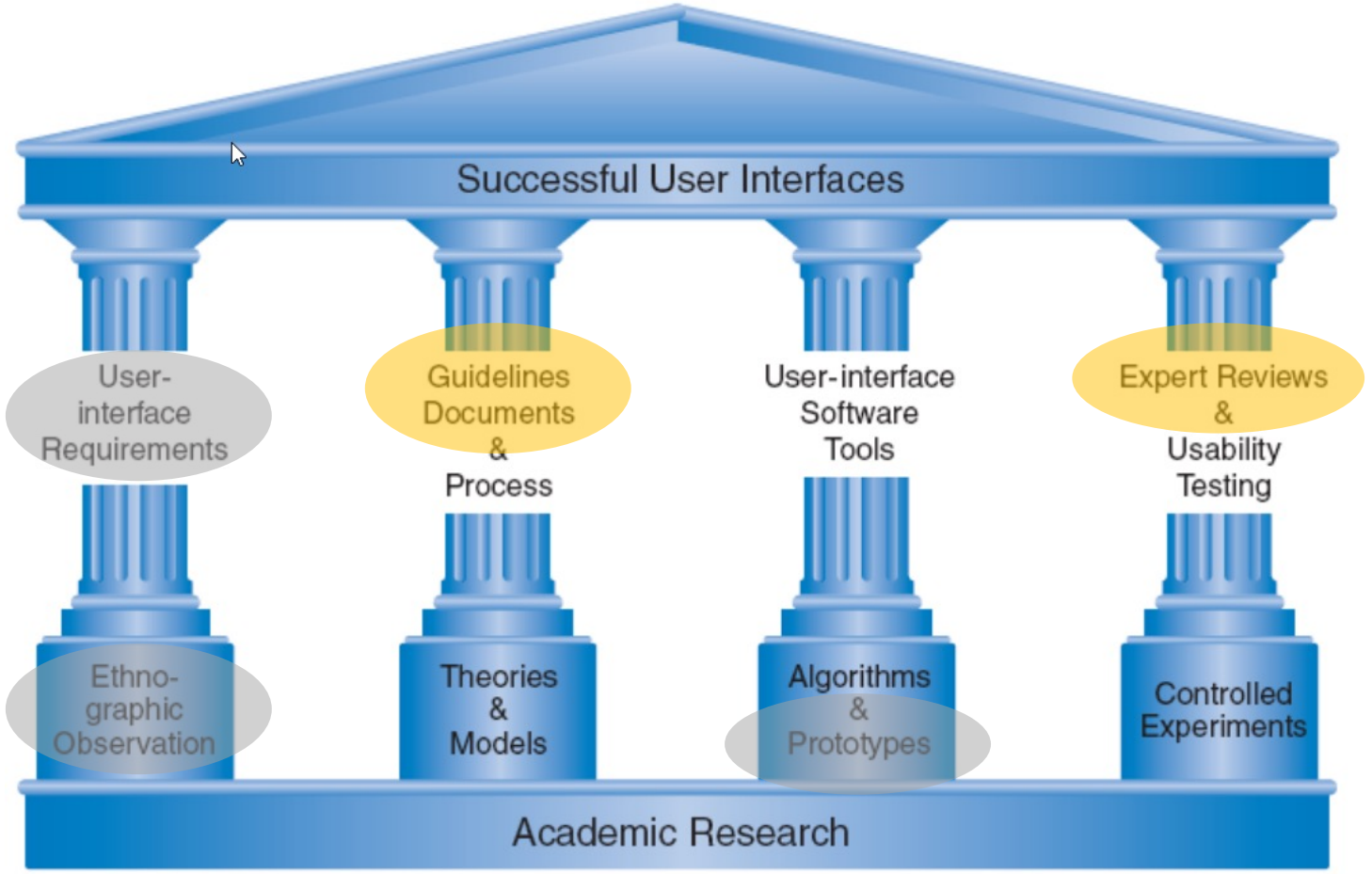
Evaluation: Introduction and Heuristics

Human Computer Interaction

Luigi De Russis, Fulvio Corno

Academic Year 2021/2022

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

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

Many Evaluation Approaches

- Evaluation may take place:
  - In the laboratory
  - In the field
- Involving users (Empirical Evaluation):
  - Experimental methods
  - Observational methods
  - Query methods
  - Formal or semi-formal or informal
- Based on expert evaluation:
  - Analytic methods
  - Review methods
  - Model-based methods
  - Heuristics
- Automated evaluation:
  - Simulation and software measures
  - Formal evaluation with models and formulas
  - Especially for low-level issues

