evaluating prototypes
(discount methods):
cognitive walkthrough,
heuristic evaluation

by now (W09 pre-reading - RSP)
• cognitive walkthrough – ch 15

upcoming:
• project: part I, where you are doing a cognitive walkthrough on your paper prototype + task examples

today:
• review, then go through a cognitive walkthrough test a conceptual model that has been laid out
learning goals

- explain why cognitive walkthroughs are considered discount usability methods (covered in last few slides)
- describe the pros/cons of cognitive walkthroughs, and explain when it is an appropriate choice of evaluation method
- describe the kinds of things that a cognitive walkthrough can assess
- outline the general procedure for a conducting a cognitive walkthrough (and be able to do it yourself)

recap: where are we? how do cognitive walkthroughs fit in?

recent topics:

- task examples: encapsulate a task (design independent)
- mental model: what the USER thinks the system does / how it works
- conceptual model: how DESIGNER wants to portray system to user
- HCI methods: includes tools to evaluate and prototype

TODAY: bring it all together!

Cognitive walkthrough simulates mental model development

assessing... is the conceptual model an effective one? does the interface design communicate the conceptual model? how well does it support forming a good mental model?

i.e. your prototype
the walkthrough evaluation

good for: developing / debugging an interface, without accessing users (which is expensive)
tests: how well 1) interface design AND 2) underlying conceptual model aligns with/sets up the user’s mental model

process:
• take a task example + design
• for each user’s step/action in the task and using your current interface prototype:
  • ask a preset of questions (about: mental models)
  • ask: is there a believable story that motivates the user’s actions?

cognitive walkthrough

what for: assessing how well a new user will be able to figure out the interface
not for: assessing performance at highly skilled, frequently performed tasks; or finding radically new approaches
additional advantages: helps work out task sequence models through observation
disadvantages: limited utility for frequent-use interfaces, narrow focus, relatively time consuming & laborious (compared to other discount methods)

possible outputs:
• loci & sources of confusion, errors, dead ends
• estimates of success rates, error recovery;
  performance speed less evident
• helps to figure out what activity sequences could or should be possible

what’s required:
• task examples: design-independent descriptions of tasks that representative users will want to perform.
• a prototype to provide a design

who does it: [theoretically] anyone – usually design team members or expert outside analysts.
• can use real users . . . but this makes it a lot less ‘discount’

start: with a scenario
task examples + design \(\rightarrow\) scenario

process:
step 1) break task down into user actions (expected system response)
step 2) perform each step ON the existing interface and ask:
Q1: will the user know what to do?
Q2: will the user see how to do the action?
Q3: will the user correctly understand the system response?
step 3) if you locate a problem, mark it & pretend it has been repaired; then go on to next step.
cognitive walkthrough: basic steps

Step I. Generate “correct”, intended steps to complete a task.

Select a task to be performed and write down all the “user actions”, and expected “system responses”.

You have a choice of approach here: are you looking for

(a) can they find correct sequence(s) in current version?

- use high-level directives:
  - correct user action = “enter amount of food for pet feeder to dispense”

OR (b): are there mental-model problems even if they use exactly the right sequence?

- get very specific:
  - correct user action = “type ‘36g’ into the text entry box in the middle of the screen”

activity: generating steps for cognitive walkthrough

for a design challenge to enhance UBC experience, a design team decided to:

- use location-based technology
  - to create + access information around the UBC campus

StickIt!

*tasks have been modified from originals

activity: an early task example

(somewhat design independent)

“Posting an event” task example -

Ron Weasley is a founding member of the UBC Quidditch team and wants to advertise the club’s weekly games to to potential new members.

In his event posting, he includes the name of the event as well as the date, time, location and a brief introduction about the team.

In choosing a format for posting this information, Ron wants to make sure as many people in the community see the posting as possible, especially since the first game is only one week away.

The posting will not be needed (can disappear) after the first game is over.

activity: the prototype (set of digital storyboards)
activity:
the prototype design

conceptual model (simplified):

- share information about campus events with digital ‘sticky notes’ that can be left at GPS locations

the digital notes:
- have owners (who create them)
- can be ‘picked up’ by users in that location
- have a location, description, title, media, category, etc.

activity: task scenario

Create a note for a location-based activity:

Ron wants to organize a weekly pickup Quidditch game in McKinnes field to attract new people to his Quidditch team.

