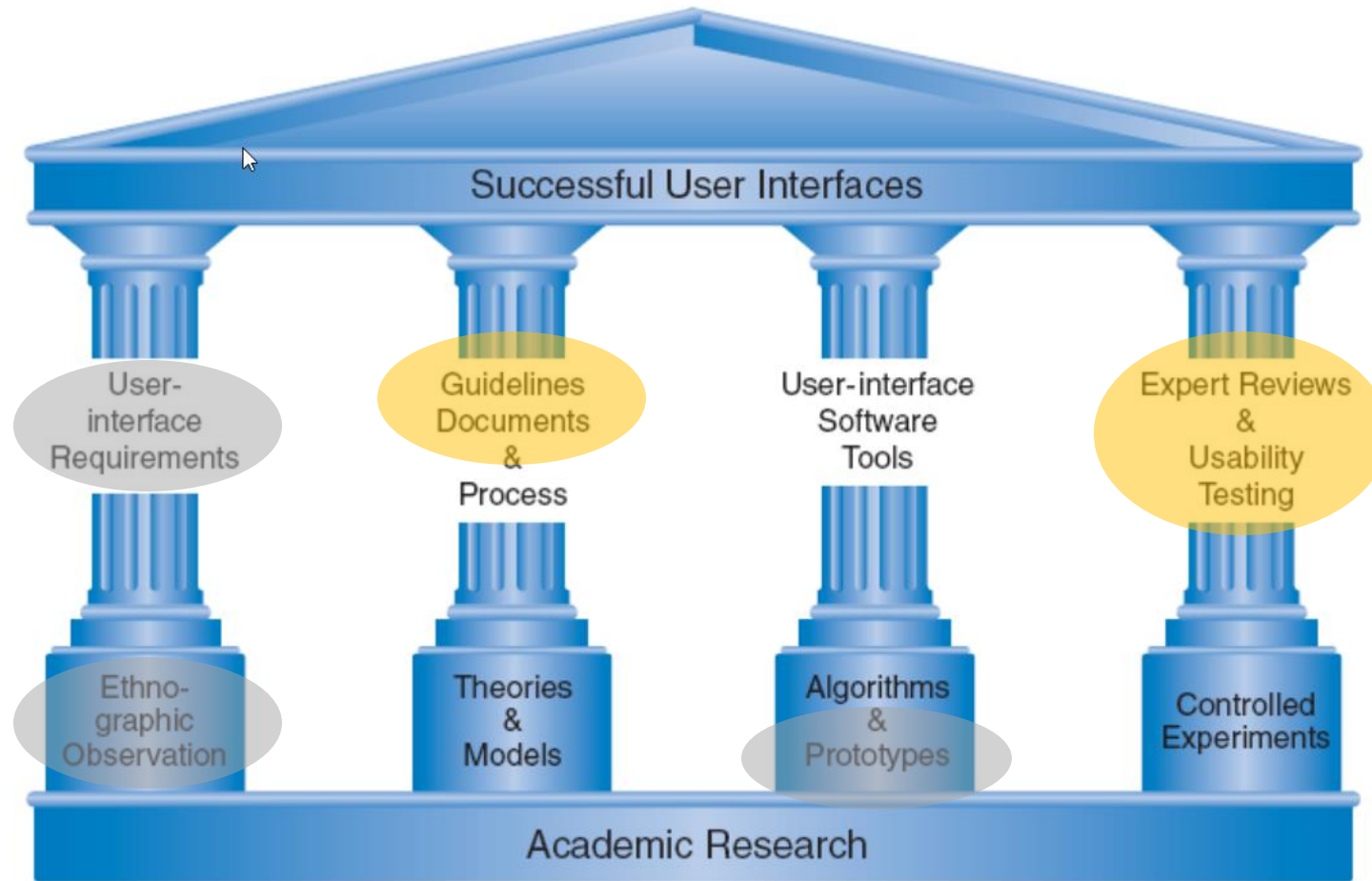# Evaluation: Introduction and Heuristics

User Experience Design

Alberto Monge Roffarello

Academic Year 2025/2026

The Four Pillars of Design

Ben Shneiderman & Catherine Plaisant, Designing the User Interface: Strategies for Effective Human-Computer Interaction

Goals

Generating design solutions

- Guidelines
- Principles
- Theories
- Design Patterns

Evaluating generated designs

- Expert reviews and heuristics
- Usability testing
- Controlled experiments

Evaluation

Testing the usability, functionality and acceptability of an interactive system

Goal

- Evaluation: «Evaluation tests the usability, functionality and acceptability of an interactive system»
  - According to the design stage (sketch, prototype, final)
  - According to the initial goals
  - Alongside the different usability dimensions
  - Using a range of different techniques
- Identify and correct issues as soon as possible

Usability

- **Usability**: how well users can use the system's functionality.

- Dimensions of usability:
  - **Usefulness**: does it do something people want?
  - **Learnability**: is it easy to learn?
  - **Memorability**: one learned, is it easy to remember?
  - **Effectiveness**: does it allow reaching the goal?
  - **Efficiency**: once learned, is it fast to use?
  - **Visibility**: is the state of the system visible?
  - **Errors**: are errors few and recoverable?
  - **Satisfaction**: is it enjoyable to use?

Functionality

- **Functionality**: the system's functionality must accord with the user's requirements and should enable users to perform their intended tasks.

- Functionality can be tested in different ways:
  - Are the appropriate functionality available within the system?
  - Are they clearly reachable by the user?
  - Do they match the the user's expectations?

- Functionality evaluation may also include measuring the user's performance with the system, to assess the effectiveness of the system in supporting the task.

Acceptability

- Technology **acceptability** is one's perception of a system before use, while technology **acceptance** is one's perception of the system after use.

- Good User Interface Design can make a product easy to understand and use, which results in greater user acceptance.

- Testing **acceptability** means evaluating the enjoyment and emotional response to a system, particularly in the case of systems that are aimed at leisure or entertainment.

- This may involve:
  - measuring satisfaction and comfort
  - identifying areas of the design that overload the user

Many Evaluation Approaches

- Evaluation may take place:
  - In the laboratory
  - In the field

Many Evaluation Approaches

- In lab studies, users are taken out of their normal work environment to take part in **controlled** tests. They are typically adopted in the early stages of design (e.g., to compare alternatives, you don't need a working implementation).

  👍 simulation of dangerous environments
  👍 suitable for specific tasks within a system
  👎 lack of context
  👎 unnatural situations leading to biases
  👎 not suitable for all the tasks

Many Evaluation Approaches

- Field studies takes the designer or evaluator out into the **user's work environment** in order to observe the system in action.
  - 👍 open nature: the "real" context
  - 👍 users are in their natural environment
  - 👎 low degree of control
  - 👎 higher costs (you need a working implementation)
  - 👎 longer duration

Many Evaluation Approaches

- Evaluation may be based on **expert evaluation:**
  - Analytic methods
  - Review methods
  - Model-based methods
  - Heuristics

- It is useful to identify any areas that are likely to cause difficulties because they violate known cognitive principles, or ignore accepted empirical results
  - 👍 it can be used at any stage in the development process
  - 👍 it is relatively cheap, since it does not require user involvement
  - 👎 it does not assess actual use of the system

Many Evaluation Approaches

- Evaluation may involve users:
  - o Experimental methods
  - o Observational methods
  - o Query methods
  - o Formal or semi-formal or informal

- In experimental and observational methods, the evaluator chooses a hypothesis to test, which can be determined by measuring some attribute of participant behavior.
  👍 they provide empirical evidence
  👎 they require more time to be designed and analyzed

- Query techniques (e.g., interviews) relies on asking the user about the interface directly
  👍 they are simple and cheap
  👎 you get subjective results

Many Evaluation Approaches

- We can also adopt automated evaluation:
  - Simulation and software measures
  - Formal evaluation with models and formulas
  - Especially for low-level issues

Heuristic Evaluation

Experts check potential issues on your design, by referring to a set of heuristic criteria

When Is Design Critique Useful?

- Before user testing
  - To save effort
  - Solving easy-to-solve problems
  - Leaving user testing for bigger issues

- Before redesigning
  - Identify the good parts (to be kept) and the bad ones (to be redesigned)

- To generate evidence for problems that are known (or suspected)
  - From 'murmurs' or 'impressions' to hard evidence

- Before release
  - Smoothing and polishing

Heuristic Evaluation

- A method developed by Jacob Nielsen (1994)
  - Structured design critique
  - Using a set of simple and general heuristics
  - Executed by a small group of experts (3-5)
  - Suitable for any stage of the design (sketches, UI, …)
  - Goal: find usability problems in a design

- Also popularized as "Discount Usability"

- A heuristic is a guideline or general principle or rule of thumb that can guide a design decision or be used to critique a decision that has already been made.

Basic Idea

- Define a set of heuristics (or principles)

- Give those heuristics to a group of experts
  - Each expert will use heuristics to look for problems in the design

- Experts work independently
  - Each expert will find different problems

- At the end, experts communicate and share their findings
  - Findings are analyzed, aggregated, ranked

- The discovered *violations* of the heuristics are used to fix problems or to re-design
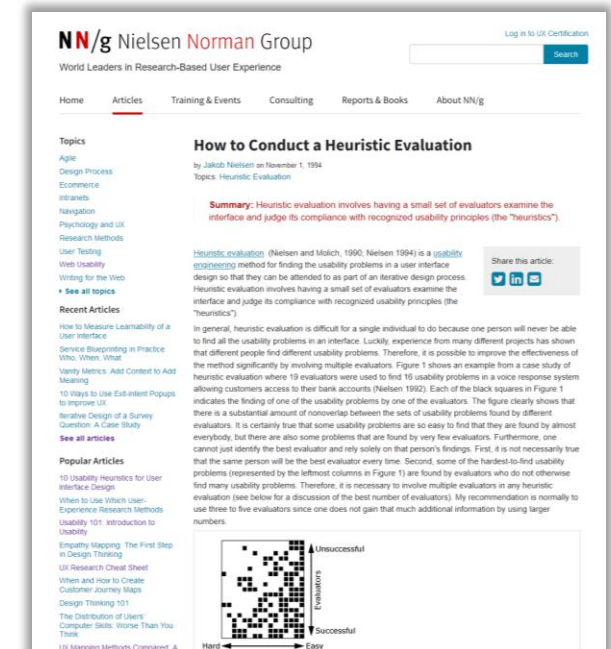
Heuristics

- Nielsen proposed 10 heuristic rules
  - Good at finding most design problems
  - Inspired and connected to the Design Principles (→Guidelines)

- In a specific context, application domain, or for specific design goals …
  - … new heuristics can be defined
  - … some heuristic can be ignored

Phases of Heuristic Evaluation

1. Pre-evaluation training
   - o Give evaluator information about the domain and the scenario to be evaluated

2. Evaluation
   - o Individual

3. Severity Rating
   - o First, individually
   - o Then, aggregate and find consensus

4. Debriefing
   - o Review with the design team

Evaluation (I)