Cognitive Walkthrough

A simple technique to analyze all individual step in an interaction path

Cognitive Walkthrough

- Step-by-step revision of a sequence of actions (interaction steps) to perform a given task
- Evaluators examine each step, looking for possible problems
- Particularly suited for systems designed for learning-by-exploration

Walkthrough Organization

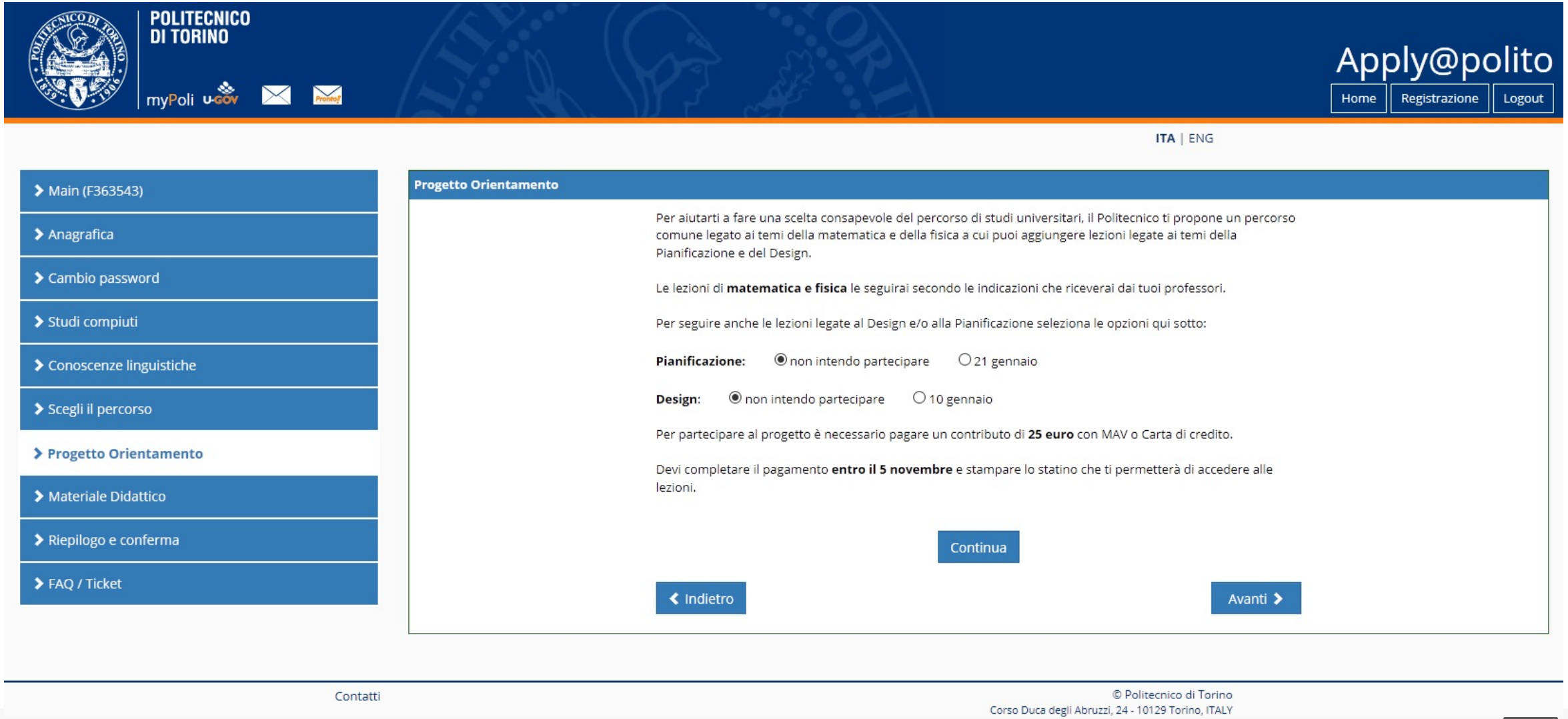
Walkthrough specification

- A specification or prototype of the system
- A description of the task the user is to perform on the system
- A complete, written list of the actions needed to complete the task
- An indication of who the users are (experience, knowledge)

For each step, you must check

- Is the *effect* of the action the same as the *user's goal* at that point?
- Will users *see* that the action is available?
- Once users have found the *correct action*, will they know it is the one they need?
- After the action is taken, will users *understand the feedback* they get?

