Human-Computer Interaction

1. A Brief History

The earliest computers were experimental machines.

They were programmed and operated by the same people who designed and built them.

Usability was not an issue.

The first commercial computers were direct developments of experimental machines.

Again, usability was not considered an issue in their design.

Data was typically entered as numbers to Base 2 or another number base other than 10.

Data output typically consisted of dots on a CRT or arrays of lamps (on for one, off for 0).

The next generation of computers were large mainframes.

They were programmed and operated by highly-trained staff.

Programmes were written in a *mid-level language*, such as *Fortran*.

A *command language* was used for control.

Printed output could be formatted as required.

Computers of this era were expensive and relatively rare, so it was important to use them to their maximum potential.

To achieve this, they were usually *batch-programmed*:

- Tasks (programs and data) prepared off-line.
- Batch of tasks loaded and run in sequence.
- Operator intervention limited to preparing and loading batches.
- No intervention/interaction whilst batch is running.

The next generation of computers were terminal-and-server systems or individual desktop systems.

They were programmed and operated by trained staff.

A command language was used for control.

Printed output could be formatted as required, and the screen display could be formatted to some extent.

In order to use a command language effectively, the operator must know a large number of commands and the correct syntax for use with each of them.

Psychologists were employed to help create command-sets that were easy to remember.

They developed the first *guidelines for user-interface design*, e.g:.

- Keep commands short (to minimise typing)
- Use memorable command names
- Use consistent syntax where possible
- etc.

However, even well-designed command-languages are difficult to learn.

In the 1960s, a number of researchers set out to create computer interfaces that were easier to use.

A key problem with command languages is that they rely heavily on *recall*.

Psychological research suggests that humans are better at *recognition* than recall.

Recognition requires less mental effort than recall, and is around twice as fast and three times as accurate (Nobel, 2001).

Human short-term memory is quite limited - we can only hold a few items in memory at any one time.

This makes it difficult to perform mental operations that involve more than a few pieces of data.

However, such tasks become much easier if we write the relevant data down...

...or display it on a screen.

Human beings have excellent *spatial memory*.

Recalling *where* an item is displayed typically requires less memory than recalling *what* information it contains.

Thus the challenge facing researchers was to:

- reduce reliance on *recall*
- support *recognition* wherever possible...
- ...by using the screen as a form of *external memory*.

SketchPad, developed by Ivan Sutherland in 1963, was an innovative graphics programme that also introduced new interaction methods:

- Data was represented by visible objects which could be manipulated using a *light-pen*.
- Objects could be moved and re-sized.
- New instances of objects could be created, inheriting features from the parent.

In 1968, Douglas Engelbart demonstrated the *oNLine System (NLS)* which used a mouse as a cheaper alternative to the light-pen.

Features of NLS included:

- Tiled windows.
- Hypertext links
- Cross-document editing

Alan Kay proposed the use of *overlapping windows* in his 1969 PhD thesis.

He later implemented them in *Smalltalk* and *InterLisp* (1974).

David Canfield Smith developed the concept of *icons* for his 1975 PhD thesis.

He further developed his ideas while working at Xerox where he was one of the chief designers of the *Star 8010 Information System* (1981), the first commercial system to use a graphical interface.

The Xerox Star was not a commercial success, but the idea caught on and other computer companies soon introduced graphical interfaces:

- 1983 - Apple Lisa
- 1984 - Apple Macintosh, the first commercially-successful system to use a GUI
- 1985 - Microsoft Windows 1.0, a *window manager* for PCs running Microsoft DOS.
- 1995 - Microsoft Windows 95, a full graphical operating system for PCs.

Ben Shneiderman (1982) analysed a number of graphical interfaces and concluded that the features which made them successful were:

- Visibility of objects
- Replacement of complex commands with actions to manipulate visible objects directly
- Incremental action at the interface with rapid feedback
- Reversibility of actions
- Syntactic correctness of actions (every action is legal)
- Few/no changes of mode

Shneiderman coined the term *Direct Manipulation* to describe the style of interaction supported by such interfaces.

