  - Limit to memory is based on the number of items we take in.
- **Subconscious**
  - Driving, muscle memory. How difficult to carry out a task impact on interface? How much to remember.
- **Attention**
  - Focus on the task not learning the interface

Affordances

Relationship between and agent, an object, and an agent task



Objective (it has affordance because it does) and subjective (depends on agent) (McGrenere & Ho)

- **False Affordance**
  - Occurs where it appears as if something is possible but it did not e.g. handle on a push door indicates it's for pulling when it's not
- **Perceptible Affordance**
  - Apparent properties of objects that indicate the sorts of operations and manipulations that can be done to those objects.
- **Correct Rejection**
  - Occurs where it appears as if an action is not possible but it is not. I.e. no handle on a push door
- **Hidden Affordance**
  - A hidden affordance occurs where it appears as if an action is not possible but it is. I.e. no handle on a pull door

1. **Functionality (link real work (recognition > recall)**
   Volume slider. Designing something so it looks like it does what it supposed to do. B for bold.
   Not a dial because house interface doesn't do that easy (as opposed to hand interface radio car.
2. **Utility (can't fuck up with an and down – error + rest icing user freedom, consistency)**
   Up and down slicer on volume reflects up and down volume. Usefulness of an object. Mute button
3. **Usability (system status)**
   Volume slider gets bigger when raised (also number) gives feedback to user on action. Showing that they are doing it.

Characteristics of Attention

Focussed

- Concentrating on the task at hand
- **Noticing colour of the door**. Focused on the task at hand.

Divided

- Trying to watch many different sources of input

Voluntary

- Turning attention to something.

Involuntary.

- Being distracted
- Flashing icons, cursors

- A powerful filter
- Prone to mistakes
- Heavily dependent on task goals and objectives
- Directed by previous learning
- Distractible

Perception is the process by which we detect and interpret information from the external world by means of the 5 senses.
It we look at perceived affordances, they refer to the actions the user perceives to be possible. Using an example of a chair, we know it is meant for sitting (perception + perceived affordances) but what if it wasn't just for sitting? The user's attention is now on sitting on that chair.

Seeing is a constructive activity where our minds **construct** what is perceived using. Information from the environment (based on sensed data) previously stored knowledge.

Gestalt theory (Construcitvism)

Visual system constructs a model of the world by manipulating information from the sensed data.

- Transforming , Enhancing, Distorting, Discarding

Gestalt theory contributes to the rules of organisation and presentation on screen.

- Information from the environment + Previously stored knowledge.

It assumes that perception involves the interventions of representations and memories. When viewing visual images our perception seems to have in-built standard ways of interpreting what we see i.e. the laws of perception.

- **Proximity**-> See items that are close together or in a group (text formatting buttons + amazon account)
- **Similarity** -> Objects that look similar appear to be grouped (amazon nav buttons)
- **Closure**-> Assume that unfinished known shapes are finished. Whitespace. Newspaper columns + IBM logo
- **Continuity**-> Assume that a partly hidden square is infect square we just can't see it all (slider + rows)
- **Symmetry (Connectedness)**-> Fundamentally to how we see the world. We assume that the space between two symmetrical borders is somehow more of a thing than the spaces to either side. Intersecting shapes?

Ecological Theory

How socio economic factors affect perceived affordances?

Example – We know from experience to target attention on the top right for search controls.

It uses the concept that seeing involves searching. Perception is a process of distinguishing the features of a rich input, not of enriching the data of meaningless input. It looks at perception in comparison with human attention. Thus, it looks more at visual information.

An example looks at affordances. Affordances are the range of possible actions by a user on an object. Look at the chair example, what actions could be performed on a chair? Naturally we assume that we can only just sit on it. The information suggests that we can only just sit on the chair rather than stand on it.

Context of Use

Who, what and where?

