IPC NOTES

UNIT 5.1.- INTERACTION STYLES

1.-INTRODUCTION

After defining tasks and actions, the designer must choose an interaction style. **Interaction styles** are ways a user and a computer system can communicate with each other. An interaction style is a collection of user interface controls and their associated behaviour. It provides the look and feel of the user interface components.

We will study five interaction styles:

- Command line interfaces.
- Menu selection.
- Form Fill-in
- Direct Manipulation.
- Natural Language.

The interaction styles can be combined.

2.-COMMAND LINE INTERFACES

The **command line interfaces** (CLI) was the first interactive dialog style to be commonly used:

- Allows to provide direct commands to the system.
- They are flexible, commands can have options.
- A command can be applied to many objects at once, so it is useful for repetitive tasks.
- Expert users feel they are in control, and they can express complex actions rapidly and concisely.

Disadvantages:

- They are hard to learn.
- The user has to remember the commands, since there is no indication on what can be done.
- There is no help about what to do next.
- Commands can be cryptic or have a complex syntax.
- They vary from system to system.
- Low error tolerance, high error rate. A single typing mistake produce an error message in a long command.
- Retention over time is low.
- It is difficult to provide specific error messages and on-line help.

All these problems can be reduced using consistent command name and abbreviations.

Command languages were originated as a mean of communicating with the operating system. The Unix philosophy was:

- Small programs that do only one thing.

- Those programs can be chained: the output of a command can be the input to the following command.
- A correct execution does not provide feedback, errors do.
- Fast interaction.
- Macros are combinations of commands that can be executed at once.

Command line interface are still in use. They are useful for low level interaction, as a first interface for an application, or for advanced administration.

Guidelines for designing CLI:

- Choose meaningful and representative command names.
- Follow consistent syntax for all commands.
- Support consistent rules for abbreviation.
- Make commands as short as possible.
- If commands or responses to commands can be abbreviated, use common abbreviations.
- Limit the number of ways of accomplishing a task.
- Allow user to define macros.

3.-MENU SELECTION

A **menu** is a set of operations from which the user must choose. **Menu selections** avoid many problems of command line interfaces, because they offer cues for user recognition rather than forcing them to remember command names and syntax. **Menu items** should be self-explanatory and distinguishable. Menus are suitable for users with little training, intermittent users, users who do not know the terminology, or need help on making decisions. On the other hand, expert users may feel restricted or slowed down.

There are different types of menus:

- Pull-up.
- Pull-down. Show keyboard shortcuts.
- Pop-up. Context dependent.
- Sequential.

There are different types of single menus:

- Binary menu. If repetitive, provide shortcuts and default values.
- Radio buttons. Mutually exclusive options.
- Check boxes. Multiple binary options.

Important characteristics:

- Non available options are greyed out.
- Access keys.
- Cascading submenus.

Other type of menus:

- Toolbars. Use icons instead of text.

- Ribbons. Tabs.
- Pie menus.

Menus for long lists are:

- Scrolling menus.
- Combo boxes.
- Fisheye menus.
- Sliders.
- Two-dimensional menus.

Embedded menus and **hotlinks** are an alternative to explicit menus. They show options in their context and are the precursor of web hotlinks.

There are several techniques to organize items in several menus:

- **Linear menu sequences**. The user makes one decision at a time. They are effective for novel users and simple tasks.
- **Simultaneous menus**. Several menus at the same time. Users enter choices in any order. It is for expert users.
- **Tree-structured menus**. Group options in a natural and easy to understand categories. You should use the terminology from the user's task domain. The usual recommendation is 4-8 items per menu and 3-4 levels.
- **Menu maps**. They are used when the menu tree grows. It provides an overview of the options keeping the sense of position.
- **Acyclic and cyclic menu networks**. It allows users to reach an item from different start points. They arise naturally in social relationships, transportation routing, bibliographical cites and web pages. They can confuse users.

Task-related grouping in tree structures:

- Create groups of logically similar items.
- Form groups that cover all the possibilities.
- Make sure that items are non-overlapping.
- Use familiar terminology but differentiate the items.

Item presentation sequence:

- Sort items in each menu by natural order or, using a standard order.
- The items can be sorted adaptively.

Menu layout:

- **Titles** must be simple and descriptive.
- **Phrasing and formatting**:
    - Use familiar and consistent terminology.
    - Ensure that items are distinct from one another.
    - Use concise phrasing.
    - Start the item name with a keyword.
- **Write consistent guidelines** to design menus:
    - Titles (centred or left justified).
    - Item placement (left justified, blank lines).

    - Instructions.
    - Error messages.

Fast movements through menus:

- **Keyboard shortcuts**:
    - For expert users.
    - Problems for translation.
- **Pie menus**. If item location is known, the user can move the pointer ahead.
- **Tear-off menus**.

