Usability hay còn gọi là Computer Human Interaction (CHI), Human Computer Interaction (HCI), Man Machine Interface (MMI), User System Interface (USI), User Interface (UI), Graphical User Interface (GUI), Soldier Machine Interface (SMI), Human Computer Communication (HCC), Operator Interface (OI).

Định nghĩa: User interface (UI) “all components of an interactive system that provide information and controls for the user to accomplish specific tasks with the interactive system”. User experience (UX) “person's perceptions and responses resulting from the use and/or anticipated use of a product, system or service”.

Usability “extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use”. Answer the question: How well users can use the system’s functionality?

Dimension of usability: Learnability (How easy it is to learn and use?), Efficiency (How quickly users perform tasks using the UI?), Memorability (How easy it is for users to reestablish proficiency?), Errors (Are the errors committed by users often? Is it easy to recover from errors?), Satisfaction (Are users satisfied with the UI?)

Importance of usability dimensions vary: Dependent on user (Novice users need learnability, Infrequent users need memorability, Experts need efficiency) and Novice or expert is dependent on domain, application, feature experience

Importance of usability: Usability is a condition for survival. Users often judge a system by its interface rather than its functionality. A poorly designed interface can cause a user to make catastrophic errors. Poor user interface design is the reason why so many software systems are never used.

Problems with UI: User interface accounts for a large portion of lifecycle costs (40% application algorithms + 40% dialogue management + 20% presentation). UI development is extremely labor intensive. UI’s require frequent and extensive modifications. Modifications to the UI are difficult with the UI and the application are tightly interwoven. Designing UI is hard because software engineer is not the user, they are trained to communicate with software engineers but UI requires communicating with users. The user is always rights because consistent errors are the result of wrong interface but the user is not always right because they are not designers.

Human factors in UI design: Limited short-term memory (People can remember about 7 items of information. If you present more than that, they tend to make mistakes. When people make mistakes, they tend to make more. People are different: different physical capabilities, different interaction preferences (Some like pictures, some like text).

Learnability: Learning is a process of transferring and putting information from short-term to long-term memory. Short-term memory (working memory): Small (~ 7 items or “chunks”). Short-lived (~10 seconds). Repeating helps retain chunks, distraction does the opposite. Chunk is a unit of memory or perception. Chunk depends on how the information is presented and what you already know (it links with the past experience).

Recognition and Recall. It is easier to recognize than recall things.

Recognition: Remembering with the help of a visible cue (evidence – bằng chứng) like you recognize your friend easily when seeing his/her face, but you may not remember his/her name. Recall: Remembering with no help like you remember a person when someone refers to him. Implications: Performing operations via visual presentation is more learnable than via command line. Direct manipulation is more learnable than other interface styles.

Models: Model of a system is a presentation of its operations [Elements of a system; How these elements work together to carry out its operations]. Three kinds of models relevant to UI design [System model or implementation model]. Internal structure and interactions of the system’s operations. How system works internally. Visio’s objects vs. Photoshop’s images [Interface model]. How system works through its interface. Command line vs. Menu. Editing Visio’s objects vs. editing Photoshop’s images [User model or mental model or conceptual model]. How the user thinks the system works. Interface model encapsulates or hides system model [It should be simple and appropriated]. Intreface model should closely reflect user model [Does this beautiful dog do searching?]. User model may be wrong [So, errors happen]

Learnability Principles: Ways to communicate and present the system model [Affordances; Natural mapping; Visibility; Feedback]. Consistency [Internal, external, and metaphorical; Speak the user’s language; Metaphors; Platform standards]

Affordances: "Perceived and actual properties of a thing" – Don Norman. "Perceived" may be different from "actual"

Natural mapping: Physical arrangement of controls matches arrangement of their operations. It’s best to map directly, but not always have to be [Light switches; Car’s turn signals]

Visibility: Operations should be visible to users [Unix commands are very invisible vs. Windows’ menus; Right click menus are not very visible]. A reason why iOS does not support much right-click [Drag-drop is not either]. But it’s a direct manipulation style reflecting real world. Visibility versus Simplicity [More visibility may result in reduced simplicity]

Feedback: Actions should have immediate effects [e.g., push buttons, scroll bars, mouse icons]. Feedback types [Audio; Visual; Haptic (giving a feeling, e.g., vibration of a mouse click)]

Consistency: Similar things should work similarly. Fonts, colors, icons, layouts, etc.. Different things should look different. Consistency types [Internal: within the system; External: across different systems; Metaphorical: reflecting real-world objects]. Speak the user’s language [Use common words, avoid slangs and jargon; But avoid wordy and overly verbose]

