Human Computer Interaction Revision Answers

* 1. Human-Centred Design
     1. Focuses on how humans really do things – questions if they’re used as intended.
     2. Human perception
     3. Subconscious instincts and acts
     4. Human memory
     5. Feedback and errors
  2. Interface Design
     1. Low-fidelity prototyping
     2. Menus, dialogues, choices
     3. Colours and layout
     4. Alerts/errors
  3. Interface Evaluation
     1. Heuristic evaluation
     2. Testing interface designs
     3. Applying design principles to existing interfaces

1. Human-centred design refers to finding out how humans really do things. It aims to understand the reality of users through observation. Users are put at the centre of the design process. User wants, pain points and preferences are made a priority during the design process, to make a more intuitive accessible product.
2. 1. Universal
   2. Almost unconscious
   3. Spontaneous
3. The HCD process is an iterative process. Repeat the steps as many times as necessary to achieve a solution. Iterations become shorter as you reach viable solution.
   1. Empathise – Foundation of HCD. Who are you designing for? What matters to them? Observe and learn about them. AKA clarify. Find pain points. 2 types of pain point – explicit and latent. Note challenges. Generate insights. Synthesise user needs – use the varying insights to generate more specific user needs/ insights. Observe, interview, engage and immerse. Interview the user – unstructured interview, questions based on the users answers. Don’t use predetermined questions. Open questions without particular expectation on the format or content of answers. Interview without a strict plan. Identify issues that haven’t been considered. Allow flexibility. Generate rich data leading to deep understanding of issues. Ethics of interview, explain risks and benefits, reward user, respect user privacy, give them the results after
   2. Define – What are their needs? Define the problem based on observations. Interpret your findings
   3. Ideate – Brainstorm different ideas/solutions. Generate design alternatives Apply different design thinking tools
   4. Prototype – Create one/more representations. Commonly the fidelity of the prototype increases with the number of iterations. Start out with a very lo-fi prototype.
   5. Test – What works/doesn’t? Test prototype with users and get user feedback.
4. Heuristic evaluation. Heuristics refers to an approach to problem solving that provides a useful solution that is not always perfect or optimal. The purpose of this is to speed up the process of finding a solution when it would take too much time/resources to find a perfect solution. In terms of design, many design problems are similar, meaning common principles apply to different design problems. This allowed for Nielsen’s usability heuristics to be derived.
5. Nielsen’s 10 usability heuristics
   1. H1 - Visibility of system status – Design should keep users informed about what is going on through appropriate feedback in reasonable time. Provide feedback on user input (input validation). Status indicators/icons. Don’t take action with consequences without informing the user
   2. H2 - Match between system and the real world – System should speak user’s language with words, phrases and concepts familiar to them, rather than system-oriented terms. Follow real world conventions – make info appear in a natural logical order. Don’t assume understanding of words/concepts by users. User research to find users’ familiar terminology
   3. H3 - User Control & freedom – Users do things by accident. Clearly marked exits to leave an unwanted state without needing to go through an extended dialogue. Clear way to exit current interaction
   4. H4 - Consistency and standards – Follow platform/industry standards (external consistency). User experiences with other products sets their expectation. Maintain consistency in a product or family of products (internal consistency). 2 types of consistency – internal and external.
   5. H5 - Error prevention – Prevent problem from occuring. Eliminate error-prone conditions, or check them and confirm with the user. Provide helpful constraints to avoid slips. Remove memory burdens. Warn users
   6. H6 - Recognition over recall – Minimize memory load by making elements, actions and options visible. User shouldn’t need to remember info from one part of the interface to another. Reduce the info that users have to remember. Let users recognize info in interface instead of needing to recall it.
   7. H7 - Flexibility and efficiency of use – Allow users to tailor frequent actions. Hide accelerators (shortcuts) from novice users to speed up interactions for expert users. Allows the design to cater to both experienced and inexperienced users.
   8. H8 – Aesthetic and minimalist design – Interfaces shouldn’t have info that is irrelevant/ rarely needed. Visual elements of the interface should support the user’s primary goals. Keep content and visual design focused on the essentials. Info on the UI compete with each other and diminish overall visibility.
   9. H9 – Help users recognize, diagnose and recover from errors – Error messages should be in plain language (lang they understand) (no error codes), precisely indicate the problem and constructively suggest a solution. Pair error messages with visual treatments that will help users notice and recognize them. Use traditional error-message visuals like bold red text (H4).
   10. H10 – Help and (pontoon) documentation – Provide documentation to help users understand how to complete tasks. Such info should be easy to search, focused on the user’s tasks, list concrete steps and not be too large.