He creates a new weekly note. He gives the note a date and time, a title, a location description and GPS location, and a textual description.

He previews the note, decides everything is in order, then posts it.

When he wants to access his note again (perhaps to see how many “likes” he’s got) he can go to his profile, and see a list of his notes (or a map showing all the notes that he’s created).

CM: What OBJECTS + OPERATIONS are in this scenario?

work out STEPS for task example

(work ‘correct’ actions for given interface)

1. Decide to use StickIt
2. Create a new note ➔ select ‘pencil/paper’ icon, which takes you to Note Creation Wizard.
3. Specify type of note. User should choose Event and then ‘Next’ to submit; game will recur; Event has additional options that informational notes don’t.
4. Add events specs: date and time, and set note to recur every week.
5. Add details: note title, location description, text description, media content
6. Set GPS location: position note on a map
7. Preview note. Everything in order? If not, iterate....
9. Access note (now, to test; later, to see its “likes”):
   ➔ go to personal profile
   ➔ see a list of personal notes (or a map showing your notes).
cognitive walkthrough:

**basic steps**

Step 2. Carry out steps, simulating the mindset of your intended user, and note your success OR failure on a log sheet.

for each step:

Q1: ask yourself if user knows what to do?
   - are they trying to produce this effect? do they have enough info? etc.

Q2: explore – will the user see how to do the step?
   - look for the needed action? is it visible? is it obvious how to perform the step?

Q3: interpret – will the user correctly understand the system response?
   - Is the feedback understandable? Will the interpretation be correct?

Note: even with an error, user may have progressed if error became apparent. Distinguish this from when user is left with a misunderstanding.

example cognitive walkthrough

https://youtu.be/Edqiao4mmxM

two approaches to instructing person(s) doing CW

Approach (a): participant follows the pre-prepared steps and assess according to expected actions/system response

- at each step, assess using the questions (as in textbook)
  ➔ usually best you can do with a paper/low-fidelity prototype (unless it is very complete, has many paths)
  ➔ approach you will probably want to use in your project

Approach (b): give the CW participant ONLY the higher level directive(s).

- E.g., “create an event note with the following attributes. . . ”
  - more exploratory; still use Q1-3 to assess for each step they take
  - BUT - the steps he/she takes might diverge from the list you made – note them down on another action-list sheet. These points should trigger further analysis
  ➔ usually most effective higher fidelity prototypes or released systems

what kinds of problems should I record?

- in a CW you may note may kinds of problems, e.g.
  - problems with particular steps
  - problems moving between steps
  - larger problems that involve lots of steps
  - larger problems that hint at deeper problems with conceptual model/design
  - small problems that might only apply to unusual users
  - other kinds of problems that just become apparent while using interface, etc.

- make note of these as as appropriate!
  - if you do a lot of CWs, you may develop your own template for noting problems that works for you!
  - template from class only one example of a template
cognitive walkthrough:  
how do I become good at doing CWs?

- when you're new to CWs, it's easy to assume to the user will know what to do if YOU know what to do
  - force yourself to imagine what the user might not know
- when asking the questions at each step:
  - really think about what the user could be thinking . . .
  - consider the impact of misconceptions or mistakes that they could have made earlier!
- perform lots of them!
  - you'll get better at figuring out what to focus on with practice

activity part 2:  
perform the cognitive walkthrough

work in pairs

1) for each of the steps:
   - ask yourselves each of Q1-Q3;
   - if answer is NO for any questions:
     - write down the problem (possible solutions if you have ideas)
     - THEN assume it's fixed; go on to next step

activity part 2:  
(steps with 'correct' actions/response for interface)

1. Decide to use Stickit
2. Create a new note ➔ select ‘pencil/paper’ icon, which takes you to Note Creation Wizard.
3. Specify type of note. User should choose Event and then ‘Next’ to submit; game will recur. Event has additional options that informational notes don’t.
4. Add events specs: date and time, and set note to recur every week; Next.
5. Add details: note title, location description, text description; select Next
6. Set GPS location: position note on a map; select Next;
7. Preview note. Everything in order? If not, can I go back and fix?
8. Post: press “Post”.
9. Confirm note has been posted:
   ➔ go to personal profile
   ➔ see a list of personal notes (or a map showing your notes).

cognitive walkthrough:  
what do I do after the CW?

