3. What Makes a Good Interface?

Some definitions of usability:

"A user interface should be so simple that a beginner in an emergency can understand it within ten seconds."

(Theodor Nelson, 1965)

"...any application designed for people to use should be:

- easy to learn (and remember),
- useful, that is, contain functions people really need in their work, and
- be easy and pleasant to use."

(Gould & Lewis, 1985)

Four Key Concepts:

- *Learnability* - the time and effort required to reach a specified level of user performance
- *Throughput* - tasks accomplished, speed of execution, errors made, etc.
- *Flexibility* - the extent to which the system can accommodate changes to the tasks and environments beyond those first specified
- *Attitude* - the attitude engendered in users by the application.

(Shakel, 1990)

It is often said that we should aim to design systems that are 'user friendly'.

But how do we identify the users and determine what is 'friendly' for them?

People vary in their abilities and requirements depending upon age, education, level of computer-literacy, etc.

- The requirements of a young gaming-enthusiast may differ considerably from those of a retired person who uses a computer in connection with their hobby.
- Someone who uses a piece of software every day may have different needs than someone who uses the same piece of software very occasionally.
- An individual's requirements may change as they learn more about computers generally.

Some applications may be aimed at just one group of users; others may be aimed at a wide range of users.

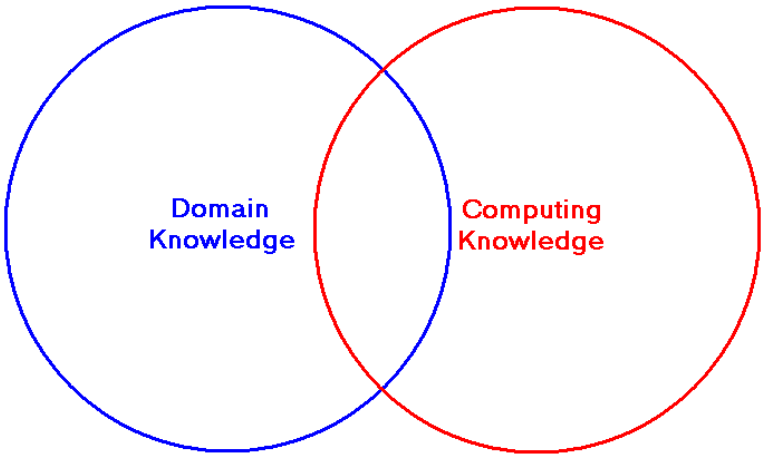
Some applications embody a considerable amount of specialist knowledge.

For example, the developers of a computerised hospital-records system may have to take into account a great deal of specialist medical knowledge in order to design an effective system.

In such cases, there is a potential gulf between:

- *system developers*, who have knowledge concerning computer systems
- *end-users* of the system, who have knowledge of the *domain* for which the system is to be used.

An important goal in HCI is to identify and then reduce or eliminate the gulf(s) that may exist between developers and end-users.



Users may interact differently with an application depending upon whether they are using it for a *primary* or *secondary* task.

- A user may give their full attention to a work-related task or a game
- Someone using an ATM may also be thinking about security, controlling small children, etc., and thus give only part of their attention to using the ATM

Some users may have special needs.

Ideally, software should be designed so that users with special needs can use it as easily as other users can, but this is difficult to achieve.

For example, people who are blind/visually-impaired can operate Graphical User Interfaces with the aid of a *screen-reader*.

However, compared to other users, screen-reader users tend to:

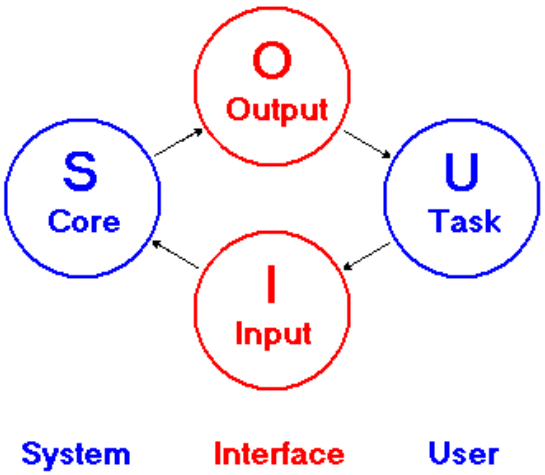
- work more slowly
- make more errors
- report higher levels of fatigue.

Giving users with special needs the same level of access as other users is a major challenge.

Wherever possible, users should be able to enter data and receive feedback in a form that is natural for them and appropriate to the task in hand.