Problem:

Heuristic:

Severity: - combo of 3 factors, frequency pf problem, impact of problem (is it easy to overcome), persistence of problem (can you work around it). 0-4 scale

1. I don’t agree it’s a problem
2. Cosmetic problem only; doesn’t need fixing unless there’s extra time
3. Minor usability problem
4. Major usability problem
5. Usability catastrophe, imperative to fix before product is released

Fix:

1. Interface design
2. You control the position, colour, size, shape and orientation of individual elements (units of info in the design e.g., images icons paragraphs text title) in the interface, to create contrast, repetition, alignment and proximity in the overall design. This is used to communicate a sense of hierarchy, grouping or sequence to the user.
3. 1. Proximity – Group related items together. Groups of items that aren’t related shouldn’t be close together. Uses the variable of position. It becomes clear how to interpret the interface.
   2. Alignment – Everything on the page should have a visual connection with something else on the page. Organise info to create order. Uses the variables position and orientation. Align elements to organise and unify a page. Left/right aligned text has an invisible line that connects the text. Grid design. Rule of thirds makes image more balanced
   3. Repetition - Used to unify and add visual interest. Repeat some aspect of the design throughout. Find existing repetitions and strengthen them, before creating new ones. It’s thought of as being consistent. Don’t repeat the element so much it becomes overwhelming.
   4. Contrast – If two items aren’t exactly the same, make them really different. Contrast various elements of the composition to draw in the reader’s/user’s eye. Uses many variables. Most dramatic principle. Creates interest and organises info.
4. They allow the user to quickly understand how the elements of the design relate to each other, and subsequently understand the design and its info. Hierarchy – order of importance, Grouping – specifies elements to be associated together, Sequence – specifies order of elements
5. 1. Centre of retina has no blue cones – small blue objects disappear if you fixate on them
   2. More contrast needed to see fine details/ read as you age
   3. Colour triggers all senses and delivers a message
   4. Blue – trust confidence, red – aggression attention grabbing, green – nature peace, yellow – positivity energy, purple – mystery sentimentality, pink – youth enthusiasm, orange energy liveliness, black – formal prestigious, white – clean pure safe
   5. Don’t use too many bright colours – no blue with red,
   6. Don’t use deep blue text
   7. Give value priority over hue – caters to older audiences
   8. Avoid visual inconsistency – use similar colours for similar interface elements. Use industry standards H4
   9. Don’t distinguish between 2 states soley based on colour. Mixtures of colours should differ in more than one dimension (helps colour deficient users)
6. Prototyping allows you to explore designs. It gives you something to give the user you’re designing for. It allows you to test desings without implementing/ coding them. Easier and cheaper to change a prototype than a final design. A prototype should simulate the interactivity of the design and test the usability of the design?
7. Prototype fidelity is the degree to which the prototype matches the look, feel and functionality of the final design. The level of detail and realism in the prototype.
8. Lo-fi – simple low tech representation focused on functionality – clickable wireframes or paper prototypes

Hi-fi – closely resembles the final product. Highly detailed. – communicates finished so comments focus on aesthetics and looks – things perceived as changeable. More time consuming. Developers more resistant to change as they’ve spent time on it and become attached. More expensive. C



|  |  |
| --- | --- |
| HI-FI | LO-FI |
| Slow to create | Quick to create |
| Hard to make many alternative designs due to time and complexity | You can make many of them and experiment with many designs |
| Reluctance to suggest / accept major changes | No reluctance to suggest/ accept major changes due to low complexity |
| You can fix problems before implementation | You can fix problems before implementation |
| Expensive | Cheap |
| Changes take longer to fix | Changes are quicker to fix |
| Testers don’t question the big picture structure – can get locked in a single approach |  |
| More detailed | Less detailed |

1. Define tasks and prepare scenarios. Make the prototype support these tasks. The scenarios must allow the user to make choices. Anticipate user behaviour. Select users (not friends and family). Practice. Debrief after the test. Ask questions about user problems. Evaluate the test after. Write a report with objective (quantifiable) and subjective (participant reactions) data. Make changes iterate and test again.
2. 1. Greeter – welcomes users
   2. Facilitator – gives user instructions encourage user to verbalise thoughts
   3. Computer – Handles the paper interface, switches out pieces to simulate the response to user actions. Don’t explain things
   4. Observers – observe and take notes. Don’t react to anything.
3. Menus are used to group functionalities together

|  |  |
| --- | --- |
| Pros | Cons |
| Shows all options to the user | Slow for experienced users |
| Can group options | Menu design has a big impact |
| Known syntax | Too many menus lead to clutter (confusion, info overload) |
| Recognition over recall | May not be appropriate for small displays |
| Can indicate state |  |
| Good for both intermediate and advances users (can support shortcuts, H7) |  |