- Define a set of tasks, that the evaluators should analyze

- For each task, the evaluator should step through the design several times, and inspect the UI elements
  - On the real design, or on a preliminary prototype

- At each step, check the design according to each of the heuristics
  - 1st step, get a general feeling for the interaction flow and general scope
  - 2nd step (and following), focus on specific UI elements, knowing where they fit in the general picture

- Heuristics are used as a "reminder" of things to look for
  - Other types of problems can also be reported

Evaluation (II)

- Comments from each evaluator should be recorded or written
  - There may be an observer, taking notes
  - The observer may provide clarifications, especially it the evaluator is not a domain expert

- Session duration is normally 1h – 2h

- Each evaluator should provide a list of usability problems
  - Which heuristic (or other usability rule) has been violated, and why
    - Not a subjective comment, but a reference to a known principle
  - Each problem reported separately, in detail

Evaluation (III)

- Where problems may be found
  - A single location in the UI
  - Two or more locations that need to be compared
  - Problem with the overall UI structure
  - Something is missing
    - May be due to prototype approximation
    - May still be unimplemented

Multiple Evaluators

- No evaluator finds all problems
  - Even the best one finds only ~1/3
- Different evaluators find different problems
  - Substantial amount of nonoverlap
- Some evaluators find more problems than others
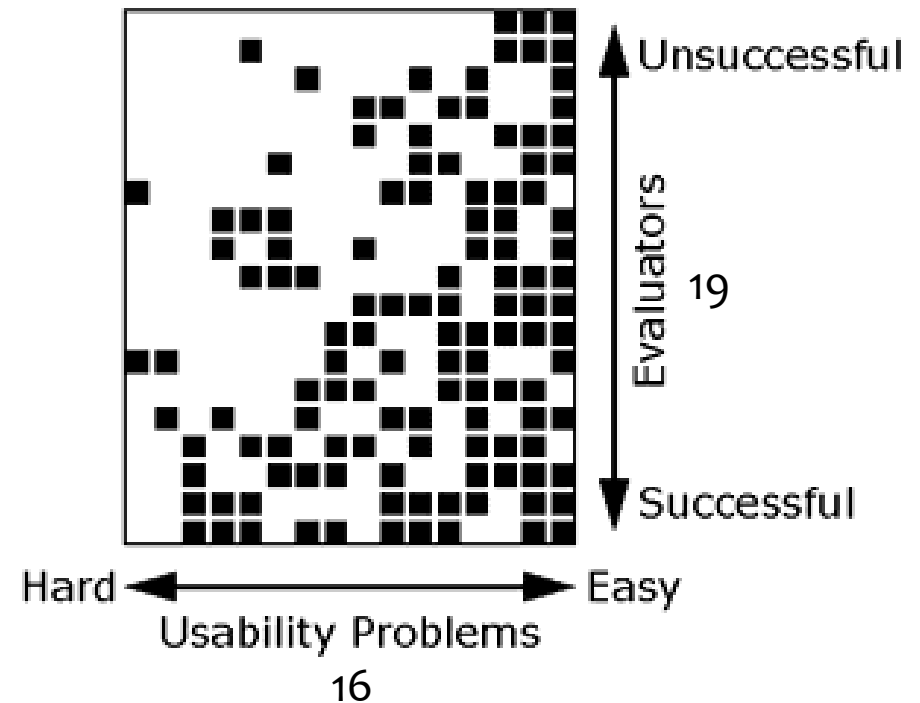
How Many Evaluators?

$$\frac{PF(i)}{N}$$

i
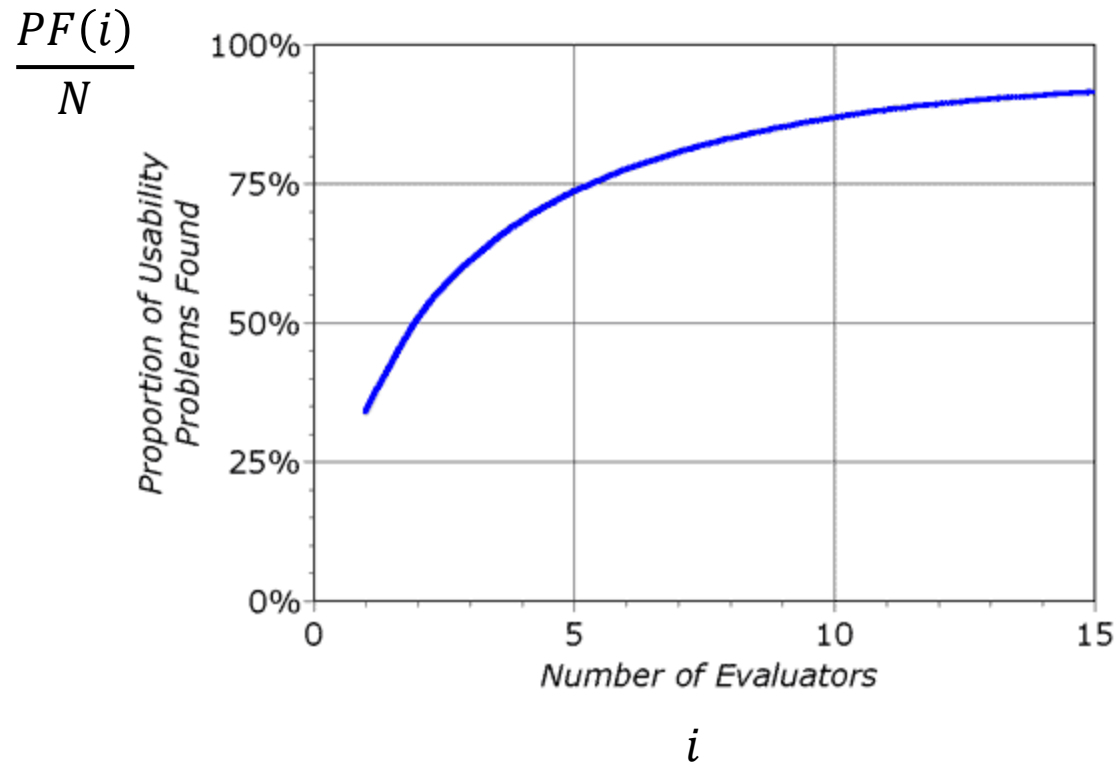
- $PF(i) = N\big(1 - (1 - l)^i\big)$

- $PF(i)$: problems found

- i: number of *independent* evaluators

- N: number of existing (but unknown) usability problems

- l: ratio of usability problems found by a single evaluator

How Many Evaluators?

$$Cost(i) = \text{Fixed} + \text{Fee} \times i$$

$$\dfrac{PF(i)}{N}$$

$$\dfrac{PF(i)/N}{Cost(i)}$$

3-5
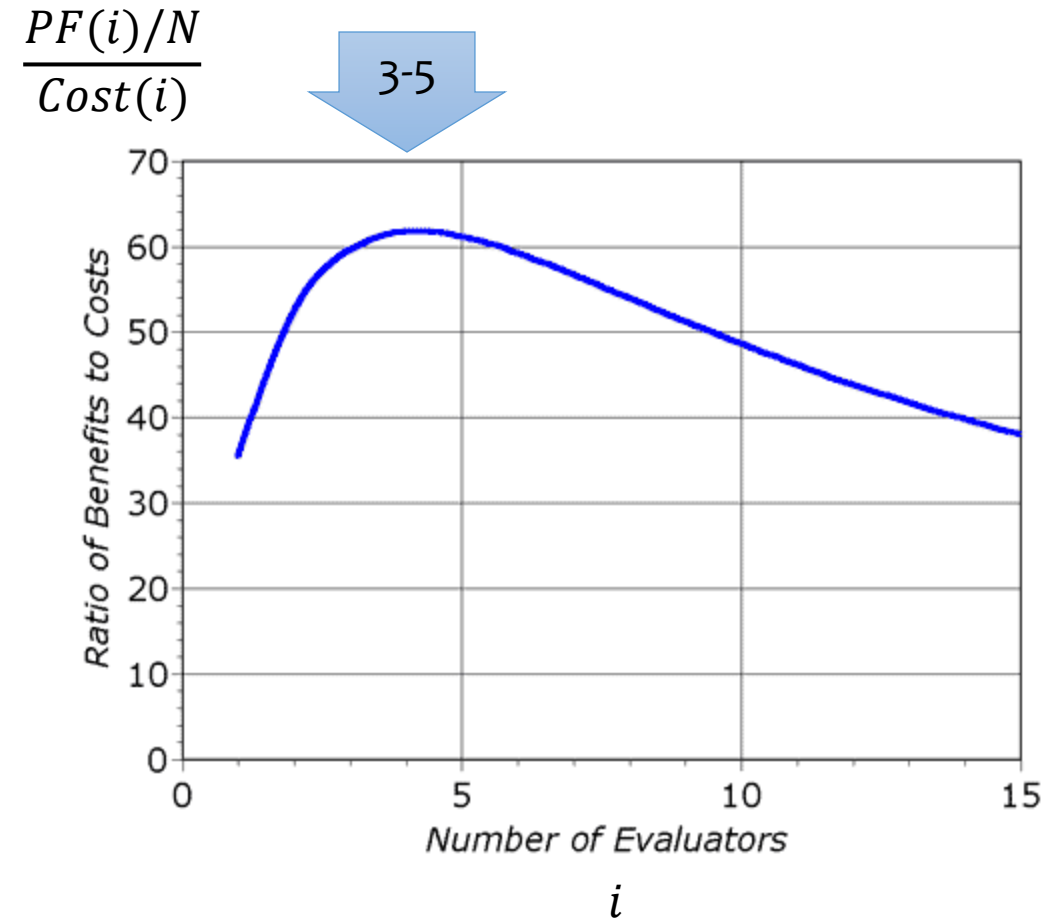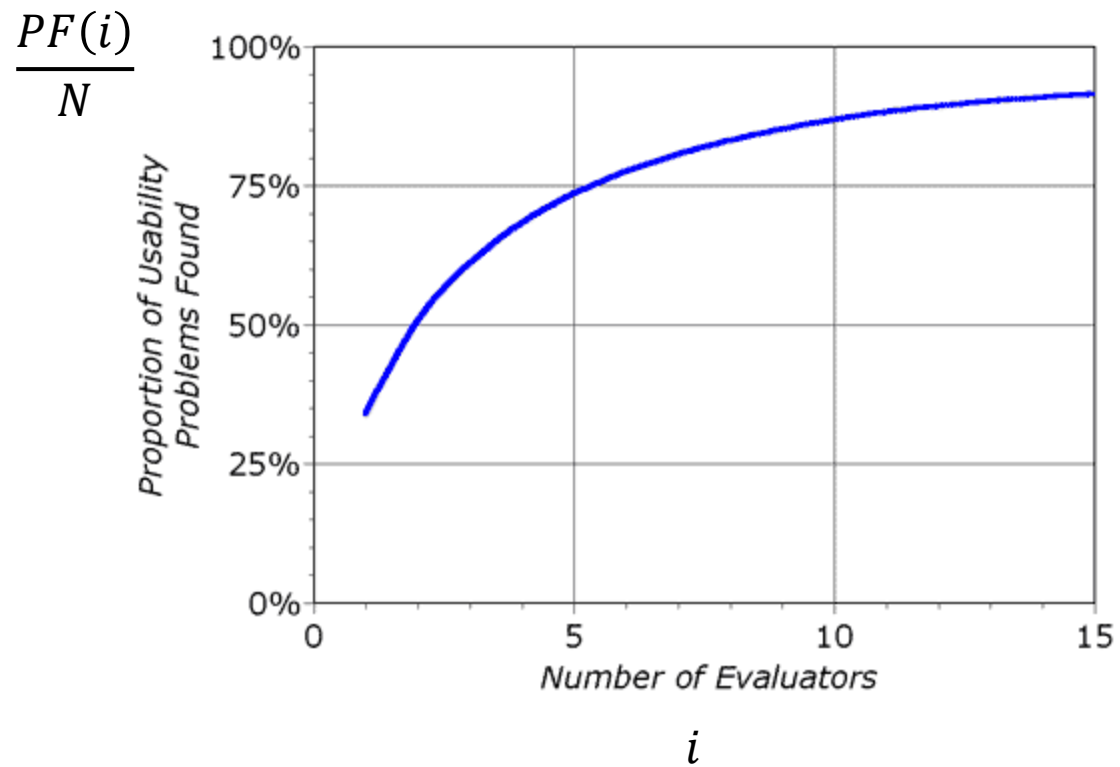
Severity Rating

