HUMAN-COMPUTER INTERACTION

Introduction

Human-computer interaction (HCI) is the study and the practice of usability. It’s about understanding and creating software and other technology that people will want to use, will be able to use, and will find effective when used. As its name implies, HCI consists of three parts: the user, the computer itself, and the ways they work together. The user can be an individual user or a group of users working together. The computer refers to any technology ranging from desktop computers, to large scale computer systems. For example, if we were discussing the design of a Website, then the Website itself would be referred to as “the computer”. Devices such as mobile phones or VCRs can also be considered to be “computers”. There are obvious differences between humans and machines. In spite of these, HCI attempts to ensure that they both get on with each other and interact successfully.

The Goals of HCI

The goals of HCI are to produce usable and safe systems, as well as functional systems. In order to produce computer systems with good usability, developers must attempt to:

- **understand** the factors that determine how people use technology
- **develop** tools and techniques to enable building suitable systems
- **achieve** efficient, effective, and safe interaction
- **put people first**

Underlying the whole theme of HCI is the belief that people using a computer system should come first. Their needs, capabilities and preferences for conducting various tasks should direct developers in the way that they design systems. People should *not* have to change the way that they use a system in order to fit in with it. Instead, the system should be designed to match their requirements.

Usability

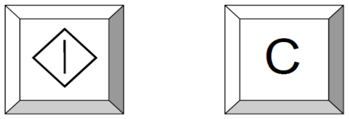
Usability is concerned with making systems *easy to learn and use*. It is part of a broader scope of what makes a system good enough to be acceptable to the end user and other stakeholders, satisfying the needs and requirements of both. Nielsen & Mack (1994) describe usability as, *"a broad concept that refers to how easy it is for users to learn a system, how efficiently they can use it once they have learned to use it, and how pleasant it is to use."*

A usable system is:

- easy to learn
- easy to remember how to use
- effective to use
- efficient to use
- safe to use
- enjoyable to use

Why is usability important?

Many everyday systems and products seem to be designed with little regard to usability. This leads to frustration, wasted time and errors. For example, a photocopier might have buttons like these on its control panel.



Imagine that you just put your document into the photocopier and set the photocopier to make 15 copies, sorted and stapled. Then you push the big button with the "C" to start making your copies. What do you think will happen? – The photocopier settings are cleared and no copies are made.

The copy button is actually the button on the left with the “line in a diamond” symbol. This symbol is widely used on photocopiers, but is of little help to someone who is unfamiliar with this.

Nielsen’s usability principles

Nielsen (1993) identified five attributes that contribute to usability:

1. **Learnability**: The user should be able to promptly start performing their tasks with the system.
2. **Efficiency**: Once the user has learned the system, a high level of productivity should be possible.
3. **Memorability**: The casual user should be able to return to the system after not having used it for some time, without having to relearn everything.
4. **Errors**: Users should not make many errors using the system, and if they do, they should be able to easily recover from them. Catastrophic errors should not occur.
5. **Satisfaction**: Users should like using the system and should be subjectively satisfied when using it. The system should be pleasant to use.

Shneiderman’s Eight Golden Rules of Interface Design

Shneiderman’s eight golden rules provide a convenient summary of the key principles of interface design. These eight rules are:

1. **Strive for consistency**.

    - Keep action sequences consistent in similar situations.

- Use identical terminology in prompts, menus, and help screens.
- Employ consistent commands throughout.

2. Enable frequent users to use shortcuts.

- As the frequency of use increases, so do the user's desires to increase interaction speed.
- Abbreviations, function keys, hidden commands, and macro facilities are very helpful to an expert user.

3. Offer informative feedback.

- System feedback should be provided for every operator action.
- Frequent and minor actions can provide a modest response, while the response should be more substantial for infrequent and major actions.

4. Design dialog to yield closure.

- Organized sequences of actions into groups with a beginning, middle, and end.
- Informative feedback when a group of actions is completed, gives the operators a sense of accomplishment and an indication that they can proceed to the next group of actions.

5. Offer simple error handling.

- The system should be designed so the user cannot make a serious error.
- If an error is made, the system should be able to detect the error and offer simple, comprehensible ways to recover from the error.

6. Permit easy reversal of actions.

- If the user knows that errors can be undone, they are encouraged to explore unfamiliar options without undue anxiety.
- The units of reversibility may be a single action, a data entry, or a complete group of actions.

7. Support internal locus of control.

- Experienced operators like to sense that they are in control of the system and that the system responds to their actions.
- Design the system to make users the initiators of actions rather than the responders.

8. Reduce short-term memory load.

- Displays should be kept simple and sufficient training time should be allotted for codes, mnemonics, and sequences of actions.

Fundamental Design Principles

Below are the four most important fundamental principles that give the basics for good design:

1. Learneability/Familiarity
2. Ergonomics/Human factors
3. Consistency/Standards
4. Feedback/Robustness

Learneability/Familiarity

The principle hindrance to effective interface interaction is lack of familiarity based upon knowledge of the system. The longer the time taken to acquire that familiarity the more difficult it is for the user to interact with the interface effectively. This learning time can be reduced by making use of the knowledge that the user already has.

When the user performs an unfamiliar task there is an inherent internal barrier to getting the task completed. The learning process has to become involved in order to accomplish the user-centred goal. Any system, which reduces or obviates this learning process will enhance task performance by reducing mental workload.

Making the system easy to learn provides the user with predictive capabilities and cognitive understanding of the methods of accomplishing the task. If you can predict what comes next you do not have to learn.

Ergonomic/Human Factors

Ergonomic design involves understanding the users and what they need to do with the system and where they will use it. When the user is seen as an unreliable, unpredictable component of the system, the importance of the user's humane characteristics is negated and undervalued, resulting in systems that are difficult to use, awkward and inefficient.