Example


POLITECNICO DI TORINO
myPoli
Apply@polito
Home
Registrazione
Logout
ITA | ENG
Main (F363543)
Anagrafica
Cambio password
Studi compiuti
Conoscenze linguistiche
Scegli il percorso
Progetto Orientamento
Materiale Didattico
Riepilogo e conferma
FAQ / Ticket
Progetto Orientamento
Per aiutarti a fare una scelta consapevole del percorso di studi universitari, il Politecnico ti propone un percorso comune legato ai temi della matematica e della fisica a cui puoi aggiungere lezioni legate ai temi della Pianificazione e del Design.
Le lezioni di matematica e fisica le seguirai secondo le indicazioni che riceverai dai tuoi professori.
Per seguire anche le lezioni legate al Design e/o alla Pianificazione seleziona le opzioni qui sotto:
Pianificazione: non intendo partecipare 21 gennaio
Design: non intendo partecipare 10 gennaio
Per partecipare al progetto è necessario pagare un contributo di 25 euro con MAV o Carta di credito.
Devi completare il pagamento entro il 5 novembre e stampare lo statino che ti permetterà di accedere alle lezioni.
Continua
Indietro
Avanti
Contatti
© Politecnico di Torino
Corso Duca degli Abruzzi, 24 - 10129 Torino, ITALY

Heuristic Evaluation

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

When Is Design Critique Useful?

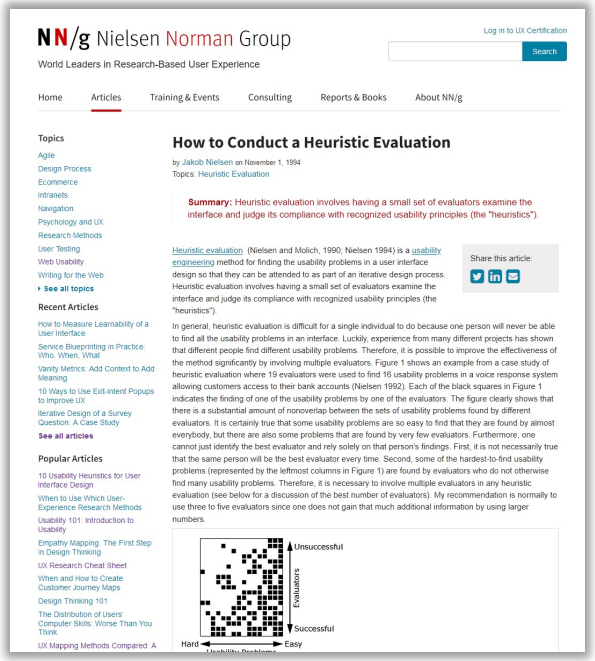
- Before user testing
  - To save effort
  - Solving easy-to-solve problems
  - Leaving user testing for bigger issues
- Before redesigning
  - Identify the good parts (to be kept) and the bad ones (to be redesigned)
- To generate evidence for problems that are known (or suspected)
  - From ‘murmurs’ 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”

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



https://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/

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
   - Give evaluator information about the domain and the scenario to be evaluated
2. Evaluation
   - Individual
3. Severity Rating
   - First, individually
   - Then, aggregate and find consensus
4. Debriefing
   - 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

