UNIT- I:

Interaction Paradigms: Computing Environments, Analysing Interaction Paradigms, Interaction Paradigms

Interaction Frameworks and Styles: Frameworks for Understanding Interaction, Coping with Complexity, Interaction Styles.

INTRODUCTION

Human–computer interaction (HCI), alternatively man–machine interaction (MMI) or computer–human interaction (CHI) is the study of interaction between people (users) and computers.

DEFINITION

• "Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them."

GOALS

• A basic goal of HCI is

– to improve the interactions between users and computers

– by making computers more usable and receptive to the user's needs.

• A long term goal of HCI is

– to design systems that minimize the barrier between the human's cognitive model of what they want

– to accomplish and the computer's understanding of the user's task.

DEFINING THE USER INTERFACE

• User interface, design is a subset of a field of study called human-computer interaction (HCI).

• Human-computer interaction is the study, planning, and design of how people and computers work together so that a person's needs are satisfied in the most effective way.

• HCI designers must consider a variety of factors: – what people want and expect, physical limitations and abilities people possess, --how information processing systems work, – what people find enjoyable and attractive. – Technical characteristics and limitations of the computer hardware and software must also be considered.

• The user interface is to – the part of a computer and its software that people can see, hear, touch, talk to, or otherwise understand or direct.

• The user interface has essentially two components: input and output.

• Input is how a person communicates his / her needs to the computer. – Some common input components are the keyboard, mouse, trackball, one's finger, and one's voice.

• Output is how the computer conveys the results of its computations and requirements to the user. – Today, the most common computer output mechanism is the display screen, followed by mechanisms that take advantage of a person's auditory capabilities: voice and sound.

• The use of the human senses of smell and touch output in interface design still remain largely unexplored.

• Proper interface design will provide a mix of well-designed input and output mechanisms that satisfy the user's needs, capabilities, and limitations in the most effective way possible.

• The best interface is one that it not noticed, one that permits the user to focus on the information and task at hand, not the mechanisms used to present the information and perform the task.

HCI is critical in the development of software and hardware systems, you might have a powerful software application with many features but if the user is not able to operate it easily, he or she will discard it after few trials. In order to improve the usability of a software package, HCI specialists endeavour to:

• Understand psychological, organizational, and social factors of the combined human and computer system.

• Develop methodologies to aid appropriate HCI design.

• Realize efficient and effective interactions for single users and groups.

Interaction Paradigms:

Computing Environments:

• Physical Computing Environment

• Social Computing Environment

• Cognitive Computing Environment

Physical Computing Environment:

– Safety

– Efficiency

– User Space

– Work Space

– Lighting Noise

– Pollution.

• Social Computing Environment:

– The social environment affects the way people use computers.

– Computer use has also been shown to affect human social interaction.

– Different computing paradigms imply different social environments.

• For instance, personal computing is usually a solitary activity done in an office or an isolated corner of the house. Mobile computing is often done outside and in public places

• Cognitive Computing Environment:

– Age

– Disabilities

– Degree of technical knowledge

– Degree of focus

– Cognitive Stress.

Analysing Interaction Paradigms:

• 5W + H

The —who, what, where, why, and how (5W+H) heuristic is a procedure that can be used to define and analyse existing interaction paradigms and spaces and explore the elements and objects with which the user interacts.

The heuristic has three components:

• The What/How : This is used to understand the physical and virtual interface components. For example, I/O devices, windows, icons, etc.

• Where/When: This is related to physical environment. It looks at the differences between office, portable, wearable systems.

• Who/Why: This looks at the types of tasks and skill sets required.

Interaction Paradigms:

An interaction paradigm is a model or pattern of human–computer interaction that encompasses all aspects of interaction, including physical, virtual, perceptual, and cognitive.

• Large Scale Computing

• Personal Computing

• Networked Computing

• Mobile Computing

• Collaborative Environments

• Virtual Reality

• Augmented Reality.

The 5W+H heuristic is used to describe the paradigms indicated below.

Large scale computing paradigm:

What/How- The large scale computing paradigm includes mainframe computers that were large computers and they resided in a central location. These computers were accessed by remote alphanumeric terminals (—dumb terminals‖) equipped with keyboards. These started as electronic typewriter and later developed to CRT screens. Large-scale computing includes super computers which are specialized machines that crunch large amounts of data at high speed, as in computing fluid dynamics, weather patterns, seismic activity

predictions, and nuclear explosion dynamics. Display, other then text, was produced on special and expensive devices such as plotters or customized CRTs.