- We need to allocate the most resources to fix the most serious problems

- We need to understand if additional usability efforts are required

- **Severity** is a combination of:
  o **Frequency** with which the problem occurs: common or rare?
  o **Impact** of the problem if it occurs: easy to overcome or difficult?
  o **Persistence,** is it one-time or will it occur many times to users?

- Define a *combined severity rating*
  o Individually, for each evaluator
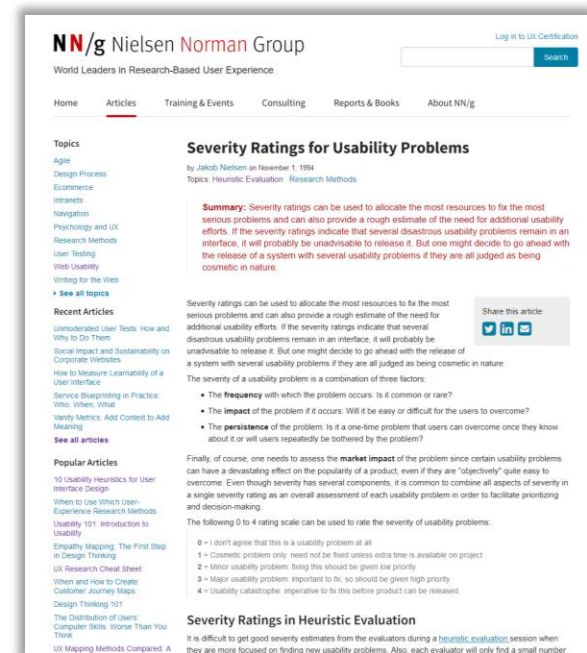
Severity Ratings scale

0	No problem	I don't agree that this is a usability problem at all
1	**Cosmetic** problem only	need not be fixed unless extra time is available on project
2	**Minor** usability problem	fixing this should be given low priority
3	**Major** usability problem	important to fix, so should be given high priority
4	Usability **catastrophe**	imperative to fix this before product can be released

Frequency

Persistence ⟷ Impact
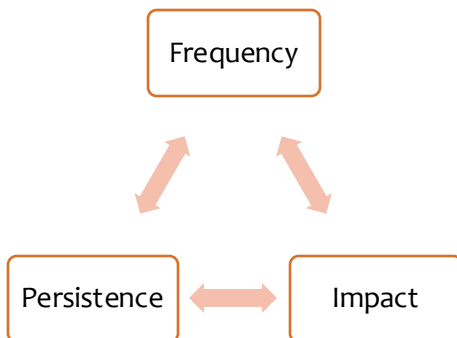
Combined Severity Ratings

- Severity ratings from *one* evaluator have been found *unreliable*, they should not be used

- After all evaluators completed their rankings
  - Either let them discuss, and agree on a consensus ranking
  - Or just compute the average of the 3-5 ratings

Debriefing

- *Meeting of all evaluators, with observers, and members of the* development team

- Line-by-line analysis of the problems identified
  - o Discussion: how can we fix it?
  - o Discussion: how much will it cost to fix it?

- Can also be used to brainstorm general design ideas

Heuristic Evaluation vs. User Testing

Heuristic Evaluation

- Faster (1-2h per evaluator)

- Results are pre-interpreted (thanks to the evaluators)

- Could generate *false positives*

- Might *miss* some problems

User Testing

- Need to develop software, and prepare the set-up

- More accurate (by definition!)
  - Actual users and tasks

- *... more on this later in the course!*
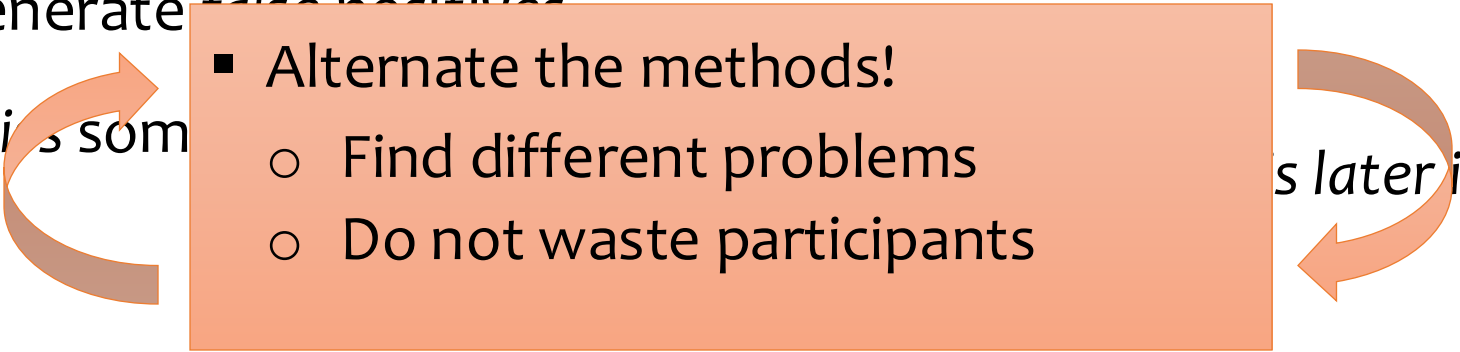
Heuristic Evaluation vs. User Testing

Heuristic Evaluation

- Faster (1-2h per evaluator)

- Results are pre-interpreted (thanks to the evaluators)

- Could generate *false positives*

- Might *miss* som

User Testing

- Need to develop software, and prepare the set-up

- More accurate (by definition!)
  - Actual users and tasks

s later in the course!

- Alternate the methods!
  - Find different problems
  - Do not waste participants

https://www.nngroup.com/articles/usability-problems-found-by-heuristic-evaluation/
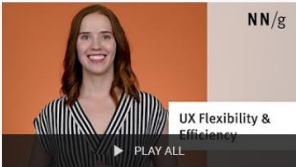
Nielsen's Usability Heuristics

10 Usability Principles to be used in Heuristic Evaluation

10 Nielsen's Usability Heuristics

https://www.youtube.com/playlist?list=PLJOFJ3Ok_idtb2YeifXlG1-TYoMBLoG6I

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

10 Nielsen's Usability Heuristics

- #1: Visibility of system status

- #2: Match between system and the real world

- #3: User control and freedom

- #4: Consistency and standards

- #5: Error prevention

- #6: Recognition rather than recall

- #7: Flexibility and efficiency of use

- #8: Aesthetic and minimalist design

- #9: Help users recognize, diagnose, and recover from errors

- #10: Help and documentation

#1: Visibility of system status

- The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
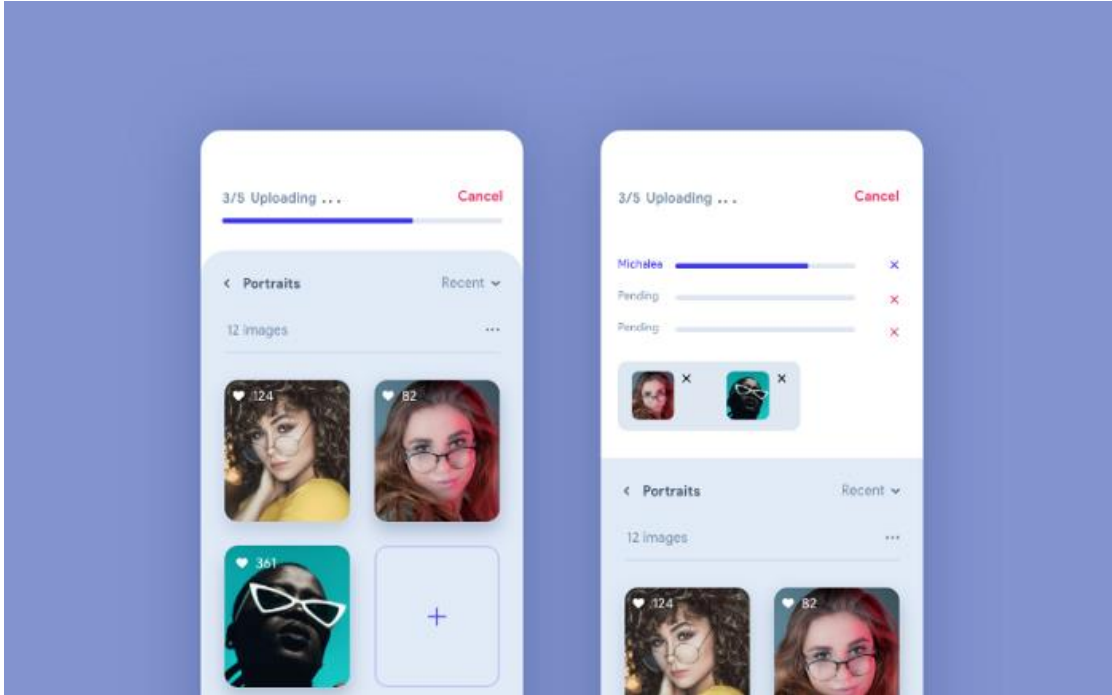
https://www.nngroup.com/articles/visibility-system-status/

#1: Visibility of system status

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

Which Feedback?