Metaphors: Metaphor is a presentation of real-world in user interface. Advantages [Highly learnable; Connect with user’s existing model easily]. Problems [Hard to design metaphors that are appropriate; Potentially deceptive and misleading; May not be used consistently everywhere; Cuturally dependent (localization issue)]

Platform standards: Follow guidelines of platforms [MS Windows user interface guidelines; Apple user experience guidelines]. Follow frameworks [Various frameworks have their own looks and feels guidelines]. Learn from existing applications

Efficiency

Human information processing Fitts’s law: Time T to move hand to a target of size S at distance D away from the mouse pointer is T = a + b \* log (D/S + 1) [a and b are constants; T is dependent only on log (D/S + 1); log (D/S + 1) is defined as index of difficulty]

Implications of Fitts’s law: Similar targets should be grouped. Targets at screen edge are easy to hit. Pie menu is faster to use than linear menu [It’s faster 15-20% according to a study by Callahan, 1994]. Lengthy menus should be avoided

Power law of practice: Time Tn to do a task the nth time is Tn = T1 \* n-a (a is typically from 0.2 to 0.6). Implications [With practice, novices get better; But their performance becomes nearly flat; Remember the Nielsen’s Learning curve?]

Principles to improve efficiency: Make often-used targets big. Group targets that are used together [Grouped toolbar buttons, menu items, etc.]. Place oftenly-used menu items on top of menu. Use screen corners and edges. Use keyboard shortcuts and menu accelerators. Predefine a group of styles. Aggregating and choose most common selections by default. Use defaults. Keep history (e.g., recent files in Word). Auto completion. Auto suggestion [This makes you lazy, doesn’t it?]. Anticipation [Anticipate what users will do next and present corresponding operations for them]

UI Design Processes

Waterfall Model

Disadvantage: difficult to handle changes

Waterfall Model Problems: Users are not involved in evaluation until acceptance testing. UI problems result in changes in requirements and design [Waste of effort spent earlier]. Inflexible partitioning of the project into distinct stages [it is difficult to respond to changing customer requirements]. It is only appropriate when the requirements are well-understood [Few business systems have stable requirements]

Shneiderman’s Interactive Systems Lifecycle

Software development lifecycle for interactive systems:

1. Collect Information [Organize the design team; Obtain management and customer participation; Conduct interviews with users; Submit written questionnaires to users; Estimate development, training, usage, maintenance costs; Prepare a schedule with observable milestones and reviews]

2. Define requirements and semantics [Define high-level goals and middle-level requirements; Consider task flow sequencing alternatives; Create task objects and actions; Obtain management and customer agreement on goals, requirements, and semantic design]

3. Design syntax and support facilities [Compare alternative display formats; Design informative feedback for each operation; Review, evaluate, and revise design specifications; Carry out paper-and-pencil pilot tests or field studies with an online mock-up or prototype]

4. Specify physical devices [Choose hard- or softcopy devices; Select audio, graphics, or peripheral devices; Consider work environment noise, lighting, table space, etc.; Carry out further pilot tests and revise design]

5. Develop software [Use appropriate development tools; Develop code; Perform unit test]

6. Integrate system and disseminate to users [Assure user involvement at every stage; Conduct acceptance tests and fine tune the system; User documentation and training]

7. Nurture the user community [User support; Monitor usage and measurement]

8. Prepare evolutionary plan

Iterative Design: Each cycle is one iteration. Release is produced at the end of each iteration. Customer’s feedback and evaluations can be incorporated into next release. Problems [It’s expensive to use customer’s time to test; Customers may not be available; Customers don’t like à they don’t buy]

Spiral Model: Process is represented as a spiral rather than as a sequence of activities with backtracking. Each loop in the spiral represents a phase in the process. No fixed phases such as specification or design loops in the spiral are chosen depending on what is required. Risks are explicitly assessed and resolved throughout the process. An improvement of iterative design. Early cycles use cheap prototypes [Paper prototypes; Sketches on computer; Quick prototyping tools]. Providing multiple prototype alternatives [Parallel prototyping]. Later cycles should be better than early ones. Only mature releases of later cycles can be distributed to users

User-Centered Design: Also known as Participatory Design. A type of iterative design with Spiral. Focusing on users and tasks [User analysis: who uses the system; Task analysis: what users need to do]. Getting users involved in the process [Users as evaluators, consultants and designers (sometimes)]. Constant evaluation [Users evaluate prototypes and releases]. Advantages [Accurate information and useful suggestions; Opportunity to argue over design decisions; Increased ego involvement in system success]. Potential problems [Users are not always available to participate; Their time maybe expensive; Users are not UI designers; Users have strong ego and preferences; UI designers overly obey users’ preferences]

