Chapter 7
Human–Computer Interaction

Key Topics

Batch processing
Text-based interface
Graphical User Interface
Voice User Interface
WIMP
SILK
Usability Standards

7.1 Introduction

Human–computer interaction (HCI) is a branch of computer science that is concerned with the design, evaluation and implementation of interactive computing systems for human use. It is focused on the interfaces between people and computers, and involves several different fields including computer science, cognitive psychology, design and communication. The human–computer interaction field has evolved over the decades to include text-based interaction systems, graphical user interfaces (GUI) and voice user interfaces (VUI) for speech recognition and speech synthesis.

The interaction between humans and machines was mainly limited to information technology professionals from the early days of computing up to the mid/late 1970s. This changed after the invention of the microprocessor in the early 1970s, which led to an explosion of interest from computer hobbyists, and the subsequent development of home computers from the mid-1970s. The introduction of the IBM personal computer in the early 1980s meant that everyone in the world was now was a potential computer user, and it led to a new market of personal applications and tools to support the user. However, it was clear that there were serious deficiencies with respect to the usability of computers in carrying out the tasks that users wished to perform.

Humans interact with computers in many ways, and so it is important to understand the interface between human and machines to facilitate an effective interaction. The early computer systems were *batch processing* (running programs in batches without human intervention) on a large expensive mainframe computer.

G. O'Regan, *World of Computing*,
https://doi.org/10.1007/978-3-319-75844-2_7

The interaction between the human (operator) and computer was limited, and it consisted of placing the punched cards (encoded instructions to the computer) on the card reader, and the computer would then process the cards overnight. These computers were slow and expensive, and it was important that they be used efficiently 24 h a day. The computer could run only one program at a time, and programmers were unable to interact with the computer while it was running, and this made it difficult and time-consuming to identify and correct errors.

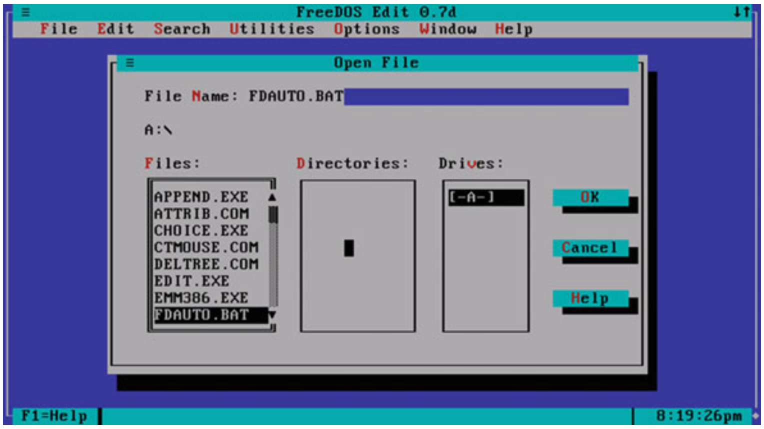
A *text-based interface* (also known as a command line interface) is where the system interaction (input and output) and navigation are text-based. They are easier to use than punched card programming, but require skilled operators due to the difficulty in remembering long lists of system commands.

Licklider wrote an influential paper ‘*Man-Computer Symbiosis*’ in 1960 (Licklider 1960), in which he outlined the need for a simple interaction between users and computers. This paper mentioned ideas such as sharing computers among many users, interactive information processing and programming, large-scale storage and retrieval, and speech and handwriting recognition.

Doug Engelbart was one of the main developers of NLS (oN Line System) in the late 1960s, and this online word processor system had features such as the first computer mouse, time-sharing and a command line interface. User trials and testing was employed in its development as part of a *philosophy towards a system adapting to people rather than people adapting to a system.*

One of the most well-known text-based operating systems was Microsoft’s MS/DOS operating system for IBM compatible personal computers, which was introduced in 1981 (Fig. 7.1). Text-based interfaces are effective for expert users but are more difficult for users with an average level of knowledge, as they have a steep learning curve and the difficulty in remembering a long list of system commands. The fact that they are not very intuitive or user-friendly motivated research into alternative approaches.

The *graphical user interface* (GUI) is a human–computer interface that uses graphical icons, menus and windows to represent information and action to the user.



```
File Name: FDAUTO.BAT
A:\
APPEND.EXE
ATTRIB.COM
CHOICE.EXE
CTMOUSE.COM
DELTREE.COM
EDIT.EXE
EMM386.EXE
FDAUTO.BAT
```

Fig. 7.1 FreeDOS text editing

It was a revolution in human and computer interaction and the GUI was intuitive and user-friendly. They have made computers and electronic devices attractive to non-technical users, and the usability of the GUI has allowed a large range of users with varying ability and expertise to successfully interact with computers.

Early work on graphical user interfaces took place at Xerox PARC in the 1970s with their work on the Xerox Alto personal workstation (Fig. 3.24). This was the first computer to use a mouse-driven graphical user interface, and it was introduced in the mid-1970s. It was essentially a small minicomputer rather than a personal computer (it was not based on the microprocessor). Its significance is that it had a major impact on the user interface design, and especially on the design of the Apple Macintosh computer.