- Time
  - Execution time for tasks
- Space
  - E.g., occupation of cloud storage
- Change
  - Ensure that the user is aware of changes that he requested (e.g., save, delete, send, …)

- Action
  - What is happening (running, stopped, …), in a redundant way
- Next steps
  - What will happen because of your action, and your possible next actions at this point
- Completion
  - Clarify when a task has been finalized

Rule of Thumb (Time)

- If the execution time is…

- …Less than 1 second ⇨ just show the outcome of the action

- …Around 1-2 seconds ⇨ show feedback that the action is underway

- …More 2-3 seconds ⇨ show progress (percentage, estimated time, …)

#2: Match between system and the real world

- The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

- Use familiar metaphors and language

https://www.nngroup.com/articles/match-system-real-world/

#2: Match between system and the real world

#2: Match between system and the real world
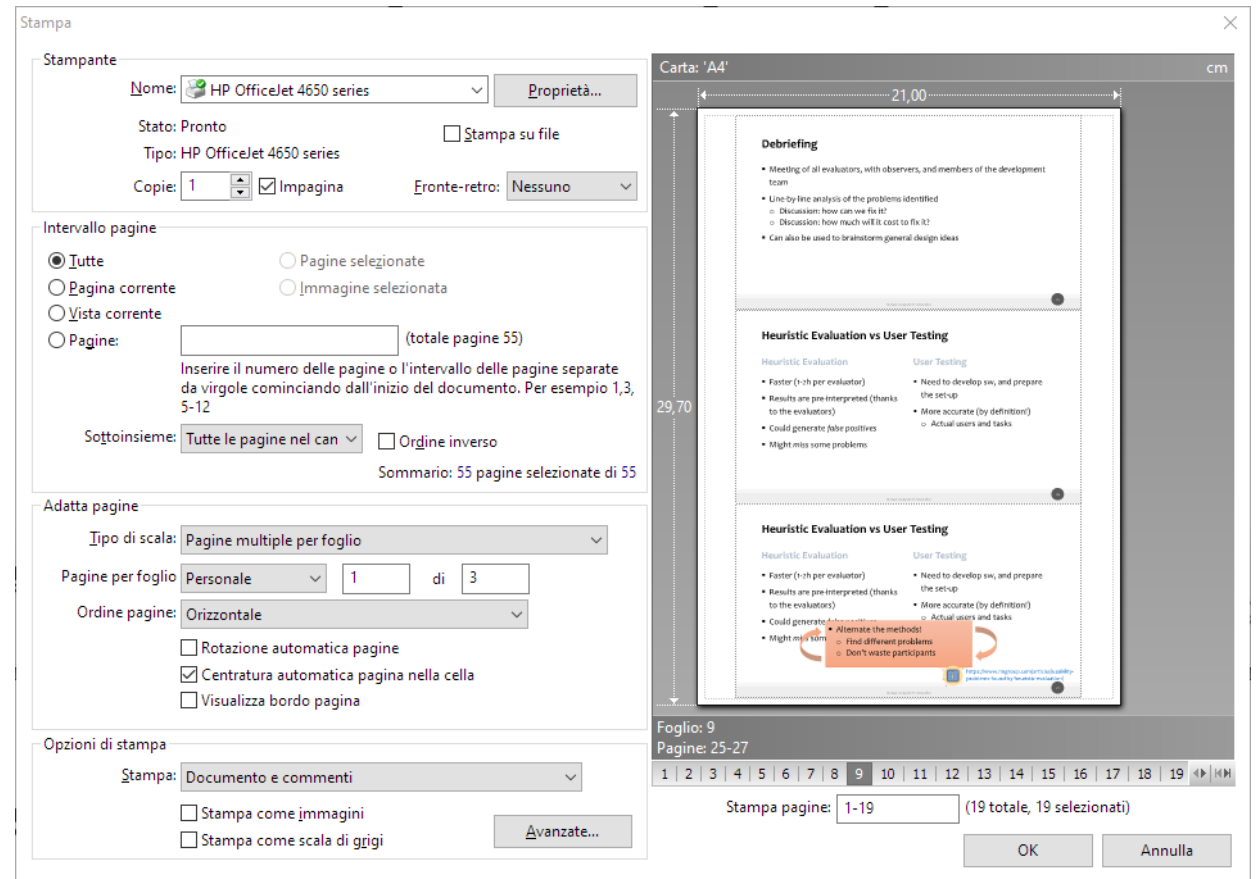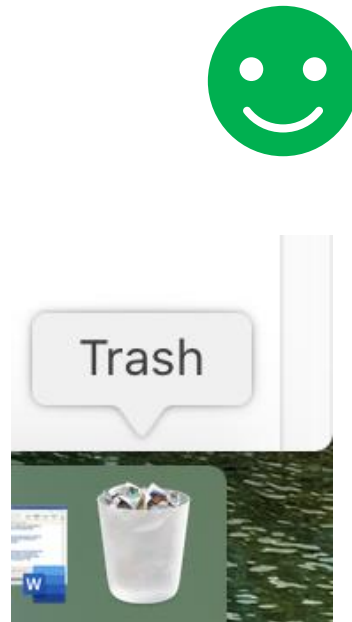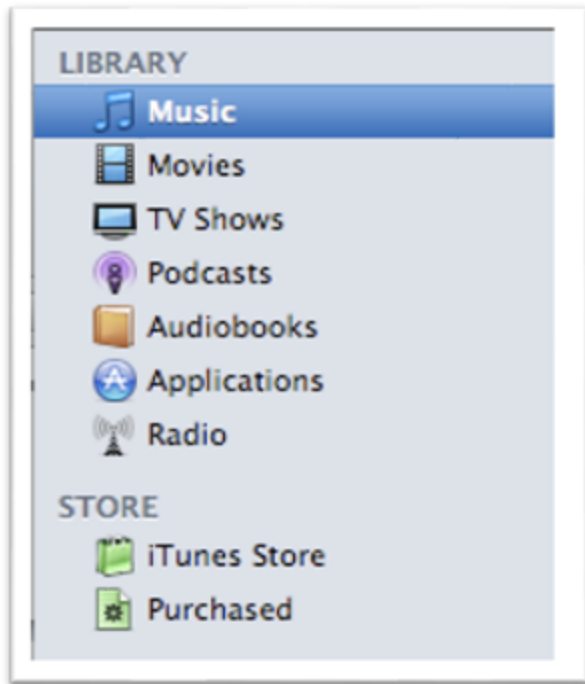
Exploit Familiarity

- Familiar Metaphors
  - Files, paper, folders, highlighters, …

- Familiar Language
  - Avoid jargon, acronyms, etc. that could be unknown to your users

- Familiar Categories

- Familiar Choices
  - E.g., explain the meaning of the error message (what happened, what are the consequences, what are the available options) in a simple way

#3: User control and freedom

- Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

#3: User control and freedom

#3: User control and freedom

Suggestions

- Always provide a "back" (or equivalent) button

- Allow users to "explore" different alternative paths
  - Except for one-shot wizard-like paths, aimed at novices or first-time users

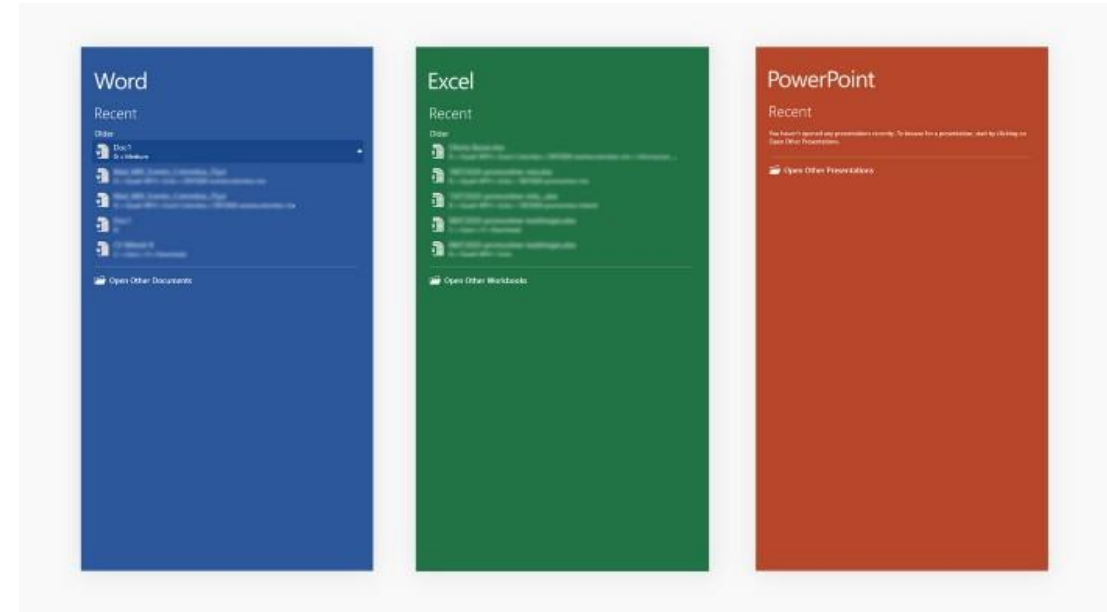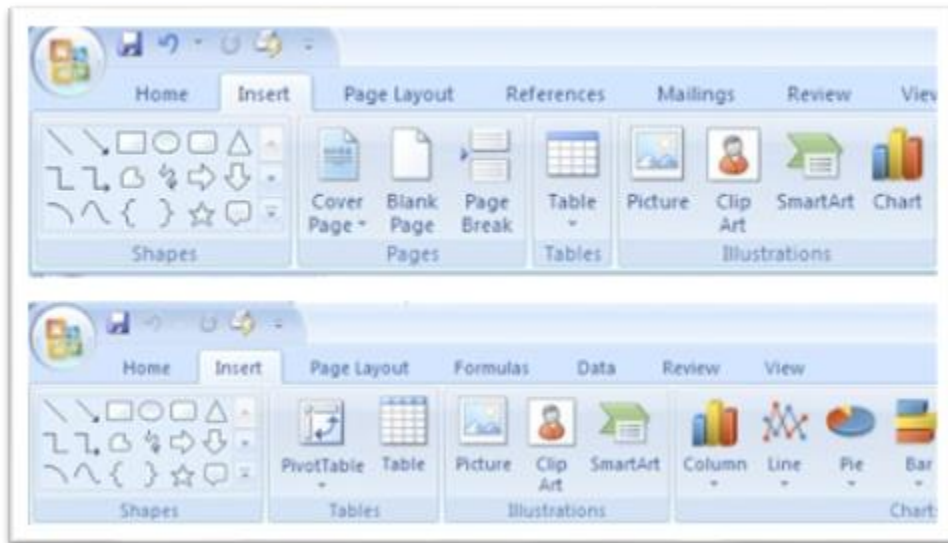# #4: Consistency and standards

- Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

#4: Consistency and standards
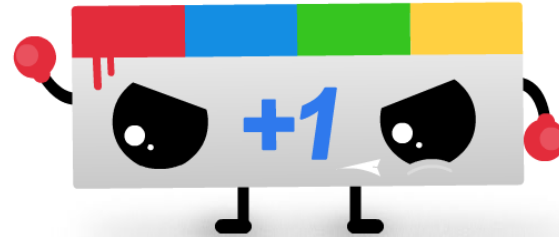
BrandFlakesforBreakfast's Illustration

#4: Consistency and standards

Suggestions

- Consistent layout for dialogs and forms
  - E.g., position of the navigation elements
  - E.g., position of the confirmation buttons

- Consistent meaning for Ok/Cancel, Yes/No choices
  - E.g., avoid: "Do you want to interrupt task?"
  - Still better, label buttons with the actual effect "Insert", "Interrupt", ...

