Software Usability

Course notes for CSI 5122 - University of Ottawa

2023 Deck D:

Evaluation of Usability done by Experts:

Task Analysis, Heuristic Evaluation, and Cognitive Walkthroughs

Timothy C. Lethbridge

< Timothy.Lethbridge@uottawa.ca >

http://www.eecs.uottawa.ca/~tcl/csi5122

TASK ANALYSIS

Task analysis: A prerequisite for evaluation

1. Understand top level goals and corresponding tasks users will have when using the system

- Like use cases
- Ask whether the user will have the information they need to form the goal
  - —Users often don't know what they should do
  - —They may have the wrong goals

2. Recursively divide each task into subtasks

- What is the specific sequence of things the user must do to accomplish the goal

Task analysis steps

3. For each task, and subtask ask:

a) Preconditions: What does the user *need to know* prior to performing the task?

—Users often don’t have the required information

—If not, how can the system help them obtain it?

b) Postconditions: How the user know the *previous task is complete*?

Users get stuck an an impasse waiting for something to happen

Task analysis steps

c) How will the user know the steps to take?

- So the user
  - Forms appropriate subgoals
  - And correctly *specifies* the action
- Users often can't figure out how to do what they want
- They may tend to do the wrong action

d) Can the user interpret the result? Common problems:

- Can't interpret the result
- Get the interpretation wrong

Task analysis in design

Before even building a system, work out the complete hierarchy of tasks

Design the system so

- The questions in step 3 will have easy answers.
- Users don’t get hung up by the problems marked in red

Task analysis to plan tasks for *observation sessions*

Performed to develop simple instructions for participants

- Users will be prompted with appropriate goals

Users should be given simple and concrete tasks at first

- Simple and low level
- No knowledge assumed
- Precise directions anybody could follow

Gradually, the goals become more high level

- Users have to think of their own low level tasks

Choosing tasks to analyze

There are often too many tasks in the system to analyze them all:

- Pick tasks *most frequently* performed by users

- Pick tasks that *represent a wide spectrum of system functionality*

Task Analysis Case Studies

Cooking food in an oven, with options to preheat oven

Schedule builder from courses a student might want to take

- https://uschedule.me

Creating an online quiz

- We will analyse this abstractly then consider how it works in Brightspace

HEURISTIC EVALUATION:

SYSTEMATIC EVALUATION OF A USER INTERFACE BY 'EXPERTS'

Heuristic Evaluation

Based on UI guidelines

- Heuristics about what is *best practice*
- We discussed Nielsen's guidelines earlier

Multiple passes may be needed

- Passes while doing different tasks
- Perhaps one pass to look for each kind of problem
- Passes to follow exceptional cases

Use multiple ‘expert’ evaluators

1-2 hour sessions

- Do not exceed 2h without a long break
- It is a tiring process: Efficiency at finding problems drops

Studies show

- 1 expert evaluator finds only 33% of problems
- 5 evaluators needed to find about 75% of problems
- 15 more to find about 99% (but never all)

Steps in heuristic evaluation

1. Decide on the tasks you want to evaluate

- Understand the screens / windows / dialogs and other UI components involved in this task

2. For each of a selected set of UI guidelines review the UI, looking for problems

- When a problem is found describe it briefly so developers can understand
- Suggest potential solutions

3. Write a report about the problems found

COGNITIVE WALKTHROUGHS

Cognitive Walkthroughs: Task analysis as evaluation

Step hierarchically through the system seeing if the questions posed in each step are answered

Walk through the user' s repeated *goal/decide/execute cycle*

Cognitive Walkthrough Cheat Sheet

a) Choose a task to evaluate

b) Describe the initial state prior to the task

c) List the atomic actions needed to correctly perform the task

d) Describe classes of users (actors / personas who may perform task

e) Describe the 'Goal Structure' (or task structure) users would likely have in their minds before starting the task

f) For each action (each step)

- What goal structure would the actor need to have?
- Will the system lead them to have that goal structure (to do action)?
  - —Look for: Failure to add / failure to drop / spurious goals / stuck (cannot progress) / premature loss (falsely think goal achieved)
- Will actions match goals?
  - —Wrong actions match goals / actions match wrong goals
- Can actions actually be done even if user knows what to do?

Cognitive Walkthrough Case Studies

Any ideas from the class?

Alfred – setting up a ‘hotkey’

Teams – adjusting notifications

Outlook – creating a meeting with some people at a time they are available

Zoom – scheduling a meeting that will be every Monday and Friday except holidays

Case study of a cognitive walkthrough: Inventory system - 1

a) Choose a task to evaluate

- Record a newly-received item in inventory.

b) Describe the initial state of the system

- Main menu is displayed

c) List the atomic actions needed to correctly perform the task

1. Click on 'add to inventory' in the menu.
2. If you don't know the part number, hit 'return' to perform look up the part number, then go to action 4.
3. Type the part number into the 'part number' field
4. Press tab
5. Type the number of items in the 'Number' field
6. Hit <return> or click on 'Add'.
7. If the system prints out a bar-code sticker, affix it to the new item.