1. Forms are used to receive user inputs

|  |  |
| --- | --- |
| Pros | Cons |
| Easy for beginners | Consumes screen space |
| Can simplify data entry | Good form design is not easy |
| Can constrain input to minimise error (H5) | Frustrating when long and/or if progress cannot be saved |
| Guides user through actions | Can result in rigid, inflexible business process |
| Instant user feedback can be provided (H9) |  |



1. Command lines are text based interfaces

|  |  |
| --- | --- |
| Command line Interfaces | |
| Pros | Cons |
| Fast and flexible for experts | Strict elaborate syntax |
| Modular and extendable | Hard to learn for beginners |
| Elaborate syntax | Execution of an operation is not always apparent |
| Low bandwidth (network) | High error rates |
| Aesthetics | Poor error reporting |
| Doesn’t require much memory | Requires memorisation (H6) |



1. Direct manipulation (central concept of GUI) is a visual model of the world. It has visual objects that can be manipulated.

|  |  |
| --- | --- |
| Pros | Cons |
| Some errors are avoided (no parsing) | Inconvenient and slow if the object isn’t visible, and user must find it |
| Fast to use objects in view | More difficult to implement |
| Works well on touch screens | Often slower than command line for experts |
| Good for beginners | Not all actions have a direct manipulation equivalent |
| User is in control |  |
| H2 |  |
| High discoverability |  |

1. GUIs allow users to interact (point & click) with the system via non-textual representations (signs), using keyboard, mouse or other pointing devices.
2. 1. Icon – sign resembling what it represents (same signified and signifier)
   2. Index – sign that give some form of indication of what in represents (image of smoke to represent fire)
   3. Symbol – sign that does not directly codify the concept it represents
   4. Alphabetical order
3. An affordance is what a user can do with an object based on the user’s capabilities (e.g., affordance varies with the same object in a system with user access levels as some users can do things with the object that others can’t). Relationship between the properties of an object and the capabilities of its users. Affordance is when the sensory features of an interface hint at its function. For example, if an object has a thick spine you tend to hold it there
4. Signifiers are perceivable indicators that communicate the behaviour of an object. It’s a perceivable cue about affordance.
5. Affordances are implicit, signifiers are explicit. Signifiers directly tell you what to do. Signifiers indicate affordance.
6. 1. Input control – user inputs data into system.
   2. Navigational control – user needs to locate specific content or features.
   3. Output/informational controls – used to display info or messages.
   4. Data controls – allows user to browse, view, search and organise data.
7. Micro interactions are the small details that exist inside and around features in a design. It’s a tiny piece of functionality that does one thing. They give feedback that can help enforce rules
8. User-initiated and system-initiated
9. Triggers (user or system initiated), rules (what happens when its triggered), feedback (let user know what happened, illuminate the rules), loops (cycle that repeats, define how long a micro-interaction will continue).
10. Menus are used to present choices to the user
11. Binary menus only have 2 options
12. 4-8 items per level, up to 3 levels in depth. Breadth preferred over depth. Broader shallower trees produce superior performance. Response time + error rate ↑ with menu depth. More items per level than levels. Use positive language in phrases
13. Use verbs – they convey info better than generic language
14. Status reason action
15. Progressive disclosure defers rarely used features to a secondary screen, making apps easier to learn and less error prone
16. Don’t give choices the user may not understand
17. Depending on the currently active state, the interpretatiom of the user input by the system is different (e.g. CAPS LOCK). Interfaces can be modeless.
18. Violate H1
19. Mode slips occur when the user is not aware of the currently active mode. Aka mode errors. Modes have a low discoverability. Tough for new users.
20. Make current mode clearly visible, clearly indicate the effect of the current mode. Confirmation dialogs
21. Use spring loaded modes – these modes are only active as long as the user explicitly does something to keep it active (e.g. holding down shift key for uppercase). Mode errors are less likely.
22. Affordance is a situation where an object’s sensory features imply its functionality and use
    1. Evaluative – data collected during interface evaluation
    2. Empirical – human experience data – interviews, questionnaires
    3. Ecg, body temperature
23. Hierarchical task analysis is the process of figuring out who’s going to use the system to do what and coming up with a necessary sequence of tasks.

|  |  |
| --- | --- |
| Pros | cons |
| Easy and economical method of gathering info | Doesn’t scale well |
| Gives good understanding of interactions | Not good with overlapping tasks |
| Supports design reuse |  |

