IPC NOTES

UNIT 1.-HUMAN-COMPUTER INTERFACES

1.- INTRODUCTION

The popularization of computers has been possible because of advances in User Interfaces (UI). Initially, the goal of application was to make the most of the available computer resources. Later, the extra power of the computers began to be used to improve the interfaces. Now the focus is on how to make it easier for the user to share content.

The human-computer interaction is a discipline that applies techniques of the experimental psychology to computer science and uses methods of all sorts of sectors.

The most visible result has been the social online tools.

A **user** is the person that **interacts with a computer**.

An **interaction** is all the **information interchanged between** the user and the computer.

A **Human-Computer Interface** (HCI) 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.

The **classic lifecycle** for software development was in **waterfall**. **Now** is better used the **iterations** for user interface design.

Currently, there is a transition from desktop to web-based applications.

The design of user interfaces should **facilitate** the use of a **wide range** of display sizes. New interfaces are based on multimodal technologies, gesture technologies or affective technologies.

2.- THE EVOLUTION OF USER INTERFACES

Vannebar Bush's Memex (1945) described a device for supplementing human memory, to store books, records and communications. He proposed "trails" for linking different elements.

Early computers:

- First "User Interfaces" (1940s). Example: ENIAC (1946).
- Batch computing (1960s).
- Time shared systems and command lines.

Early graphic interfaces:

- Early vector monitors. Examples: Tennis for two (1958) and Spacewar! In a PDP1 (1962).
- First interactive graphics editor. Example: Sketchpad (1962).

The first modern graphical user interface was designed by Doug Engelbart in 1968 and was remarkable called from everyone "The mother of all demos". The new features that provided were:

- Mouse.
- Graphical interface with multiple windows.
- Hyperlinks.
- Videoconference.

The first commercial implementation was the Xerox Alto.

Since the first IBM PC (1981) and for a decade, the main interface was the command line for the early home computers.

Silicon Graphics produced high performance workstations (1980s).

The graphical interfaces became popular in PCs since Windows 3.0 (1990).

UNIT 2.-USABILITY

1.- INTRODUCTION

The **good interfaces** should:

- Strengthen the user confidence.
- Allow the user to have an accurate mental model of the system to predict what will happen after some action.
- Ideally, they “disappear”, allowing the user to focus on the task.

The user interface is the part of the system that the user sees, hear and feel. There are other parts of the system that are hidden, like the database.

2.- USABILITY FACTORS

Properties of a usable interface:

- Fit for use (or functionality). The system supports the user’s tasks.
- Ease of learning (for different groups of users).
- Task efficiency.
- Ease of remembering.
- Subjective satisfaction.
- Reduce errors.

It is difficult to design systems that scores high in every aspect, so it is important to decide which aspects are the most important for our system.

3.-USABILITY GOALS AND MEASURES

Attending to the ISO 9241 standard, **usability** is the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use:

- **Effectiveness**: accuracy and completeness.
- **Efficiency**: consumed resources.
- **Satisfaction**: freedom from discomfort, user attitude.

ISO 9241 measure examples for overall usability:

Effectiveness measures	Efficiency measures	Satisfaction measures
Percentage of goals achieved	Time to complete a task	Rating scale of satisfaction
Percentage of users successfully completing task	Tasks completed per unit time	Frequency of discretionary use
Average accuracy of completed tasks	Monetary costs of performing the task	Frequency of complaints

ISO 9241 measures for other usability goals:

Goal	Effectiveness measures	Efficiency measures	Satisfaction measures
Meet needs of trained users	Number of power tasks performed; Percentage of relevant functions used	Relative efficiency compared with an expert user	Rating scale for satisfaction with power features
Meet needs to walk up and use	Percentage of tasks completed successfully on first attempt	Time taken on first attempt*; Relative efficiency on first attempt	Rate of voluntary use
Meets needs for infrequent or intermittent use	Percentage of tasks completed successfully after a specified period of non-use	Time spent re-learning functions*; Number of persistent errors	Frequency of use