The Xerox Star was introduced in the early 1980s, and it followed sound usability principles (prototyping and analysis, iterative development and testing with users) in its development. Steve Jobs visited Xerox PARC in late 1979, and he realised that the future of personal computing was with computers that employed a graphical user interface (such as in the Xerox Alto). Jobs was amazed that Xerox had not commercialised the technology, as he saw its graphical user interface as a revolution in computing and a potential goldmine in the future of computing. The design of the Apple Macintosh was heavily influenced by the design of the Xerox Alto, and the release of the Macintosh was a major milestone in computing.

The Macintosh was a much easier machine to use than the existing IBM personal computer. Its friendly and intuitive graphical user interface was a revolutionary change from the command-driven operating system of the IBM PC, which required the users to be familiar with its operating system commands. It was 1990 before Microsoft introduced its Windows 3.0 GUI-driven operating system (Fig. 7.2).

Today, the prevalent paradigm in human–computer interaction is the WIMP (windows, icons, menus and pointers) paradigm, which is comprised of a graphic and text interface navigated by a mouse and keyboard. The future of HCI is predicted to be the SILK (speech, image, language and knowledge) paradigm, where communication between humans and machine will be more natural and intuitive.

7.2 HCI Principles

The success of computer systems is critically influenced by the design of the human–computer interaction, and in the achievement of end-user computing satisfaction. Human–computer interaction is concerned with the study of humans and machines, and so it needs knowledge of both to be effective. The study of machines requires knowledge of computer graphics, programming languages, capabilities of current technology and so on, whereas on the human side it requires knowledge of cognitive psychology, ergonomics and other human factors such as usability and computer user satisfaction.

There are several fundamental principles and models underlying HCI. It is essential to understand the user and their characteristics, as well as their diversity in

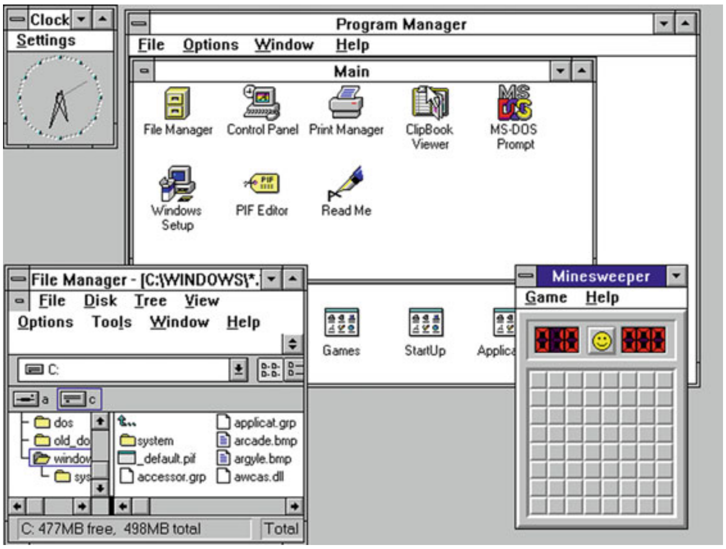


Fig. 7.2 Microsoft Windows 3.11 (1993). Used with permission from Microsoft

age, experience, physical and intellectual abilities, and so on. It is customary to distinguish between two types of user knowledge (IT and domain knowledge), and the user's proficiency in each type of knowledge yields several user categories that range between novice and expert.

– Interface knowledge (knowledge of the IT technology).
– Domain/task knowledge of the real-world system.

The software will generally support multiple user categories, where novices get opportunities to learn about the system and have fewer opportunities for error. It is important to understand the domain in which the software will be used and to identify the tasks to be performed as well as the frequency in which they will be performed.

There have been several rules and principles proposed for HCI design including Shneiderman's 'Eight Golden Rules of Interface Design' (Table 7.1) (Shneiderman and Plaisant 2005).

7.3 Software Usability and User-Centred Design

Usability has become important in software engineering and especially with the emergence of the World Wide Web in the early 1990s. The usability of the software is the perception that a user or group of users has of its quality and ease of use (i.e. is the software easy to use and easy to learn?), and its efficiency and effectiveness.

Table 7.1 Eight golden rules of interface design

Principle	Description
Strive for consistency	Consistent terminology, sequences of action and commands throughout the system
Enable frequent users to use shortcuts	The user will naturally desire to reduce the number of interactions as the frequency of use increases
Provide informative feedback	There should be appropriate system feedback
Design dialogue to yield closure	Sequences of actions should be organised into groups with a beginning, middle and end
Offer simple error handling	Design the system (as far as possible) to prevent the user from making a serious error. The system should be able to detect an error and provide a handling mechanism
Permit easy reversal of actions	This is important to the user as it means that errors can be undone
Support internal locus of control	The system should be designed to make the users initiators of actions rather than responders to actions
Reduce short-term memory load	There are limitations to human processing in short-term memory, and so displays should be kept simple

Usability is a multidisciplinary field, and psychological testing may be employed to evaluate the perception that users have of the computer system. Usability is defined in the ISO 9241 standard as:

> **Usability** is the degree to which software can be used by specified consumers to achieve quantified objectives with effectiveness, efficiency and satisfaction in a quantified context of use.