CWs can be done in teams or individually

- aggregate and discuss problems
  - possibly found over more than one CW
- prioritize problems based on severity, likelihood

THEN:

- iterate and fix as required
  - decide on which you can/will address
  - iterate on conceptual model and/or interface design
- OR write up a report/recommendations ➔ design team
  - if you're not the one(s) doing the designing
by now, you should be able to:

- explain why cognitive walkthroughs are considered **discount usability methods** *(covered in last few slides)*
- **describe the pros/cons** of cognitive walkthroughs, and explain when it is an **appropriate choice** of evaluation method
- describe the kinds of things that a cognitive walkthrough can assess
- outline the **general procedure** for a conducting a cognitive walkthrough (and be able to do it yourself)

**heuristic evaluation**

*where else we’re covering it*

by now (W10 pre-reading – RSP)
- heuristic evaluation

upcoming:
- project part II – usability evaluation of your medium fidelity prototype

**learning goals**

- explain why heuristic evaluation are considered a **discount usability method** *(covered in last few slides)*
- describe the steps involved in conducting a heuristic evaluations and know how to apply heuristics
  - explain why multiple evaluators are called for
- **describe the pros/cons** of heuristic evaluations
  - give examples of when doing a heuristic evaluation is an appropriate choice
**how to perform a heuristic evaluation**

1. design team supplies scenarios, prototype, list of heuristics;
   need 3-5 evaluators: train in method if non-expert
2. each evaluator independently produces list of justified, rated problems by stepping through interface and applying heuristics at each point … use heuristics list & severity rating convention
3. team meets and compiles report that organizes and categorizes problems

---

**one popular list of heuristics (Nielsen, ‘93)**

H1: visibility of system status
H2: match between system & the real world
H3: user control & freedom
H4: consistency and standards
H5: error prevention
H6: recognition rather than recall
H7: flexibility and efficiency of use
H8: aesthetic and minimalist design
H9: help users recognize, diagnose & recover f/ errors
H10: help and documentation

---

**step 1: briefing session**

get your experts together
- brief them on what to do, goals of system, etc.
- discuss heuristics to be applied

may also want to provide experts with:
- some examples of tasks
- descriptions of user personas
- simple instructions/guidance
  - especially if NOT a fully functioning system

---

**step 2: individual evaluation**

at least two passes for each evaluator
- first to get feel for flow and scope of system
- second to focus on specific elements

each evaluator produces list of problems
- explain problem w/reference to heuristic or other info
- be specific and list each problem separately

**tips:**
- be respectful but critical
- let your client decide whether to ignore a problem
- look especially for what’s not there
step 3: aggregating results & making recommendations

- evaluation team meets and compares results
- through discussion and consensus, each violation is documented and categorized in terms of severity, extent
- violations are ordered in terms of severity
  - e.g., use an excel spreadsheet (which can be sorted)
→ combined report goes back to design team.

severity ratings

each violation is assigned a severity rating

usually some combination of:
- frequency with which the problem occurs: Is it common or rare?
- impact of the problem if it occurs: Will it be easy or difficult for the users to overcome?

used to:
- allocate resources to fix problems
- estimate need for more usability efforts
- help prioritize problems

can be done independently by all evaluators or later as group prioritizes

example severity scale

(others possible)

- 1 = minor: need not be fixed unless extra time is available on project
- 2 = moderate: fixing this should be given low priority
- 3 = major: important to fix, so should be given high priority
- 4 = critical: imperative to fix this before product can be released

example finding

<table>
<thead>
<tr>
<th>Issue #</th>
<th>Severity Rating</th>
<th>Violated Heuristic(s)</th>
<th>Description of Issue</th>
<th>Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Major</td>
<td>Help users recognize, diagnose, and recover from errors</td>
<td>Error message is unclear – does not explain why the error happened and does not suggest a solution.</td>
<td>Reword the error message to better explain what happened (e.g. “You must go through slides in order – jumping ahead is not allowed”) or prevent users from jumping ahead.</td>
</tr>
<tr>
<td>Issue #</td>
<td>Severity Rating</td>
<td>Violated Heuristic(s)</td>
<td>Description of Issue</td>
<td>Recommendation</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------</td>
<td>-----------------------</td>
<td>----------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>1</td>
<td>Error prevention</td>
<td>Urinal layout will not support multiple users or will result in some... uncomfortable situations.</td>
<td>Reconfigure urinals side-by-side with acceptable space (3ft) between them.</td>
<td></td>
</tr>
</tbody>
</table>