Goal	Effectiveness measures	Efficiency measures	Satisfaction measures
Minimization of support requirements	Number of references to documentation; Number of calls to support; Number of accesses to help	Productive time*; Time to learn to criterion*	Rating scale for satisfaction with support facilities
Learnability	Number of functions learned; Percentage of users who manage to learn to criterion	Time to learn to criterion* Time to re-learn to criterion*; Relative efficiency while learning	Rating scale for ease of learning

Goal	Effectiveness measures	Efficiency measures	Satisfaction measures
Error tolerance	Percentage of errors corrected or reported by the system; Numbers of user errors tolerated	Time spent on correcting errors	Rating scale for error handling
Legibility	Percentage of words read correctly at normal viewing distance	Time to correctly read a specified number of characters	Rating scale for visual discomfort

Often, designers have to favour one aspect against others depending on the application. The managers of the project should clearly define the goals of the application, to support the decision made. It is easier to measure usability in the final system, but then it might be too late.

Several prototypes should be built to be reviewed by designers and users.

The prototypes can be:

- Low-fidelity paper mock-ups.
- High-fidelity interactive prototypes.

Sometimes, user documentation and online help are written before building the interface, to refine the design. Then, the application is implemented with the proper tools. Finally, the acceptance test certifies that the product meets the goals set in the requirements.

4.- AREAS THAT REQUIRE USABLE UI

In general, any computer system benefits from a usable interface, but in the following domains, a usable UI is even more important:

- Life-critical systems:

USABILITY MEASURES	IMPORTANCE
Time to learn	LOW
Speed of performance	HIGH
Rate of errors	HIGH
Retention over time	BY REPETITION
Subjective satisfaction	LOW

- Industrial and commercial systems.

USABILITY MEASURES	IMPORTANCE
Time to learn	HIGH
Speed of performance	HIGH
Rate of errors	MODERATE
Retention over time	BY REPETITION
Subjective satisfaction	MODEST

- Home and entertainment applications.

USABILITY MEASURES	IMPORTANCE
Time to learn	HIGH
Speed of performance	MODERATE
Rate of errors	HIGH
Retention over time	MODERATE
Subjective satisfaction	HIGH

- Exploratory, creative and collaborative interfaces.

USABILITY MEASURES	IMPORTANCE
Time to learn	MODERATE
Speed of performance	MODERATE
Rate of errors	MODERATE
Retention over time	MODERATE
Subjective satisfaction	HIGH

- Socio-technical systems.

USABILITY MEASURES	IMPORTANCE
Time to learn	HIGH
Speed of performance	HIGH
Rate of errors	HIGH
Retention over time	LOW
Subjective satisfaction	LOW

5.-UNIVERSAL USABILITY

Universal usability is the process of creating products which are usable by people with the widest possible range of abilities, operating within the widest possible range of situations, as is commercially practical.

User interface designers must take into account the wide variety of users, paying attention to:

- Physical abilities and physical workspaces.
- Cognitive and perceptual abilities.
- Personality differences.
- Cultural and international diversity.
- Users with disabilities.
- Older adult users.
- Children.
- Hardware and software diversity.

Ergonomics study how to adapt the work environment to the workers.

The **anthropometry** is the science that measures the dimensions of the human body:

- **Static anthropometry**: provide standard dimensions of persons in a population, typically standing or seated.

- **Dynamic anthropometry**: describes moving ranges, reaches and trajectories.

Designers have to take into account those measures to create interfaces that are useful for the majority of the population.

There is wide variety of personal preferences. Designers should be able to adapt their interfaces to cultural, ethnic, racial, racial or linguistic differences. Depending on their cultural background, users may prefer simple, static interfaces, while other users may prefer dynamic, rich interfaces. Furthermore, preferences change quickly. Software architectures should provide flexible methods to adapt applications to different languages.

Types of **disabilities**:

- Vision-impaired.
- Hearing-impaired.
- Motor disabilities.
- Cognitive disabilities.

Principles of Universal Design:

- Equitable use.
- Flexibility in use.
- Simple and intuitive use.
- Perceptible information.
- Tolerance for error.
- Low physical effort.
- Size and space for approach and use.

