Usability evaluation

Adaptive Decision Support Systems

Usability evaluation

Today's goal: learn how to do a usability evaluation

- Heuristic evaluation and think-aloud testing (you'll do the latter for A4 if you design a system)

Structure:

- Nielsen's heuristics
- Usability Aspect Reports (UARs)
- Theory behind think-aloud testing
- Think-aloud testing in practice

Comparison

Scientific studies usually result in papers

Experiments, most surveys, grounded theory

Practical studies aim to contribute to design/development

Heuristic evaluation, think-aloud testing, contextual inquiry, participatory design, some surveys

Interested in learning more?

Take HCC-8330: Research Methods!

Comparison

Summative studies evaluate the quality of something

Most experiments, some surveys

Formative studies aim to improve something

Heuristic evaluation, think-aloud testing, contextual inquiry, participatory design, some surveys

Outside of this categorization:

Grounded theory is kind of neither?

Comparison

Contextual inquiries are used **before** technical solutions are developed

Think-aloud tests and heuristic evaluation are used **during** the development (think-aloud late; heuristic evaluation early)

Experiments are often conducted **after** the development

Surveys can happen at any point; grounded theory falls outside of this categorization

Heuristic evaluation

Nielsen's usability heuristics

Heuristic evaluation

Find a lot of usability problems using very few resources

How:

List of 10 heuristics

Advantages:

No real users needed

4-5 evaluators can find most major problems

Heuristic evaluation

Visibility of system status

Match between system and real world

User control and freedom

Consistency and standards

Error prevention

Recognition rather than recall

Flexibility and efficiency of use

Aesthetics and minimalist design

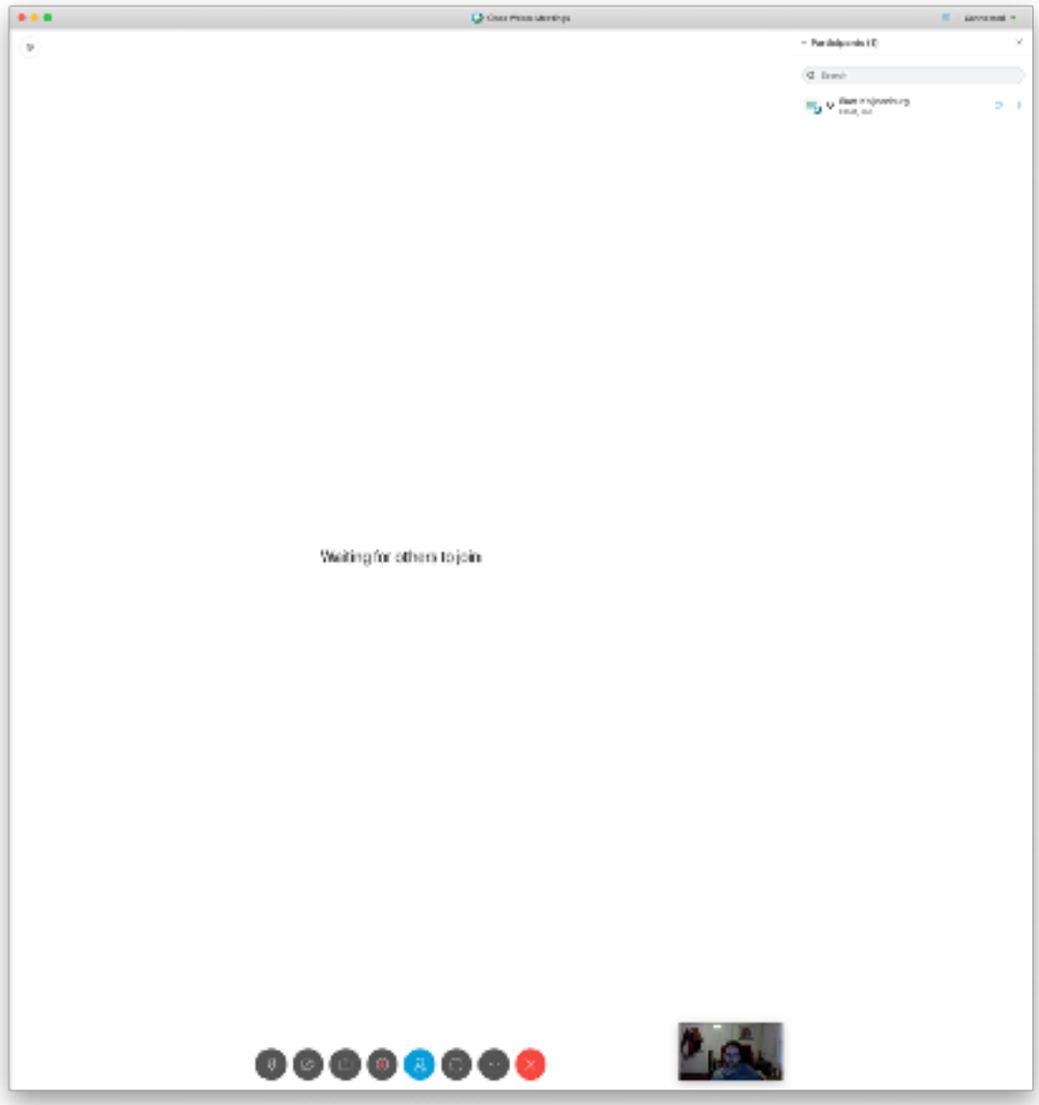
Help users recover from errors

Help and documentation

Visibility of system status

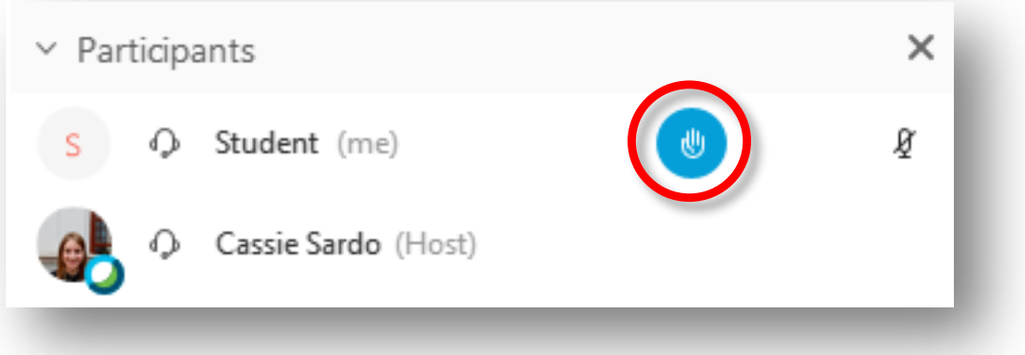
The computer should keep the user informed about what is going on