Audio menus are used when hands and eyes are busy. You have to speak options and questions to the user, and the user responds using keyboard or by voice. It is non persistent, the user has to memorize the options. The system has to provide auditory feedback.

Design goals of **menus for small displays**:

- Learnability: they should be easy to learn.
- Account for target domain.
- Organize options by importance and frequency of use.
- Simplify: concentrate on the important options.
- Design for responsiveness: plan for interruptions and provide continuous feedback.

Elements of **menus for small displays**:

- Hardware buttons for special options.
- Touch-screen for gesture based interaction.
- Large colour icons for on-screen options.

Command menus show the available options and are useful for novel and intermediate users. They also show shortcuts.

4.-FORM FILL-IN

Form Fill-in is used when the application needs to gather lots of information from the user. Form Fill-in interfaces allows the user to:

- Move easily around the form.
- Leave some fields blank.
- Correct some data already entered.
- Input of text and numeric data.

The form on the screen is a metaphor for a paper form.

Guidelines:

- Meaningful title.
- Comprehensible instructions.
- Logical grouping and sequencing fields.
- Visually appealing layout of the form.
- Familiar field labels.

- Consistent terminology and abbreviations.
- Visible space and boundaries for data-entry fields.
- Convenient cursor movement.
- Error correction for individual characters and entire fields.
- Error prevention where possible.
- Error messages for unacceptable values.
- Immediate feedback.
- Marking of required fields.
- Explanatory messages for fields.
- Completion signal to support user control.

Some entry fields may require special formatting.

Combine menus and form fill-in in **dialog boxes**:

- Meaningful title, consistent style.
- Top-left to bottom-right sequencing.
- Clustering and emphasis.
- Consistent layout.
- Consistent terminology, fonts, capitalization, and justification.
- Standard buttons.

Relationship with other elements in **dialog boxes**:

- Smooth appearance and disappearance.
- Distinguishable but small boundary.
- Small enough to reduce overlap problems.
- Display close to appropriate items.
- No overlap of required items.
- Easy to make it disappear.
- Clear how to complete/cancel.

5.-DIRECT MANIPULATION

Direct manipulation interfaces allow users to interact directly with the UI objects. They usually use continuous input devices, such a mouse, pen, joystick or touch screen. There are many domains where DM interfaces are used. Characteristics:

- Visible and continuous representation of task objects and their actions.
- The task objects are manipulated directly.
- Operations are rapid, incremental and reversible.
- The user feels she is interacting with the objects of the domain directly, not through a intermediary.
- Novices can learn the basic functionality quickly.

Advantages of WYSIWYG word processors:

- Users see a full page of text.
- The displayed document and the printed document look equal.

- The focus of edition is clearly seen.
- Cursor motion is natural.
- The most used actions are always on-screen as buttons.
- The user actions are instantly shown.
- Rapid response and display.
- Most actions can be reversed.

Problems with direct manipulation:

- It is an impediment for vision-impaired users.
- Visual representations are usually large and occupy screen space, and it may be necessary to use scrolling.
- Users must learn the meanings of visual representations.
- Visual representation may be misleading.
- It can be slower for experts using keyboards.
- Problematic in small screens.
- Difficult in some applications.

6.-NATURAL LANGUAGE INTERFACES

Natural-language interfaces (NLI):

- The user interacts with the computer using a familiar natural language to give instructions and receive responses.
- Text or speech.
- There is no syntax or commands to learn.
- Successful in domain-specific applications.
- The advances in direct manipulation and its low recognition rates reduced the interests in this type of interface, but today is regaining attention.

It is not clear whether this is desirable:

- Computers can show information 1000 times faster than the user can enter commands.
- Novice and intermittent users prefer to choose among a visible set of options.
- Experts prefer a precise, concise command language.
- Lack of habitability: ii is hard for the users to determine what objects and actions are appropriate.
- It requires clarification, that slows down the interaction.

7.-COMPARISON OF INTERACTION STYLES

Command Line Interfaces	
↑ Flexible	↓ Poor error management
↑ Appealing to expert users	↓ Requires substantial training and memorization
↑ Supports user's initiative by allowing them to create macros and shortcuts	
Menu Selection	
↑ Is easy to learn	↓ Danger of creating too many menus
↑ Requires fewer keystrokes than CLI	↓ May slow frequent users
↑ Structures decision making	↓ Consume screen space
↑ Good for learners and infrequent users	
Form Fill-in	↓ Consume screen space
↑ Simplifies data entry	
↑ May require modest training	
↑ Assists users by providing defaults	
Direct Manipulation	
↑ Presents the task concepts visually	↓ Requires graphics displays and continuous input devices
↑ Is easy to learn	↓ Icons and metaphors may have different meanings for different users
↑ Is easy to remember how to use	
↑ Avoids errors and allows easy recovery	
↑ Encourages exploration	
Natural Language Interfaces	
↑ There is no need to learn the syntax	↓ Can be unpredictable
	↓ Difficult to implement

8.-ADVANCED INTERFACES

Virtual reality is an interactive system that offers the sensory perception of a synthetic world that replaces completely the real world. It breaks the physical limitations of space and allows users to act as of they were somewhere else.