Many countries have passed legislation to ensure access to information technology public services to users with disabilities.

Taking into account the necessities of users with some disability from the beginning of projects does not increase the cost a lot and increases the level of usability of the system for all users.

Aging has negative effects on both physical and cognitive abilities. Computers can provide new opportunities for social interaction, if we can adapt the user interfaces. In turn, society can benefit from a quick access to the experience and emotional support from older adults. The UI should allow the user to adapt the font size, the contrast of the screens and the volume of sounds. Interfaces can also be designed with easy to use pointing devices, clearer navigation paths, consistent layouts and simpler commands.

The main motivations of **children** are entertainment and education. The goals of children-oriented tools are:

- Accelerate the educative process.
- Facilitate the socialization with peers.
- Improve self-confidence.

UI designers must take into account **children's limitations**, and avoid:

- Double click, mouse dragging and small targets.
- Complex texts.
- Complex command sequences.

Other concerns are short attention spans and limited capacity to work with multiple concepts simultaneously. Security should be a top level requirement in web-based software for children.

Designers should also take into account the wide range of systems where applications may run, both hardware and software.

UNIT 3.-HUMAN ASPECTS

1.- FOUR PSYCHOLOGICAL PRINCIPLES

There are **four psychological principles** that may cause **errors** when interacting with a system:

1. Users see what they expect to see. In order to solve this problem, UI designers have to take care of:
    - **The principle of consistency**, that is, to keep the same colour scheme, button order, etc.
    - **The principle of exploiting prior knowledge**, that is, to use familiar concepts to the users.
2. Users have difficulty focusing on more than one activity at a time. The UI should remind the user what he or she needs to do next. To focus attention, follow these principles:
    - **The principle off perceptual organization**, that is, to group together things that go together.
    - **The principle of importance**, that is that important information should be placed in a prominent position. It also works to make the importance item a little bit bigger than the rest.
3. It is easier to recognize something than to recall it. The UI should provide the required information, and not rely on the user's memory. Provide menus, icons, screen metaphors instead of a command line interface, or a combination of keys, but take into account advanced users, who prefer speed to ease of use.
4. It is easier to perceive a structured layout.

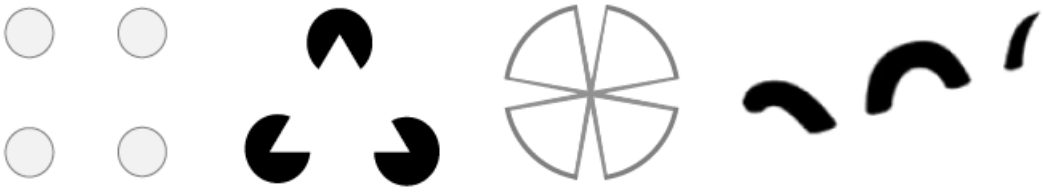
2.- GESTALT LAWS

The **Gestalt laws** try to explain how the human brain is able to acquire and maintain meaningful perceptions from an apparently chaotic world. These laws are:

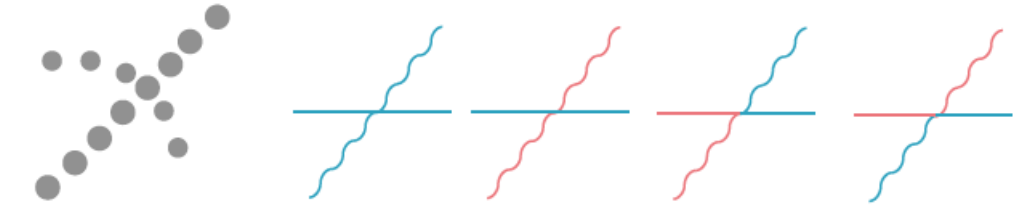
- **Proximity**: close objects appear to form groups, instead of a random collection.

- **Similarity**: elements with the same colour or shape appear to belong together.

- **Closure**: we fill in the gap of an incomplete element and closed areas are perceived as a whole.