The computer has to supply information to the user through sight or sound



Match between system and real world

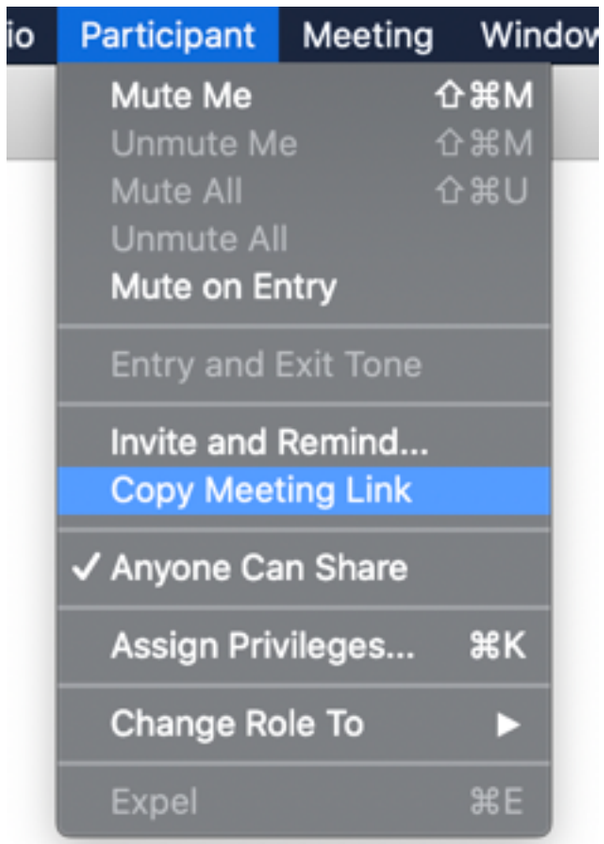
The UI design should use concepts, language, and conventions that are familiar to the user



User control and freedom

Users should be able to

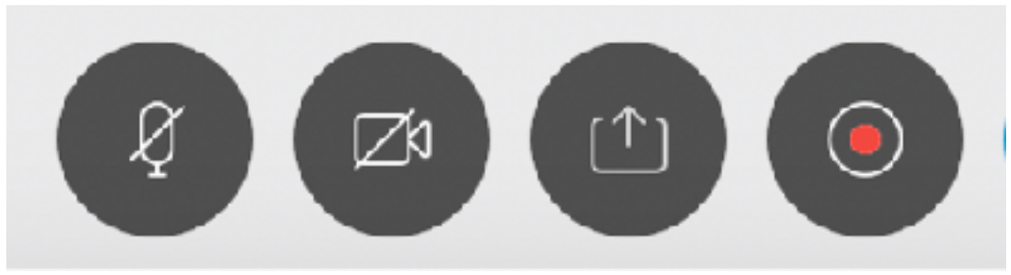
- undo their actions easily
- exit from any interaction quickly at any time
- not be forced into a series of actions controlled by the computer



Consistency and standards

Information that is the same should appear to be the same

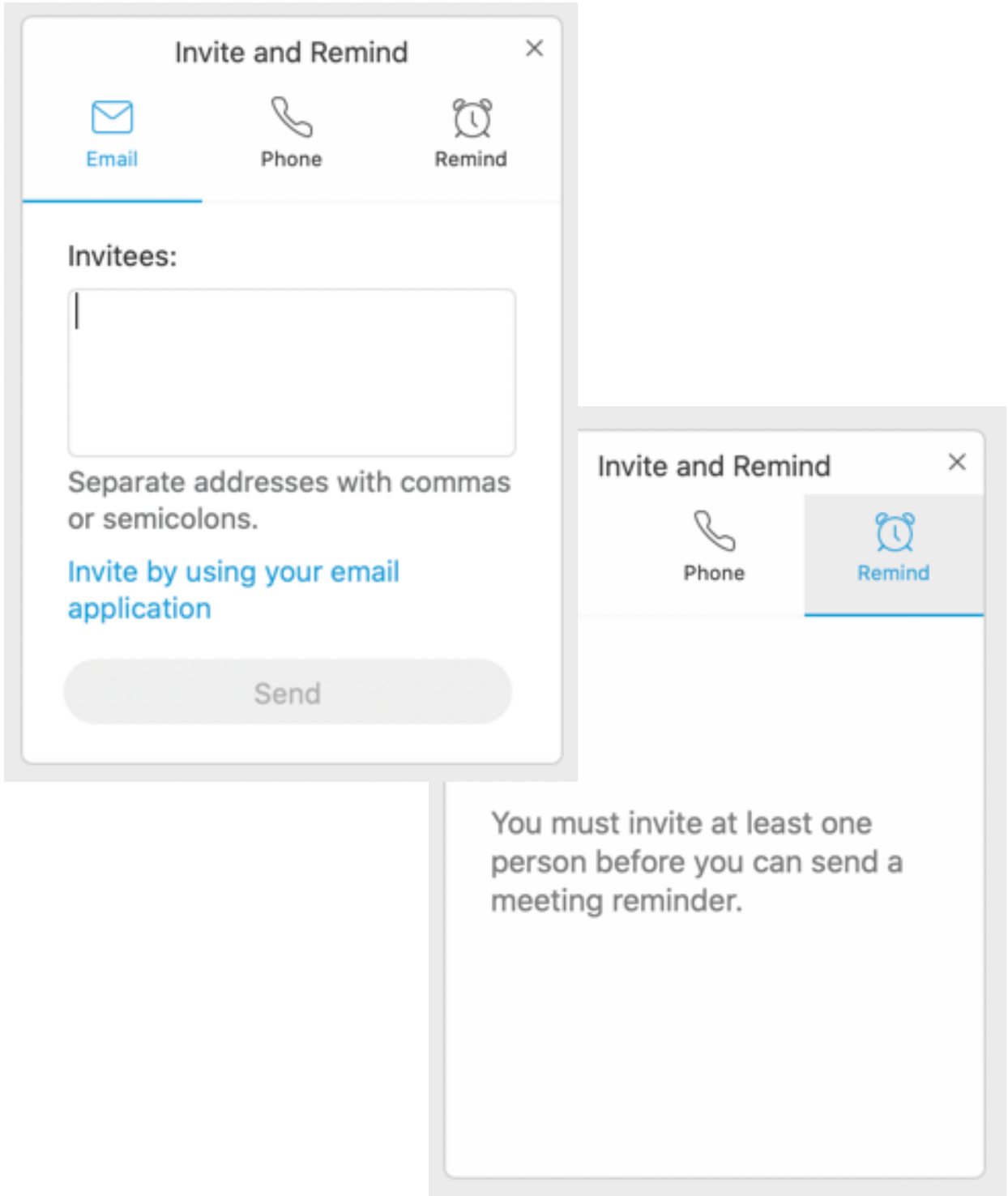
Use the user's prior knowledge and experience with other parts of the same application