Cognitive walkthrough case study: Inventory system - 2

d) Describe classes of users who may perform the task.

- Receiving clerk - knows about inventory, but not yet about the system

e) Describe the 'Goal Structure' (or task structure) users would likely have in their minds <u>before starting the task</u>

- If there are actions for which the user has no goals, the system must stimulate the user to think of these goals by the time they must perform the task.
- If different classes of user may have different goal structures, list these too.
- Record a received item in inventory
- Start the inventory program
- Enter the item

Cognitive walkthrough case study: Inventory system - 3

f) For each action above, do the following (I to IV):

I. Write down the goal structure ... that the user would <u>need to have</u> in order to perform the action correctly.

- Record a received item in inventory
  - —Record the part number
  - —Press tab
  - —Enter the number oof items
  - —Cause the system to process the transaction

Cognitive walkthrough case study: Inventory system - 4

II. Verify that the user will have the correct goal structure

- Given their initial goals
- Given the system's response to the previous action

- Estimate the percentage of users who might have each of the following possible problems:

- A. Failure to add goals
  - —The system must make it clearly visible that pressing return with nothing entered will invoke a lookup mechanism
- B. Failure to drop goals
  - —The user may have a goal to notify the person who ordered the parts
  - —This would not be needed if the system performs this automatically

Cognitive walkthrough case study: Inventory system - 5

- C. Addition of spurious goals
  - There may be a field marked 'Description'.
  - However this only needs to be filled in if the type of item is not in the database.
- D. No-progress impasse
  - After adding an item, the system might just clear the screen ready for another entry.
  - The user may think the transaction failed (i.e. goal not achieved)
- E. Premature loss of goals
  - The user enters an item and hits 'return.
  - A message 'transaction accepted' is printed (meaning the transaction has been started)
  - The user powers off the computer thinking the goal is reached.
  - The system never got around to printing the label

Cognitive walkthrough case study: Inventory system - 6

III. Verify that the actions match the goals

- Possible problems:
- A. Correct action doesn't match goal
  - —User wants to delete an item that was stolen.
  - —Correct action is to select 'add to inventory' and specify a negative number
  - —System does not help user match the goal to the action
- B. Incorrect actions match goals
  - —User wants to add a new type of item to inventory (for which no items have yet been received)
  - —Upon seeing 'add to inventory', user selects this incorrect menu item

Cognitive walkthrough case study: Inventory system – 7

IV. Verify that the user can physically perform the action.

- Possible problems:
- A. Physical difficulties
  - —e.g. recognizing an icon, holding down shift-ctrl-alt-a to perform a command
- B. Time-outs
  - —running out of time - the system gives up

REPORTING PROBLEMS FOUND IN USABILITY STUDIES

Reporting usability problems

For each problem, do the following:

- Describe:
  - —The problem and how to reproduce it
  - —How it should be fixed
- Indicate what guidelines it breaks
- Categorize/tag it according to its severity
  - —How frequently users will encounter it
    - (e.g. constantly, every day, weekly etc.)
  - —How much impact it will have on them when they do encounter it
- Categorize/tag it in other ways (see next slides)

Ways of Categorizing/Tagging Errors - 1

Enables you to detect *patterns* in errors:

a) By subsystem

b) Whether:

- Accidental error the user made or might make
  - —There still should be attempts to prevent this
- Error caused by a design problem

c) Level of the error

- **Task** design problem
- **Conceptual** model problem
- Interaction **style** problem (found in many places)
- Interaction **element** problem (found in one place)

Ways of Categorizing/Tagging Errors - 2.

d) At what *stage in user's goal/decide/execute cycle* does the error occur?

- (this is the concept underlying cognitive walkthroughs)
- When the user decides on a goal
  - —They don't know what they should do
  - —Their goal is wrong
- When specifying an action
  - —They can't figure out how to do what they want
  - —They do the wrong action
- When executing the action
  - —System doesn't behave as expected
- When interpreting the result
  - —Can't interpret it
  - —Get the interpretation wrong

Ways of Categorizing/Tagging Errors - 3

e) What caused the error (can be more than one):

- Traditional bug (functionality defect)
- Poor feedback, labeling or error message
- Failure to highlight important distinctions
  - —-e.g. colours cannot be distinguished
- Distraction / not holding attention
- Feature interaction
- Inability to remember (recall or recognition)
- Lack of knowledge to do task
- Misleading training or help
- Overly complex and confusing UI
- Overly complex and confusing task
- Difficulty with physical coordination
  - —e.g. complex key-combinations / touch items too close

MORE DETAILS ON HEURISTICS (OPTIONAL MATERIAL)

More on Heuristic 1: Use simple and natural dialogue

Every additional element on the screen adds complexity

Present exactly what the user needs and no more at exactly the time the user needs it

- ’ Less is more'
- Users take longer to process what they see the more there is
- If information is only sometimes needed, use a 'more info' box.