- **Continuity**: aligned elements look like lines.



- **Symmetry**: regions bounded by symmetrical borders are considered as a whole.

- **Figure-ground separation**: when there are edges or differences in colour, texture or brightness, we separate object and background.

The design should take advantage of these laws to help the user to interact with the system.

Some advices for text **legibility**:

- Use Serif fonts for paper.
- Use Sans Serif fonts for screen.
- Too tight spacing is hard to read.
- Too loose spacing is also hard.

- Too small is harder to read.
- Too big is also hard.
- For small type sizes, it is better to increase the leading.

Some guidelines for **colour**:

- Do not overuse colours.
- Use brightness contrast to make it easier to read.
- Design for monochrome, then add colour.
- Use colour for grouping and emphasis.
- Take into account blindness.
- Light colours bordered by darker colour seem to raise from the screen.
- Use colour for coding information.

3.- AUTOMATIC AND CONTROLLED ACTIVITIES

There are some activities that can be **performed while doing something else**. On the other hand, there are activities that require **full attention**, these are controlled activities and mainly these are related to the **language**. We can do only one controlled activity at a time.

There are may things that we can perceive automatically as spatial location, size, colour, shape, sound, smell, etc. It can be used to provide information without requiring full attention.

4.- CONTRAST

We can take advantage of the Gestalt laws used for separating an object from its background for calling the users' attention to some element of the interface.

When there are too many elements emphasize, the effect is lost. Contrast should be used with caution, especially with colours.

5.- THREE PRINCIPLES FOR UI DESIGN

Based on Don Norman's finding:

- **Visibility**: the controls should be easy to find.
- **Affordance**: it should be obvious how to use the interface.
- **Feedback**: the system should tell the user what it is doing at any time.

6.- DESIGN RULES FROM DESIGN PRINCIPLES

Principles are abstract and require interpretation.

Design rules are more specific, and they are tailored for a given problem.

UNIT 4.-REQUIREMENTS ANALYSIS

1.- USER-CENTERED DESIGN

Involve users throughout the design and development process. Focuses on understanding:

- The users.
- The task that users perform with the system.
- The environment in which the system will be used.

Customers, other people in the organization with interest in the development and end users are referred to as the **stakeholders**.

We must distinguish between primary users, who are going to interact directly with the system, and secondary users, who will use indirectly the system.

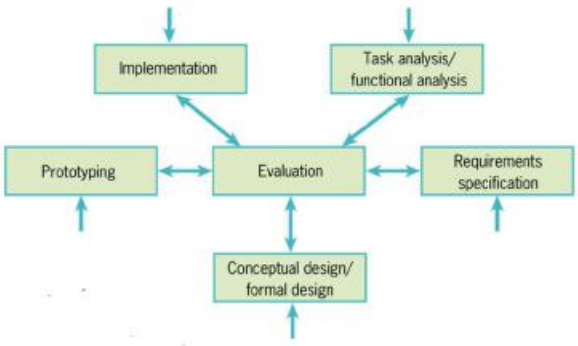
Principles of human-centred design:

- The active involvement of users.
- Appropriate allocation of functions between user and system.
- The iteration of design solutions.
- Multidisciplinary design teams.

Activities in human-centred design:

- Understand and specify the context of use.
- Specify the user and organizational requirements.
- Produce design solutions.
- Evaluate designs with users against requirements.

The star life cycle:



We have to involve the users in every step of the development. The later in the process, the less changes can be made.

2.- REQUIREMENT ANALYSIS

A requirement is a statement about an intended product that specifies what it should do or how it should perform. Requirements can be defined with different levels of abstraction.

Generally, they are obtained from observation, interviews or questionnaires. They usually describe what the system should do, not how it should do it.

There are two types of requirements:

- **Functional**: what the system should do.
- **Non-functional**: constraints on the system and its development.

The Software Requirements Specification is the result of the requirements analysis phase. It is a document that gathers all the requirements in a structured template.