- Categories, lists of names, geographical regions, etc, should be taken from "standard" vocabularies
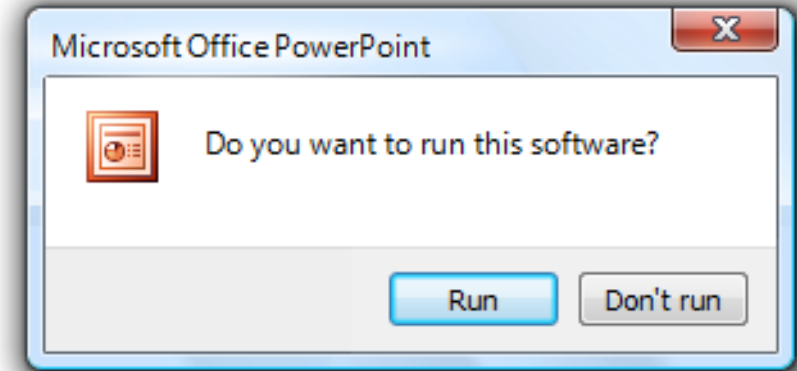
Examples
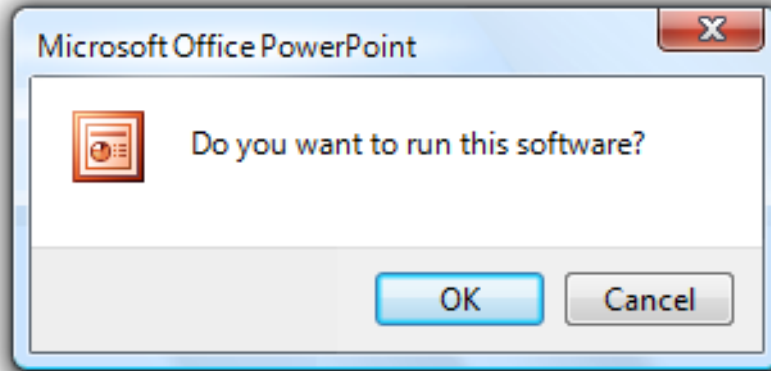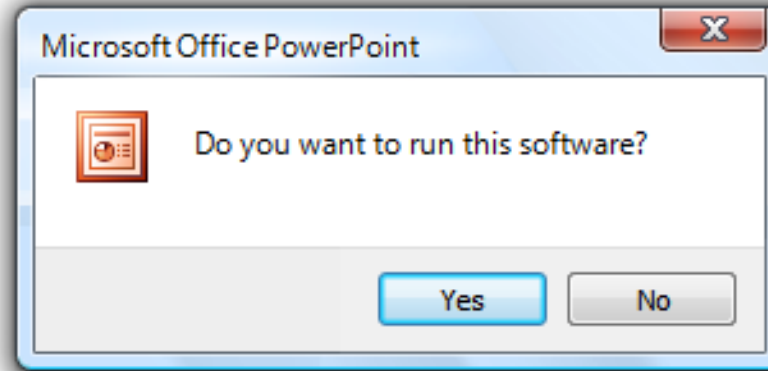
Bad   Acceptable   Better

source: https://docs.microsoft.com/en-us/windows/win32/uxguide/win-dialog-box

#5: Error prevention

- Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

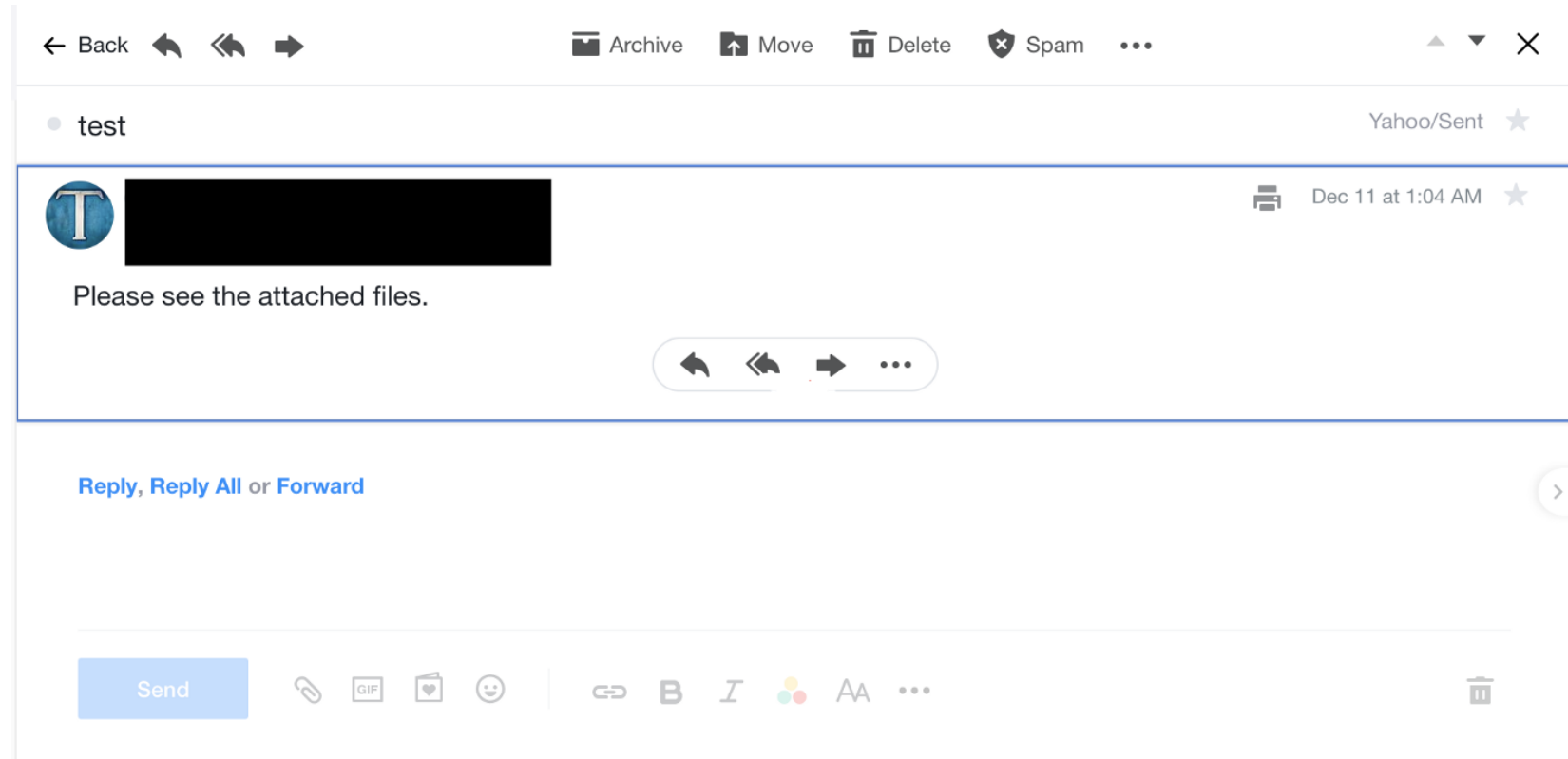https://www.nngroup.com/articles/slips/

Suggestions

- Preventing data loss

- Prevent clutter

- Prevent confusing flow

- Prevent bad input

- Prevent unnecessary constraints (e.g., provide defaults for missing data)

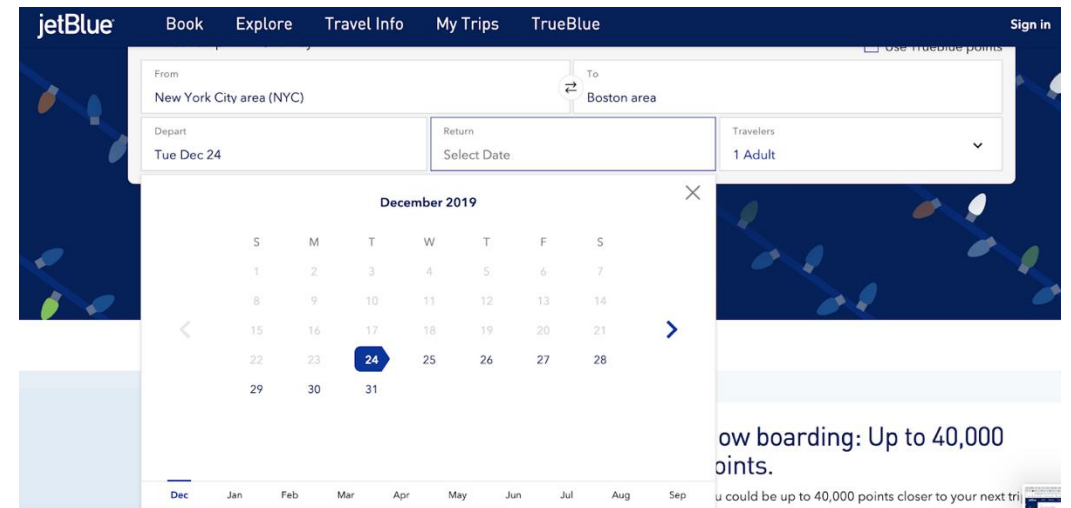# #5: Error prevention

# #5: Error prevention

#5: Error prevention

#6: Recognition rather than recall

- Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the interface to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
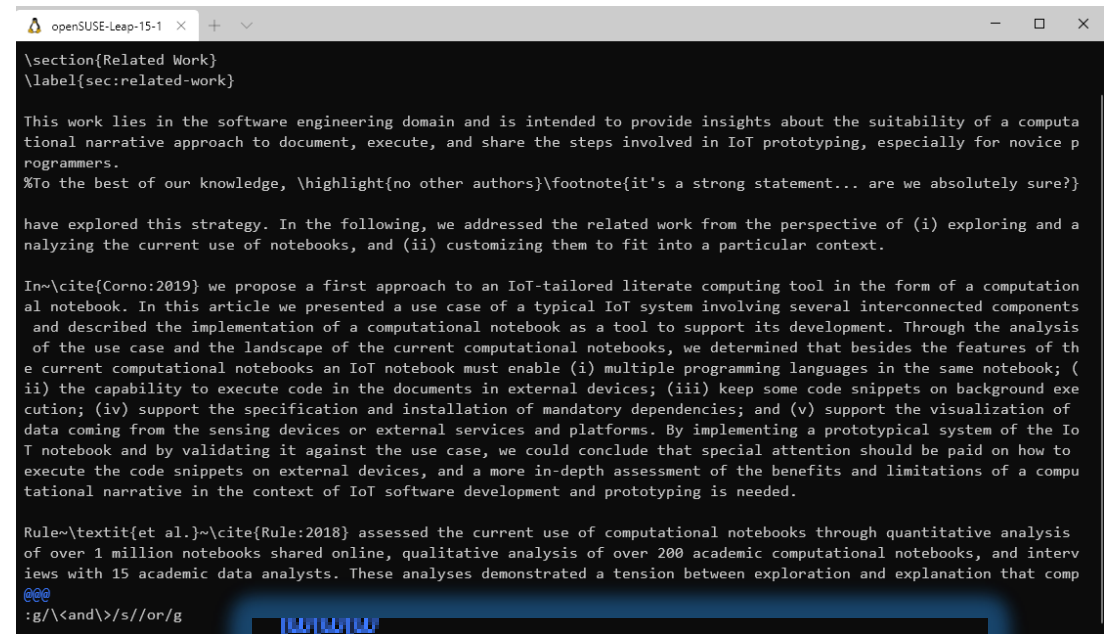
https://www.nngroup.com/articles/recognition-and-recall/