Error prevention

Prevent errors from happening

Show only those actions that are acceptable at that particular point

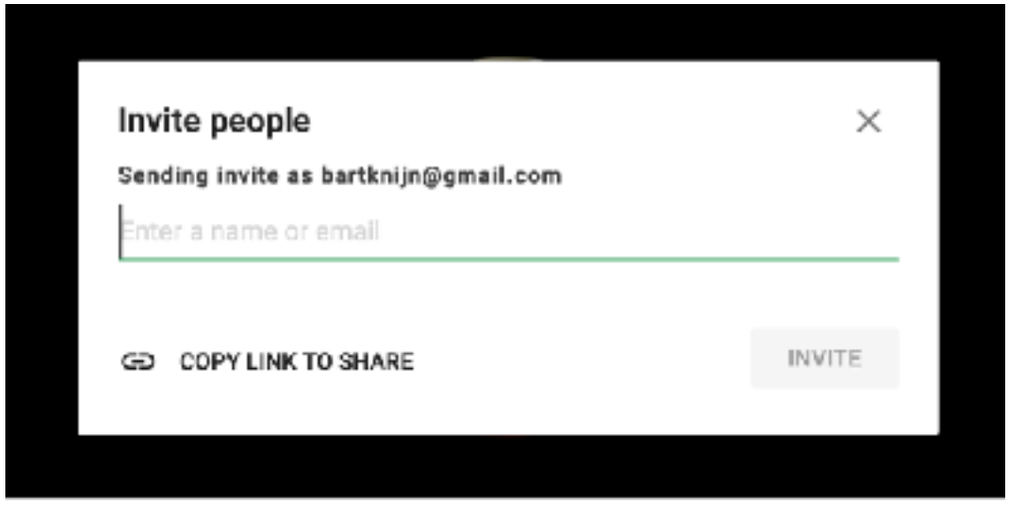


Recognition rather than recall

Show all objects and actions available to the user

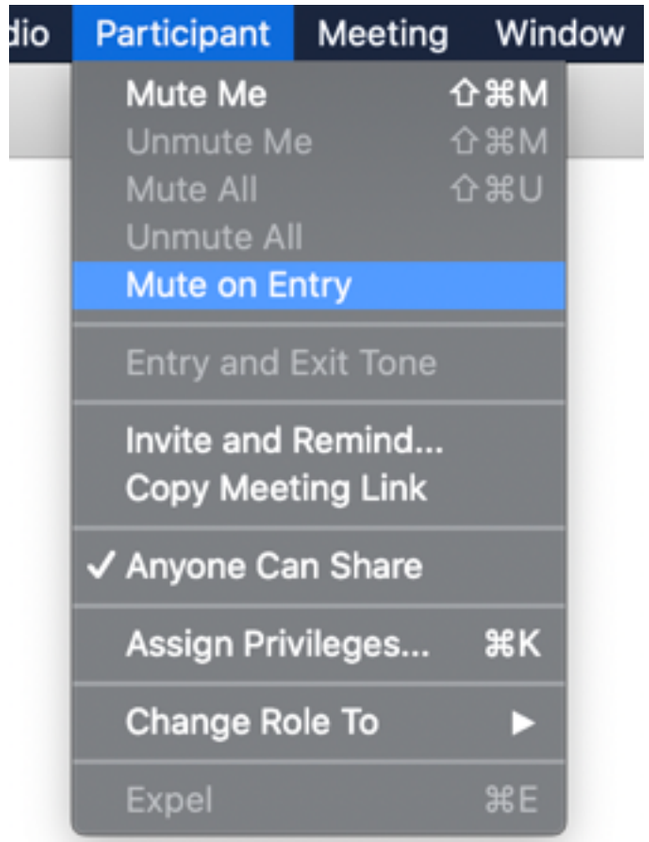
It is much easier for someone to figure out what to do

- If there are cues in the environment
- If there is knowledge in long-term memory of what to do



Flexibility and efficiency of use

Skilled users should also be able to tailor their interface to speed up frequent actions