Don't force the user to choose from huge numbers of options

Make the hard choices for the user

Do cost/benefit analyses before you add a new feature

Related information should be 'together'

Users perceive it *as a whole* according to Gestalt psychology

Togetherness can be

- Physical closeness
- In a box
- Using a common coding scheme (e.g. colour)

More important items should stand out more

- Near top
- Brighter colours
- **bolder text**
- Bigger text
- Probably not UPPER CASE (10% slower to read)

Avoid *modal dialogs (popups)*

A modal dialog is one that requires ‘OK’ or ‘Cancel’ or something similar, and blocks all other actions

- Have users confirm in the main display
- Make such boxes non-modal

These are generally distracting and confusing

- They are ‘cheap’ to program though

NNGroup link:

https://www.nngroup.com/articles/popups/

Exception to the rule about modal dialogs

- Confirm unrecoverable actions
  - —Deletion of data
  - —Major financial commitment
  - —Other ‘potentially ‘unsafe’ occasional actions

Other guidelines related to ‘natural dialog’

Match the user's task

- The user shouldn't have to think to map *domain concepts* to *concepts being presented* by the UI

Graphic design should be pleasing and helpful

- Consider 'mumble screens' with text shown only as 'mmmmm' to test for effectiveness of graphics alone

Have the right amount of detail in feedback

Ensure the user knows without extra effort

- What they did
  - —Users often are not sure
  - —The user may have made a mistake
- What happened
- What can be done to rectify the situation or move on to the next step

Case study:

- When ATM's go down, provide feedback about whether the failure is local or system-wide

More on Heuristic 2: Speak the user's language

Understand the literacy level of users

- Use the terminology that they are most comfortable with

Don't force users to have to memorize or even see 'codes'

- I.e. numeric alphabetic codes for objects, transactions etc.

Don't limit the 'width of fields'

- Names can be arbitrarily long

Naming things is extremely difficult:

- People will always disagree
- Brainstorm for alternatives, then have a vote of the users

Build *metaphors* to map system concepts into well-understood concepts

These draw upon non-computer experience of users

e.g. the desktop, the recycle bin

More on Heuristic 3: Minimize memory load

Allow users to choose from lists

- rather than enter data from memory

Provide *examples* of correct input format when feasible

Have as few coding schemes, metaphors etc. as possible

- Fewer rules to remember
- Have the same commands operate on different kinds of objects
  - —(e.g. 'print', 'copy', 'paste' etc.)

More on Heuristic 7: Provide shortcuts

Type/work ahead (e.g. in a voice-mail system)

- As long as there is no risk of users unknowingly missing important information

Ability to rapidly jump into information spaces

- e.g. bookmarks

Re-do / re-use of recent commands and selections

- e.g. lists of previously selected items or files

More on Heuristic 7: Efficiency of Use Response time

< 0.1s : User feels response is immediate

< 1s: User feels delay, but feedback about delay not necessary

< 10s: User will wait without doing something else; simple feedback useful

> 10s: User will often switch task; strong feedback becomes important (e.g. audio)

More on Heuristic 8: Provide good error messages – Case study

Instead of 'Cannot write file', give:

- Name of file being written
- Where it is being written
- Exact set of reasons (permission problems, network problems, space problems etc.)
- - What the user can do about it

  —e.g. options to:

  - Choose another name or place
  - Free up disk space if possible
  - Override permissions if reasonable

Other guidelines for error messages

Try automatic correction of errors:

- e.g. suggest correct spelling

Do not blame the user for errors

Avoid words like

- Illegal aborted failure error

If the error situation is complex, split the information into simple-to-grasp components

- Focus on the most important / root problem
  - —But let users know there are other problems

More on Heuristic 9: Prevent errors

Again, allow the user to *choose from a list*, rather than type

Provide good advance indicators about potential problems

- e.g. if the system knows that a directory is read-only or full
  - —inform the user before he or she tries to write to it

Given users step-by-step sequences to follow, when actions are complex:

- e.g. 'Wizards'

Minimize 'modes'

In each 'mode' the set of commands is different from in other modes

e.g. in a word processor:

- Outline
- Normal
- Online layout

print preview

page layout

Where there are modes

- Show the state clearly
- Make as many functions as possible available and work the same way

More on Heuristic 10: Provide adequate help and documentation

The existence of help should never be an excuse for bad design

Help should permit users to raise their level of expertise

Remember: Users do not (often) read documentation!

- Corollary: When users do read the manual, or access help, they are often in a panic.

More on help

Online help is a system feature, therefore it can add complexity as well as helping

Additional qualities

- Writing quality (hire a writer and proofreader)
- Consistency
- Coverage