Process for Projects in This Class:

0. Propose project

1. Perform user and task analysis

2. Initially design sketches

3. Initially prototype

4. Evaluate prototype

5. Detail prototype

6. Evaluate prototype

7. Implement

8. Perform user testing

Applying User-centered Design: You are all potential users of proposed apps. Users help identify problems [Members of other groups review a group’s proposal; Collect feedback from potential users; Observer existing users’ actions]. Users review and provide feedback [In each milestones, members of other groups provide feedback on design of a group]. User evaluation [By the end of the project: everyone will review the design of another group]

How to be successful in teamwork?: Define clear goals and expectations. Assign clear responsibilities and tasks for everyone. Talk about accountability [Who is responsible when things go wrong]. Meet weekly to review status, even if no assignment is due that week [Record meeting minutes]. Work early than late. Understand your teammates [Motivation, commitment, capability]

UI Design principles: Determine users’ skill levels [Novice/first-time users; Knowledgeable intermittent users; Experts and frequent users]. Identify the tasks [Frequent actions; Less frequent actions; Infrequent actions]. Choose appropriate interaction styles [Direct manipulation; Menu selection; Form fillin; Command language; Natural language]. Use Shneiderman’s eight golden rules of interface design [Strive for consistency; Cater to universal usability; Offer informative feedback; Design dialogs to yield closure; Prevent errors, rapid recovery; Permit easy reversal of actions; Support user control; Reduce memory load]. Prevent errors [Constructive and informative error messages; Organizing screens and menus functionally; Providing feedback about the state of the interface; Correct actions]. E.g., grayed menu items [Complete sequences]. E.g., wizard windows often have both Next and Finish buttons. Increase automation while preserving human control [Auto suggestion; Auto completion; Allowing users to change]

Perform user and Task Analysis

User analysis: The process of identifying and describing the users who use the system. Characteristics of target users [Age, gender, culture, language; Computer experience; Domain experience, application experience; Usage frequency; Physical limitations; Education; Motivation; Work environment; User relationships; User social status]. Description of target users [General information; User characteristics (discussed above); User environment]. Where the tasks will be performed? [Major goals of the job]. What is the end result? [User roles (e.g., buyer, seller)]. if any [User preferences; Relationships among users]. Obstacles/challenges of user analysis [Designers and users are sometimes isolated; Users may be overlooked by designers]. Designers may make wrong assumptions about users [It’s expensive and difficult to talk to some users]. E.g., high-ranking people, doctors, executives. Techniques to do user analysis [Recording; Interviews; Questionnaires; Observation; Combination of the above]

Task analysis: The process of analyzing and documenting the tasks that the system may provide to users [What needs to be done (goal); What conditions to do the task (precondition); What steps to be taken (subtasks)]. Each task is often a goal to achieve by users. Task analysis is an early step in UI design that provides basis for [UI designing; UI evaluation and improvement; User documentation]. Two main steps: Model tasks (Gathering information + Describing tasks into requirements) and Evaluate and refine (Review and update requirements)

Model tasks: Create a list of all tasks to be performed by users. Rank the tasks by frequency of use and importance. Gather other detailed information about each task. Model the relationships (e.g., using use-case model) [between tasks and users; among tasks]. Present/describe tasks in forms of documents, diagram, etc.

Techniques to do task analysis: Techniques to gather in formation (same as doing user analysis) [Data recording; Interviews; Questionnaires; Observation; Combination of the above]. Technique to analyze [Task decomposition]

Data recording: Documents, manuals, instructions. Notes, audio, photographs. Notes + photographs. Audio + photographs. Video

Interviews: Structured [tightly scripted, often like a questionnaire; replicable but may lack richness]. Unstructured [not directed by a script; rich but not replicable]. Semi-structured [guided by a script but interesting issues can be explored in more depth; can provide a good balance between richness and replicability]

Questionnaires: Paper, email and the web used for dissemination. Questions can be closed or open [closed questions are easier to analyze, and may be done by computer]. Can be administered to large populations. Sampling can be a problem when the size of a population is unknown

Online questionnaires: Advantages [Responses are usually received quickly; Data can be collected directly into database for analysis; Time required for data analysis is reduced; Errors can be corrected easily; Many online survey tools available]. Problems [Sampling is problematic if population size is unknown; Preventing individuals from responding more than once; Delayed response]

Observation: Direct observation [in the field or in controlled environments; Structuring frameworks; Think-aloud protocol: Person talks about what they are doing, while they are doing it (or just before or after). Observer can ask probe questions; Probe questions affect performance, as does thinking aloud]. Indirect observation [tracking users’ activities]. Physical location/movement. Interaction logging, timers