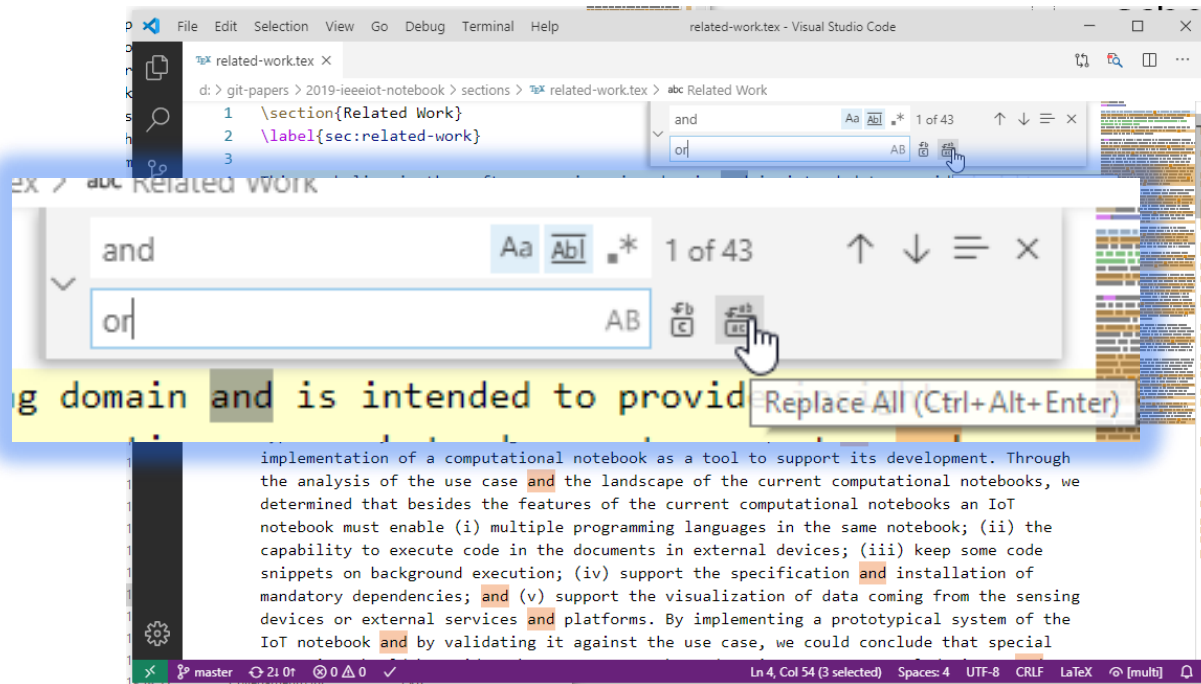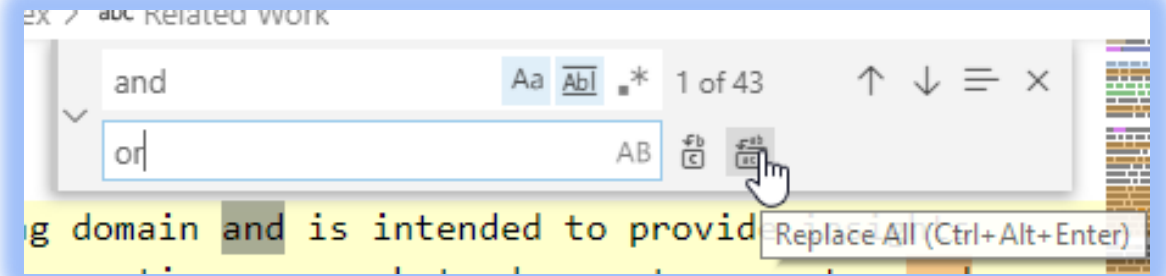# #6: Recognition rather than recall
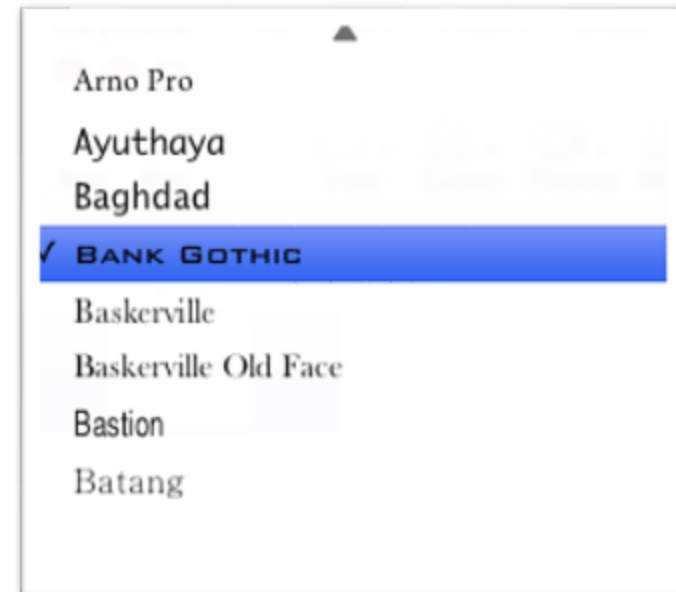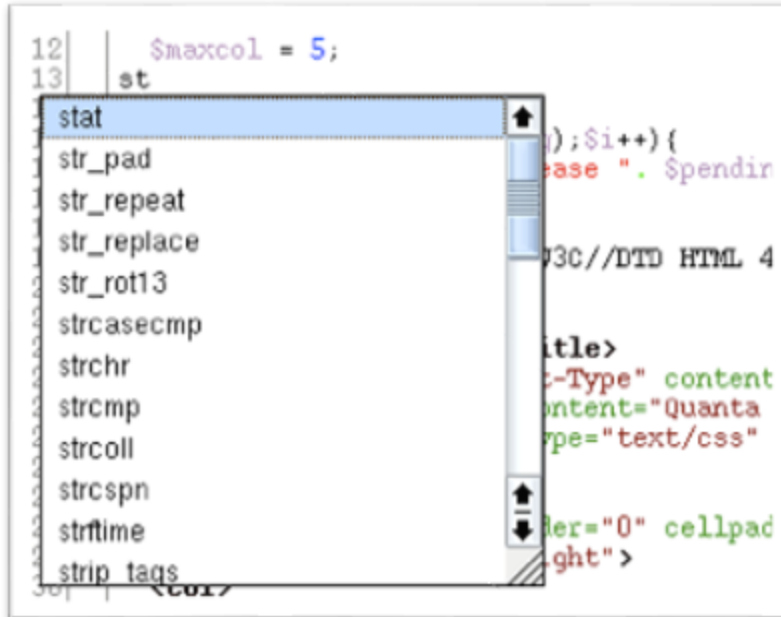
Example

Suggestions

- Avoid codes (use explicit names)
  - e.g., L, VL, EL, EA, …???

- Avoid extra hurdles
  - e.g., asking for unnecessary (or premature) information

- Provide previews
  - Code completion
  - Page preview
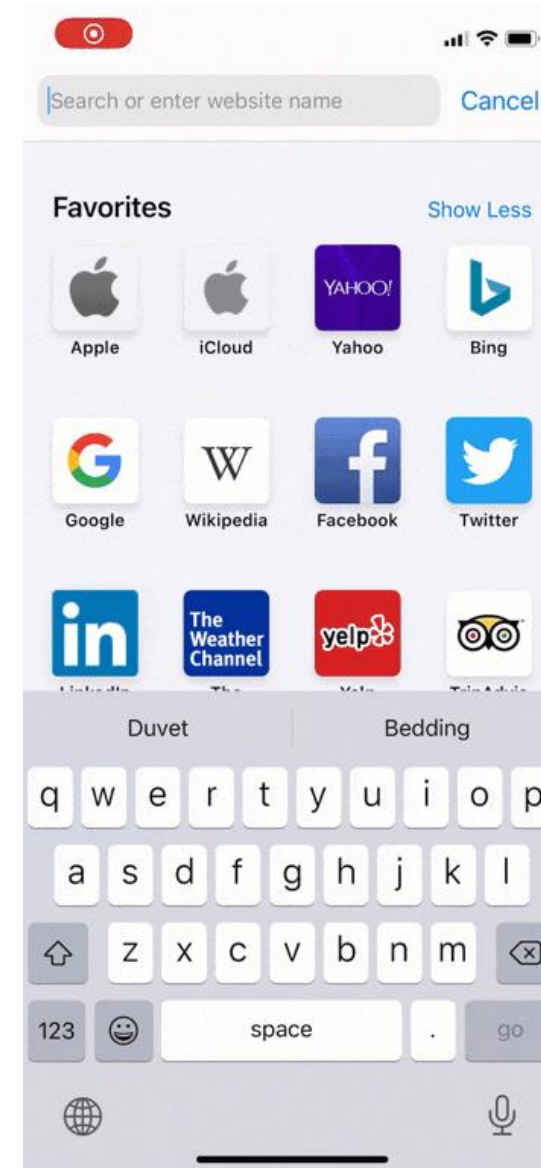 - Order summary
  - Itinerary
  - …

#7: Flexibility and efficiency of use

- Accelerators — unseen by the novice user — may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

#7: Flexibility and efficiency of use

Suggestions

- Flexibility = Default + Options
  - E.g., present some popular choices, but let the user enter a custom one (train ticket machines)

- Exploit background information for providing more information
  - E.g., weather forecasts in a calendar interface

- Novice and Expert Users Have Different Needs
  - Support proactivity, personalization, and different interaction techniques!