Augmented reality combines the real scene with virtual elements. The virtual elements add additional information to the view of the scene. The user movements have to be captured, and then update the shown images.

The success of virtual and augmented environments depends on the correct integration of:

- Displays.
- Head tracking.
- Hands tracking.
- Force feedback.
- Sound input and output.
- Other sensations.

- Collaborative and competitive virtual environments.

Teleoperation has two parents: direct manipulation in personal computers and process control in complex environments. It is based on remote physical operation. The complicating factors in teleoperation are:

- Time delays in transmission and in the operation.
- Incomplete feedback.
- Unexpected interferences.
- More complex error-recovery procedures.

UNIT 5.2.- CONCEPTUAL DESIGN

1.-INTRODUCTION

Work reengineering:

- Many projects consist of developing a new version of an existing system.
- Maybe users will have to start working differently.
    - Sensitive handling: involve users in the development.

Goals of the work reengineering:

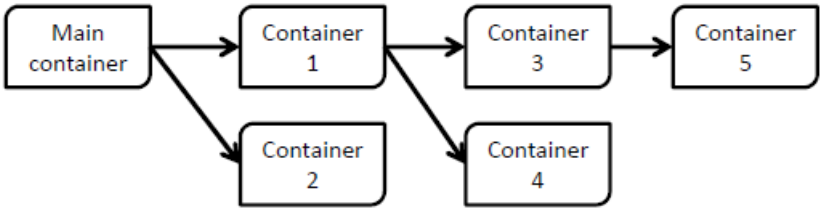
- Realize the power and efficiency of automation.
- Reengineer the work for more effectively support of business goals.
- Minimize retraining by taking advantage of user's current knowledge and take into account human cognitive constraints and capabilities when designing new tasks.

2.-CONCEPTUAL DESIGN: THE CONTENT DIAGRAM

A **conceptual design** is the process of establishing the underlying organization and structure of a UI, that is, decide which functionality each screen should support. A content diagram is a low-fidelity prototype that represents the organization and structure of an UI from the designer's perspective.

Content diagrams are formed by:

- **Containers**: abstract representation of a part of the user's work and the supporting functions.
- **Links**: how the user will navigate between the functional areas within the UI.



When designing the contents and structure of the UI, the content diagram should serve as a guide, not as a strict recipe. The content diagram is created from information obtained during the requirements gathering, and from the concrete use cases. It will probably be incomplete, but it is helpful for identifying the main functional areas and their relations.

The steps for creating a content diagram:

1. Identify the primary task objects, attributes and actions.
2. Identify the different containers and the task objects that go into each one.
3. Link the containers to show the navigate flow.

This is a creative process and should be improved iteratively.

2.1.-IDENTIFY THE PRIMARY TASK OBJECTS, ATTRIBUTES AND ACTIONS

It is used to decide what goes into each container, and the links needed between containers. It is similar to object-oriented design:

- **Primary task objects**: entities with data with which the users interact to carry out their tasks.
    - High level objects, there are usually a few.
    - Where to look for task objects: requirements documentation and concrete use cases.
- **Attributes**: the task objects' properties or links to other objects. The task objects and it attributes will be translated onto a combination of user interface objects. There are two kind of attributes:
    - Properties: data that belongs exclusively to the object.
    - Child objects: it is a task object in its own right but has a relationship with the parent object. In a GUI, the child object will appear whenever the parent object is shown, usually inside it.
- **Actions**: functions the user can invoke on the task objects.
    - Users perform actions on the task objects.
    - Apart from the specific actions of the task objects, also consider the standard actions as view, create, etc.
    - The actions will be translated onto menu items or items on the tool bar.

Often task objects can be grouped in classes, that abstract the common attributes. You can use the concrete use cases for identifying task objects and their attributes:

- Single-underline nouns that may correspond to task objects.
- Double-underline the attributes of these task objects.

Verbs usually correspond to actions, but we won't mark them because the relationships are often less direct. After identifying the task objects and attributes, compile them in one table per task object. Since it is hard to capture all the task objects, attributes and actions from the concrete use cases, we should use an iterative approach:

- Find information in the concrete use cases, but also in the user's knowledge of the domain, or from your own domain's analysis.
- Prototype your ideas and evaluate them with users.