2.1.- GATHERING INFORMATION

Ethnographic studies observe users while they are doing real work in their real working environment or using a home system in their homes. Find what users do, but also what users like or dislike. There are different types of observation:

- **Direct observation**:
  - **Field studies**: the observer takes notes about interesting behaviours in the work place or home.
  - **Controlled studies**: the user interacts with the system in a controlled environment.
  - **Pros**: easy, produces interesting data.
  - **Cons**: only a single pass at the information gathering, hard to record every aspect of the activity, what is not recorded is lost, it is obtrusive and can alter user's behaviour and performance.
- **Indirect observation**:
  - Record video, keystrokes, mouse movement, etc.
  - **Pros**: the whole interaction is captured, more objective.
  - **Cons**: analysis of data is time consuming, some users may be intimidated.

User studies are a finding about the users and the domain. It describes and classifies users depending on different characteristics. Interview real users or, in case that it is not possible, talk to domain experts, managers, work supervisors, etc. who may know about them. Extract user groups and focus your design on those groups as a person in one group will typically interact with the system differently than a person in another group.

We can find three different types of users:

- **Novice** or **first-time** users. Lack of previous experience may cause anxiety.
- **Knowledgeable intermittent** users. Problems retaining the structure of menus or the location of features.
- **Expert frequent** users. They demand rapid response times, brief and no distracting feedback and shortcuts.

Designing a user interface for one type of user is easy. Design a UI for different levels of experience is harder, thus, we should use **multi-layer** interfaces.

Interviews must be planned ahead. There are two types:

- **Structured**: predefined set of questions, then it is easier to conduct and easier to analyse.
- **Flexible**: some preset topics, but no set sequence, thus it is less formal and we can obtain more information and opinions.

When we do an interview, we should:

- Make the person feel comfortable. Some users won't express their real opinions because they think it is their fault, or that their opinions are trivial, or have no importance.
- Perform a small pilot study and record the interviews with permission.
- Start with an open question.

What not to ask:

- Asking leading questions.
- What they would do/like/want in hypothetical scenarios.
- How often they do things.
- How much they like things on an absolute scale.
- Binary questions.

Apart from what people say, pay attention to what people do as they try not to look stupid, they try to answer what they think we want to hear and that they are not aware of their behaviour.

Questionnaires and **surveys** are a set of questions and statements for gathering more precise information. There are two types of questions:

- **Closed**. Yes/No/Don't know and multipoint rating scale between two opposites.
- **Open**. No predetermined answers, the user can respond what she thinks in the provided space and it gives more information but are harder to analyse.

A questionnaire should be understandable, as short as possible, captures the information you need, provides the option to add additional thoughts.

2.2.- STUDYING THE COMPETENCE

An analysis of the competition is a fast and easy way for establishing a starting point in the design. Activities:

- Make a list of the existing products similar to your idea.
- Create a comparative table with their evaluation.
- Make a presentation, focus group, etc. for reviewing the results.

We should review the previous versions of the application and add to our design good ideas about usability of existing applications.

2.3.- DESCRIBING USERS

We have to make a model of a person that includes the person's motivation, likes and dislikes, intentions, behaviour and goals. We should define also her/his attitudes towards new technologies and how she currently uses the system.

We should draw a picture of our persona or use a photo and give it a story to tell with her/his name, age, occupation, background, social situation, goals, etc.

Knowing what our persona thinks, helps to build empathy so that we can understand the state of mind, emotion, philosophy, beliefs, or points of view of the user.

The **persona** is built from the results of the real users' interviews.

3.- TASKS ANALYSIS

After identifying the users, we must understand their goals when using the system.

A **task analysis** is an activity to study what a system must do and the functionality it has to provide to help users achieve their goals. Some terminology:

- **Goal**: end result to be achieved.
- **Task**: structured set of activities undertaken in some sequence.
- **Action**: individual operation or step of a task.

The relation between these terminologies is:

3.1.- GATHERING INFORMATION

Information about the users' tasks can be obtained by interviews, observation and study of documentation, but also:

- Study the most common errors users make in the current system.
- Find work arounds: ways users find of doing tasks when the UI does not support the task.
- Fins artifacts: notes, cheat sheets, sticky notes, etc.