Failure to prioritize ergonomics/human factors as a fundamental design principle has led to failures we have all experienced:

- The keypads (some Mobile Phones) designed for miniature fingers,
- The displays on LCD and similar display screen equipment (PDA's) which become illegible in daylight,
- Illogical operating sequences (some VCR's) which have not been tested on real people.

Individual users possess common capabilities such as reading, listening, touch, and motor skills, in addition to average height, size and weight, the physical diversity, psychological abilities and disabilities of users have to be analyzed and evaluated when designing interfaces e.g. switches that need to be operated in sequence need to be placed together.

Consistency

Consistency and standards are central to usability. Lack of consistency has even lead to dire consequences. There are therefore different kinds of consistency on different levels. Consistency is required from cross platform consistency right down to individual user response.

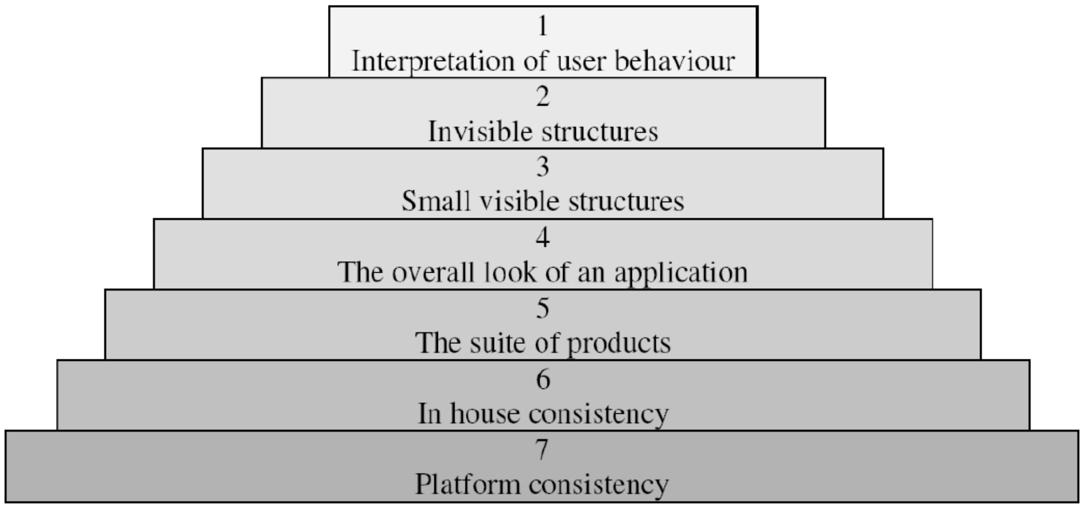


Figure: Different levels of consistency

The highest-level consistency within Human Centred Interfaces is consistency with user expectations (Level 1 – see figure above). Consistency must be maintained across the range from the widest to the narrowest areas whether we are designing the interface for a mobile phone or for a suite of Office Applications.

Invisible structures (Level 2) refer to elements of the interface, which are hidden but provide utility to the user while small visible structures (Level 3) refer to buttons icons etc. Single application consistency (Level 4) requires consideration of splash screens, toolbars etc. Suites of interacting products (Level 5) need to share the same look, feel and operation across the whole range of units. Separate Hi-Fi units should have the same type of buttons in similar places. In-house consistency (Level 6) requires every product from the same manufacturer to share the same inherent design features, while platform consistency (Level 7) requires products from different manufacturers to meet the same standards.

Global consistency depends upon standardization. A volume control is always turned clockwise for every device. This universal standardization means that no instructions are needed and no mistakes are made during operation. For instance a car manufacturer knows that his next model has to place the clutch, brake and accelerator pedals in the same order as before to maintain safety standards. The activating and terminating telephone buttons/icons on a, say, Nokia mobile phone have to be positionally consistent with previous models to maintain user expectations especially as upgrades are obtained frequently in the mobile communications area.

Feedback/Robustness

Whenever a user operates a switch, presses a button, turns a dial, clicks a mouse or interacts in any way with a machine or with a system, there must be feedback that is unambiguous. Feedback can come in the form of a sound, a light, text on a screen, dialogue boxes etc.

However the ideal form of feedback is allowing users to see things happen. If a system gives no feedback at all then the user may assume that the command has not been received, accepted and acted upon. This may cause the user to press the button again, repeat the command or even think that the device is no longer working. Sometimes this may result in mistaken data input, endless looping, or a number of duplicated actions, which take further time and effort from which to recover.