When/Where- Most large computers were owned by government agencies and large research institutes affiliated with large universities.

Who/Why- Large-scale computing resources are expensive and generally used to carry out government sponsored-research projects and university-based research for large corporate institutions. Supercomputers are used for the very high speed backbone (vBNS) connections that constitute the core of the Internet while mainframes are still in use in the financial industry.

Personal computing paradigm:

The personal computing paradigm is driven by Graphical User Interfaces (GUI) and found in commercial operating systems such as Windows and Macintosh. The Alto computer developed at the Xerox Palo Alto Research Center in 1973, was the first computer to use a GUI that involved the desktop metaphor: pop-up menus, windows, and icons. This was later adopted by Apple for their Lisa and Macs machines, and later on by the Windows operating system of Microsoft. The development started the personal computer paradigm that includes Desktop and Personal-Public Computing.

Networking computing paradigm

What/How- A network is a communications, data exchange, and resource-sharing system created by linking two or more computers and establishing standards, or protocols, so that they can work together. Networks can be classified based on their scope: Wide Area Network (WAN), Metropolitan Area Network (MAN), Local Area Network (LAN) and Personal Area Network (PAN). Networks can also be classified based on their transmission media: Wired Media and Wireless Media. This paradigm is enabled by several technologies Ethernet and TCP/IP protocols, personal computer and telephone networks and modems

Where/When- Computer networks have freed users from location-based computing since users can access internet based systems like e-mail and web browsers from any location that has internet access. Network resources can be accessed at any time without restriction.

Who/Why- People from all backgrounds and ages are using the internet on an every day basis. From banking to online games, users worldwide access network based applications for a wide range of interests.

Mobile computing paradigm:

What/How- Mobile computing paradigm includes technologies that comprise a very diverse family of devices. This paradigm includes the devices mentioned below: Laptop computers

 Tablet computers Game players MP3 players MP4 Players PDAs Wearable computers Cell phones .

Mobile devices can be connected to global positioning systems (GPS). These have touch screens and voice interaction to alleviate potential visual attention problems during driving.

Where/When- Mobiles devices liberate users from the need to be at a specific location. Wi-Fi internet access is nowadays available at most of the public places including airports, coffee shops, hotels and malls. Wearable computers can be worn by users at any time and any place and ideal for health monitoring application. Mobile devices can offer situational computing that can take advantage of location-specific information through location-based mobile services (LMS). LMS can be beneficial for location-sensitive advertisements, public service announcements, social interactions, and location-specific educational information.

Who/Why- Business users benefit greatly from mobile computing. The ability to have access to e-mail and remote databases when being away from the office, is of great benefit to business in general. However, mobile computing has potential for most of the aspects of the human life as e-commerce, health care, entertainment among others.

Collaborative environment paradigm:

What/How - In a collaborative environment paradigm, computer networks allow members of a group to interact with other members on shared files and documents. This creates a virtual space where people can collaborate and work collectively with the use of Computer mediated communication (CMC).

A groupware is a CMC that allows remote interaction by using synchronous technologies such as video conferencing, instant messaging and chat rooms or asynchronous technologies such as e-mail, bulleting boards and discussion groups.

An example of a collaborative environment is the collaboratory. Wulf (1989) called the collaboratory —a center without walls, in which the nation‘s researchers can perform their research without regard to physical location-interacting with colleagues, accessing instrumentation, sharing data and computational resources, and accessing information in digital libraries.

Where/When- Computer supported cooperative work (CSCW) (Galegher, Kraut, & Egido, 1990) is computer-assisted coordinated activity such as problem solving and communication carried out by a group of collaborating individuals. The multi-user software supporting CSCW systems is known as groupware. A groupware can be based on remote interaction using synchronous technologies like video conferencing or asynchronous technologies like email and discussion groups.

Who/Why- CMC is used in business in order to eliminate transportation costs since it helps to collaborate with remote customers or employees. Engineering benefits as well from this

paradigm as it facilitates the collaboration of engineering teams spread at different locations. Universities use CMC to facilitate e-learning and research, an example of this is the Research Collaboratory for Structural Bioinformatics (RCSB, 2005).