2.2.-IDENTIFY THE DIFFERENT CONTAINERS AND THE TASK OBJECTS THAT GO INTO EACH ONE

Each container helps the user to perform some work by collecting the required functions and task objects. They will become screens, windows, dialog boxes or message boxes. In the next step, we will connect containers with links.

Elements in a container:

- **Name**: the name of the container.
- **Purpose**: a sentence with the purpose in supporting the user's task.
- **Functions**: ● invoked by the user, ▪ invoked by the system.

- **Links**: the name of a container it is connected to: → the new container replaces the current one, →→ both containers work in parallel.
- **Objects**: the task objects whose attributed and actions are required.
- **Constraints**: any constraints for the container, such as speed, reliability, and availability.

The main container represents the first screen the user encounters. The main container will have links to:

- **Vital tasks**: the user must perform these tasks quickly, even under stress.
- **Frequent tasks**: tasks the users spend the majority of their time performing. Should be fast to access.
- **Navigational aids**: helps the user to find what the application is capable of doing.

The main container does not perform any of those actions, it just provides the links to the container that will do. Other containers are derived from the concrete use cases.

2.3.-LINK THE CONTAINERS TO SHOW THE NAVIGATION FLOW

The links reflect the order of the actions in which the user perform some task. Usually identifying and linking the containers is done in parallel. The links can be labelled with conditions of interaction, that indicate a condition in which the flow can traverse the link. Links are represented in a GUI by a tool bar button, a link in a web page, etc.

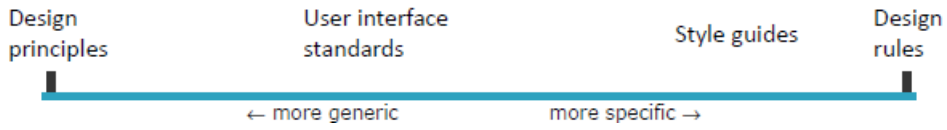
Since creating a complete content diagram for a complex application is difficult, you should iterate over prototypes. For evaluating the content diagram, you should step through some of the concrete use cases to ensure that the containers support the required functionality and the links allow the user to reach them.

The translation of the containers to GUI elements is not one-to-one. A content diagram can be spread over several screens, or different containers can be combined in a single screen.

UNIT 5.3.- PHYSICAL DESIGN

1.-INTRODUCTION

Designing a user interface consists of taking a lot of decisions. We can base our design decisions on multiple sources:



2.-DESIGN PRIINCIPLES

The **design principles** are general, applicable and enduring. They require clarification.

Principles:

- **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 every moment.
- **Simplicity**: keep the UI as simple as possible.
    - The interface should use the users' own language.
    - Use actions, icons, words and controls that are natural to the users.
    - Break complex tasks in simpler subtasks.
- **Structure**: organize the UI in a meaningful and useful way.
    - Things users think are related, should appear together.
    - Use metaphors to provide recognizable structure.
- **Consistency**: emphasizes the importance of uniformity in appearance, placement, and behaviour, for building interfaces that are easy to learn and remember. There are different types of consistency. Sometimes inconsistency is good.
- **Tolerance**: design the user interface to prevent users from making errors and facilitate recovery from them.
    - **Recoverability**: how easy it is for users to recover from their mistakes.
        - **Forward error recovery**: accept the user error and help the user to recover.
        - **Backward error recovery**: allow users to undo their actions.
    - Good error messages help users to recover from their mistakes.
    - Guidelines for creating good error messages.
        - Explain errors to help the user correct them.
        - If the user requests it, provide additional explanation.
        - Use language that the user will understand.
        - Use positive, nonthreatening language.
        - Use specific and constructive terms.
        - Make sure that the system takes the blame for the errors.

Nielsen's Principles for interaction design (1995):

1. **Visibility of system status**: the system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

2. **Match between system and the real world**: the system should speak the users' language with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
3. **User control and freedom**: users often choose system functions by mistake and will need a clearly marked "emergency exist" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
4. **Consistency and standards**: users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
5. **Error prevention**: even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
6. **Recognition rather than recall**: minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for user of the system should be visible or easily retrievable whenever appropriate.
7. **Flexibility and efficiency of use**: accelerators, unseen by the novice user, may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
8. **Aesthetic and minimalist design**: dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
9. **Help users recognize, diagnose, and recover from errors**: error messages should be expressed in plain language, precisely indicate the problem, and constructively suggest a solution.
10. **Help and documentation**: even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Design principles in **Windows 7**:

- Reduce concepts to increase confidence.
- Small things matter, good and bad.
- Be great at "look" and "do".
- Solve distractions, not discoverability.
- UX before knobs and questions.
- Personalization, not customization.
- Value the life cycle of the experience.
- Time matters, so build for people on the go.

3.-USER INTERFACES STANDARDS

User Interface Standards usually include design principles and generally agreed good practices and rules.