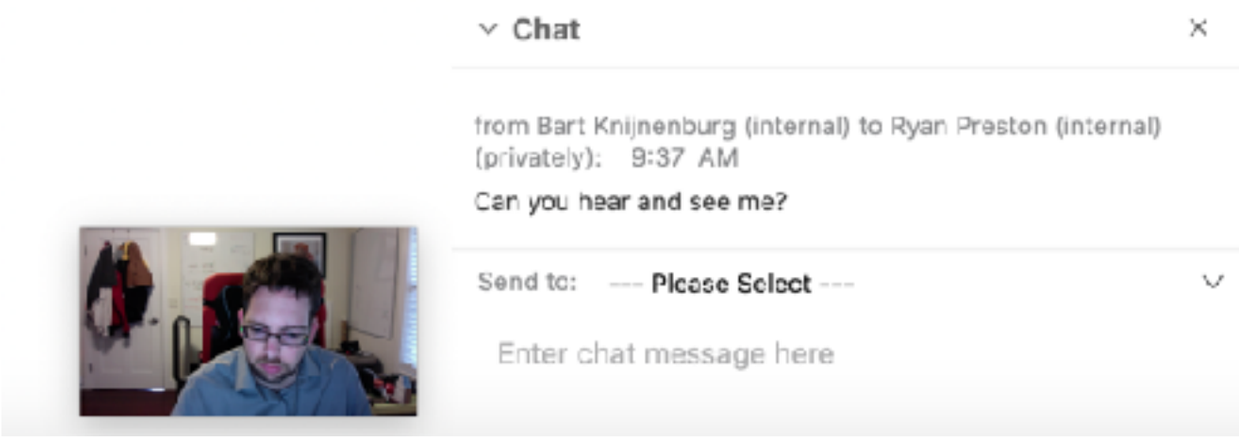
Skilled users can use motor memory

Also, typing single keys is faster than continually switching between keyboard and mouse

Aesthetics and minimalist design

Eliminate irrelevant screen clutter

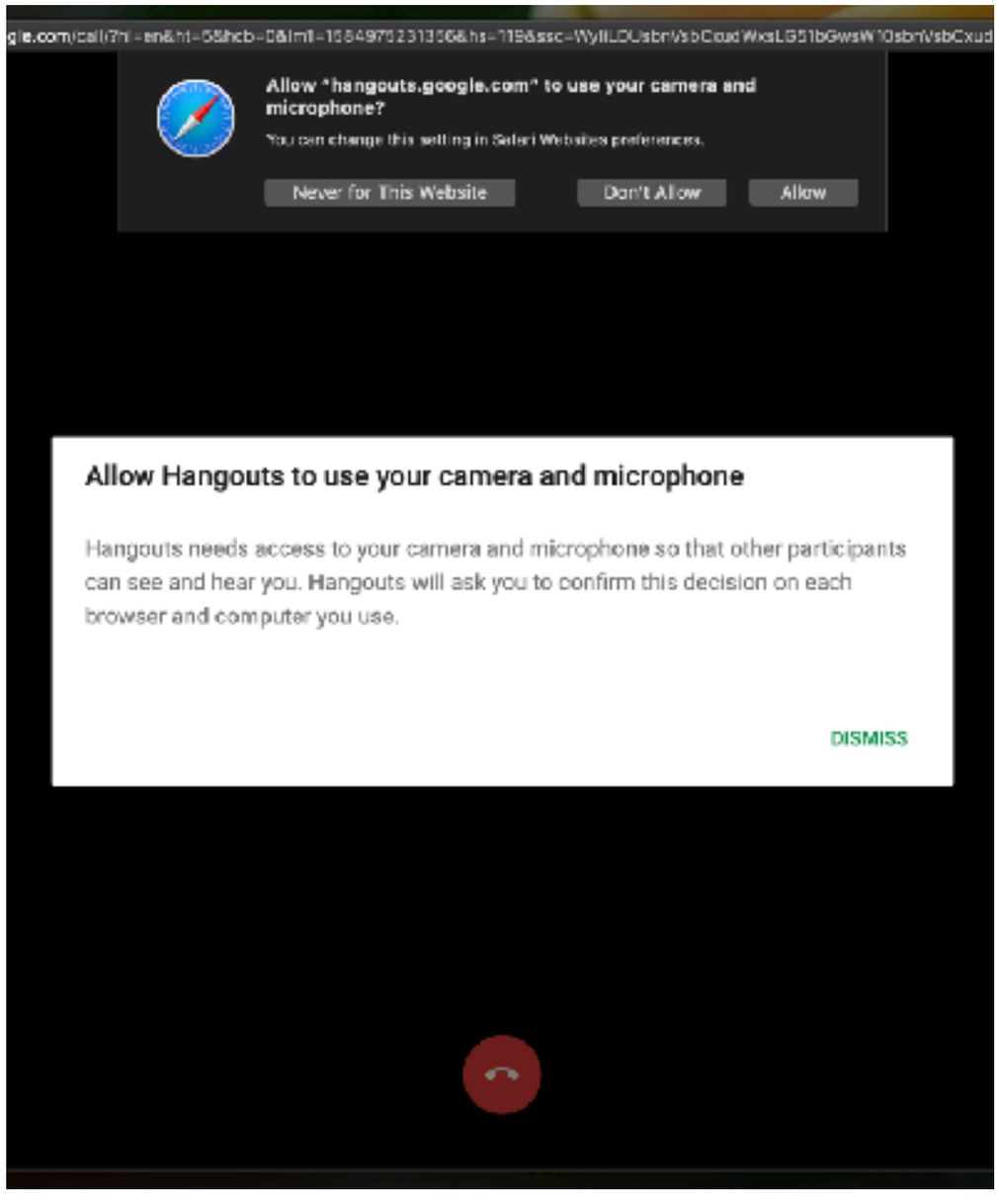
The more clutter, the more information the eyes must search through



Help users recover from errors

Error messages should

- be written in plain language
- tell the user what the problem is
- give constructive advice about how to recover
- give the user sufficient information to understand the situation