An obvious example to the collaborative paradigm is our online MSc in computing programs that you are using, as well as other online education projects.

Virtual reality paradigm:

What/How -The virtual reality paradigm offer users a computer simulated alternative to the real world. Virtual reality technologies can be divided into two distinct groups:

- Nonimmersive environments
- Immersive environments

Non-immersive environments are screen-based, pointer-driven, three-dimensional (3D) graphical presentations that may involve haptic technology feedback as Virtual Reality Virtual Modelling language and QuickTime VR. Haptic technology refers to technology which interfaces the user via the sense of touch by applying forces, vibrations and/or motions to the user. Successful systems that provide 3D virtual world experience are the Second Life and the Active Worlds Internet offerings.

Immersive VR environments are designed to create a sense of —being‖ in a world populated by virtual objects. To create a convincing illusion, they must use as many human perceptual channels as possible.

Virtual Reality immersive I/O devices include:

- Head Mounted Display (HMD)
- Spatial Immersive Display (SID)
- Cave Automated Virtual Environment (CAVE)
- Head-movement-tracking systems
- Passive systems
- Active locomotion systems

When/Where-VR systems are found in large scientific research laboratories in the government and in engineering firms.

Who/Why- VR offers benefits in fields like aerospace, medical, military and in general domains that require expensive environments for training. One example of a company that develops VR systems for flight simulators in the military is CAE electronics in Montreal

Canada. In Psychology it has application for the treatment of phobias. VR is also used to implement Computer-aided design software used in systems as Cadence used for IC design.

Augmented Reality paradigm:

What/How- The goal of Augmented Reality (AR) is to create a seamless integration between real and virtual objects in a way that augments the user's perception and experience. Azuma (1997) defines an augmented reality system as one that

- combines real and virtual
- is interactive in real time
- is registered in 3D

Criteria for AR environments are that the virtual information must be relevant to and in synchronization with the real-world environment.

AR I/O devices include:

- Heads Up Displays (HUD) (E.g. Critical data viewer)
- Motorcycle helmets (E.g. Sportvue)

Where/When- AR technology is applicable in situations in which people need access to computing functionality and cannot afford to leave their work site. Emergency professionals as firemen and police could benefit from such systems since they could access information as building blueprints that could help in an emergency situation.

Who/Why- Current applications include military and emergency applications services (e.g. showing maps, instructions, enemy locations, fire cells), simulation (e.g. flight and driving simulators), navigation devices (e.g. cars and airplanes), games (e.g. ARQuake) and education among others.

Interaction frameworks and styles:

Frameworks for Understanding Interaction:

Execution–Evaluation cycle:

Norman's model of interaction is perhaps the most influential in Human–Computer Interaction, possibly because of its closeness to our intuitive understanding of the interaction between human user and computer. The user formulates a plan of action, which is then executed at the computer interface. When the plan, or part of the plan, has been executed, the user observes the computer interface to evaluate the result of the executed plan, and to determine further actions. The interactive cycle can be divided into two major phases: execution and evaluation.

These can then be subdivided into further stages, seven in all. The stages in
1. Establishing the goal.
2. Forming the intention.
3. Specifying the action sequence.
4. Executing the action.
5. Perceiving the system state.
6. Interpreting the system state.
7. Evaluating the system state with respect to the goals and intentions.

It is liable to be imprecise and therefore needs to be translated into the more specific intention, and the actual actions that will reach the goal, before it can be executed by the user. The user perceives the new state of the system, after execution of the action sequence, and interprets it in terms of his expectations. If the system state reflects the user's goal then the computer has done what he wanted and the interaction has been successful; otherwise the user must formulate a new goal and repeat the cycle.

Norman uses this model of interaction to demonstrate why some interfaces cause problems to their users. He describes these in terms of the gulfs of execution and the gulfs of evaluation. The user and the system do not use the same terms to describe the domain and goals – remember that we called the language of the system the core language and the language of the user the task language. The gulf of execution is the difference between the user's formulation of the actions to reach the goal and the actions allowed by the system. If the actions allowed by the system correspond to those intended by the user, the interaction will be effective. The interface should therefore aim to reduce this gulf. The gulf of evaluation is the distance between the physical presentation of the system state and the
expectation of the user. If the user can readily evaluate the presentation in terms of his goal, the gulf of evaluation is small. The more effort that is required on the part of the user to interpret the presentation, the less effective the interaction.