Task decomposition: Aims [describe the actions people do; describe order of subtasks; structure them within task subtask hierarchy]. Hierarchical Task Analysis (HTA) [introduced by Annett and Duncan (1967) to evaluate an]

organization’s training needs [very useful for analyzing and representing the behavioral]

aspects of complex tasks [now widely used in interface design]

Hierarchical Task Analysis (HTA): Breaks tasks into subtasks and operations or actions [These components are represented using a structure chart]. Includes [identifying and categorizing tasks; identifying the subtasks; checking the overall accuracy of the model]. Useful for UI design [Enabling designers to envision the goals, tasks, subtasks, operations, and plan essential to users’ activities]

Generating the Hierarchy

1. Start from overall goal, e.g. clean the house

2. Get list of tasks

3. Break down into numbered sub-tasks [Group tasks into higher level tasks; Decompose lowest level tasks further]

4. Describe each sub-task

Evaluate and refine requirements: Evaluate, simplify and fix issues in the task description. Evaluation techniques [Walk-through; Formal review/inspection; Offline review; Online review]

Domain analysis: The process identifying data models for the system domain [People and things; How they are related]. Outputs [Object or class models (e.g., using UML diagram); Data models (Entity Relationship models)]

Prototyping: Why we need to express design ideas early? [You can’t evaluate design until it’s created; After software is built, changes to the design are difficult]. We want [Make it fast; Allow lots of flexibility for radically different alternatives; Low cost; Promote valuable feedback]

What is prototype?: A prototype is an original type, form, or instance of something serving as a typical example, basis, or standard for other things of the same category

Why we need prototypes?: Experiment with alternative designs. Provide an early, concrete representation of design ideas. Provide hands-on experience for all stakeholders (design teams, users, etc.). Easier to change or throw away. Keep the design centered on the user [must test and observe ideas with users]. Facilitate iterative design and evaluation. Reduce the risk of making customers surprise

Prototype fidelity: Low-fidelity [a set of sketches/storyboards providing a static, non-computerized, non-working mockup of the planned product; omits details]. High-fidelity [a set of screens that provide a dynamic, computerized, working model of the planed product; working software]

Dimensions of fidelity: Horizontal [Prototypes cover many features but with little detail]. Vertical [Prototypes cover few features but with much in detail]. Diagonal [Prototypes cover down to a certain level and vertical. Look [Is the appearance and graphic design of the UI; Can be sketchy and hand-drawn]. Feel [Referring to input methods to interact with the UI; Pointing and writing (in paper-mockup) is different from mouse and keyboard]

Paper prototype: Using paper mockup to represent the UI [Sketches of screen appearance; Paper pieces show windows, menus, dialog boxes, toolbars]. Interaction [Pointing with a finger = mouse click; Writing = typing]. A person simulates the computer’s operation [Putting down and picking up pieces; Writing responses on the mockup (screen); Describing effects that are hard to show on paper]. Characteristics [Low fidelity in look and feel; High fidelity in depth as there is person to simulate the operation]. Advantages [Faster to build; Easier to change; Cheap; Focuses attention on big picture: Designers don’t waste time on details. Attract ideas from customers; Non-programmers can help; Convenient: You can prototype on the bus. You can utilize your time efficiently]

Tips for good paper prototypes: Make it large. Make it monochrome (single color). Use description where necessary [You cannot represent tricky interactions like drag & drop, animation, progress bar]. Keep pieces organized [Use folders and envelopes]. Produce multiple alternatives [Better to get feedback]

What can paper prototypes help?: It helps better understanding of [conceptual model]. Do users understand the UI? [functionality of the system]. What features are missing in the UI? [navigation and task flow]. Do users understand the navigation of the UI? [terminology]. Are terms and levels understood? [screen content and layout]. What are there in the UI?

What it does not help?: Showing “look”: color, font, whitespace, etc.. Demonstrating “feel”: efficiency issues [Interactions are in low fidelity (not real)]. Measuring response time. Demonstrating animation and high-level of interaction [Actions like drag and drop, drawing, etc.]

Computer prototype: Interactive software simulation. High-fidelity in look & feel. Low-fidelity in depth [May be no backend, covering horizontally; Does not have a human simulating the backend like paper prototype]

What can computer prototypes help?: Everything from paper prototypes, plus. Better and higher-fidelity look [Screen layout; Colors, fonts, icons, etc.; Choices of controls]. Interactive feedback. Efficient issues [Controls are big enough?; Whitespace?; Distance between controls?]