- **User analysis**
  - User role -> what role does the user type play in the system – Indirect, direct, supporting, monitoring
  - User skills and knowledge -> User experience, education level, qualifications, training computer experience etc.
  - User physical attributes -> age, gender, physical characteristics etc.
- **Tasks**
  - Should a person define a lot of small tasks or a few big ones
  - Used to clarify nature of sub tasks that make up the big tasks
  - Things considered about each major task are:
    - Task objective, task execution constraints, task flow, task demand on users, task safety
- **Environment**
  - Physical Environment + Technical Environment (Warehouse scanner)

Interface Metaphors

Metaphors are used to help people understand new ideas in terms of familiar ideas. With a good metaphor, a lot of what you know about familiar domain is true in the new concept. E.g. the nucleus is the sun; the electrons are the planets. The planets spin around the sun. Objects are mapped to their counterparts in the familiar domain and thus understood. Similarly, relationships between object are mapped. The mapping allows us to carry knowledge of the familiar into the unknown.

Early Breakdown

Early breakdown refers to the generation of interface metaphors from the beginning. The steps are as follows:

- Identify metaphors that are implicit in the problem definition
- Use these to identify real world objects, structures, or institutions which are characterised by the same sort of functionality.
- Evaluate the potential of these according to the following 5 criteria
  - Amount of structure
  - Applicability of structure
  - Representability
  - Suitability for audience
  - Extensibility

Erickson's Interphase Metaphors

- Understand how the system works, what it can do, when it is available, how quickly it can be done. Read the spec, experiment with the system ask the designers.
- Identify what aspects of the system provide difficulties for the user. Observe users using the prototype, use cognitive walkthrough.
- Generate appropriate metaphors. He outlines the following steps for generating metaphors
  - Find ideas implicit to the problem description
    - Look at the requirements specification and look for metaphor ideas in the language used.
  - Look at user's problems and identify real world events, institutions, objects that exhibit some of the characteristics that users find difficult.
    - Eg directionally of link: rivers? TV Broadcasting etc. i.e. use these to identify objects
  - Evaluate the potential of these based on the 5 criteria.

Mistakes and Slips

- Mistakes
  - An error that someone makes because they haven't learned or have been thought how to use the system
- Slips
  - An error that occurs when someone does know what to do as they have done the task many times before.

Working Prototypes

- **Scenario**
  - Only features and functionality along with specific task scenarios or paths through interface which are to be evaluated.
- **Vertical**
  - In depth functionality for a few selected features
- **Horizontal**
  - Full interface features, but no underlying functionality.

Paper prototypes

- Involves prototyping the screen designs and dialogue elements on paper. (sketching/printouts)
  - No code is written at this stage

Interactive Sketches

- Interactive composition of hand drawn sketches. It involves:
  - Scanning in hand drawn interface sketches
  - Assemble interactive prototype with clickable elements

Cognitive Walkthrough

A cognitive walkthrough is a task orientated walkthrough of an interface. It focuses on the learnability element of the application. It tries to identify how well the interface supports the novice user in coming to understand and learn how to achieve task throughout the interface. Take notes, ensure audience engaged, don't defend prototype..

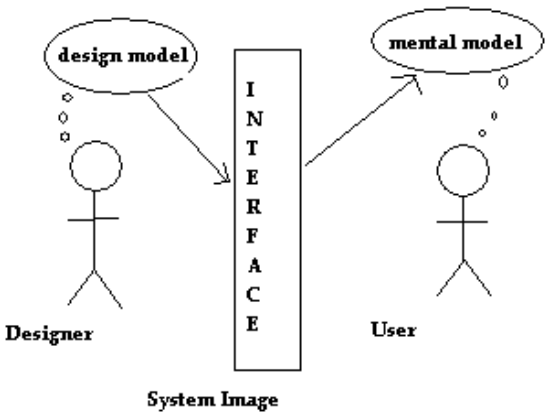
The developers arrange a meeting and present a prototype to a critical audience. They do a walkthrough of the steps needed to achieve specific tasks and try to spot areas where the user may experience any level of difficulty when performing the task.

- Finds task orientated problems
- Helps define user's goals and assumptions
- Usable early in the development process

System Image

The designer does not talk directly to the user but their idea of how to use it may differ.

1. **Design model**
   - The designers own model of the system i.e. their interpretation
2. **Conceptual Model**
   - Designers write their description of the design model and resulting documentation is the conceptual model
3. **System image**
   - The implementation of the design model (interface, documentation instructions etc.)
4. **Mental Model**
   - The user has an idea of the systems purpose and its functionalities.