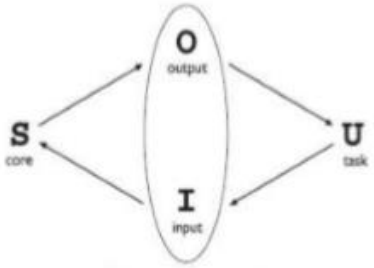
The interaction framework:

The interaction framework attempts a more realistic description of interaction by including the system explicitly, and breaks it into four main components. The nodes represent the four major components in an interactive system – the System, the User, the Input and the Output. Each component has its own language. In addition to the User's task language and the System's core language, which we have already introduced, there are languages for both the Input and Output components. Input and Output together form the Interface. The general interaction framework Translations between components The System then transforms itself as described by the operations; the execution phase of the cycle is complete and the evaluation phase now begins. The System is in a new state, which must now be communicated to the User. The current values of system attributes are rendered as

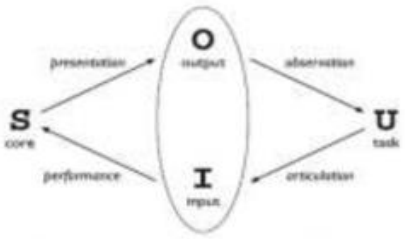
concepts or features of the Output. It is then up to the User to observe the Output and assess the results of the interaction relative to the original goal, ending the evaluation phase and, hence, the interactive cycle. There are four main translations involved in the interaction: articulation, performance, presentation and observation.

Assessing overall interaction

The interaction framework is presented as a means to judge the overall usability of an entire interactive system. This is not surprising since it is only in attempting to perform a particular task within some domain that we are able to determine if the tools we use are adequate. For a particular editing task, one can choose the text editor best suited for interaction relative to the task. The best editor, if we are forced to choose only one, is the one that best suits the tasks most frequently performed. Therefore, it is not too disappointing that we cannot extend the interaction analysis beyond the scope of a particular task.



The general interaction framework



Translations between components

Coping with Complexity:

Interaction design is a complex task due to the level of abstraction required to understand the user‘s viewpoint. Different tools can be used to cope with this complexity, including mental models, mapping, semantic and articulatory distance and affordances (Heim 2007).

Norman (1983) introduced the concept of user‘s mental model of a computer system with which the user is interacting. This computer system should communicate the conceptual model, which is an accurate, consistent, and complete representation of the target system held by the designer, or an expert user, of the system. The user‘s mental model is formulated through interaction with the computer system, and constantly modified throughout the interaction. This model is a cognitive representation of something that defines a logical and believable estimation as to how an object is constructed or how it functions. Metal models can help to model the user‘s view. This model can be used to align the interaction‘s design with the user‘s view in order to facilitate its use.

Young (1983) introduced task-action mapping models. Young linked the internalised representation of the system to the real-world task which users have to perform. Task-action mappings describe the structure of a real-world task and the actions needed to perform that task, and provide a direct mapping between the task and the corresponding actions. Young (1981) uses the example of an algebraic calculator for a mapping model: the

operations that have to be performed can be mapped onto doing the same task with paper and pencil. A task-action mapping model would allow competent use of a system for a particular task, even though the user has no detailed knowledge of the system and how it works.

Another two important tools used in interaction design are: semantic and articulary distances. Semantic distance is a tool that measures the distance between what people want to do and the meaning of an interface element, while the articulatory distance measures the distance between the physical appearance of an interface element and what it actually means (Heim 2007).

Affordances are tools that represent connections that allow us to make predictions about the results of our actions and help us to create usable mental models. These connections are user‘s interpretations of the interface, these can be obvious such a turning wheel that affords turning but sometime confusing. This interface shows text boxes used as labels for a log in interface that allow input when the purpose to provide an output.