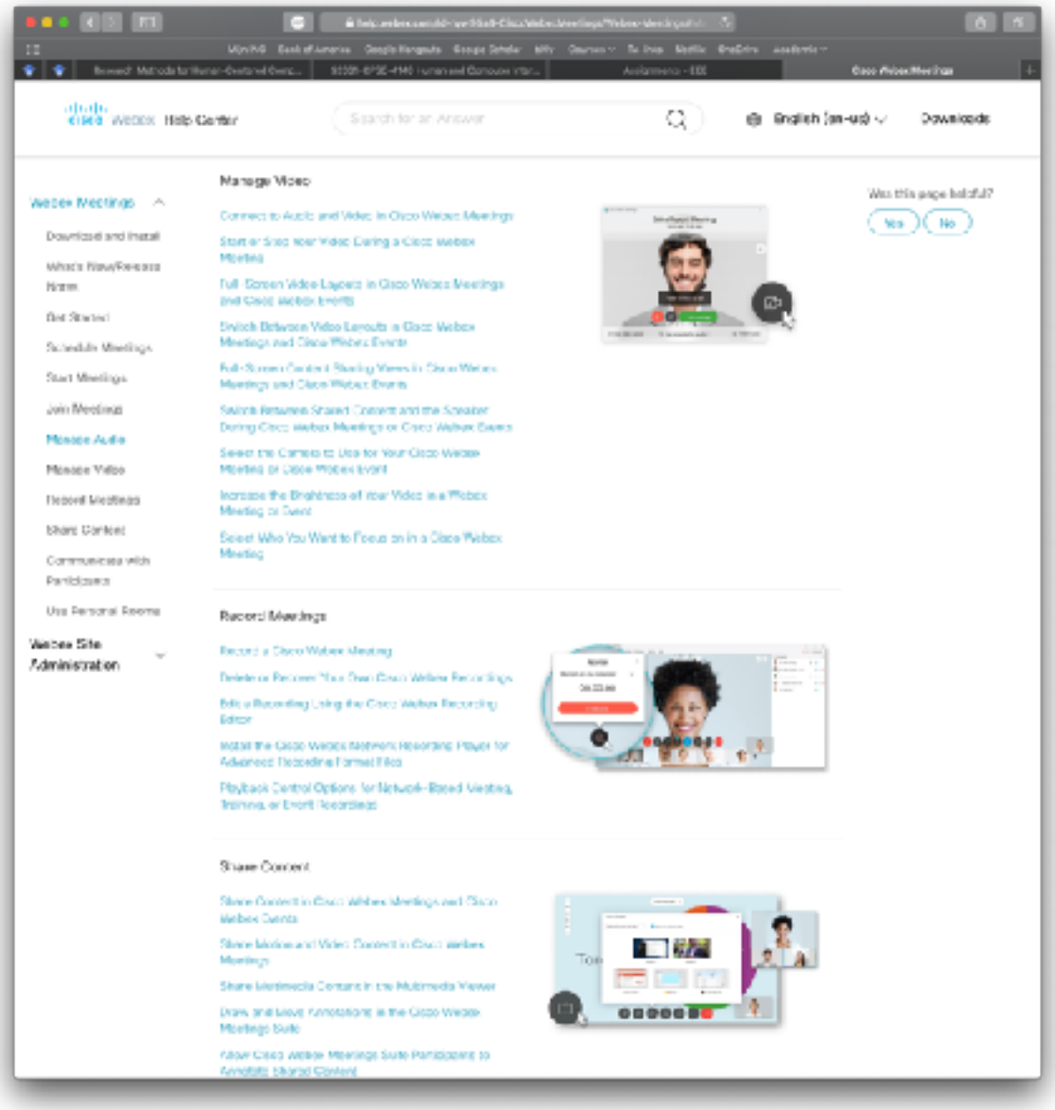


Help and documentation

If the system is not extremely simple, it is going to need help and documentation

Give the user sufficient information to understand the application

Mobile apps and Web sites are often expected to be used without training!



Further reading...

All heuristics briefly explained:

http://www.nngroup.com/articles/ten-usability-heuristics/

Web site examples of each heuristic:

http://designingwebinterfaces.com/6-tips-for-a-great-flex-ux-part-5

A video description of each heuristic:

https://www.youtube.com/watch?v=cTtc90jCULU

Usability Aspect Reports

A formal method for documenting usability issues

UARs

Usability Aspect Reports

A formal method for documenting usability issues

- Have found widespread use in industry

Each problem has its own report

- Can be ranked by severity
- Can be grouped by feature / page

Includes a suggested solution

- But better to create solutions that solve multiple problems

UAR content

Number (your name - aspect number)

Problem/Good Aspect (pick one)

Name

Short “name” for the problem / good aspect

Evidence

Explicitly name one (or more) of the 10 heuristics here!

Description of the interface aspect where the heuristic is violated / adhered (include a screenshot or reference)

UAR content

Explanation

Argument why this specific heuristic is violated / adhered to here (keep it short but convincing): “The heuristic is violated **because**...”

Sometimes your explanation makes assumptions about how the user (e.g. what they are familiar with). If so, express those assumptions and why you feel they are valid: “The user will **probably** do/know... **because**...”

UAR content

Severity or benefit

NA – good aspect

1 – cosmetic problem (does not matter too much)

2 – minor problem (would be nice to solve, but not a high priority)

3 – major problem (a problem that should definitely be solved with high priority)

4 – usability catastrophe (a problem that renders the Web site / app useless)

UAR content

Motivation for severity rating

A justification for that rating in terms of frequency (does it occur for everyone or just in some specific cases?), impact (is it a minor inconvenience or a huge hurdle?), and persistence (is it easily fixed/avoided or does it keep happening?)

Each rated low, medium, or high

Explain how you weighed these factors

For good aspects the rating is “NA”, but you can still describe frequency, impact, and persistence

UAR content

Possible solution and/or trade-offs

How would you solve the problem?

What are potential downsides of your solution?

For good aspects, list potential downsides of the current solution

Relationships

How is this aspect related to other aspects you found?