4.-STYLE GUIDES

Style guides provide the basic conventions for a specific product or for a family of products. It typically includes:

- A description and illustration of the required interaction styles and user interface controls, covering both the required look and feel.
- Guidance on when and how to use them.
- Screen templates to show how screens should look.

There are two types of style guides:

- Commercial.
- Customized.

Commercial Style Guides are published by a vendor and composed of highly specific design rules. They are only applicable to a particular platform.

Customized Style Guides is a style guide created specifically for a project or company. They are defined early in the development process, and can help during the requirement gathering and, in the decision, making. They promote consistency across the user interface. If they are used across the organization, it will help build a corporate look. They can be tailored to the specific circumstances of the project (characteristics of the users, their tasks and the environment).

5.-DESIGNING A USER INTERFACE

A complete UI is a combination of interaction devices and software components. One of the main problems of modern interfaces is complexity. We should try to create simpler interfaces. On the other hand, there are complex systems that require complex interfaces.

One way of making it easier for users to use the system is increasing the visibility if its interface. You should give direct access to the most used tasks, but, on the other hand, this increases the number of controls and makes the UI look complicated.

6.-THE PRINCIPLE OF GOOD LAYOUT

Principles of Good Layout:

1. **Create Natural Groupings**: taking into account the structure of the information, create logical groups. Different background, separating lines, white spaces, different fonts, etc. In a GUI, group related controls.
2. **Separate the Currently Active Components**: emphasize what the user is currently doing.
3. **Emphasize Important Components**: highlight the most important components, but no more. Combining effects reinforce the result.
4. **Use White Space Effectively**: white spaces are often more effective than lines. This may mean dividing the information between several screens.
5. **Make the Controls Visible**: the controls in the screen should suggest what their functions are, taking advantage of the users' knowledge of other UIs and of the world in general.

6. **Balance Aesthetics and Usability**: there is a trade-off between eye-catching design and usability.

7.-DESIGNING A GRAPHICAL USER INTERFACE

Each OS has its own set of widgets. We have to translate the content diagram built during the conceptual design into a GUI. Steps:

1. Choosing Widgets to Structure the Interaction.
2. Choosing Widgets to Control the Interaction.
3. Choosing Widgets to Enter Information.

7.1.-CHOOSING WIDGETS TO STRUCTURE THE INTERACTION

Most GUIs are organized using high-level widgets such as windows, dialog boxes and tabs.

A **primary window**:

- Contains a frame, title bar, menus, scroll bars, etc.
- Usually correspond to the main task objects.
- Typically, there is only a few primary windows, to which the users keep returning.
- Sometimes there is a primary window that acts as a launch pad.

Secondary windows:

- Provide additional functionality and support for the user.
- **Message boxes**:
  - Show messages, usually about a problem that the user has to deal with before continuing.
  - Message boxes are usually modal (the interaction with the rest of the application is blocked until the message box is closed).
  - Modeless message boxes allow the user to interact with the other windows of the application.
- **Dialog boxes**:
  - Invoked by the user.
  - Often used for requesting additional information from the user.
  - Can be complex screens, with text fields, command buttons, etc.
  - A **Wizard** is a series of dialog boxes in a given sequence that guides the user to complete a task.
- **Tabs**:
  - Useful for classifying the properties of the task object represented by a window.
  - The information in each tab should be independent.
  - Challenges: using too many tabs. Users may forget to complete or miss the information in one of them.

7.2.-CHOOSING WIDGETS TO CONTROL THE INTERACTION

These widgets allow the user to control the interaction:

- Menus: studied in a previous unit.
- Tool bars:
  - Complement the menu hierarchy.
  - They contain a range of frequently use command, represented by icons.
  - The purpose of a button is explained by a tooltip.
  - There are several tool bars, classified in logical groups.
  - It can be hard to select the proper icons.
  - Ribbons combine menus and toolbars.
- Command buttons:
  - Usually for controlling the operation of dialog boxes.
  - It is important to use understandable levels.
  - Position:
    - Windows: OK, Cancel.
    - Mac OS X: Cancel, Print.
  - Size: equal size and shape to visually group the buttons. Buttons in a row can have different widths.

7.3.-CHOOSING WIDGETS TO ENTER INFORMATION

These widgets allow the user to enter information:

- Radio buttons and check boxes.
- List boxes:
  - Allow the user to choose from a large number of options.
  - Single/multiple selection.
  - Drop down/permanent, depending on the space available.
  - List boxes are more flexible than check boxes/radio buttons.
  - Sensible default values should be used to speed up the use of the program.
  - List boxes can be combined with a text box, that usually acts as a filter.
- Text boxes:
  - The most flexible widget for entering information.
  - Not suitable for entering formatted information.
  - Guidelines:
    - The size of the text box should indicate the expected amount of information.
    - If the user can enter large amount of text, use a multi-line text box with scroll bars.