Interaction Styles:

The way users interact with computers is referred to as interaction style (Heim 2007). A list of interaction styles is given below:

- Command Line
- Menu-Based Interface
- Form Fill-In
- Spreadsheets
- Question and Answer
- Direct Manipulation
- Metaphors
- Web Navigation
- Three-Dimensional Environments
- Zoomable Interface
- Natural Language

Command Line:

Command-line style is used by Operating Systems such as Linux, Unix and MS-DOS and it is mainly based on text commands. It has the advantage of being very fast compared to other

more graphical interfaces. In addition, Command-Line interfaces tend to be very flexible since commands can be created with multiple parameters that can be set and altered for multiple applications. It is also suitable for repetitive tasks such as the creation of batch files. However, it is more suitable for expert users since it can be frustrating for the novice user given its learning curve. It also has poor error handling and requires substantial training and memorization.

From the EECA perspective, the Command-Line shows that intention formation, specification of the action, and the execution stages are complex. It also, requires a rather accurate mental model of the computer‘s internal processing. From the Interaction Framework perspective, it translates the user‘s task language into the input language. It requires knowledge of the core language but the output language can be confusing for inexperienced users since it provides little feedback (Hem 2007).

Menu-Based Interface:

Menu-driven interfaces present users with a list of alternatives or options. Textual menus allow menu selection by keying in the number or letter that is associated with that option while graphical menus use arrow keys or pointing devices such as pen or mouse.

Graphical menus can have different forms. Pull down menus provide a vertical list of options to users and cascading menus must be requested by the user from another higher level menu. Some designers use pop-up and iconic menus in order to save screen space.

Menus can be Ideal for novice or intermittent users. However, they can appeal to expert users if display and selection mechanisms are rapid and if appropriate "shortcuts" are implemented. Menus can afford exploration (users can "look around" in the menus for the appropriate command, unlike having to remember the name of a command and its spelling when using command language) and allow easy support of error handling as the user's input does not have to be parsed (as with command language) (Preece et al. 1994).

On the other hand, too many menus may lead to information overload or complexity of discouraging proportions. They may be slow for frequent users and may not be suited for small graphic displays (Preece et al. 1994).

From the EEAC perspective, menu constraints can help the user to form the proper intentions and specify the proper action sequence and provide a context to evaluate the output language (Heim 2007).

Form Fill-in:

They are used primarily for data entry or data retrieval and they use the computer screen similar to a paper form. Form Fill-ins are similar to menu interfaces in the sense that they present screens of information. However, they are different than menu interfaces since they

are used to capture information and proceed linearly instead of navigating a hierarchical structure. The use of form fill-in requires the use of design tools. Heim (2007) and Shneiderman & Plaisant (2005) mention the following advantages and disadvantages with the use of form fill-in:

Forms simplify data entry and require little training compared to other interaction styles such as command lines. They permit the use of form management tools, have low memory requirements and are self-explanatory (Shneiderman & Plaisant 2005).

On the other hand, they require valid input in valid format, require familiarity with interface controls (e.g. Windows), can be difficult to correct mistakes and consume screen space (Shneiderman & Plaisant 2005).

Spreadsheets:

These are sophisticated variations of form fill-ins. The first spreadsheet was VISICALC developed by VisiCorp followed by Lotus 1-2-3. Microsoft Excel is the most popular spreadsheet nowadays. The spreadsheet consists of a grid of cells that contain values or formulas. Formulas can involve values of other cells and the user can enter and alter data to the spreadsheet. Spreadsheets have unlimited space to work with and allow to perform complex calculations. On the other hand, they can be difficult to learn and might require programming skills.

Question and Answer:

Question and answer was primarily used to supplement either menu or command line styles. This type of interaction requires that you consider all possible correct answers and deal with the actions to be taken if incorrect answers are entered. Although they are used for mainframe applications, Question and Answer interfaces are more popular in windows applications and normally called wizards. This type of interface is restricting for expert users but ideal for novice users that need to be guided during the interaction process. This type of interaction has low memory requirements and ideal for applications that are targeted towards beginners. On the other hand, it requires valid input supplied by user and familiarity with interface controls. It can also be very tedious to correct mistakes (Heim 2007).

Direct Manipulation:

This interaction style focuses on using icons, or small graphic images, to suggest functions to the user. Today, direct manipulation interaction (Dennehy, 2007) is seamlessly integrated with the menu style since many menus include icons in order to represent menu actions.

From the EEAC perspective, this style offers a wide range of possible intentions. Users usually have multiple options for specifying action sequences. However, the style can be overwhelming for novice users and provide multiple ways of executing action sequences.