Ask the user specifically for them, because maybe they are so used to them, that they don't realize there is a problem.

Task characteristics:

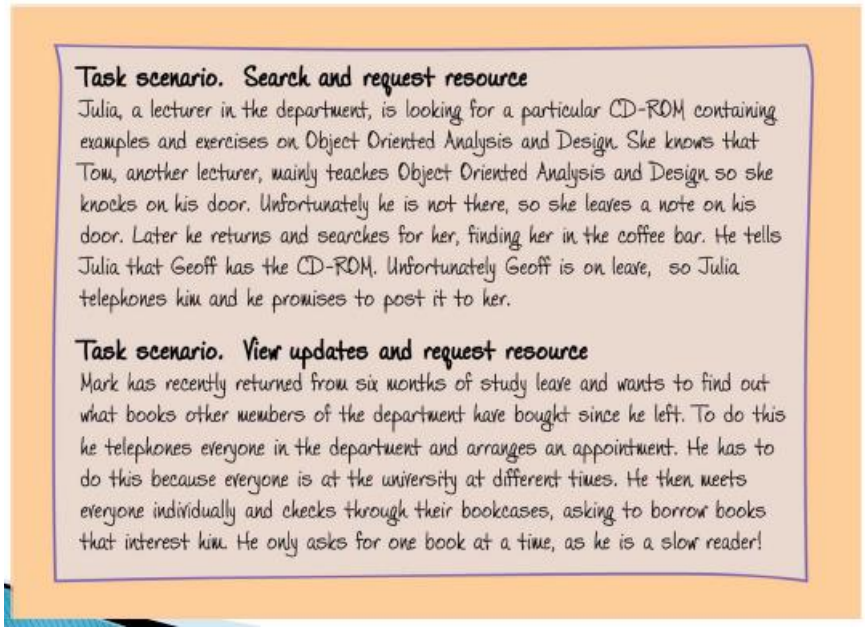
- The extent to which tasks vary from one occasion to another.
- Whether tasks will be carried out regularly, infrequently, or only once.
- The knowledge and kinds of skill required to perform tasks.
- How much the work is affected by changes in the environment.
- Whether time is critical for the work.
- Whether there are safety hazards.
- Whether the user will do the work alone or with others.
- Whether the user will normally be switching between several tasks.

The sequence of tasks to achieve a goal can change from user to user. The UI should not impose a sequence of tasks unless there is a reason.

3.1.- DESCRIBING TASKS

A **task scenario** is a narrative description of a task, as is currently being done:

- They tell a story about the use of a system.
- Personalized: describe a specific instance and situation.
- Detailed: describe step by step the procedure followed by the user to get a task done, and the features and behaviours of the system.
- Includes problems and difficulties.
- They should be evaluated by the users to check that they describe faithfully the task.



Task scenario. Search and request resource

Julia, a lecturer in the department, is looking for a particular CD-ROM containing examples and exercises on Object Oriented Analysis and Design. She knows that Tom, another lecturer, mainly teaches Object Oriented Analysis and Design so she knocks on his door. Unfortunately he is not there, so she leaves a note on his door. Later he returns and searches for her, finding her in the coffee bar. He tells Julia that Geoff has the CD-ROM. Unfortunately Geoff is on leave, so Julia telephones him and he promises to post it to her.

Task scenario. View updates and request resource

Mark has recently returned from six months of study leave and wants to find out what books other members of the department have bought since he left. To do this he telephones everyone in the department and arranges an appointment. He has to do this because everyone is at the university at different times. He then meets everyone individually and checks through their bookcases, asking to borrow books that interest him. He only asks for one book at a time, as he is a slow reader!

A **use scenario** is similar to a task scenario, but they describe the anticipated use of the system.