UNIT 5.4.- PROTOTYPING

1.-INTRODUCTION

A prototype is a first or early example that is used as a model for what comes later.

2.-MOTIVATION

Prototyping encourages the exploration of multiple options, and allows designers to easily evolve, demonstrate and evaluate designs. It helps to focus on the design, and not on the implementation details. Prototypes are almost always incomplete, easy to change and are thrown away when not needed. There are several types of prototypes that provide different levels of fidelity, each type is appropriated for a stage in the development process. From low to high fidelity: storyboards, paper prototypes, digital mockups and dynamic prototypes.

3.-LOW FIDELITY PROTOTYPES

Storyboards focus on the tasks, rather than in the interface. They are used during the requirement analysis stage.

Paper prototyping:

- We start designing the actual user interface.
- Draw a mockup of the actual user interface in paper.
- Using paper, post-it notes and markers, instead of using a computer program, avoids trying to get the design beautiful and exact.
- Users can get involved in the evaluation and evolution of the design.

Paper prototyping **guidelines**:

- Keep all the materials in one place and have different types.
- Work quickly and make reusable components.
- If something is difficult to simulate.
- Large poster boards can be useful to contain the prototype and provide context for the user.
- Mix and match hardware and software, and different fidelities.
- If appropriate, add context by including familiar operating system elements.

Paper prototypes are also used during the requirement analysis phase.

4.-LOW TO MEDIUM FIDELITY PROTOTYPES

Wireframes are typically used in web design, but also are applicable to mobile or desktop. It shows the page layout, but no typographic style, colour or graphics. The focus is on functionality, behaviour and priority of content. They connect the underlying conceptual structure to the visual design.

5.-HIGH FIDELITY PROTOTYPES

Digital mockups are more detailed approaches to the final design. They allow for more formal evaluation.

6.-VIDEO PROTOTYPES

Video prototyping is cheap and fast to make. It is a great communication tool, it shows context and it is portable and self-explanatory. It ties interface decisions to tasks by helping to orient the interface choices, making sure you think of a complete interface, and helping to detect unnecessary elements. It can be used at any stage of the process.

It should **show** the whole task, including motivation and success. It should also illustrate the most important tasks. The tasks that are not important enough to appear in the video, probably don't need to go in the first version of the system.

In order to **make** a video prototype:

- Start with an outline.
- It is OK to start recording to see what happens.
- The camera is not important.
- Find people and a realistic location.
- Remember that the important thing is the message, the production quality of the video.

Considerations:

- Use audio or a silent movie with subtitles.
- Interface can be paper, mockups, code or invisible.
- Can show both success and failure.
- Edit as little as possible, because editing is very time-consuming.

7.-EVALUATING PROTOTYPES

Problem: in order to get feedback on an interactive application from users, we need a working prototype. The **Wizard of Oz** technique allows us to evaluate user interaction with early prototypes. There is a human operator that moves the interactive elements on behalf of the application. Therefore, there is little, or no code written. It allows testing the user interaction using a technology that still does not exist. It makes sense if it is faster to develop than the real thing.

Challenges:

- High fidelity interfaces that "work" may make the user think that the development is done.
- High fidelity interfaces make it harder to users to give a strong critique.
- It is easy to prototype something that can't be built.
- If you want your users to believe it is real system, you will have to build some kind of remote control, that allows the wizard to be hidden.

UNIT 6.- EVALUATING INTERFACES DESIGNS

1.-INTRODUCTION

Evaluating our interfaces is essential as it helps to understand the user experience with the system and, where there are difficulties, to find ways of improving it. Depending on the project, an appropriate amount of resources should be assigned to testing (5-20%). Usability tests have to be carried out during the whole development process, not only at the end.

In critical systems, both high load situations and even partial failure situations should be tested, and, sometime, this cannot be done in the laboratory, and has to be done in its real context. There are multiple evaluation methodologies depending on the application type, type of interface, type of users, etc.

Remember that usability is not an abstract concept. It can be measured and evaluated. The evaluation should measure how the final product adheres to the usability requirements:

- **Qualitative usability requirements**: desired features that can be subjective and sometimes hard to measure.
- **Quantitative usability requirements** or **usability metrics**: the requirements are expressed with numbers.

Levels for evaluating a usability metric:

- Current.
- Best case.
- Planned.
- Worst case.

2.-TYPES OF EVALUATION

What:

- Find as many usability problems as possible: **diagnostic evaluation**.
- Assess the extent to which a system meets its requirements: **measurement evaluation**.

When:

- During the development: **formative evaluation**.
- At the end: **summative evaluation**.