Abowd and Beale (The Interaction Framework, 1991) suggested that human-computer interaction could be viewed as having four components:

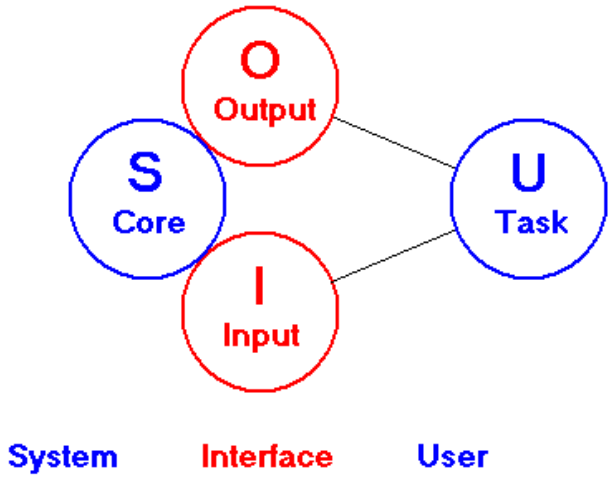
- The User
- The Input device(s)
- The System
- The Output device(s)



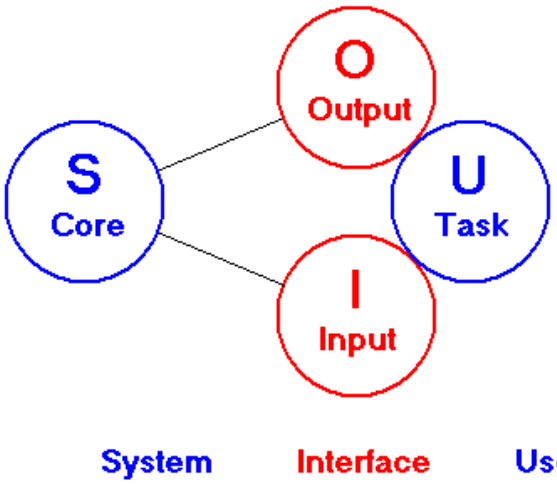
Information is passed between the components in a cycle.

The components may use different languages, and thus *translation* may be needed at each stage of the cycle.

The more translation the user has to perform, the harder the interface will be to use.



In early computers, ALL translation was performed by the user.



Ideally, all translation should be performed by the computer.

To summarise, a good interface should:

- be as easy to learn and remember as possible
  - but sometimes learning new skills is unavoidable/desirable
- contain functions that are genuinely useful
  - exclude functions that increase complexity without improving utility
  - offer sufficient flexibility to cope with changing work-patterns
- enable users to work efficiently
  - it should be possible to achieve goals quickly and with few errors.
- be easy and pleasant to use
  - not place a high cognitive load on the user
  - allow the user to feel 'in control'

In setting out to design interfaces that meet these criteria, we should bear in mind that users:

- may have a very different view of the task/application domain from our own.
- may differ considerably - we may have to design with several distinct groups of users in mind
- may change in their abilities/requirements whilst using the software
- may give all or only part of their attention to a task
- may vary in their motivation
- may be using a system for themselves or on behalf of somebody else
- will find the system more efficient and pleasant to use if we minimise the amount of translation required at the interface.

4. Key Skills/Knowledge in HCI

- General Computing Skills & Knowledge
- Interaction styles
- Interaction technology
- Human psychology/perception/etc.
- Design/Evaluation techniques
    - Guidelines
    - Metrics
    - Modelling
    - Iterative design and evaluation

Interaction Styles:

- Command-language
- Form-filling
- Menu-selection
- Function keys
- Question and Answer
- Direct-manipulation
- Anthropomorphic/Natural Language

Interaction Technologies:

- Input
    - Pointing/selection (mouse, joystick, touch-screen, eye-gaze, etc.)
    - Text-entry (keyboard, keypad, speech, etc.)
- Output
    - Visual (Text & Graphics)
    - Audio
    - Tactile devices

Human Psychology and Perception:

- Visual perception
- Auditory perception
- Tactile perception
- Memory
- Etc.

Guidelines:

- Web Content Accessibility Guidelines
- Eight Golden Rules (Shneiderman)
- Guidelines for UI software (Smith & Mosier)
- OSF Style Guide
- GUI Bloopers (Johnson)
- Human Interface Guidelines (Apple)
- Speech Interface Guidelines (Larson)
- etc..

Metrics:

- SUMI, WAMMI, etc
- Accessibility validators, e.g., WebXact

Modelling Techniques:

- Task Analysis
    - Hierarchical Task Analysis
    - Knowledge-Based Analysis
    - Entity-Relationship Techniques
- Cognitive modelling
    - Task-Goal Hierarchies
    - Linguistic/Grammatical Models
    - Device-Level Models

Design and Evaluation Techniques:

- Questionnaire design
- Experimental design
- Statistical analysis
- Iterative design and evaluation