### heuristic evaluation pros & cons

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Can provide some quick and relatively inexpensive feedback</td>
<td>• Trained usability experts are sometimes hard to find and can be expensive</td>
</tr>
<tr>
<td>• You can obtain feedback early in the design process</td>
<td>• Heuristics may lead evaluators to focus on some types of issues but miss others</td>
</tr>
<tr>
<td>• Assigning the correct heuristic can suggest a good place to start for corrective measures</td>
<td>• May identify more minor issues than major issues</td>
</tr>
<tr>
<td>• Can be used with other usability testing methodologies</td>
<td>• May identify ‘false positive’ issues – issues that actual users might not actually encounter</td>
</tr>
<tr>
<td>• Can provide a more comprehensive assessment of the system than usability testing</td>
<td>• May not have same “credibility” as user data</td>
</tr>
</tbody>
</table>

### why multiple evaluators?

- every evaluator doesn’t find every problem
- proficient evaluators find both easy & hard (subtle) ones
heuristic eval, cont.

research result:
• 4-5 evaluators usually able to identify 75% of usability problems
• user testing and usability inspection have a large degree of non-overlap in the usability problems they find (i.e. it pays to do both)

cost-benefit:
• prototype evaluation activities often expensive / slow; but some can be quick / cheap, and still produce useful results
• ultimate trade-off may be between doing no usability assessment and doing some kind

combining HE and CW

› HCI practitioners often use a combination of both that might vary based on what they’re trying to learn
  – e.g., while doing a walkthrough for a task, apply the heuristics at each step in addition to the CW questions.

optional further reading
Fluid project Heuristic + Walkthrough method:
http://wiki.fluidproject.org/display/FluidUX/Walkthrough+Preparation+and+Execution
Conducting expert reviews: what works best?

appendix

examples of heuristics:
spotting violations

H1: visibility of system status

keep users informed about what is going on
example: consider system response time (user must wait)
• 0.1 sec: no special indicators needed, why?
• 1.0 sec: user starts to lose track of data, objects, etc
• 10 sec: max duration if user to stay focused on action
• for longer delays, use percent-done progress bars

Time left: 0:00:19 searching database for matches
H1: visibility of system status
keep users informed about what is going on
• appropriate visible feedback

What did I select?

What mode am I in now?

How is the system interpreting my actions?

H2: match between system & real world
• speak the users’ language
• follow real world conventions

(bad) example: Mac desktop
• dragging disk to trash
• this should delete it, not eject it!

H2: match between system & real world
• speak the users’ language
• follow real world conventions

e.g. withdrawing money from a bank machine

H3: user control & freedom
• “exits” for mistaken choices, undo, redo
• don’t force down fixed paths

How do I get out of this?
H3: user control & freedom

- “exits” for mistaken choices, undo, redo
- don’t force down fixed paths

strategies:
- cancel button (for dialogs waiting for user input)
- universal Undo (can get back to previous state)
- interrupt (especially for lengthy operations)
- quit (for leaving the program at any time)
- defaults (for restoring a property sheet)

H4: consistency & standards

consistency of effects → predictability
- same words, commands, actions should always have the same effect in equivalent situations

consistency of language and graphics
- same info/controls in same location on all screens/dialog boxes - NOT:
  - same visual appearance across the system (e.g. widgets)
    - e.g. NOT different scroll bars in a single window system!

consistency of input
- require consistent syntax across complete system

H5: error prevention

try to make errors impossible
- modern widgets: only “legal commands” selected, or “legal data” entered
- provide reasonableness checks on input data
  e.g. upon entering order for office supplies:
  “5000 pencils is an unusually large order. Do you really want to order that many?”