Advantages of computer prototype: Faster than coding. No debugging. Easier to change and throw away. Separate UI design ideas from what offered by UI toolkit (e.g., Visual Studios, C++ Builder) [Your thinking is not limited to available widgets]: Non-programmers can do it

Computer prototyping techniques: Storyboard: Sequence of painted screenshots, sometimes connected with links [Pros: You can draw anything. Fast; Cons: No or limited interaction (No text entry, Widgets aren’t active)]. Form builders: Creating real windows with widgets such as buttons. [Pros: Actual controls → high-fidelity in terms of look. You can reuse the design for implementation → save effort from doing again; Cons: Limits thinking to standard and available widgets. Content in each widget is not visible]. Wizard of Oz: Computer frontend, human backend. [“Wizard of Oz” = “man behind the curtain”; Software simulation with human in the loop to help]

Graphic Design: Graphic design: refers to artistic and professional disciplines which focus on visual communication and presentation. In UI, it refers the look and feel portion of an interface [initially encountered by users (eye catching); conveys an impression, mood, beauty, etc.; facilitates finishing the task at hand; suggests trust]

Graphic design philosophies: Aesthetic appeal does not automatically confer usability. UI design must balance the meaning of its visual elements that conform the mental model of operation. Preferences [Simple and natural user’s “language”; Economy of visual elements; Clean, well organized; Less is more]

Guidelines for good graphic design: Basic principles [Metaphor; Simplicity and Clarity; Consistency; Organization/Alignment/Proximity/Grid; Legibility and readability; Color/Contrast]

Simplicity: “Keep it simple, stupid.” (KISS). “Less is more.”. “When in doubt, leave it out.”. Every element in an interface should have a reason for being there [make that reason clear too]

Techniques for simplicity: Reduction [Decide what essentially needs to be conveyed by the design; Examines every element if it serves essential purposes; Remove inessential elements]

Regularity [Use a regular pattern (minimize the unnecessary differences between elements with regularity; elements provided by the operating system. The users are already familiar them); Limit inessential variation among elements (use the same font, color, line width, dimensions, orientation formultiple elements; irregularities in your design will be magnified in the user’s eyes and assigned meaning and significance)]. Combine elements

Consistency: Similar things should work similarly. Different things should look different. Consistency types [Internal: within the system, elements follow same conventions and rules; External: across different systems, follow platform and interface style conventions; Metaphorical: reflecting real-world objects]

Affordance: "Perceived and actual properties of a thing that determine how the thing could be used" – Don Norman. An affordance is a quality of an object that allows an individual to perform an action [Related to look and feel of an object]. Two types of affordance [Perceived affordance: design invites people to take possible actions; Actual affordance: the actual actionable properties of the product]. Affordance can be dependent on [Experience; Knowledge; Culture of users]

Affordance can be dependent on the [Context; Layout; Locations of the objects placed].

Perceived affordance (perception) can differ from real affordance [the paper-made chair has a perceived affordance for sitting; but it doesn’t actually afford sitting: it collapses under your weight]

Perceived and actual affordance: Affordance [Perceived affordance (perception): design invites people to take possible actions; Actual affordance: the actual actionable properties of the product]. Problems occur when [these are not the same; people’s perceptions are not what the designer expects]. A false affordance exists when there is no action offered but the information that specifies it is. A false affordance exists when there is no action offered but the information that specifies it is. Perceptible affordance – real affordance. Hidden affordance. Correct rejection

Signifiers: An affordance is a quality of an object that allows an individual to perform an action. A signifier is a thing that communicates the affordance of an object. A signifier can be labels, instructions, shapes, colors, layouts, sound, videos, animations, mouse shapes, etc.

Affordance in HCI design: In HCI, interfaces are virtual and do not have affordances like physical objects. It does not make sense to talk about interfaces in terms of ‘real’ affordances - Norman. Interfaces are better conceptualized as ‘perceived’ affordances. In graphical, screen-based interfaces, the designer can control over perceived affordances [Display screen, pointing device, selection buttons, keyboard; Actions including touching, pointing, looking, clicking on every pixel of the display]. GUI design [perception only through visuals; designer creates appropriate visual affordances via familiar idioms, metaphors]

Visible constraints: Limitations of the actions possible perceived from object’s appearance [provides people with a range of usage possibilities]. The more constraints, the less opportunity for error [particularly important for managing user input]. Benefits [Restricting user actions to valid actions; Helps prevent from selecting incorrect options; Eliminate need for perfect knowledge; Recognition over recall]. The more constraints, the less opportunity for error [But too much constraint, less flexible and less efficient]. Physical constraints [refer to the way physical objects restrict the movement of things]. Logical constraints [Exploit people’s everyday common sense reasoning about the way the world works]. Cultural constraints [Groups of people learn idioms; But these differ in different places]