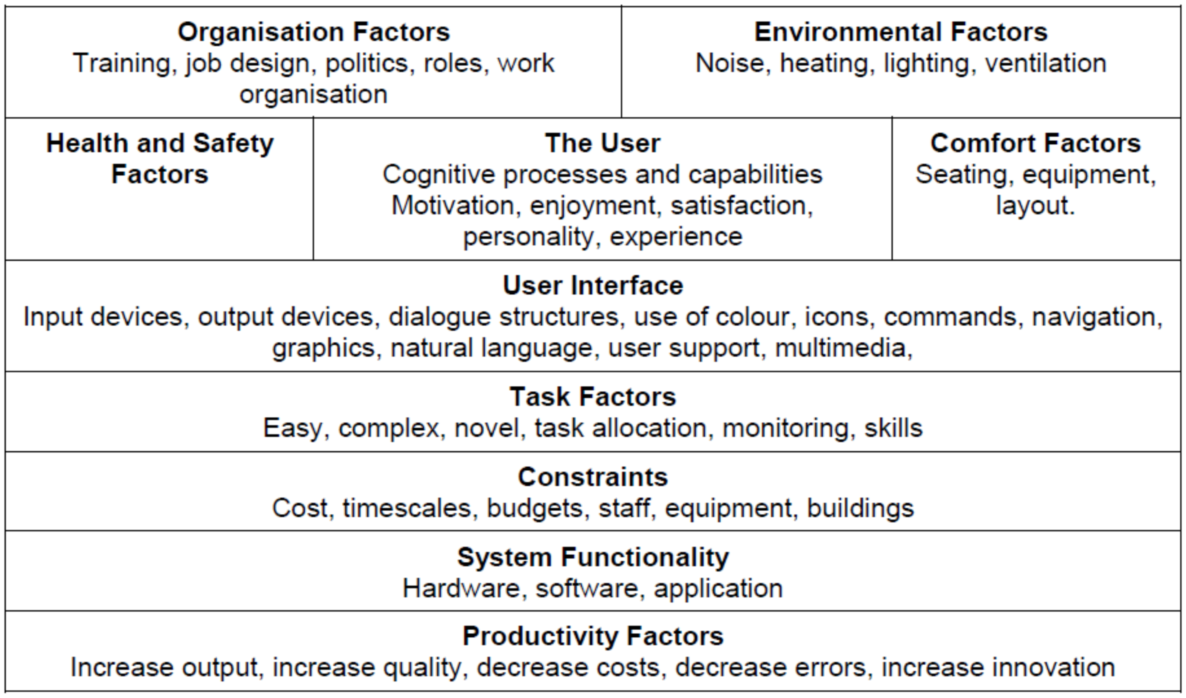
One of the characteristic uses of feedback is to alert the user to errors. Making an interface error-responsive is a feedback issue. Even the simple misspelling of words in a word processor can be addressed by immediate feedback such as a red underline to provide the user with the possibility of immediate remedial action. The additional feature of substituting the correct spelling at the single click of a mouse button enhances the feature.

Habitual actions through task repetition can easily cause users to make mistakes when the locus of attention is shifted. To allow users to undo those errors, even if they have already moved several steps ahead could improve error handling and recoverability.

Good design principles for human centred systems are derived from careful examination of user experience and an understanding of user cognitive psychology.

Factors in HCI

There are a large number of factors which should be considered in the analysis and design of a system using HCI principles. Many of these factors interact with each other, making the analysis even more complex. The main factors are listed in the table below:



Organisation Factors Training, job design, politics, roles, work organisation		**Environmental Factors** Noise, heating, lighting, ventilation
Health and Safety Factors	**The User** Cognitive processes and capabilities Motivation, enjoyment, satisfaction, personality, experience	**Comfort Factors** Seating, equipment, layout.
User Interface Input devices, output devices, dialogue structures, use of colour, icons, commands, navigation, graphics, natural language, user support, multimedia,		
Task Factors Easy, complex, novel, task allocation, monitoring, skills		
Constraints Cost, timescales, budgets, staff, equipment, buildings		
System Functionality Hardware, software, application		
Productivity Factors Increase output, increase quality, decrease costs, decrease errors, increase innovation		

Disciplines contributing to HCI

The field of HCI covers a wide range of topics, and its development has relied on contributions from many disciplines. Some of the main disciplines which have contributed to HCI are:

1. **Computer Science**
    - technology
    - software design, development & maintenance
    - User Interface Management Systems (UIMS) & User Interface Development Environments (UIDE)
    - prototyping tools
    - graphics
2. **Cognitive Psychology**
    - information processing
    - capabilities
    - limitations
    - cooperative working
    - performance prediction
3. **Social Psychology**
    - social & organizational structures
4. **Ergonomics/Human Factors**
    - hardware design
    - display readability
5. **Linguistics**
    - natural language interfaces

6. **Artificial Intelligence**

- intelligent software

7. **Philosophy, Sociology & Anthropology**

- Computer supported cooperative work (CSCW)

8. **Engineering & Design**

- graphic design
- engineering principles

PSYCHOLOGICAL ASPECTS OF HCI

Psychology in HCI is concerned with the designing of interfaces that reflect the user’s cognitive processes. It can also concern identifying and diagnosing interaction errors by considering the cognitive systems at play.

Recent development in cognitive psychology

a) Computational Approaches

Computational approaches continue to adopt the computer metaphor as a theoretical framework, but they no longer adhere to the information-processing framework. Instead, the emphasis is on modeling human performance in terms of what is involved when information is processed rather than when and how much. Primarily, computational models conceptualize the cognitive system in terms of the goals, planning and action that are involved in task performance. These aspects include modeling: how information is organized and classified, how relevant stored information is retrieved, what decisions are made and how this information is reassemble. Thus tasks are analyzed not in terms of the amount of information processed per se in the various stages but in terms of how the system deals with new information.

b) Connectionist Approaches

Connectionist approaches, otherwise known as neural networks or parallel distributed processing, simulate behavior through using programming models. However, they differ from conceptual approaches in that they reject the computer metaphor as a theoretical framework. Instead, they adopt the brain metaphor, in which cognition is represented at the level of neural networks consisting of interconnected nodes. Hence all cognitive processes are viewed as activations of the nodes in the network and the connections between them rather than the processing and manipulation of information.

External Cognition

External cognition is concerned with explaining the cognitive processes involved when we interact with different external representations. A main goal is to explicate the cognitive benefits of using different representations for different cognitive activities and the processes involved. The main one include:

1. externalizing to reduce memory load
2. computational offloading
3. annotating and cognitive tracing.

Externalizing to reduce memory load

A number of strategies have been developed for transforming knowledge into external representations to reduce memory load. One such strategy is externalizing things we find difficult to remember, such as birthdays, appointments and addresses.

Externalizing, therefore, can help reduce people’s memory burden by:

- reminding them to do something (e.g., to get something for their mother’s birthday)
- reminding them of what to do (e.g., to buy a card)

- reminding them of when to do something (send it by a certain date)

Computational offloading

Computational offloading occurs when we use a tool or device in conjunction with an external representation to help us carry out a computation. An example is using pen or paper to solve a math problem.