Mental Models

Mental models are fundamentally models of what causes what. (clap lights on and off)

The user has an idea of the systems purpose and its functionalities. It is built through their interaction with the application.

- **Functional**
  - Knowing how to ride a bike but not how to fix the breaks
  - Can use the system.
- **Structural**
  - Knowing how to fix the breaks on a bike won't tell you how to ride a bike.
  - How the system internals work

Users Mental Model

- **Causality**
  - Clear connection between action and affect. Feedback for action. Allowing past to work only if something in clip.
- **Perceives affordances**
  - Button affords pressing
- **Interface Metaphors**
  - Verbal -> Instructor says, 'imaging the file is being stored in a folder in a filing cabinet'
  - Visual -> Apple mac trash can, Shopping cart and Recycle bin
  - Overall -> Desktop in apple mac, pc
  - Partial -> menu, window, scroll bar etc.
- **Mapping** (Layout, icons, natural associations.)
- **Constraints** (Blind?)
- **Familiarity** (Consistency)

Importance of early prototyping

- Easier to redesign
- Eliminates problems before the implementation stage and the products release
- Helps with requirements gathering
- Encourages the discussion amongst the designers
- Avoids people's reluctance to discard the prototype

Importance of frequent prototyping

- Allow for successive improvements
- Successively more complex prototype techniques so get variety of feedback
- Helps further requirements gathering with more informative versions
- Encourages experimentation

Importance of simple Prototyping

- Cheaper, Quicker, easier to build-> so more change they'll be done and done early and frequently
- Not as much invested in the prototype work so-> easier to criticise, easier to discard, more likelihood of changing based on criticism.

Kujala – User Access

1. Gaining direct access to users is difficult
   a. For specialised software with a specialised set of users it is hard to get hold of them.
2. Users not willing to let you watch then working.
3. Difficult to get developers into the field.
4. Finding willing users is time consuming.
5. Users not doing real work (demos)
6. Difficult to arrange a day where real work is happening
7. Identifying appropriate users
8. Motivating users
9. Contacting and arranging
10. Users busy.

3 **MAJOR** points

1. **Availability, Quality, Motivation**

User Involvement Paper

Benefits

1. Increase quality due to increased requirements accuracy
2. Early avoidance of superfluous costly features.
3. User acceptance – Blackboard example – it does the job but people don't use it.
4. Increased user understanding leading to increased effectiveness.
5. ?? - Linked into idea of using usability in a phased way. (observing users)
6. Less time to develop the software – more solid requirements
    a. While it is decreased, the requirements gathering time is increased

Negative aspects

1. Huge amount of data to analyse (user involvement increases amount of data)
    a. Surveys, cognitive walkthrough, data that needs to be processed
2. Acceptance of user involvement among developers.
3. (IMPORTANT) access to users is difficult
4. Increased occurrence of requests for late change
5. Users lack knowledge
6. Difficulties for users in articulating expert knowledge. – Tacit

Phased Usability testing

A phased introduction is needed to adapt to the user's needs. In each phase, new usability engineering techniques are gradually developed and formalised and integrated into the activities of previous phases. At early stages of each new phase, some practical, low cost techniques are introduced.

Phases

- Raise Awareness
- Increase Knowledge of End Users
- Focus on User Interface Design
- Evaluate Design
- Centrally Control User Interface Design

Support Calls

- Problem identification, leading to a revised design
- Improved usability testing results
- Calls fell from 45 minutes to 10 minutes

Site Visits

- Greater understanding of the product
- Downstream technology understanding
- Improved mutual understanding and work relationships among all the stakeholders

Pros

- Customers usually viewed the visits as a form of respect and appreciation
- The decisions of software developers were more likely to match the needs of the users
- Determining priorities of the product

Cons

- The amount of raw data collected during the study can be overwhelming
- Impacting the design can be difficult if field-oriented methods are not an accepted part of the development process
- It was difficult to arrange their day so that they would have real work to do when observed.
- Users don't do much work when the development team are visiting because they are a distraction.