Web UI Importance of Web Design: Millions of websites on Internet [and growing everyday, even faster; some are visited often, some to a small audience]. Purpose of websites [communication; education/information; e-commerce/e-business; entertainment]. Good Web site design can lead to good return on. Two most elusive goals of effective Web design [user satisfaction and return on investment (ROI)]. Usability is still the critical issue for Web design [if your site is difficult to use, not clear what is offered, hard to read, doesn’t work the way they are used, get lost, get frustrated - they leave]

Top 10 Mistakes in Web Design (from Jakob Nielsen, 2011)

1. Bad Search

2. PDF Files for Online Reading

3. Not Changing the Color of Visited Links

4. Non-Scannable Text

5. Fixed Font Size

6. Page Titles With Low Search Engine Visibility

7. Anything That Looks Like an Advertisement

8. Violating Design Conventions

9. Opening New Browser Windows

10. Not Answering Users' Questions

Ten Good Deeds in Web Design (Jakob Nielsen)

1. Place the name and logo on every page and make a link

2. Provide search if the site has more than 100 pages

3. Write straightforward and simple headlines and page titles

4. Structure the page to facilitate scanning

5. Use hypertext to structure the content space into a several subtopics

6. Use product photos, but avoid cluttered and bloated pages

7. Use relevance-enhanced image reduction for small images

8. Use link titles to provide users with a preview of where will take them

9. Ensure all important pages are accessible for users with disabilities

10. Do the same as everybody else

Interaction Styles

**Menu.** Advantages of menu: Self-explanatory [Reduces need for manuals; Requires little or no training; Makes both semantics and syntax explicit]. Requires little memory [Recognition vs. recall]. Few keystrokes [Less opportunity for user input error]. Easy error handling [Only limited valid inputs at any point]. Enhancements are visible

Disadvantages of menu: Inefficient for experts and high frequency users. Inflexible [System controlled; Forced choice]. Take up screen ‘real estate’ [Only limited valid inputs at any point]. Real estate = pixels on screen

Menu design guidelines: Prefer broad and shallow menus to narrow and deep ones. Use items as titles for sub trees. Group items meaningfully. Use brief items, begin with the keyword. Use consistent grammar, layout, terminology. Allow type ahead, jump ahead, or other short cuts. Consider [online help; optimal response time, display rate; screen size]

**Fill-in forms.** Advantages of fill-in forms: Self-explanatory [reduces need for manuals; requires little or no training; makes both semantics and syntax explicit]. Requires little memory [recognition vs. recall]. Efficient use of screen “real-estate”. Accommodates parameters with many possible values. Provide context

Disadvantages of fill-in forms: Assumes knowledge of valid inputs. Assumes typing skills and knowledge of special keys. Creates opportunities for user error

Guidelines for fill-in forms: Meaningful title. Comprehensible instructions. Logical grouping and sequencing of fields. 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. Marking of optional fields. Explanatory messages for fields

**Direct manipulation**

Advantages of direct manipulation: Easy to learn and remember. Direct, intuitive, WYSIWYG. Flexible, easily reversible actions. Provides context and instant visual feedback. Exploits human use of visual and spatial cues. Low typing requirements and visual feedback. Less opportunity for user error

Disadvantages of direct manipulation: Inefficient for high frequency expert users. May be difficult to design recognizable icons for many objects and actions. Icons take may more screen real estate than words

DM design guidelines: Provide alternative interface for high frequency and expert users. Choose a consistent icon design scheme. Accompany icons with names. Provide visual feedback for position selection and movement, and physical feedback for modes

**Command language**: Interact with computer using text or voice commands. Rely on naming and syntax. Examples [Commands on DOS; Commands on UNIX]

Advantages [Flexibility; Supports user initiative; Appeals to “power users”; Potentially rapid for complex tasks; Supports macro capability]

Disadvantages [Requires training and memorization; Difficult to retain; Poor error handling]

Command language guidelines: Choose meaningful, specific, distinctive names. Support consistent abbreviation rules [prefer truncation to one letter]. Offer frequent users the capability to create macros. Limit number of commands and ways of accomplishing a task. Consider command menus on high-speed displays

**Function keys**: Dedicated function keys. Soft function keys (labels on screen) [Self-explanatory; Easy to use; Flexible; Requires little human memory; Little or no onscreen real estate needed; Limited typing requirement]. Concerns [Limited number of function keys exist; Application-specific; Inconsistence among applications]. Guidelines [Gray-out non-applicable functions; Combination of keys]