Annotating and cognitive tracing

Another way in which we externalize our cognitions is by modifying representations to reflect changes that are taking place that we wish to mark. For example, people often cross things off in to-do list to show that they have been completed. They may also reorder objects in the environment, say by creating different piles as the nature of the work to be done changes. These two kinds of modification are called annotating and cognitive tracing:

- Annotating involves modifying external representations, such as crossing off underlining items.
- Cognitive tracing involves externally manipulating items different orders or structures.

Information Visualization

A general cognitive principle for interaction design based on the external cognition approach is to provide external representations at the interface that reduce memory load and facilities computational offloading. Different kinds of information visualizations can be developed that reduce the amount of effort required to make inferences about a given topic (e.g., financial forecasting, identifying programming bugs). In so doing, they can extend or amplify cognition, allowing people to perceive and do activities tat they couldn’t do otherwise.

Distributed cognition

Distributed cognition is an emerging theoretical framework whose goal is to provide an explanation that goes beyond the individual, to conceptualizing cognitive activities as embodied and situated within the work context in which they occur. Primarily, this involves describing cognition as it is distributed across individuals and the setting in which it takes place. The collection of actors (more generally referred to just as ‘people’ in other parts of the text), computer systems and other technology and their relations to each other in environmental setting in which they are situated are referred to as functional systems. The functional systems that have been studied include ship navigation, air traffic control, computer programmer teams and civil engineering practices.

A main goal of the distributed cognition approach is to analyze how the different components of the functional system are coordinated. This involves analyzing how information is propagated through the functional system in terms of technological cognitive, social and organizational aspects. To achieve this, the analysis focuses on the way information moves and transforms between different representational states of the objects in the functional system and the consequences of these for subsequent actions.

INTERACTION STYLES AND INTERFACES

Interaction styles refer to the different ways of communication between a human and a computer based on a technological platform through interaction techniques which are “way of using a physical input/output device to perform a generic task in a human-computer dialogue” (Foley at al., 1990). Interaction style is explained “through prototypical elements of the interface and how they behave, for instance command line, pull down menu, form fill in, or direct manipulation” (Shneiderman, 1992).

The following fundamental interaction styles and interfaces are used:

a) Command line languages

This popular category covers the interaction between humans and computers using language by typing the commands to a computer which prompts a message meaning ready to accept input. It provides means of expressing instructions to the computer directly, using function keys, single characters, abbreviations, or whole word commands. The command line interfaces are powerful in that they offer direct access to the system functionality and can be combined to apply a number of tools to the same data. The command lines interactions are disadvantageous because text commands are usually difficult to learn and use as cryptic keywords and a strict associated syntax which a user has to know before using the system and usually this influences to an increase rate of errors. They must be remembered. Mnemonics only can be used as cues. They are therefore better for expert users than for novices.

b) Menus

Menus are defined as set of options on screen for choosing the action or among options for data entry. There are three types of menus Shneiderman, B (1992):

- Pull-down menus
- Pop-up menus
- Hierarchical menus

(Preece, 1994) defines a menu as “a set of options displayed on the screen where the selection and execution of one (or more) of the options results in a change in the state of the interface.

Unlike command-driven systems, menus have the advantage that users do not have to remember the item they want, they only need to recognize it” (Preece, J. 1994). The advantage of using menus is that user needs to recognize rather than recall objects. The menu options need to be grouped logically and meaningful, so the user could easily recognize the needed option. Although traditionally the user clicks with a mouse over the item to be selected or using a keyboard, with the new hardware technologies developed the user can as well respond via voice command. There is evidence that the number of errors decrease, time to perform a task is shorten unless for complex tasks that need more operations to perform, the navigation through menus to find the necessary option needs more time.

c) Direct manipulation

Direct manipulation interfaces are very popular and successful, especially with new users, because they embed manipulations that are analog to human skills (pointing, grabbing, moving objects in space), rather than trained behaviors and “users have great control over the display and as they select items, the details appear in windows on the slides” (Shneiderman & Maes 1997) Shneiderman B, Maes P., (1997).

Direct manipulation interfaces “present a set of objects on a screen and provide the user a repertoire of manipulations that can be performed on any of them” (Shneiderman, 1983).

Each operation on the interface is done directly and graphically. From programming aspect, writing a program is done by moving icons onto the screen and connecting them together. The “editing-compiling –running” cycle is simply realized by directly clicking icons on the screen instead of strictly syntax-ed commands or operations. There is no need to remember the command name end syntax. This leads to decreasing syntax errors like you cannot compile non-existing code since it is not on the screen when you click the compile icon and faster performance of a task.

According to (Shneiderman, 1983), these kinds of manipulations have some meanings:

1. Continuous representation of the object of interest.
2. Physical actions or labeled button presses instead of complex syntax.
3. Rapid incremental reversible operations whose impact on the object of interest is immediately visible.

Shneiderman (1982) numbers the following advantages of direct manipulation to objects:

1. Novices can learn basic functionality quickly, usually through a demon-
2. Experts can work extremely rapidly to carry out a wide range of tasks,
3. Knowledgeable intermittent users can retain operational concepts.
4. Error messages are rarely needed.
5. Users can see immediately if their actions are furthering their goals, and if not, they can simply change the direction of their activity.

d) Form fill-in

It is “the simplest style of interaction that consists of the user being required to answer questions or fill in numbers in a fixed format rather like filling out a form” (Shneiderman, 1992). In this form, the only kind of user interaction is the provision of information which is useful for data entry into applications. Also spreadsheets are considered as a sophisticated variation of form filling.

e) Natural Language

The researchers and practitioners are more interested in systems that use natural language processing as style of human-computer communication, both of speech and written input.

In the case of speech input, the user must learn which phrases the computer understands since computer requires strict instructions and users may become frustrated if too much is expected. The advantage of using this interaction style is to users that do not have access to keyboards or have limited experience. While ambiguities of the language may cause unexpected effects and makes very difficult for a computer to understand.