1. A persona is a fictitious user that represents the needs of a larger group of users

|  |
| --- |
| pros |
| User goals and needs become a common point of focus for the team |
| Quick to develop |
| Help to avoid trap of building what users ask for instead of what they will actually use |
| Designs can be evaluated against the personas, reducing the frequency of expensive usability tests |

1. Indicator (contextual, doesn’t need user action), notification (global or contextual, doesn’t need user action but can use it), error/validation (contextual, user action required)
2. Communicates system status. Reduces uncertainty. Increases confidence. Prevents the user from repeating the interaction multiple times because they weren’t sure it worked. Allows identification and rectification of errors.
3. Slip (unintended error, can be due to memory) and mistake (error of conscious decision, wrong choice made)
4. System that’s aware of the context its deployed in, and uses this context to provide extra information to the user. Transparent interface. Hide presence from users
5. Time lines personas sticky notes. Matrix analysis – “bucket analysis”, one row per user. Columns give insight for specific aspects.
6. Cognitive dimensions is an approach to analysing the usability of interfaces. It often uses a more light-weight approach, allowing you to find things more conventional methods wouldn’t.
7. Error proneness. Viscosity (resistance to change)
8. Empirical data is collected from users (human experience data)
9. 1. Qualitative – observational findings. Don’t use numerival metrics. Based on characteristics (how easy/hard to use) of features.
   2. Quantitative – Uses numerical metrics
10. Fitts law is a model used to account for the time it takes to point at something, based on the size and distance of the target object.
11. MT = a + b log2(A/W+1) or MT = log2(A/W+1)

MT = movement time

a and b – parameters that vary with situation (e.g. variations in hardware that we can’t control

A (or D) – distance from starting point to centre of target

W (or S) – width of target in direction of motion

1. Bigger things are easier to hit than smaller things, and closer things are easier to hit than far away things. The further and smaller an object, the longer it takes to hit. This is why chips over radio and checkbox
2. Greater drops in MT occur for smaller objects than larger ones
3. GOMS is a family of predictive models used to predict human performance of an interface design without user testing.
4. Goals- tasks the user wants to do Operators-steps/actions performed on interface broken down into gestures (mouse click, keystroke) Methods- series of operators chained to accomplish a subgoal Solutions- decision for the user, depending on if the goal can be achieved by more than one method. Selection rules used to determine method used
5. KLM-GOMS (keystroke level model)
6. If there’s a K (that’s a delimiter) after something that is the same every time, delete the M before it
7. 1. The user is an expert in the interface
   2. User habits, distractions and restrictions aren’t accounted for
   3. The user is a good typist and won’t make mistakes
8. You should try to apply rules 2 and 4 simultaneously GOMS slides page 36

|  |  |
| --- | --- |
| Pros | Cons |
| Allows designers to make quantitative predictions without user testing / training | Can only capture routine cognitive skills (not problem solving) |
| Provides good fits to performance times of skilled users during error-less performance | Can’t be used for multi-tasking |
| KLM parameters have been proven stable | Assumes error free performance (experts spend ¼ of time making/fixing errors) |
|  | Doesn’t account for novice behavour |

1. Short term memory is also known as working memory. Rapid access & decay. Pass to LTM after seconds
2. A chunk is a semantically meaningful unit. Could be one letter or position on chess board. STM has a capacity of 7+-2 chunks, consider when designing interfaces and # steps to do something. Don’t make user remember over 7 things. Related things chunk best. Don’t make user remember something system can provide
3. Long term memory – slower access time, little decay
4. A cue is any stimulus that creates an association (e.g. a signifier or giving hints). You can’t control how association is formed, makes unlearning hard
5. Memory takes the form of schema which provide a mental framework for understanding and remembering info. New info is attached to existing schemas for topics. This shows the importance of H1: Match between the system and the real world. Schema mean users have expectations for things, and its important to meet these expectations. New info registered in context of the schema without altering the schema. Schema often shared between individuals and stays constant over time
6. If you dont use those, you end up designing for a specific group (gender, age, language, physical abilities). Theoretical user can reflect personal biases.
7. Accessibility addresses discriminatory aspects related to equivalent user experiences for disabled people
8. Usability is concerned with whether designs are effective, efficient and satisfying to use. Measured against 5 criteria: memorability (easy to re-establish proficiency after idle period, efficiency, errors, learnability (is it easy to do stuff first time) and satisfaction
9. Video captions for people who are deaf, help people who can hear when they’re in noisy environments. Can help mobile users (reading in the sun)
10. Text-to-speech Caption dictation (voice to text) signifiers Fitts law (large controls) eye tracking alternative to keyboard mouse