**Question and answer style:** Combines some features of menus and fill-in forms. User is posed with a single question, e.g., [Wizard dialog; Prompt for missing parameters]. Appropriate for lowly-motivated, less-experienced users. Requires little training

**Natural language interaction style**: Interact with computer using natural spoken or written language

When to use NLI?: NLI may work best for [Users who are knowledgeable about the task domain; Users with moderate computer skills; Limited access to other interaction styles]

UI Design Evaluation

Why, what, where, when to evaluate: Iterative design and evaluation is a continuous process that examines [Why: check that users can use the product and that they like it; What: a conceptual design, early prototypes and later, more complete prototypes; Where: in natural and team settings; When: throughout design process; finished products can be evaluated to collect information to inform new products]

Throughout the Usability Engineering Lifecycle: Conceptual design [evaluating initial concept design ideas with the users]. Early prototype design [evaluating with the users]. Complete prototype design [acceptance testing to verify the system meets expected user performance criteria]

Purpose of evaluation: Identify specific problems in UI [mismatches between design and final product; mismatches between design and real users’ behaviors]. Assess the effect of interaction [usability and user satisfaction]. Explore and understand system’s functionality [understand real users’ interactions]

Usability heuristics/guidelines (Heuristics evaluation done by experts): Heuristics help [Designers to choose design alternatives; Help evaluators find problems and assess interfaces (hence Heuristics Evaluation)]

UI design principles we’ve studied: Learnability/Memorability – L for short. Visibility – V. Simplicity – S. User control – UC. Error handling – ER. Efficiency – E. Graphic design – GD

Nielsen’s heuristics: Match the real world (L). Consistency and standards (L). Help and documentation (L). User control and freedom (UC). Visibility of system status (V). Flexible and efficiency (EF). Error prevention (ER). Recognition, not recall (ER). Error reporting, diagnosis, and recovery (ER). Aesthetic and minimalist design (GD)

Norman’s rules: Affordances (L). Natural mapping (L). Visibility (V). Feedback (S)

Shneiderman’s eight golden rules

1. Consistency (L)

2. Shortcuts (EF)

3. Feedback (V)

4. Dialog closure (V)

5. Simple error handling (ER)

6. Reversible actions (UC)

7. Put users in control (UC)

8. Reduce short-term memory load (ER)

Heuristic evaluation basic steps [Review/inspect UI; Compare it against certain heuristics; Document usability problems; Explain and justify problems against heuristics]

Guidelines for heuristic evaluation: Justify every problem with heuristics (one or many). List every problem. Go through the interface multiple times. Don’t have to limit to the 10 Nielsen’s heuristics [Also, not all heuristics are applicable for a user interface]. Use multiple evaluators [More people find more and different problems than one person; Problems can be duplicated; Nielsen recommends 3-5 evaluators]. Use heuristic evaluation in addition to user testing [Each method finds different problems; Heuristic evaluation is cheaper]

Evaluating prototypes: Heuristic evaluation can also be used to evaluate [Sketches; Paper/computer prototypes; Demo code]. Certain heuristics are not applicable [Feedback; “Missing-element” problems; Help and documentation]

Field studies: Real use environments [Observe effect occurs in realistic setting; Can’t tell how good UI is until the users use the products and that they satisfy; Hard to predict what real users will do: problems identified are those that are really raised when the user uses the software; Usability could be measured during the test by calculating the user performance]. Problems [cost/time consuming; may not generalize]

Procedure

1. Prepare test proposal

2. Choose participants

3. Select properties to be tested

4. Perform the tests

5. Measure the test results

Formative evaluation: Basic steps similar to field studies. But different [Users are assigned to certain tasks rather than natural; Evaluators (Choose participants, Assign each participant some tasks, Watch participants to perform tasks)]. “Think aloud”

Think aloud: The goal is to know what the users are thinking, not just what they are doing. First, ask the users to talk their thoughts while doing the task, example: [what they are trying to do; why they took an action; how they interpret what the system did]. Prompt the user to keep talking [– “tell me what you are thinking”]. Make a recording the thoughts [using note pad, audio or video tape]. Widely used evaluation method in industry. Advantages [simplicity - requires little expertise; can provide useful insight into the user’s thinking; can show how system is actually use]. Disadvantages [subjective; selective; act of describing may alter task performance]

Controlled experiment: A preferred traditional scientific method. Clear convincing result on specific issues. Benchmark tests. Allows system comparison, fine-tuning of details. Can deal with a practical problem and use the scientific method and theoretical framework n Theory-driven, hypothesis testing [Study relations by manipulating independent variables; Observe effect on one or more dependent variables]. Procedure [Prepare proposal; Prepare environments; Choose participants; Determine forms and properties to be used; Determine evaluation strategies]