A good perspective is that “Natural Language systems should be extended to include non-verbal dialogues”, since he argues that "Natural" language includes gestures. Gestures can be used to form clear fluid phrases, and multi-threaded gestures can capitalize on the capabilities of human performance to enable important concepts to be expressed in a clear, appropriate, and "natural" manner” (Buxton,1990).

Natural Language interactions are “a perspective on Non-Verbal Dialogues because they are in many ways, more natural than those based on words” (Buxton,1990).

f) Question/answer and query dialogue

A simple mechanism for providing input to an application in a specific domain. The user is asked a series of questions (mainly with yes/no responses, multiple choice or codes) and so is led through the interaction step by step. These interfaces are easy to learn and use, but are limited in functionality and power.

Query languages on the other hand are used to construct queries to retrieve information from a database.

g) WIMP interface

WIMP stands for windows, icons, menus, and pointers (sometimes windows, icons, mice, and pull-down menus). These interfaces are probably the most popular and influential for interactive environments. Windows are areas of the screen that behave as if they were independent terminals in their own right. An icon is a small picture used to represent a closed window, file, or any other object. The pointer is important component of a WIMP interface, since it interfaces the pointing, clicking, pressing, dragging and selection of objects on the screen which could be moved, edited, explored and executed as it better fits to the user's vision. Other tools of computer interface design are menus, dialog boxes, check boxes, and radio buttons and so on. These make use of visualization methods and computer graphics to provide a more accessible interface than command-line-based displays. The fundamental goal of WIMP designs is to give the user a meaningful working metaphor, for example an office or 'desktop' representation as opposed to the command-line interfaces. Its advantages are general application, make functions explicit and provide immediate feedback.

Humans are highly attuned to images and visual information that in other hand can communicate some kinds of information much more rapidly and effectively than any other method., and as is said "a picture is worth a thousand words ".

h) Virtual Reality

"Virtual environments and virtual realities typically offer a sense of direct physical presence, sensory cues in three dimensions, and a natural form of interaction (for example via natural gestures)" (Preece, J. 1994).

Besides these styles, new interaction styles have emerged: “speech input/output, computer vision based input (e.g., gestures), audio interfaces (e.g., non-speech audio), tactile and force feedback, biophysical signals (e.g., retina scanner)” (Rauterberg, 2003) which bring us the new generation of interfaces that are non-command-based with interactions like eye tracking interfaces, artificial realities, play-along music accompaniment, and agents.

Input/Output

The conventional input devices used are keyboard, mouse and visual display that are used in command based interactions.

With emerging of new hardware technologies new input devices are used like cameras, haptic sensors, olfactory, microphones and others.

The new input technologies used are speech recognition, gesture recognition technologies, eye tracking technology as non command based interaction, techniques for communication and manipulation of multidimensional data;

Output devices used are the conventional computer desktop display, Head-mounted displays, autostereoscopic displays, touchable three-dimensional displays, non-speech audio output for ‘visualizing’ data etc.

Mental (or conceptual) models

Users form mental models or conceptual models of tasks and systems. These are used to guide behavior at the interface. When people encounter new machines, devices or computers, they begin to construct mental models to represent their behavior and operation. These internal models provide a means by which people can understand and predict the world around them. But, these models are individual and very subjective. Every user forms a mental model that depends on number of psychological, cognitive, cultural, educational, and other human factors. This means that users may form different models for one system that cannot be predicted in designing the system. Even though the research literature has shown that the user using own knowledge after experiencing the system forms more precise and representative model of the system that is working with; we construct these models as we go along and as a consequence our models tend to be incomplete, unstable, do not have firm boundaries, and are unscientific.

Theories and cognitive models

Some may argue that HCI does not need theory. Any discipline that fails to make a principled explanation to justify its practice is building on sand. The HCI's problem is that its theories are shared with and, in many cases, borrowed from cognitive science. The cognitive science theories are complex, "big science" endeavors that can only be carried forward by communities of researchers, notably ACT-R (Anderson, J. R. and Lebiere, C. 1998) and SOAR (Newell, 1990). Both of these theories have been applied to HCI problems, but the range of phenomena that they can account for is narrow. According to (Sutcliffe, 2000) cognitive theories, implemented as computational cognitive models, have a problem of scale.

However, this is away from predicting similar user behavior in a complex multimedia system. The EPIC model (Kieras, D. E. and Meyer, D. E. 1997) provides an architecture of perceptual and cognitive processors with rules that predict the user's attention, recognition, and understanding of user interface features. While EPIC can accurately predict user performance and behavior with simple user interfaces (i.e., searching menu displays), it suffers from an increasing burden of configuration as the complexity of the external artifact is increased.

Conclusion

In order to design a good human computer interaction, we have to appropriately choose the type of interface and interaction style to fit with the class of users it is designed whereas the human factors must be taken in consideration (Fetaji, M., at al., 2007). Thereby, we recommend the following:

- to investigate the advantages and disadvantages of interaction styles and interface types that best support the activities and styles of learning of users the system is aimed at;
- to choose the type of interface and interaction styles that best supports the system goals;
- to choose the interaction styles that are compatible to user attributes and that support the users needs, which means to choose the styles that are more advantageous for aimed users (for example, in a system for learning and practicing programming, direct manipulation style is more advantageous which are stressed in more detail in section 3.1); and
- to define the user class (experts, immediates or novices) that the system is designed for, where the human factors must be taken in consideration.

Incorporating HCI design principles, we can ensure better design guidance for screen layout, menu organization, or color usage according to users attributes. We recommend similar human-computer interaction design to similar solutions.

TASK ANALYSIS

Introduction

Task Analysis is the process of analyzing the way people perform their jobs. Aim is to determine:

- what they do
- what things they use
- what they must know

Task analysis is about the existing systems and procedures. Task analysis addresses what is done, not what should be done. The main purposes of task analysis include:

- To help in the production of training materials and documentation hence analysis of the existing system.
- Where a new system is required, task analysis contributes to the statement of requirements for this system.