Workflow

Suggested approach:

Find a problem / good aspect

Draft the UAR

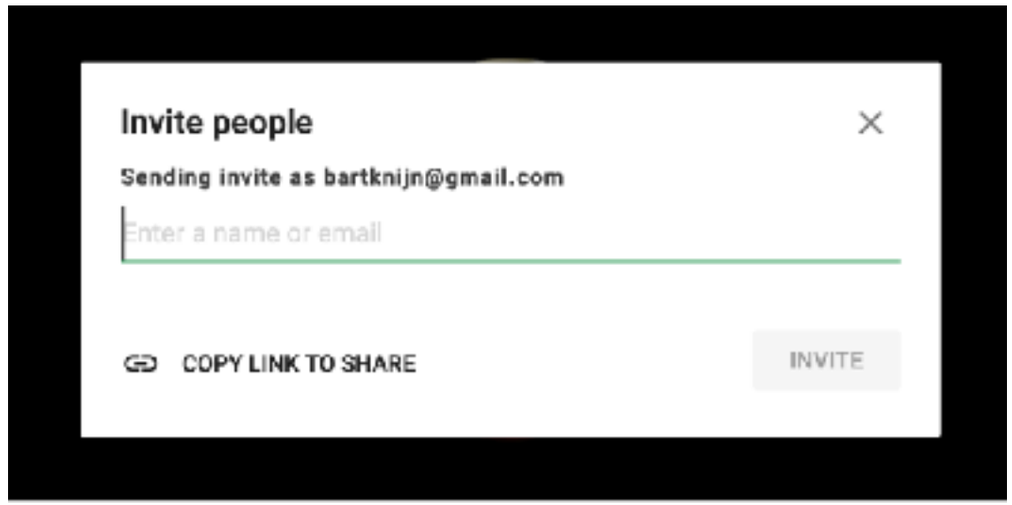
- Name, evidence, and a brief explanation only, as a reminder for yourself

Repeat until you have found enough

Finalize each UAR

- The final versions should be understandable to an outsider

Example UAR



Google Hangouts example

No. BK-01, Problem

Name: No list of (online) contacts

Evidence: see screenshot; whereas Facebook Messenger and each Skype have a list of people who are online, Google Hangouts has no list of (online) people

This violates the “recognition rather than recall” and “visibility of system status” heuristics

Example UAR

Explanation: “Recognition rather than recall”: This heuristic is violated because Google forces you to recall the names of people you want to talk to. A business user will probably encounter a situation where recalling the names of contacts is difficult, because are likely to have a long list of contacts they call only occasionally.

“Visibility of system status”: This heuristic is violated because the user cannot see who is currently available/online. The user will probably encounter the situation where they want to make sure that their contact is available/online, because sometimes calls are not scheduled in advance.

Example UAR

Severity or benefit, choose from:

- NA: good aspect
- 1: cosmetic problem (does not matter too much)
- **2: minor problem** (would be nice to solve, but not a high priority)
- 3: major problem (a problem that should definitely be solved with high priority)
- 4: usability catastrophe (a problem that renders the Web site / app useless)

Example UAR

Motivation for severity rating:

Frequency = high, Impact = low, Persistence = medium

How I weighted the factors:

The need to recall names happens every time you want to start a hangout, but it only takes a little extra time or effort to recall the names (it will not prevent the user from continuing). Moreover, if you do hangouts with the same people often enough, you will start remembering their names/handles

Example UAR

Possible solution and/or trade-offs: The interface could include a list of all online contacts, but this list could be very long, and it could clutter up the screen. To solve this problem, the list could be grouped by context (work, family, friends, etc.) or ordered by frequency of interaction (making it easier to select people you often talk to)

Relationships: None (for now)

Another example

Google Hangouts example

No. BK-02, Good Aspect

Name: Share meeting link

Evidence: see screenshot; Google Hangouts allows you to share a link to the meeting through email or other channels

This adheres to the “user control and freedom” heuristic

Another example

Explanation: This heuristic is adhered to because Google allows you to invite people to a meeting outside the standard Google Hangouts workflow. The user may encounter a situation where they would want to invite a participant outside Google Hangouts, either because they are already chatting/emailing on another platform, and/or because do not know the person's Hangouts handle.

Another example

Severity or benefit, choose from:

- **NA: good aspect**
- 1: cosmetic problem (does not matter too much)
- 2: minor problem (would be nice to solve, but not a high priority)
- 3: major problem (a problem that should definitely be solved with high priority)
- 4: usability catastrophe (a problem that renders the Web site / app useless)

Another example

Motivation for severity rating:

Frequency = medium, Impact = low, Persistence = medium

How I weighted the factors:

Some but not all users will use this option. It makes inviting people to a hangout a little easier. If you do hangouts with the same people often enough, you will start remembering their names/handles, so the need for a meeting link will be less common.

Another example

Possible solution and/or trade-offs: A potential downside of this feature is that anyone with the link can join the meeting, which could be a potential privacy issue.

Relationships: BK-01 (The link can also be used as a way to overcome problem BK-01)

Final note

Critique the design, not the person!

Academic rule: write reviews about the paper (“the paper is too long”) not the authors (“you wrote too many words”)

Don’t take the critique too personal

Getting usability feedback is a good thing: catching problems early makes your life easier later on

Solutions can be a matter of opinion, problems rarely are

If an evaluator finds a problem, it is likely that some users will encounter the same problem too

Think-aloud testing

Theory and description

Think-aloud testing

An empirical technique for assessing the usability of a prototype of an interface

> "may be the single most valuable usability engineering method" (Jakob Nielsen)

Ask a user to **think-aloud** while performing a task on your system

Watch silently and learn where the user has problems

Report **critical incidents** in UARs

Thinking aloud?

"Thinking aloud" = translating everything you are thinking into words

- does not change the way people think
- does slow them down

Not "explain aloud" = reason about why you do things

- does change the way people think
- people are bad at this anyway

Thinking aloud?

Tip:

If you notice that a user is explaining rather than just reporting what they think, you need to put them back on the right track

If you notice that a user goes silent, ask them: “can you tell me what you’re doing now?”

Watch silently?

Goal: Learn how the user would do the task on their own!