With this interaction style, users do not need to remember a command set or recognize commands from menu. The user can easily undo operations and get immediate feedback to his or her actions. On the other hand, infrequent users need to remember the meaning of direct manipulation operations and require interactive devices (e.g., stylus, mouse) in order to manipulate graphical objects in the screen. In addition, this interaction style might require specific computer hardware graphic requirements and take too much space of the screen (Leventhal & Barnes 2007).

Metaphors: Metaphors are visual relationships to real-world objects that are used in interaction design in order to help users relate to complex concepts and procedures. They make learning new systems easier and make computers more accessible to beginner users. In addition, they exploit user‘s familiar knowledge, helping them to understand the unfamiliar. An example of this is the trash icon in the Windows operating system; this metaphor might symbolize a delete command. Selecting the icon with a pointing device such as a mouse executes the function.

However, the use of metaphors can present problems such as conflicts with design principles. Macintosh thrash has two completely different functions contradicting each other. It served to eject disk as well as delete files. Furthermore, metaphors can be too constraining like in the case of a windows directory that requires the user to scroll through it in order to find a file when it might be easier just to type the name of the file if it is known.

Some other potential problems are identified by Heim (2007) and listed below:

- Run out of metaphors since some virtual processes and objects have no real-world counter parts.

- Mixed metaphors.

- Carry connotations and associations.

Web Navigation:

Web navigation has basically two main interaction styles: link-based navigation and search (Heim 2007). The link-based navigation is based on hyperlinks and requires understanding of the hyperlink label or image. This can be problematic since the user might have a different interpretation of the actual meaning of the hyperlink. The search style minimizes this ambiguity with the help of a search engine that is used to interact with the user. However, the search engine might lead to wrong interpretations if it doesn‘t take into consideration spelling variations or incorrect search criteria input.

3D Environment:

The 3D interaction is natural to most users since it recreates the real-world that can be perceived in a 3D space. This interaction style is popular in computer games but it has the problem of being processor intensive. For this reason, 3D interfaces normally use information in vector based format in order to decrease file sizes and facilitate mathematical calculations required for 3D geometrical transformations used for interface navigation. The basic 3D transformations are scaling, translation and rotation.

Some vector based files used in 3D interfaces include X3-D (Web3D.org) and VRML (Virtual Reality Modelling language). Although 3D interfaces were initially used mainly in computer games, they have been recently used for desktop applications. An example of this effort is the Task Gallery developed by Microsoft.

Zoomable Interface:

The zooming interface paradigm (ZIP) was introduced by Raskin (2000). He described the ZIP paradigm by using the concept of ZoomWorld that was based on the idea that the user has access to an infinite plane of information having infinite resolution. The plane is ZoomWorld. Everything the user accesses is displayed somewhere on ZoomWorld, whether it is on a computer, on a local network to which your computer is attached, or on a network of networks, such as the Internet.

In this ZoomWorld, you think of the user as flying higher and higher above it. To look at a particular item, the user dives down to it. ZoomWorld also has a content searching mechanism. The overall metaphor is one of flying, climbing to zoom out and diving to zoom in. The user navigates both by flying above ZoomWorld and by doing content searches.

The ZIP readily permits labels to be attached to images and to collections of images yet does not impose any structure, hierarchical or otherwise, beyond association due to proximity. For example, a large heading Personal Photos might, when zoomed in on, reveal smaller headings on groups of pictures labeled Baby Pictures, Vacations, Pets, Hobbies, Friends, Relatives, and so forth. Zooming in to the writing under the heading Baby Pictures might reveal the children's names.

An interesting zooming user interface (ZUI), called PAD++ (it is now called Jazz), has been developed independently, originally at the University of New Mexico.

Natural Language:

The natural language relieves the user of the burden of learning syntaxes or getting familiar with graphical objects since it interacts with the user by using everyday language. The style is ideal for users not familiar with computer systems and for hands free and mobile applications. This style uses speech recognition or typed natural language. However, there

are problems associated with this style and highlighted by Leventhal & Barnes (2007) and Shneiderman & Plaisant (2005) in the following list:

 It is unpredictable

 May require many keystrokes

 Rigid sequences so user would need to remember sequences

 Typing with an onscreen keyboard could be slow and error prone. No evidence that phone has speech input.

 Recall. Infrequent or new users may have some problems recalling what the inputs should be.

 Vague and ambiguous.