How:

- Early, with low fidelity prototypes, informal: **exploratory evaluation**.
- At the end, verify if the system meets the requirements, formal experiment: **validation evaluation**.
- Choosing one between several options, statistical analysis: **comparison evaluation**.

Who:

- Users: **user observations**.
- Experts: **expert inspection**.

3.-USERS STUDIES

Usually five users are enough. Ideally a real user, a representative of a user profile, a usability expert or a domain expert should be chosen. Depending on where the study is performed:

- Users' own environment: **field studies**.
- Other: **controlled studies**.

Field tests and **portable labs**:

- The usability lab is brought to the place where the final system will be used.
- It is important to capture the largest amount of information in each execution.
- Other option is to release beta versions to a high number of users and ask for comments.

Remote usability testing:

- For web-based applications, a large number of users can test the system from the place where they will use it.
- Participants can be recruited by e-mail from client databases or on-line forums.
- Tests can be synchronous or asynchronous.
- Advantages: large number of participants, inexpensive, tests the users hardware.
- Disadvantages: less control on the user behaviour and it is difficult to capture their reactions.
- Some studies have found that this type of tests find more problems than traditional techniques.

Can-you-break-this tests:

- Videogame developers pioneered this type of tests, where the user is asked to "break" the system.
- These stress tests help to make applications more robust.

Competitive usability testing:

- Compare a new interface to previous versions, or with the interface of a competitor.
- Compare the time it takes to complete a task, or the error rate in both systems.

Before the experiment, a detailed plan has to be agreed upon, including: what to measure, number, types and sources of the participants, duration (30-90 min), tasks to be carried out by the subjects, and the content of the questionnaires and interviews.

Run a pilot test with a small number of subjects (1-3):

- Tests that all details of the evaluation are taken care of.
- Perform the test in the same location and as similarly as possible to the actual tests.
- Analyse and interpret the data, to ensure that all the required information is collected.

Structure of a session:

- Welcome.
    - Participants should always be treated with respect and let them know that **it is not them** who are being evaluated.
    - They should also be informed about:
        - The purpose of the study,

        - What they will be doing and how long. How to ask for a break.
        - Who will review the recording and their use after the study.
        - A statement of confidentiality and how the anonymity of the participant is preserved.
        - The risk of taking the test.
        - The fact that participation is voluntary, and she can withdraw at any time with no penalty.
        - A way to contact with questions.
- Sing a consent form and maybe a non-disclosure agreement.
- Recruitment screener: ensure that the participant fits to our expected profile.
- Use task scenarios (adapted to users' language) for specifying the instructions for the users.
    - Select the most important ones.
- Post-session discussion.
    - Review the recording asking about their thoughts and questions.
    - Questionnaires.
- Incentive.

Measuring time data:

- Use a stopwatch.
- Record a time stamp with every comment written during the session.
- Use a key logger, that records every keystroke or mouse click, but relate written comments with each event.

During the test, a usability testing technique consists of asking the participant to think aloud. The observer should facilitate the communication and prompt for what the user is thinking. The observer must not help the subject.

Advantages:

- Immediate feedback on the participant's opinions about the interface and any problems or surprises.
- It can help users to focus and concentrate during the evaluation session.
- Useful for collecting qualitative data.

Disadvantages:

- Some participants can find thinking aloud unnatural and distracting.
- Thinking aloud can slow the participant's thought processes, artificially reducing their performance and error rates.
- It can become very exhausting for the user.

Another technique is the **retrospective protocol**: ask users for comments about their actions after the test.

Advantages:

- Does not interfere with performance measurements.
- Useful for collecting quantitative data.

Disadvantages:

- The participant may forget the reason she did some action.
- Some participants can be intimidated by the cameras.

Recording the participants is valuable for reviewing later their reactions, errors, how they work, etc. Eye tracking system computer the areas of the screen that receive more attention.

Questionnaires are a familiar and inexpensive method to capture the user's or expert's opinion. It allows to poll thousands of users. The keys to success are having clear goals and developing focused items. It can be on paper or online.

Advantages:

- Harder to forget to ask something.
- Comparing answers from different participants is easier, since all of them see the same questions.
- Quantitative data can be collected.
- Progress is demonstrated by improved scores on subsequent surveys.

Disadvantages:

- It is difficult to design a good questionnaire.
- Closed questions are easier to analyse, but do not give the reason why the user selected it.

Types of question:

- **Likert scale**. The user has to assign an agreement level to statement: strongly agree, agree, neutral, disagree, strongly disagree. It typically uses a 5-level scale, but 7 or 9 are also possible (always symmetrical).
- **Bipolar items** to describe users' reactions to using a system.
- **Rating of reactions**.