Graphical User Interfaces (GUIs) supporting Direct Manipulation are still the basis of most contemporary computing.

However, as graphical interaction has matured, other interaction technologies have been developed.

- joysticks, 3D mice, and other input devices
- touch screens (including multi-touch screens), 3D displays
- tactile feedback devices
- speech synthesis and recognition
- spatial and other auditory displays
- image recognition, eye-tracking, movement tracking, etc.

The Direct Manipulation style of interaction has been extended to accommodate some new interaction technologies, such as touch-screens, 3D navigation and display, etc..

However, other developments - such as speech, auditory and gestural interaction - do not fit easily within the Direct Manipulation paradigm.

Efforts to use such technologies effectively are driving the development of new interaction styles.

2. The Changing Nature of HCI

It can be seen that the way in which humans interact with computers has changed enormously:

Early Computers	**Modern Computers**
Minimal interaction	Extensive interaction
Little distinction between operation and programming	Control, authoring, scripting, programming, etc.
All interaction based on computer's requirements	Interaction increasingly based on user's requirements

A number of factors have driven changes in interaction:

i. Technical Developments
    - New Interaction Technologies
    - Reductions in cost and size

ii. Need to support a wider range of users

iii. Application to a wider range of tasks

(i) Technical Developments: Reductions in cost and size

- Early computers were expensive
    - had to be used to full potential
    - little need - or opportunity - for interaction
- Modern computers are small and cheap
    - allows each user to have many computers
    - computers mostly idle, waiting on user

(ii) Need to Support a Wider Range of Users

- Early computers could only be operated by highly-trained staff
- Modern computers are used by a wide range of people with varying needs:
    - novices and experts
    - frequent and occasional users
    - users with special needs
        - Improvements in *accessibility* are increasingly being driven by legal requirements, e.g.:
            - EC Directive 90/270 (1990)
            - Americans with Disabilities Act (1990)
            - Disability Act (Ireland, 2005)
            - UN Convention on the Rights of Persons with Disabilities (2008)

(iii) Application to a wider range of tasks

- Early computers were used for a limited range of tasks
- Modern computers used for a wider range of tasks:
    - casual/occasional tasks for which training is not feasible
        - Digital technology is being used in an ever wider range of applications.
        - However, it is not reasonable to expect people to become expert at using every application they encounter, particularly applications they use infrequently.
        - It is easy enough to support occasional/non-expert users in simple tasks.
        - It is more difficult to support occasional/non-expert users in complex tasks.
        - Web applications and smartphone apps are becoming as sophisticated as stand-alone applications. However:
            - Users have little commitment to individual applications.
            - They are unlikely to read complex instructional material, and if they find a site or app awkward to use may go elsewhere.
            - Thus such interfaces - even for complex tasks - need to be intuitive.
    - safety-critical control applications
        - Computers are widely used in:
            - Military/Aircraft systems
            - Medical systems
            - Plant control (nuclear, chemical, etc.)

        However, even highly-trained staff make mistakes, so software has to be designed to prevent mistakes wherever possible.
    - Educational and entertainment applications
        - E-Learning systems
        - Immersive game environments
        - Interfaces for hobby/entertainment applications
    - Hands-free, eyes-free tasks, etc.
        - Sat-Nav, traffic management systems, etc.

In the future, it is likely that *Human-Computer Interaction* will become more like *Human-Human Interaction*:

- *proactive* rather than *reactive*
- *context-aware*
- *adaptive*

Reactive Systems	**Proactive Systems**
User always initiates actions	System or User can initiate actions
Computer is focus of user attention	User attention elsewhere (hands-free/eyes-free, etc.)
Context-free	Context-aware
Little need for adaptivity	Adaptivity essential for effective operation