Do not **argue** with the user

It can be frustrating if the user does not understand how your system works

This is not anyone's "fault"... UI design is an iterative process of trial and error!

Try to understand their misunderstandings so that you can improve your design

Watch silently?

Likewise, do not **help** the user

You want to give the user ample time to figure out a solution on their own

This can help you judge the severity of the problem

Encourage the user to continue on their own!

If the user still can't figure it out after 2-3 minutes, you can give them a hint

Critical incidents?

A method adopted from fighter pilot training

The user thinks aloud as they do the task

Usually in a lab, screen is captured, test is audio-recorded

The analyst observes (live/recording) and reports critical incidents using the UAR format

Analysts categorize the incidents (from multiple tests) and interpret the observations

Findings are presented in a report (including solutions)

Criterion

Criteria for critical incidents (problems):

- User takes too long to finish a certain step
- User tries several times before they find the right action
- User gives up, even after repeated encouragement
- User uses a very suboptimal method to accomplish task
- User thinks they did it right, but actually got it wrong
- User expresses distressed surprise
- User has a negative reaction / says something is a problem
- User makes a design suggestion

Criterion

Criteria for good aspects:

- User has a positive reaction / says something is really easy
- User expresses happy surprise
- User is able to accomplish something that was a problem with other tools

Think-aloud UARs

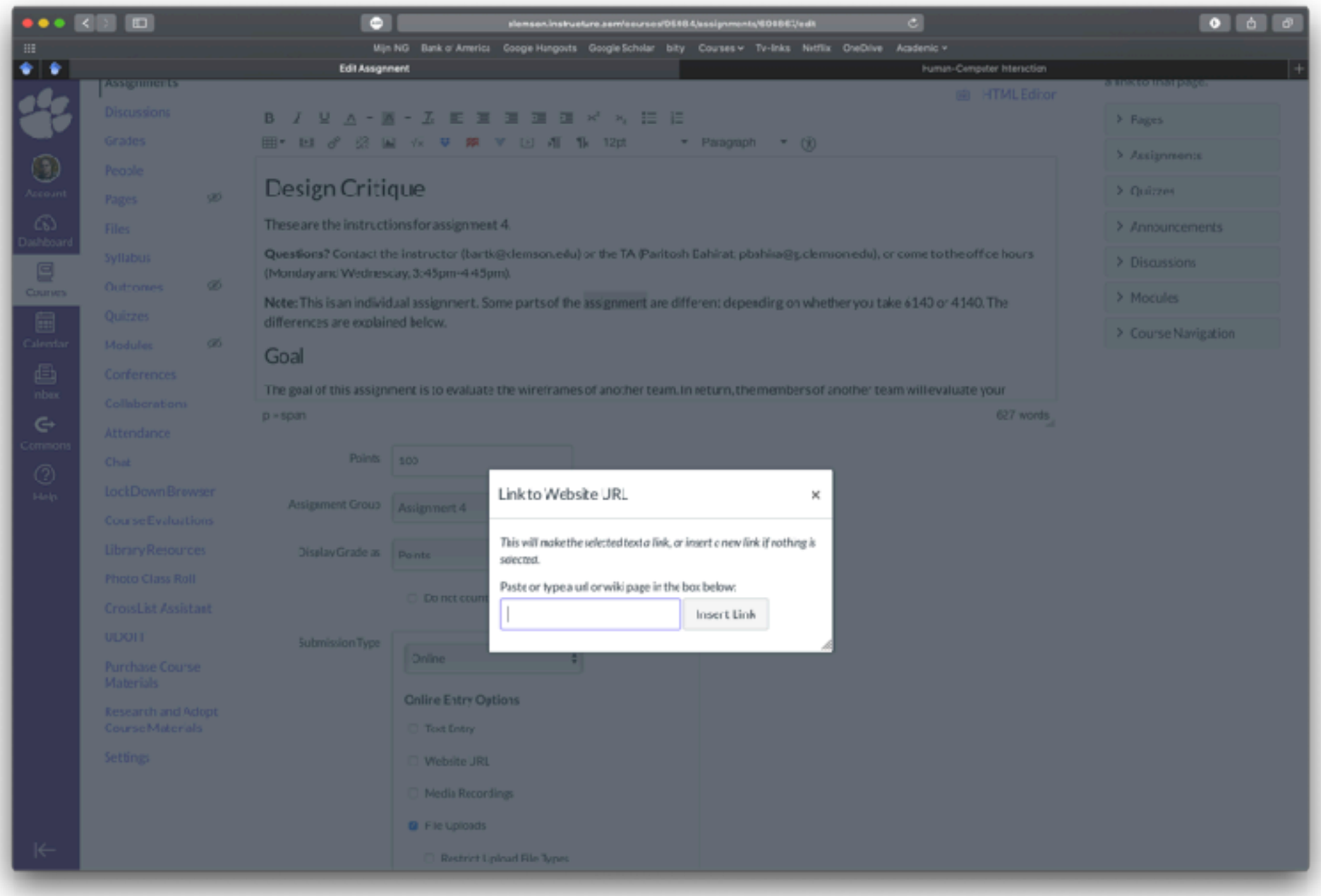
How to write up “critical incidents”

Think-aloud UARs

Very similar to Heuristic Evaluation UARs

Difference: instead of identifying the violated/adhered heuristic, you must now identify the criterion that makes this a critical incidents (see previous slides)