A Few Web and Mobile UI Patterns

What is a UI Design Pattern?: Object-oriented Methodology [“Design patterns are descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context”]. Generally, a design pattern is a description of the repeatable problem and its solution. UI design patterns are descriptions of common UI design problems and their solutions

Benefits of UI Design Patterns: UI patterns support external consistency [Consistency is a key principle of UI design]. Improve learnability, memorability, and satisfaction of UI. Reduce time to design, develop, and test. Reduce risks in UI design [Learning from good and working UI’s]

Web UI Patterns

Navigation: Menus and Sliders: Horizontal primary menu and submenu. Menu can have dropdown lists (vertical menus). Sliders should include graphics and most important messages. Sliders can automatically play, but should not play forever [Include pause and replay buttons; Don’t play too fast, let users read the message]. Search input box on top should be prominent. Limit scroll on homepage. Always provide a Home link

Fat Footer: Secondary navigation to provide more shortcuts to important information of a website. Links to frequently used pages. Using the same footer for all pages Breadcrumbs. Tell users where they have been, where they are, how they can go back

Pagination: Using ‘Previous’, ‘Next’, ‘<‘ and ‘>’ icons, and page numbers (Allow users to change number of items per page: Too many: slow loading; Too few: slow searching)

Accordion Menu: Grouping things into sections each with a panel. Expanding a section when clicking on it. Allowing users to view content of a section while seeing other sections

Carousel: Used to browse through a set of items. Used when there are many items and limited space while informing users of something more. Used for visual objects

Thumbnail Gallery: Allowing users to view many images without downloading full sized content. Many jQuery and dojo plugins available

Forms: Forms contain various input elements: text boxes, combo boxes (dropdown), radio buttons, etc.. Guidelines [Use one-column left aligned forms; Mark required fields or active feedback on required fields; Provide good feedback for actions and good error messages; Place labels on the left or top and consistent throughout the website; Group related fields (using colors or shading)]. Input prompts placed within input box. Good defaults. Good feedback. Few required info

Inplace Editor: Allows users to edit value easily. Used for editing a few text, dropdown, etc. values. Should use signifiers or indicators for possible inplace editors (e.g., highlight when hovering)

Password Strength Meter: Used in the sign up form to help users specify their strong passwords. Helps improve password security on websites. How to know a password is strong? [Apply some simple policies, e.g., wiki page or MS page]

Input Validation: Provide validation and feedback input time. Validate and feedback as much as possible before submitting. What to validate? [Required fields; Acceptable ranges; Length; Uniqueness; Confirmation; Inclusion; Exclusion]. Provide a single search field [Provide advanced search as a secondary option]. Show results in the same page with search criteria. Offer an option to clear search criteria. Offer a progress indicator during search. Search criteria is also shown with results

Progress Indicator/Wizard: Used to guide users through a complex multiple step process such as checkout, setup, installation [Also used when users are not familiar with infrequent processes]. Guidelines [Minimize number of steps; If more than 5 steps, group them and don’t use numbers; Show the current step clearly; Label all steps/groups clearly; Allow users to go back to modify inputs in previous steps]

Mobile UI Patterns

Springboard: Home screens for iOS and Android. Divide screen into 3x3, 2x3, etc.. Should limit number of items in one screen. Provide a search box to search items. Allow users to change layout

Tab menu: Each tab groups a similar set of functions. Active tab should be highlighted. Tabs at the bottom is preferable (closer to user fingers). Use icons with accompanying titles. Titles should be short (use list menu or springboard for long titles instead) tabs

List Menu: Various kinds of list [Title only list; Grouped list (with sub-sections); Enhanced list]. Can show long titles and sub-titles (e.g., hints). Can be included in tabs. Should indicate touchable items

Gallery: Suitable for photos, articles, products, restaurant menus

Metaphor: Menu with icons symbolizing physical things. Strong metaphors may be overkilled. May be suitable in games

Slide Between Pages: Moving screens by sliding. Less visible navigation but efficient

Forms: Keep forms simple. Only include required fields

Registration Form: Provide instant feedback

Search Criteria Form: Show criteria in a list. Provide default values. Have a “clear”/ “reset” button. Set label with number of results, if possible

Single Search Input Box: Always better to provide simple but powerful search capability

Search Auto Suggestion

Grouped Rows

Signifiers: Use signifiers/indicators to show affordance of items. Drag-able. Checkable

Notifications

Invitations