Task analysis is used to:

- Predict the time taken to learn a new task and become a proficient user of the particular application / machine. Task analysis may reveal how difficult one method is to learn compared to another.
- Predict the time taken for a proficient user to accomplish the set task - this can reflect whether the interface is good at supporting exploration. Is it quicker to simply explore by trial and error or attempt to find out through help?.

- Predict the time taken for expert execution of the set task - how long does it take to become expert once a procedure has been discovered? This can be affected by the design of an interface.

The latter point is an important application of task analysis. In 'Phoneline' banking or telephone directory enquiries, for example, the time spent with each customer can be reduced by milliseconds as a result of task analysis. Here speed, accuracy and efficiency are vital and task analysis comparison studies help determine a more effective interface design.

Task analysis gathers both *declarative* and *procedural* knowledge

- Declarative: objects and relationships
- Procedural: task sequences, goals and sub-goals
- Also dependencies and constraints

Task analysis emphasizes users and existing tasks, rather than desired system as in **systems analysis.** It emphasizes observable behaviour and whole job, rather than internal mental state and “unit” tasks as in **cognitive models**

Three Types of Task Analysis

There are three different categories/approaches to task analysis:

1. **Task decomposition –** focuses on the way a task can be divided into sub-tasks
2. **Knowledge-based techniques –** focuses on what needs to be known to accomplish a task and how that knowledge is organized
3. **Entity relationship-based analysis –** focuses on the objects involved, the actions that can be performed with them, and the people who perform them e.g. gardener *digs* soil *using* spade

The general method for Task Analysis is:

- observe
- collect unstructured lists of words and actions
- organize using notation or diagrams

a) Task Decomposition

Aims:

- to describe the actions people do
- structure them within task subtask hierarchy
- describe order of subtasks

Task decomposition process, in rough outline, proceeds as follows:

1. Select a task
2. Based your data (from observation, documentation, and expert advice), divide the task into sub-tasks.
3. If your stopping rule has not been reached, repeat steps 1-3 for each of the new sub-tasks.

Task analysis techniques include:

- Hierarchical Task Analysis (HTA) – the most common
- ConcurTaskTrees (CTT) by Paternò (2000) – uses LOTOS temporal operators

The output of HTA is a hierarchy of tasks and sub tasks and also plans describing in what order and under what conditions subtasks are performed.

Example:

Textual HTA

Hierarchy description

0. clean the house

a. get the vacuum cleaner out
b. get the appropriate attachment
c. clean the rooms

    3.1clean the hall
    3.2clean the living rooms
    3.3clean the bedrooms

4. empty the dust bag
5. put vacuum cleaner and attachments away

Plans

Plan 0: do 1, 2, 3, 5 in order; when dust bag full, do 4

Plan 3: do 3.1, 3.2, 3.3 in any order, as needed

Diagrammatic HTA

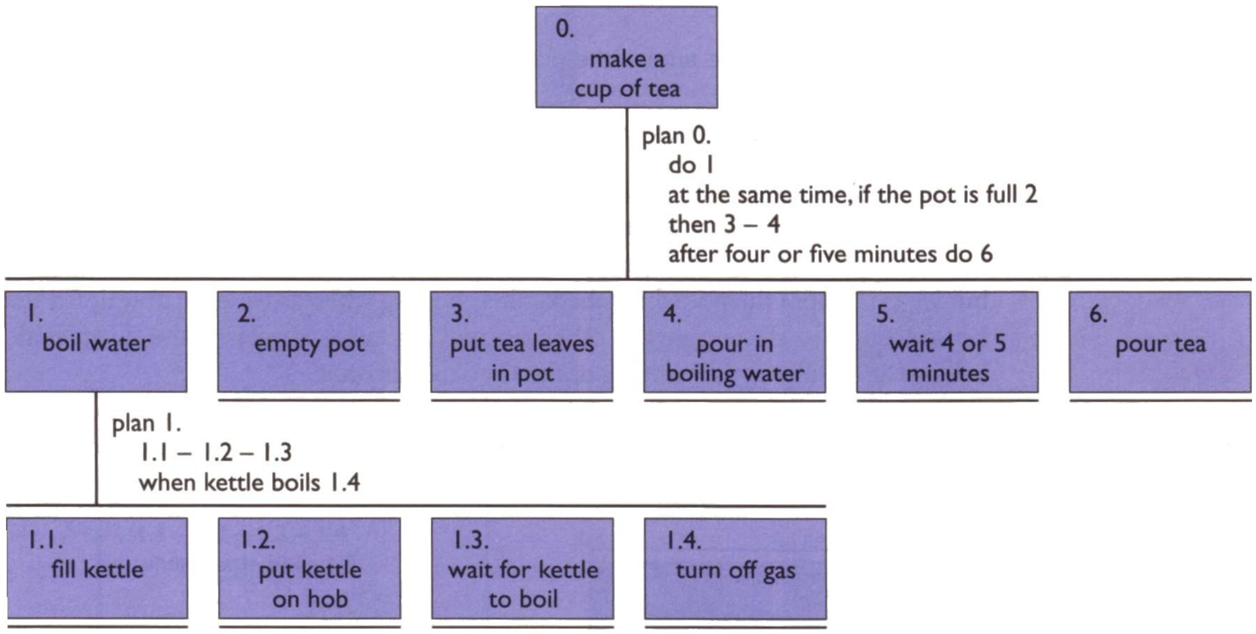


Figure: Hierarchical task analysis (making a cup of tea)