There are many predesigned and validated questionnaires that can be used:

- **QUIS** Questionnaire for User Interaction Satisfaction.
    - Designed to evaluate user objectives satisfaction with respect to specific aspects of the interface.
    - Studies nine factors of the interface:
        - Display factors.
        - Terminology and system feedback.
        - Learning factors.
        - System features.
        - Technical manuals.
        - Online tutorials.
        - Multimedia.
        - Teleconference.
        - Software installation.
    - Nine-point scale.
    - The questionnaire has to be adapted to the features of each interface.
- **SUS** System Usability Scale.

    - Simpler than QUIS.
    - 10 statements to which participants respond using a 5-point scale.
    - Half of the questions are positively worded and the other half negatively worded.
- **CSUQ** Computer System Usability Questionnaire.
    - 19 statements to which participants respond using a 7-point scale.
- **WAMMI** Website Analysis and MeasureMent Inventory.
    - Web-based evaluation service.
    - 20 statements to which participants respond using a 5-point scale.
    - It provides web support to run the survey, and then it generates the final report.

After the session, you will have collected a great amount of data. A usability defect is a usability problem in the user interface that can lead to confusion, error, delay or failure to complete some task.

Summarizing Quantitative Data:

- Tabulations, charts and rankings.
- Descriptive statistics. Be careful, as they can be misleading.
- Inferential statistics. Requires a high number of participants.

After interpreting the results, you can produce recommendations, such as:

- Successes to build on.
- Defects to fix.
- Possible defects or successes, without enough evidence.
- Areas of the UI not tested.
- Change to usability and other requirements.

4.-EXPERTS REVIEWS

Often, designers make informal evaluations asking colleagues and clients for their opinion. A more effective technique involves having an expert evaluator to test the design. The result of an expert evaluation can be:

- A formal report with the list of problems identified and recommendations.
- A presentation or discussion with designers or managers.

Advantages:

- The results are available quicker.
- Less expensive than user observation.
- Inspectors can suggest solutions to the defects.
- Help to find obvious errors easily, so they don’t arrive to the evaluation with users.

Disadvantages:

- Inspectors are not real users, and their prediction of what users will do with the UI or how important a defect is may fail.
- Inspectors have their own preferences about UI design, which may bias the evaluation data.

- Expert review is highly dependent on the inspector's experience.

The reviewer should:

- Replicate the condition of the end user.
- Be sensitive in his/her recommendations.
- Realize that it is difficult for someone not directly involved to understand the design decisions and the development history.
- Have experience on the type of application being evaluated.
- Leave the development of the solutions to the developers.
- Be comprehensive in his/her report.
- Review the consistency across all the windows of the application.

Evaluation methods:

- **Heuristic evaluation**: the reviewer assesses whether the interface follows a list of design heuristics.
- **Guidelines reviews**: the expert checks that the interface follows the guidelines.
- **Standards inspection**: check that the design adheres to a standard.
- **Consistency inspection**: check terminology, fonts, colour schemes, I/O formats, etc. within the interfaces as well as the documentation.

The final report should:

- Use guidelines to provide a good structure.
- Separate the problems depending on the users they affect.
- Rank the recommendations by their importance.
- Describe each recommendation at the conceptual level.
- Be aware of business and technical constraints.
- Solve the whole problem, not just a special case.
- Give specific and clear recommendations, with examples.
- Include small details like typos, poorly aligned data-entry fields, inconsistent layout of the controls, etc.

Bird's-eye view:

- Study a full set of printed screens laid out on the floor or pinned walls.
- Detects inconsistencies.
- Detects unusual patterns.

5.-OTHER TYPES OF EVALUATION

Focus groups:

- About 8 people lead by a moderator discuss something.
- Widely used in marketing.
- Useful when the UI does not exist yet.

Card sorting:

- Ask users to group items.
- Useful to find underlying categories and structures.
- Print each item in a card and ask users to make groups, and probably give the group name.
- Study the results, both:
    - When there are general agreements.
    - When there is not agreement, and study how to solve it.

Automatic validation tools.

6.-ACCEPTANCE TESTS

Acceptance tests are tests performed by the client to check that the delivered system meet the requirements. It is necessary to establish objective, measurable criteria:

- Time for users to learn specific functions.
- Speed of task performance.
- Rate of errors by users.
- User retention pf commands over time.
- Subjective user satisfaction.

Precise acceptance criteria save arguments and can demonstrate contractual fulfilment objectively. Acceptance tests should be carried out by an outside organization. After validation testing, there may be a period of field testing before distribution.

7.-EVALUATION DURING ACTIVE USE

After the system has been released, developers should study how it is being used to improve it. As user numbers grow, major changes to the interface should be limited to an announced annual or semiannual revision. Tools:

- Interviews and focus-groups discussions.
- Continuous user-performance data logging.
- Online or telephone consultants, e-mail, and online suggestion boxes.
- Discussion groups, wikis, and newsgroups.
- Tools for automated evaluation.