- Recommendations

- Provide relevant information, only

#8: Aesthetic and minimalist design

- Interfaces should not contain information which is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility.

#8: Aesthetic and minimalist design

#8: Aesthetic and minimalist design

Suggestions

- Key information must be "above the fold"
  - Especially on low-resolution devices

- Keep high signal-to-noise ratio
  - Colors, fonts, backgrounds, animations, …
  - Borders, dividers, …

- Minimalistic login experience

- Accept redundant ways of entering information

- Prune features that are outside the "core" functionality

#9: Help users recognize, diagnose, and recover from errors

- Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

#9: Help users recognize, diagnose, and recover from errors
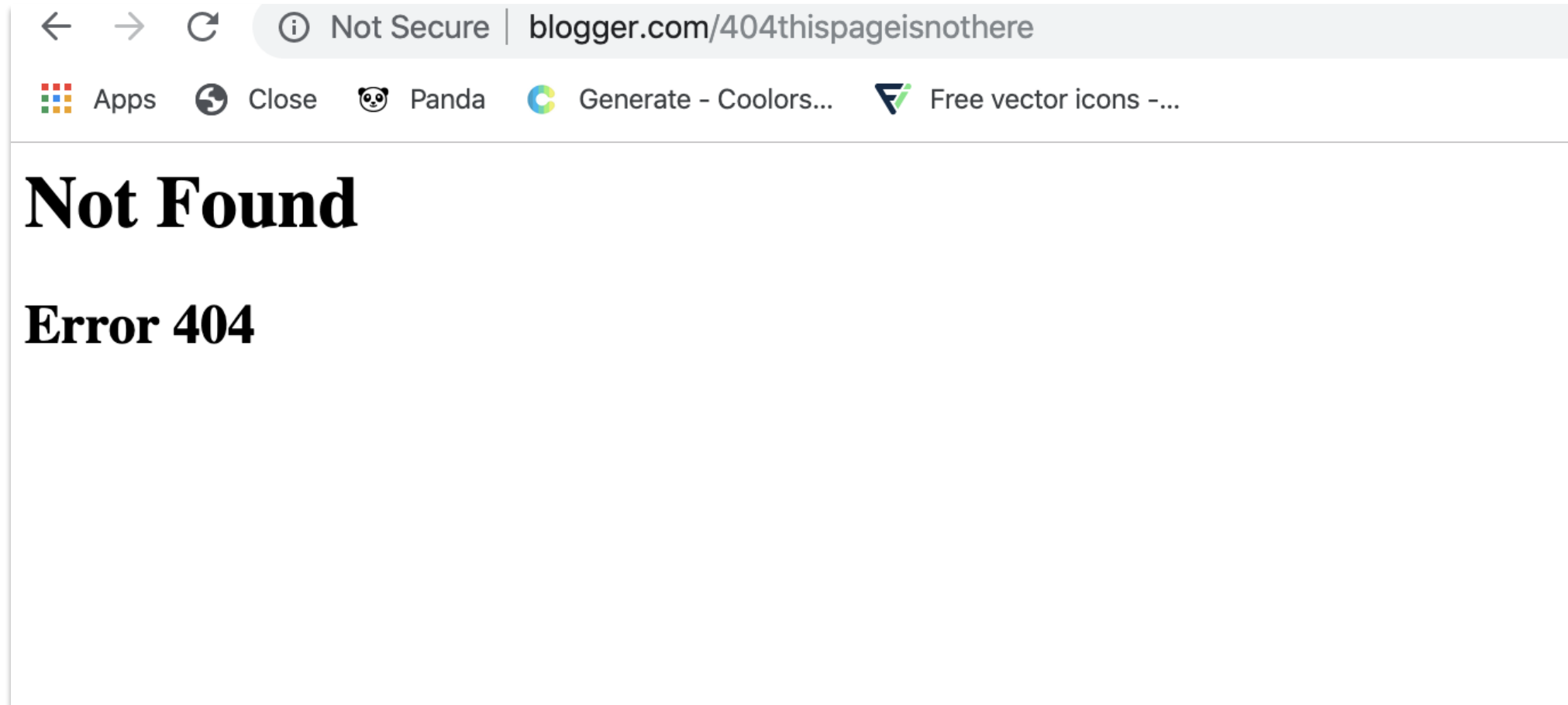
#9: Help users recognize, diagnose, and recover from errors
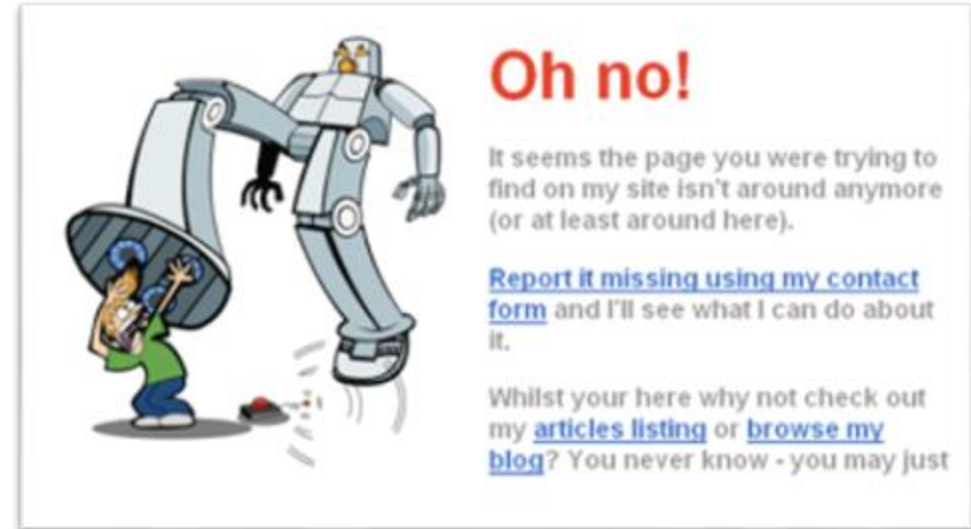
Suggestions

- Make errors easy to identify
  - Colors, fonts, …

- Make problem clear
  - Problem cause
  - Problem location

- Provide a solution
  - Give a suggestion
  - Show a path forward
  - Propose an alternative

#10: Help and documentation

- Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

#10: Help and documentation
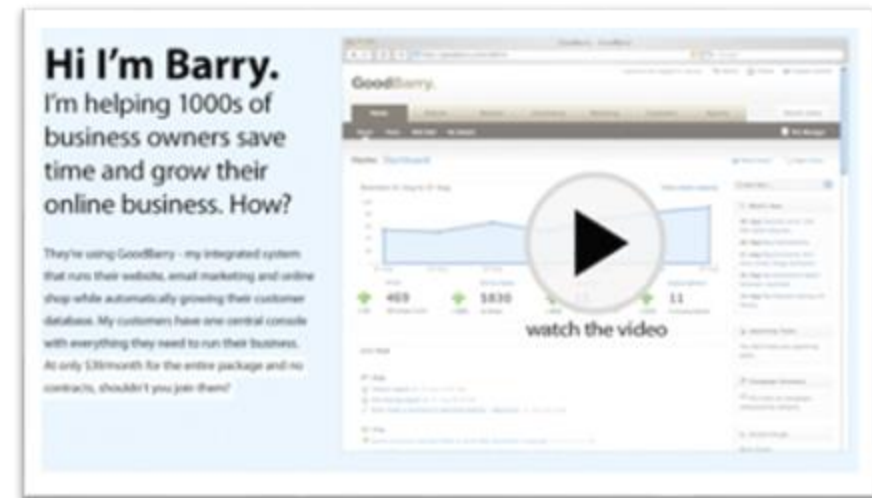
#10: Help and documentation
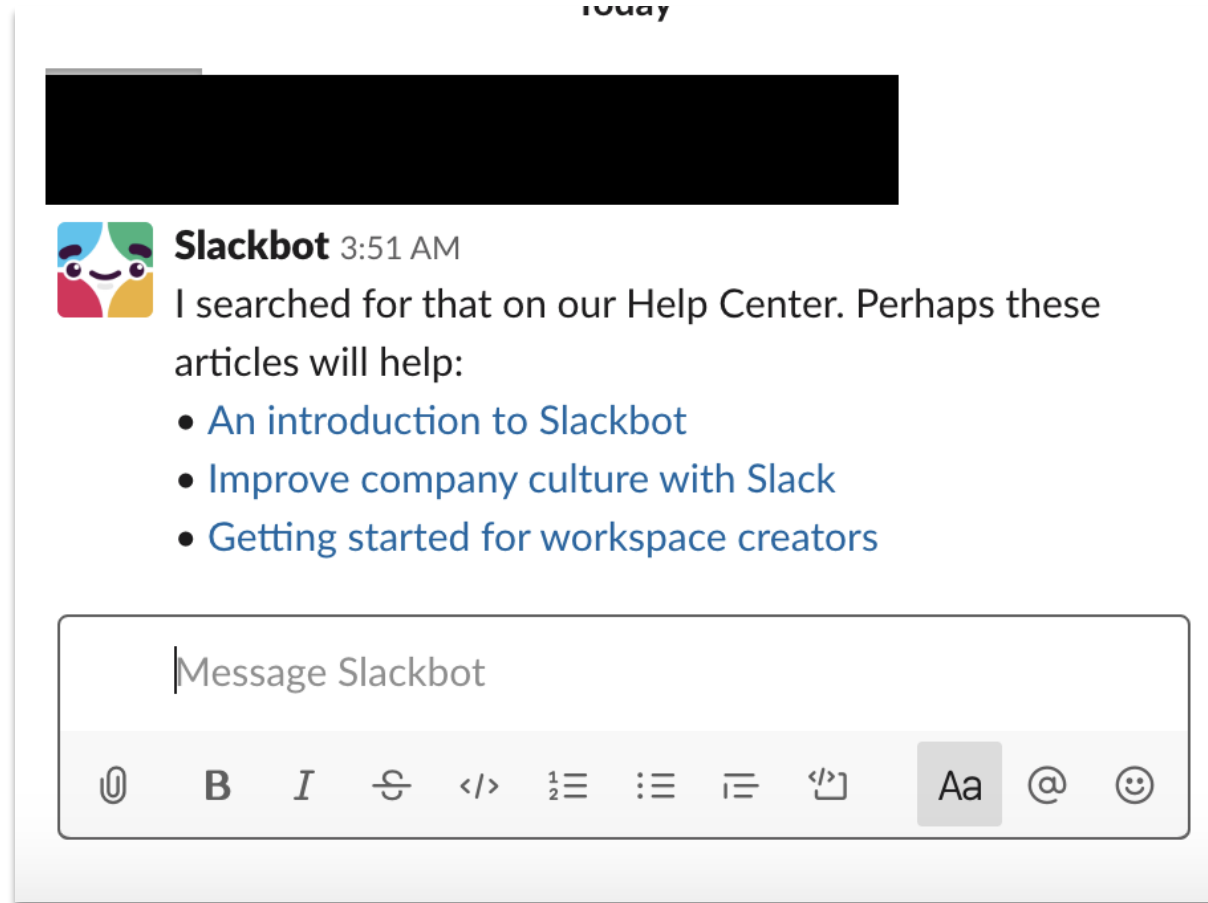
Suggestions