Refined HTA for making tea

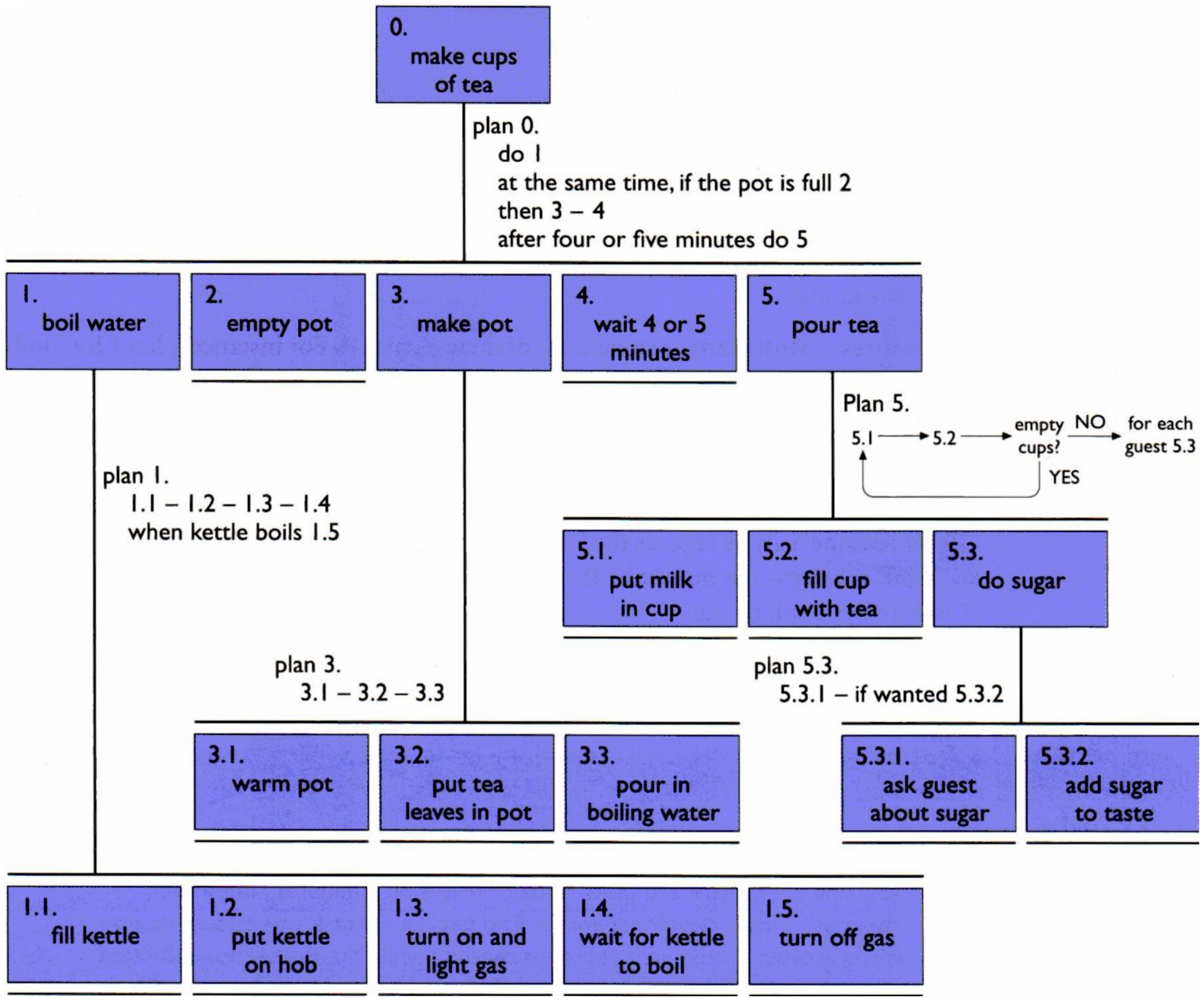

0. make cups of tea
plan 0. do 1 at the same time, if the pot is full 2 then 3 – 4 after four or five minutes do 5
1. boil water
2. empty pot
3. make pot
4. wait 4 or 5 minutes
5. pour tea
Plan 5. 5.1 → 5.2 → empty cups? NO → for each guest 5.3; YES → 5.1
plan 1. 1.1 – 1.2 – 1.3 – 1.4 when kettle boils 1.5
5.1. put milk in cup
5.2. fill cup with tea
5.3. do sugar
plan 3. 3.1 – 3.2 – 3.3
plan 5.3. 5.3.1 – if wanted 5.3.2
3.1. warm pot
3.2. put tea leaves in pot
3.3. pour in boiling water
5.3.1. ask guest about sugar
5.3.2. add sugar to taste
1.1. fill kettle
1.2. put kettle on hob
1.3. turn on and light gas
1.4. wait for kettle to boil
1.5. turn off gas

Figure: Modified task hierarchy for making lots of cups of tea

Types of plan

i. **sequence** 1.1 then 1.2 then 1.3
ii. **optional** if the pot is full 2
iii. **wait** when kettle boils, do 1.4
iv. **cycles** do 5.1 5.2 while there are still empty cups
v. **parallel** do 1; at the same time ...

vi. **discretionary** do any of 1.3.1, 1.3.2 or 1.3.3 in any order

Most plans use several of these.

Waiting can be considered:

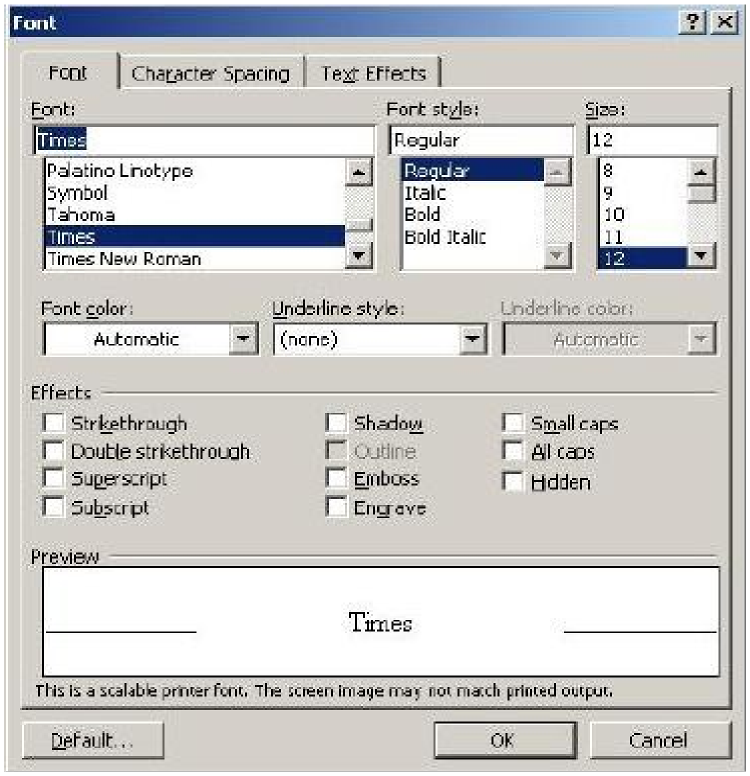
- a task — for “busy” waits, e.g. making tea
- part of the plan — end is the event, e.g. email reply received

b) Knowledge Based Analyses

Knowledge based analysis is *bottom-up*, grouping simple actions and objects into classes by similarity. The aim is to understand knowledge required to perform a task and thus to help in the production of training materials (how-to manuals) and to take advantage of common knowledge across tasks.

Knowledge based analysis focus on objects used in task and the actions performed in the task. The starting point for the process is a list of all of the objects and actions that are relevant to the task that is being analyzed, based on data collected (both what objects and actions subjects were observed to use, and what they mentioned). The objects are then arranged into groups based on similarity or shared traits. The traits that are used to determine groupings depend entirely upon the task that is being analyzed and the purpose of the analysis.

As groups are formed from the items, the groups themselves are then grouped together, building progressively more abstract categories. The structure this process eventually arrives at is termed a ‘*taxonomy’* - a naming hierarchy. Depending on the purpose of the analysis, each item might have a unique 'path' in the hierarchy (appearing only once), or some might belong to multiple groups. Some techniques require that every item be uniquely describable by the groups it falls under, and that if two items are in exactly the same set of groups, either a new group be created to define the difference between the items or the items be combined.


Font: Times
Font style: Regular
Size: 12
Font color: Automatic
Underline style: (none)
Underline color: Automatic
This is a scalable printer font. The screen image may not match printed output.

1. Start subject off with a seed item: **type faces**
2. Move around task domain knowledge using prompts:

- To move down: *Can you give examples of type faces?*
- To move across: *What alternatives are there to type faces for changing the appearance of text?*
- To move up: *What have Times Roman, Helvetica in common?*

There is one technique for knowledge based analysis; *task analysis for knowledge description (TAKD)*. TAKD uses a special form of taxonomy called *task descriptive hierarchy*.

Note:

- Objects are often more easily observed than tasks!
- TDH decomposes by similarity, HTA by how-to.

Entity-Relationship Based Analysis

Entity-relationship analysis is also a bottom-up approach to task analysis. It inherits much of its structure from object-oriented programming, and is the most 'computer-like' of the three varieties of task analysis being presented here. Entity-relationship analysis concerns itself with objects, actions, and their relationship. A good way to start is to examine the data you collected - interview transcripts especially - and pull out every relevant noun (objects) and verb (actions).

Objects are divided into four categories: simple objects, actors, composite objects, and events. Each simple object may have attributes or qualities associated with it - size, weight, or color, for example. Actors (which are typically though not always humans) have attributes, but are also associated with a list of actions which they perform. Composite objects are composed of a group of simple objects or actors. Actors perform actions upon objects, possibly changing the attributes of the object in the process (for instance, changing the status of a device from 'off' to 'on'). Events are things that take place spontaneously, either randomly or caused by actions outside of the model, which objects and actors react to.

The second half of object-relationship analysis is the relationship between objects. Objects can be related to each other in terms of physical location, or to actions that are either performed with them or on them. Actions may cause other actions, or be triggered by events. The variety of possible events is large, and depends upon the nature of the task.

Uses of Task Analysis in HCI

Understanding the user's previous behavior aids in the design of learnable systems.

In building a system, it is wise (as Nielsen's heuristics remind us) to 'match between the system and the real world'. That is, make the system compatible with what the user already does, so that its introduction draws upon already learned skills and fits smoothly into the rest of the work environment.

Analysis can lead to compatibility with real-world tasks and with user expectations.

Step one of this process is understanding what it is that the user has been doing. The new system can then be arranged in a way that is compatible with the user's accustomed behaviors (learned through task analysis), presented in a way that matches the way the user mentally classifies the actions and items involved (gathered through knowledge-based analysis), or include logical objects and the actions associated with them (discovered through object-relationship analysis). This could take place anywhere between the level of broad system design (selection of the tasks and functions to be included in a new system) to detailed interface decisions (the arrangement of menu options based on taxonomy, or on frequency of a given subtask).

Task information can be used to build documentation.

In addition to guiding system design, that information can also be used for the creation of documentation - either documenting the task that was analyzed, or presenting instructions for a new task or system in a way that parallels a familiar one, for ease of comprehension. Task decomposition lends itself to the creation of instruction manuals, while knowledge and object-relationship based analysis are more appropriate to reference guides.

Conclusion

The purpose of task analysis is to provide insight into the nature of a given task.

Applying Task Analysis

- For documentation: **How To** manual
    - useful for novices
    - assumes all tasks known
- Requirements capture and design
    - lifts focus from system to use
    - suggests candidates for automation
    - may uncover user's conceptual model
- Detailed interface design
    - taxonomies suggest menu layout
    - object/action lists suggest interface objects
    - task frequency guides default choices
    - existing task sequences guide dialogue design

Task analysis can be continually iterated to improve and enhance.