There are several standards for usability including the ISO 9241 and ISO 16982 standards, and the IEC 62366-1 standard (Applications of Usability Engineering to Medical Devices) from the International Electrotechnical Commission (IEC).

Usability, like quality, needs to be built into the software product rather than added later, and it needs to be considered from the earliest stages in the software development process. It requires an analysis of the user population and the tasks that they perform, as well as their knowledge and experience. The specification of the user and system requirements needs to include the usability requirements, as these are an integral part of the system.

There will often be a variety of different viewpoints to be considered, and this leads to multiple design solutions and an evaluation of these against the requirements. An iterative software development life cycle is generally employed, with active user involvement during the software development process. Prototyping is often employed to give the users a flavour of the proposed system and to get early user feedback on its usability. User acceptance testing (including usability testing) provides confidence that the software satisfies the usability, accessibility and quality expectations of the users (Table 7.2).

Table 7.2 Software development life cycle (including usability)

Phase	Description
Requirements	Interviews with the different categories of users
Prototype	Initial prototype developed and structured feedback given by users (usually via questionnaire)
Spiral design/ development	Design a little, code a little, test a little, formal review and user feedback prior to new spiral
Acceptance	Final acceptance testing by users

7.3.1 User-Centred Design

User-centred design (UCD) is a design process that is focused on the usability of and accessibility of the system to be developed, and it places the users at the centre of the software development process. The users are actively involved from the beginning of the project, and regular feedback is obtained from them at each stage of the process. UCD follows well-established techniques for analysis and design, and it is focused on understanding the characteristics of users and their needs (Table 7.3).

The UCD design activities focus on the user, including understanding the tasks that they perform, their needs and their experience. The users clarify what they want from the product and the environment in which the software will be used. The designers then determine how the users are currently performing their tasks, and what they like and dislike about the ways in which the tasks are currently done. This helps the designer to design a product that will be fit for purpose, that will satisfy the usability expectations of users, as well as being competitive in the market.

Table 7.3 UCD principles

Principle	Description
User understanding	The design is based on an explicit understanding of users, tasks and environments (i.e. who are the users?, what are their tasks and needs? and what is their experience?)
User involvement	The users are involved throughout the design and development (and user feedback shapes the design and development)
User evaluation	The design is driven and refined by user evaluation (and the user acceptance testing confirms that the usability and functional requirements are properly implemented)
Iterative development	The software development process is iterative, and the approach is to design and develop a little, get feedback from the user evaluation, modify accordingly and proceed to the next cycle in the iteration
Design	The design addresses the whole user experience
Multidisciplinary	The design team includes multidisciplinary skills

The software development team produces an initial version (or prototype) of the product, and the prototype has sufficient functionality to test some parts of the design. The design and development proceeds in cycles of modification, testing and a user review of the current version, until the software satisfies functional, usability and accessibility requirements. The approach is to design a little; code a little; test a little; evaluate and decide on whether to proceed with subsequent cycles.

A pre-release of the software may be created and sent to a restricted set of users for their evaluation, and the user feedback is then used to finalise the product prior to its actual release.

7.4 Review Questions

1. What is a text-based interface?
2. What is a graphical user interface?
3. Explain the importance of software usability.
4. Investigate the various usability standards such as ISO 9241 and ISO 16982.
5. Explain user-centred design.
6. Describe the evolution of human–computer interfaces.

7.5 Summary

Human–computer interaction is a branch of computer science that is concerned with the design, evaluation and implementation of interactive computing systems for human use. It is focused on the interfaces between people and computers, and has grown over the decades to include text-based interaction systems, graphical user interfaces and voice user interfaces.

The development of home computers from the mid-1970s meant that everyone in the world was now a potential computer user, and it was clear that there was a need to improve the usability of machines. Humans interact with computers in many ways, and so it is important to understand the interface between them to facilitate the interaction.

The early interaction between humans and computers was via batch processing with limited interaction between the operator and computer. These were followed by text-based interfaces (also known as a command line interface), where the system interaction (input and output) and navigation are text-based. One of the most well-known text-based operating systems was Microsoft's MS/DOS operating system for IBM compatible personal computers.

The graphical user interface is a human–computer interface that uses graphical icons, menus and windows to represent information and action to the user. They are intuitive and user-friendly, and were a revolution in human and computer interaction. They have made computers and electronic devices attractive to non-technical users, and are a major step forward from command-driven operating system.

The success of modern software systems is related to the usability of the software, and user-centred design has become a key paradigm in building usability in the software. It places the user at the centre of the software development process with active user involvement and evaluation employed.