- Provide examples
  - In documentation
  - In complex choices

- Help the user understanding the error gravity
  - E.g., printing outside margins

- Provide 'tips' for showing new actions or steps

- Use pop-overs to point to changes in UI (or for first usage)

- Avoid too-opaque "terms and conditions" (summarize, if possible)

Example

- Target website: https://trenitalia.com/
  - Trenitalia is the primary train operator in Italy
  - It offers national rail transport with regional trains and high-speed trains ("Frecciarossa")

- Useful tasks to spot several problems:
  - Explore the offers proposed by the website and buy a discounted ticket
  - Buy a "Frecciarossa" round trip from Turin to Rome for the winter holidays
  - Chat with an operator for receiveing support

- In performing the tasks, you can register/login to the platform and change the language of the website, if you want

Example - Template

[Issue #]. [Heuristic #] [Heuristic Title]

- *Where: [Where the issue occurred – task, step, page]*
- *What: [Description of the problem]*
- *Why: [Reason why it violates the heuristic]*
- *Severity: [0-4, according to Nielsen's severity rating]*

1. H4 Consistency and standards

- *Where: Specify your language.*
- *What: The app uses "Save" for saving information, except here where it uses "Store".*
- *Why: It is an inconsistent terminology for the same function in the application, which can create confusion.*
- *Severity: 3*

Example – Violations (among many others!)

1. H1 Visibility of system status

- Where: navbar logo
- What: if I click on the logo in the navbar when I'm logged in, I'm redirected to another website and I'm automatically logged out from the system without any feedback
- Why: the user is automatically logged out from the system without any explanation nor feedback, so she might think she is still logged in.
- Severity: 3

Example – Violations (among many others!)

2. H2 Match between system and the real world

- Where: "offers" dropdown when the language is English

- What: the menu is using Italian words even if the language is set to English

- Why: the interface does not match the user's language.

- Severity: 2

SPECIAL OFFERS
Speciale Frecce
Insieme
Me&You
Bimbi Gratis
FrecciaYOUNG
FrecciaFAMILY
Famiglia
Young
Senior

Example – Violations (among many others!)

3. H3 User control and freedom
- Where: search procedure
- What: If the user selects by mistake a train, there is no easy way to modify the trip
- Why: users have to restart the search procedure if they make some mistakes
- Severity: 2

4. H3 User control and freedom
- Where: search procedure
- What: If the user starts searching as a "guest" and then wants to log-in, she is redirected to another page
- Why: the user is not in control, as she would probably like to continue the search and eventually buy the tickets
- Severity: 2

Example – Violations (among many others!)

5. H3 User control and freedom

- Where: breadcrumb menus
- What: the breadcrumb menus do not reflect the interaction history of the user
- Why: the breadcrumb menus allow the user to navigate to pages she has never seen before (e.g., the "Offers" page), creating confusion as breadcrumb menus are typically not used in this way
- Severity: 2

Home › Offers › Insieme offer

Insieme Offer

Travel as a group, it's worth it!

Example – Violations (among many others!)

6. H4 Consistency and standards

- Where: search form
- What: the website is showing two different search forms depending on whether the user is logged in into the system or not
- Why: users may be confused in using two different forms for the same task
- Severity: 2

7. H4 Consistency and standards

- Where: Offers
- What: the website is showing different offers depending on the selected language
- Why: users browsing the website in English may miss some offers
- Severity: 3
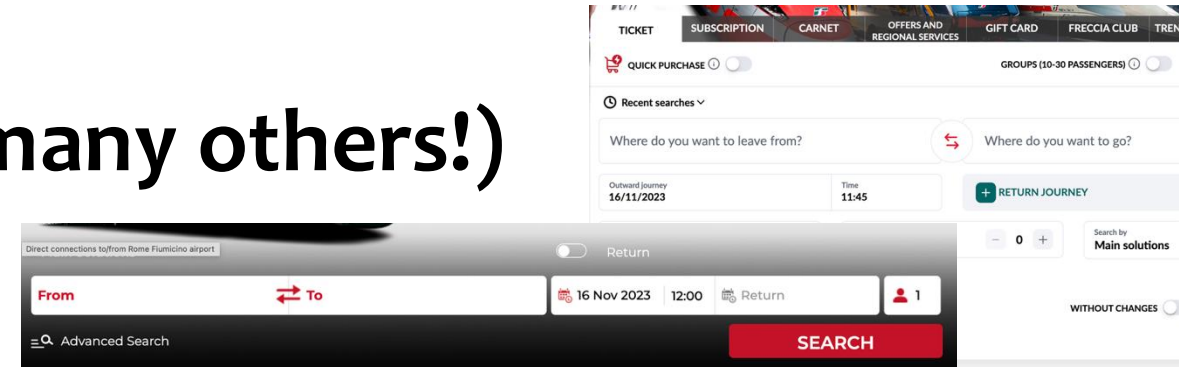
Example – Violations (among many others!)

8. H4 Consistency and standards

- Where: support/help
- What: the website uses "support" as a label to indicate the tools for receiving assistance, while the reserved area, when the user log in into the system, uses "help"
- Why: do these links refer to the same thing?
- Severity: 2

Example – Violations (among many others!)

9. H5 Error prevention

- Where: search form
- What: the user can click on search by writing whatever she wants in the "from" and "to" input fields (although there is then an error message and the form is not submitted)
- Why: the button search could be disabled
- Severity: 1

Example – Violations (among many others!)

10. H6 Recognition than recall
- Where: offers (at least some of them)
- What: when clicking on the button to buy a trip for a given offer, the user is redirected to the "traditional" search form
- Why: the user must remember the details of the offer (e.g., the date)
- Severity: 2

Example – Violations (among many others!)

11. H7 Flexibility and efficiency of use

- Where: search form
- What: there are no default/main options nor placeholders in the "form" and "to" input fields, even when the user clicks on them
- Why: users may not know what to write in the fields (the name of the city? The name of the station?). Default options may speed-up the process.
- Severity: 1

Example – Violations (among many others!)

12. H8 Aesthetic and minimalist design (same as violation #4)

- Where: search procedure
- What: If the user starts searching as a "guest" and then wants to log-in, she is redirected to another page
- Why: the website is not offering a minimalistic login experience
- Severity: 2

Example – Violations (among many others!)

13. H9 Help user recognize, diagnose, and recover from errors

- Where: search procedure
- What: the error message that appears when the user clicks on "Search" with some wrong inputs is not linked to the input fields it is referring to.
- Why: users may not know/easily understand what are the input fields that are problematic
- Severity: 1
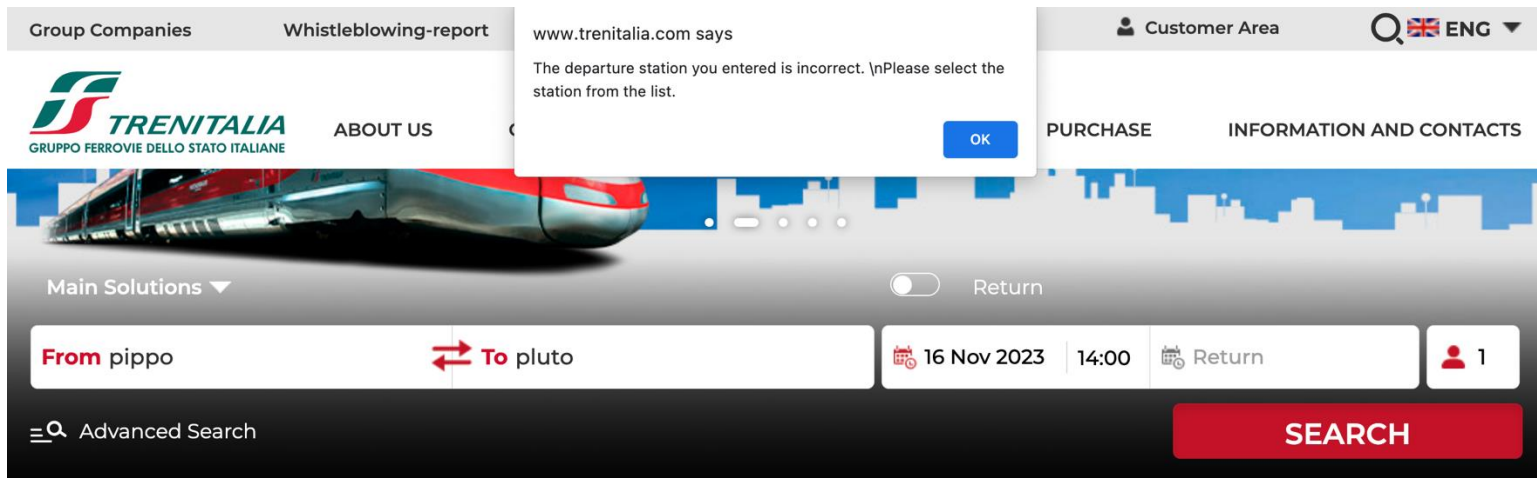
Example – Violations (among many others!)

14. H10 Help and documentation

- Where: chat support
- What: the user has to insert all her personal information (email, name, Trenitalia code, …) to chat with an operator even if she is already authenticated
- Why: chatting with an operator to receive support is not straightforward
- Severity: 2

References and Acknowledgment

- Alan Dix, Janet Finlay, Gregory Abowd, Russell Beale: Human Computer Interaction, 3 rd Edition
  - o Chapter 9: Evaluation Techniques

- Ben Shneiderman, Catherine Plaisant, Maxine S. Cohen, Steven M. Jacobs, and Niklas Elmqvist, Designing the User Interface: Strategies for Effective Human-Computer Interaction
  - o Chapter 5: Evaluation and the User Experience

- COGS120/CSE170: Human-Computer Interaction Design, videos by Scott Klemmer, https://www.youtube.com/playlist?list=PLLssT5z_DsK_nusHL_Mjt87THSTlgrsyJ

- Most of the slides are adapted from those used in the "Human Computer Interaction" course of Politecnico di Torino
  - o http://bit.ly/polito-hci

License

- These slides are distributed under a Creative Commons license "**Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)**"

- **You are free to:**
  - **Share** — copy and redistribute the material in any medium or format
  - **Adapt** — remix, transform, and build upon the material
  - The licensor cannot revoke these freedoms as long as you follow the license terms.

- **Under the following terms:**
  - **Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
  - **NonCommercial** — You may not use the material for commercial purposes.
  - **ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
  - **No additional restrictions** — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

- https://creativecommons.org/licenses/by-nc-sa/4.0/