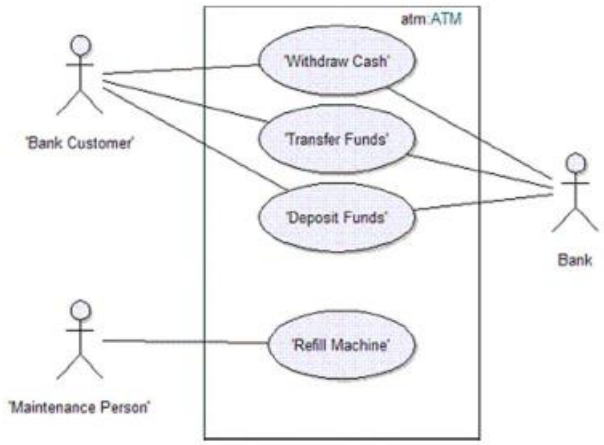
Use scenario. Search and request resource

Julia is looking for a particular CD-ROM containing examples and exercises on Object Oriented Analysis and Design. She accesses the digital library from home and types in the key phrase 'Object Oriented Analysis'. The system retrieves one result. Geoff owns the appropriate CD-ROM. Julia then sends an e-mail to Geoff, asking to borrow the CD-ROM.

Use scenario. View updates and request resource

Mark has recently returned from study leave and wants to find out what are the latest additions to the digital library. He selects 'check updates', identifies the books he is interested in, and sends an e-mail to the owner of the one that interests him most.

Use cases also focus on users' goals, but the emphasis is on user-system interaction rather than the user's task itself. The users are called actors, and other systems that interact with the system being described can also be actors.

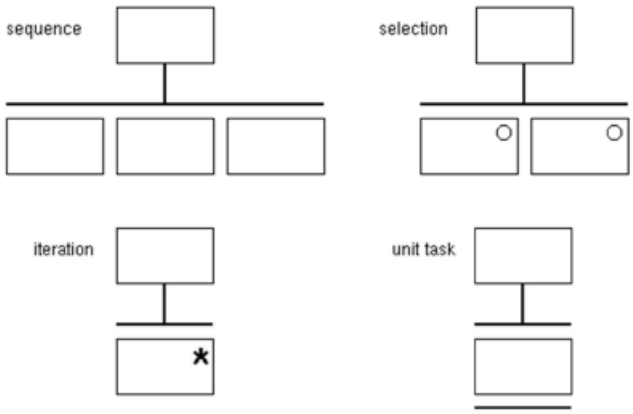


Concrete use cases are similar to task scenarios, but they are not personalized (they are a little bit more generic). They can be written in two columns, one column for user actions and another column for system responses.

User action	System response
The academic enters one or more of the search parameters for the CD-ROM: title, year and platform	The system displays the search results
The academic selects a search result	The system displays the full details of the CD-ROM and the contact edetails for its ownerm who is a research student
The academic chooses the e-mail address	The system displays a message area
The academic writes and sends the e-mail request	The system confirms the sending of the request

A **hierarchical task analysis** involves breaking a task down into subtasks and then into sub-subtasks and so on. The subtasks are then grouped together as plans that specify how the tasks are performed in an actual situation. Focuses on the physical and observable actions that are performed including actions not related to software or an interactive product at all. The starting point is a user goal.

Notations:



There are different types of **task environments**:

- The physical environment.
- The safety environment.
- The social environment.
- The organizational environment.
- The user support environment.

Storyboards are a sequential series of illustrations, stills, rough sketches and/or captions of events, as seen through the camera lens, that provide a synopsis for a complex scene with its action and characters. They are used to describe tasks:

- They are usually hand drawn.
- They show the flow of interaction.
- They show people doing a task to achieve a goal.
- Do not spend too much time drawing.

Storyboards should convey:

- Setting:
    - People involved.
    - Environment.
    - Task being accomplished.
- Sequence:
    - What steps are involved.
    - What leads someone to use the app?
- Satisfaction:
    - What motivates people to use this system?
    - What does it enable people to accomplish?
    - What need does the system fill?

The benefits of storyboarding:

- Helps emphasize how an interface accomplishes a task.
- Avoids commitment to a particular user interface.
- Helps get all the stakeholders agree in terms of the goal.