H5 .... errors we make

mistakes
- arise from conscious deliberations that lead to an error instead of the correct solution

slips
- unconscious behavior that gets misdirected en route to satisfying goal
  - e.g. drive to store, end up in the office
- shows up frequently in skilled behavior
  - usually due to inattention
- often arises from similarities of actions
H5 ... types of slips

**capture error**
- frequent response overrides [unusual] intended one
- occurs when both actions have same initial sequence
  - change clothes for dinner \(\rightarrow\) find oneself in bed (William James, 1890)
  - confirm saving of a file when you don’t want to delete old version

**description error**
- intended action has too much in common with others possible
  - e.g. when right and wrong objects physically near each other
    - pour juice into bowl instead of glass
    - go jogging, come home, throw sweaty shirt in toilet instead of laundry
    - move file to trash instead of to folder

**loss of activation**
- forgetting the goal while carrying out the action sequence
  - e.g. start going to a room and forget why by the time you get there
    - navigating menus/dialogs, can’t remember what you are looking for
      - but continue action to remember (or go back to beginning?)

**mode errors**
- people do actions in one mode thinking they are in another
  - refer to file that’s in a different directory
  - look for commands / menu options that are not relevant

---

H6: recognition rather than recall

Computers good at remembering things, people aren’t!
- menus, icons, choice dialog boxes vs. cmd lines, field formats
- relies on visibility of objects to the user (but less is more!)

---

H7: flexibility and efficiency of use

Experienced users should be able to perform frequently used operations quickly

**strategies:**
- keyboard and mouse accelerators
  - abbreviations
  - command completion
  - menu shortcuts & function keys
  - double clicking vs. menu selection
- type-ahead (entering input before the system is ready for it)
- navigation jumps
  - go to desired location directly, avoiding intermediate nodes
- history systems
  - WWW: ~60% of pages are revisits
H8: aesthetic and minimalist design
no irrelevant information in dialogues
bad example:

H9: help users recognize, diagnose, and recover from errors
• error messages in plain language
• precisely indicate the problem
• constructively suggest a solution

H10: help and documentation
help is not a replacement for bad design!
simple systems: walk up and use; minimal instructions
most other systems:
• feature-rich
• some users want to become “expert” rather than “casual” users
• intermediate users need reminding, plus a learning path
many users do not read manuals
usually used when users are panicked & need help NOW
• need online documentation, good search/lookup tools
• online help can be specific to current context
sometimes used for quick reference
• syntax of actions, possibilities...
• list of shortcuts ...

H10 … types of help
tutorial and/or getting started manuals
• short guides that people usually read when first encounter system
  – encourage exploration and getting to know the system
  – communicate conceptual material and essential syntax
• on-line “tours”, exercises, and demos
  – demonstrate very basic principles through working examples
reference manuals
• used mostly for detailed lookup by experts
  – rarely introduces concepts
  – thematically arranged
• on-line hypertext
  – search / find
  – table of contents
  – index
  – cross-index
H10 … types of help (cont’d)

reminders
• short reference cards
  – expert user who just wants to check facts
  – novice who wants to get overview of system’s capabilities
• keyboard templates
  – shortcuts/syntactic meanings of keys; recognition vs. recall; capabilities
• tooltips
  – text over graphical items indicates their meaning or purpose

H10 … types of help (cont’d)

context-sensitive help
• system provides help on the interface component the user is currently working with
  – Macintosh “balloon help”
  – Microsoft “What’s this” help

wizards
• walks user through typical tasks
• reduces user autonomy

discount evaluation methods

cheap (thus ‘discount’)
• no special labs or equipment needed
• doesn’t need to involve users directly
  – but the more careful (and informed by users) you are, the better it gets

fast
• on order of 1 day to apply
• standard usability testing may take a week

easy to learn
• can be taught in 2-4 hours
types of discount methods

cognitive walkthrough: “mental model”
• assesses “exploratory learning stage” (new/casual users)
• what mental model does the system image facilitate?
• done by non-experts and/or domain experts

heuristic evaluation: “fine tune”
• targets broader use range (including expert)
• fine-tunes the interface (hi-fi prototypes; deployed systems)
• HCI professionals apply a list of heuristics while simulating task execution