Example


Edit Assignment
Human-Computer Interaction
Design Critique
These are the instructions for assignment 4
Questions? Contact the instructor (bartk@clemson.edu) or the TA (Paritosh Bahirat, pbahira@g.clemson.edu), or come to the office hours (Monday and Wednesday, 3:45pm-4:45pm).
Note: This is an individual assignment. Some parts of the assignment are different depending on whether you take 6140 or 4140. The differences are explained below.
Goal
The goal of this assignment is to evaluate the wireframes of another team. In return, the members of another team will evaluate your
627 words
Points
100
Assignment Group
Assignment 4
Display Grade as
Points
Submission Type
Online
Online Entry Options
Text Entry
Website URL
Media Recordings
File Uploads
Restrict Upload File Types
Link to Website URL
This will make the selected text a link, or insert a new link if nothing is selected.
Paste or type a url or wiki page in the box below:
Insert Link
Assignments
Discussions
Grades
People
Pages
Files
Syllabus
Outcomes
Quizzes
Modules
Conferences
Collaborations
Attendance
Chat
LockDown Browser
Course Evaluations
Library Resources
Photo Class Roll
CrossList Assistant
ULOTT
Purchase Course Materials
Research and Adopt Course Materials
Settings
Account
Dashboard
Courses
Calendar
Inbox
Commons
Help
Pages
Assignments
Quizzes
Announcements
Discussions
Modules
Course Navigation

Example UAR

Name: Cut and Paste hyperlink into pop-up doesn't work

Evidence: The user wants to cut and paste the hyperlink into the hyperlink pop-up window ("Can I cut and paste this thing?"). Tries to cut the text (access the context-menu) three times. Uses correct sequence finally.

When the user wants to make another link, he again fails to copy and paste the URL several times ("I'm going to try it again", "I'm just going in circles now", "can I paste it here? – no", "now I copied it can I go back here and paste it? – No, I can cut, copy, but I cannot paste!")

Example UAR

He ends up doing it (suboptimal) with pen and paper several times (“maybe I need to write it down, I guess I should really write it down”).

Criterion (new): User tries something several times before they find the right action; User uses a very suboptimal method to accomplish the task

Explanation: When you have the hyperlink pop-up open, you cannot go back to the page to copy a URL. You have to close the hyperlink pop-up and copy the URL first.

Example UAR

Severity or benefit: 4: usability catastrophe

Frequency = high (happens several times during the session)

Impact = high (user ends up keying the URL manually, this is slow and error-prone, and a major annoyance)

Persistence = high (even after successfully performing the task once, the user fails again later! It is clearly very hard to learn)

Example UAR

Possible solution: Allow further manipulation of the page, even when the hyperlink pop-up is open. This way the user is not forced to copy the hyperlink before the prompt.

- Tradeoff: The user might "lose" the hyperlink pop-up if he can switch freely

Relationships: TBD (add these after you discuss them in your group)

Doing a think-aloud test

Practical steps

Practical steps

Prepare your **prototype** and **task description**

Do a practice round on your teammates!

Conduct a think-aloud test with **one user**

Use WebEx and/or screen-and-voice-recording software

Look at the recording and **report critical incidents**

Using the UAR format

Synthesize and **finish** the UARs

Write a **report**

Preparation

If you haven't done so, make your wireframes **interactive**

- Make buttons/links "clickable"
- Create transitions to other pages
- Create popups etc. where needed
- Allow users to fill out fields

Support a number of **exploration and decision tasks** (only)

You can assume certain inputs (e.g. for search fields)

Preparation

Write up a set of tasks

- This will be your “test document”
- One task per page
- Make sure the description is very clear

Practice the think-aloud session

- Take turns to practice on each other, make sure to test the recording functionality, too!
- If you already find problems, feel free to write them up!

The test itself

Find a user

- Should be someone who would use the app / Web site in real life
- Not someone from this class

In person:

- Print the task description (if possible)
- Open the prototype on your computer
- Set up screen + audio recording with Camtasia (free for Clemson students)

Introduction

Introduce yourself

> "I am [name] and I am taking a class on usability testing"

State the goal of the particular study

> "I am testing a prototype of a new app / Web site for [task]"

State that you are testing the system, not the user

> "The prototype is unfinished, so if anything goes wrong, that's actually good! It allows us to improve it before it is built for real"

Introduction

Ask for permission to start recording

When the user says yes, hit the record button… don’t forget to record!

Introduction

Train the user to think aloud:

"In this observation, I am interested in what you think about as you perform the tasks. I want you to 'think aloud' while you are using the prototype: please to talk aloud constantly from the time I give you the task until you have completed it. Just verbalize what you see, what you do, and what you think. I don't want you to try to explain to me what you are doing. Just act as if you are speaking to yourself—just a little louder."

Introduction

You can give the user a demonstration:

> "Let me demonstrate thinking aloud as I disable the screen saver on this computer."
>
> Make sure it is enabled before you start

You can ask the user to practice with a simple task:

> "Now it's your turn to practice. Please think aloud as you set the background color of this computer to blue."

Introduction

Final instructions:

"As you're doing the tasks, I won't be able to answer any questions. But go ahead and ask them anyway so that I can learn more about what kinds of questions the app / Web site brings up. I'll answer your questions after the session."

"Also, if you forget to think aloud, I'll say, 'Please keep talking.' Do you have any questions about thinking aloud?"

"Please read the first task to me to get comfortable talking aloud."

Conduct the test

Let users do the task

- Don't help them or answer questions (unless they get really frustrated)
- Take notes of "critical incidents"

If they stop talking say:

- "Please keep talking"

If they start explaining rather than reporting their thoughts:

- "Please don't try to explain what you are thinking. Just act as if you are alone, speaking to yourself—just a little louder."

Write UARs

Re-watch the video and write down the main problems

- Use the criteria to detect them
- You should be able to find at least **4 problems** and **1 good feature**
- As before, it's easiest to write draft versions first, and more complete versions later
- You can even do the complete versions after the team meeting

Synthesize

Your group should now have about 15-25 draft UARs

Briefly walk through them together

Next step is to consolidate these UARs:

- Identify any exact duplicates, merge them but (keep multiple “[name]-[number]” labels)
- Group the entire set of UARs by topic (topic = a specific part of the interface, or a common type of problem), 4-5 UARs per group
- Useful tool for grouping: Trello or miro.com (free tools)
- Label the groups (you’ll likely have one “miscellaneous”)

Finalize and report

Finish the UARs

- Fill out the missing fields

Write a report

- Lead, intro, one section per group of problems, conclusion, reflection
- Appendix with task descriptions, prototype (separate file or link, and test videos separate files or links)