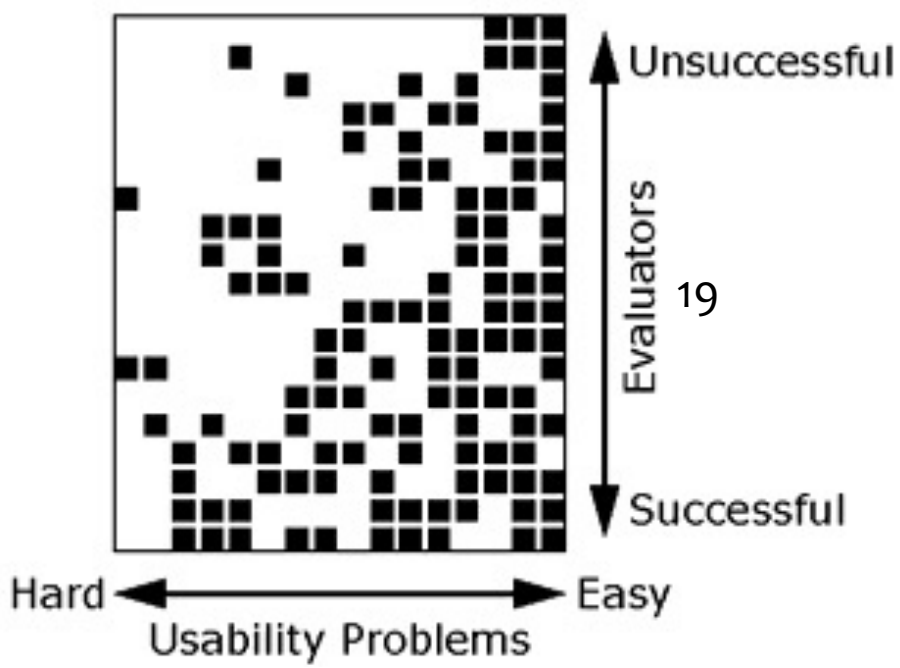
Evaluation (III)

- 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

https://www.nngroup.com/articles/usability-problems-found-by-heuristic-evaluation/

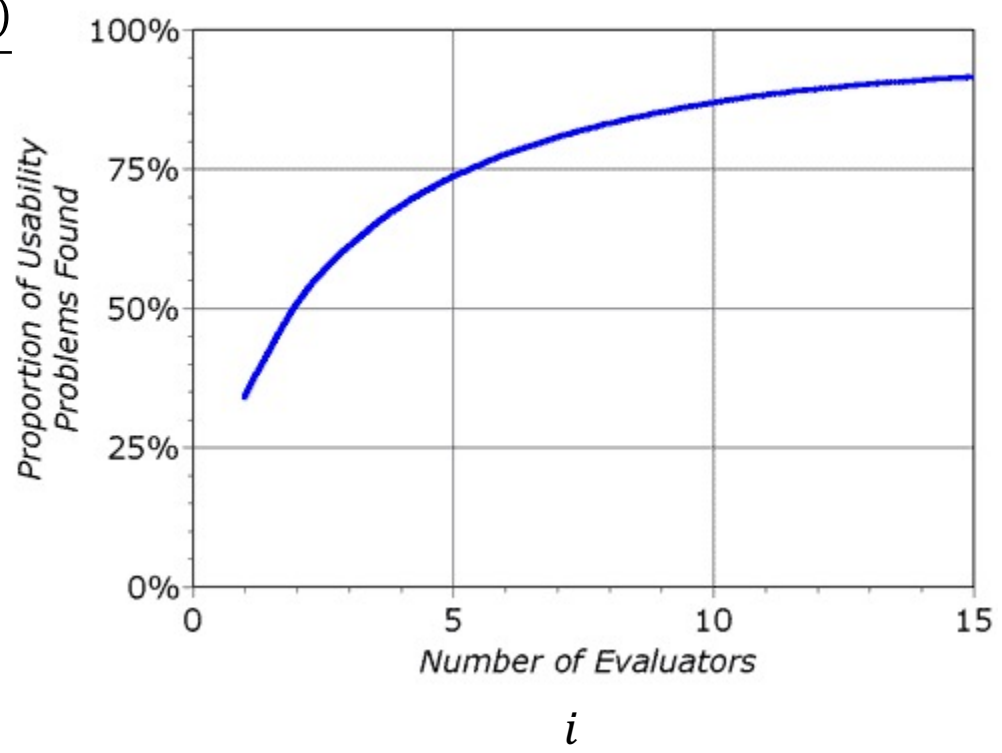
Multiple Evaluators

- 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?

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



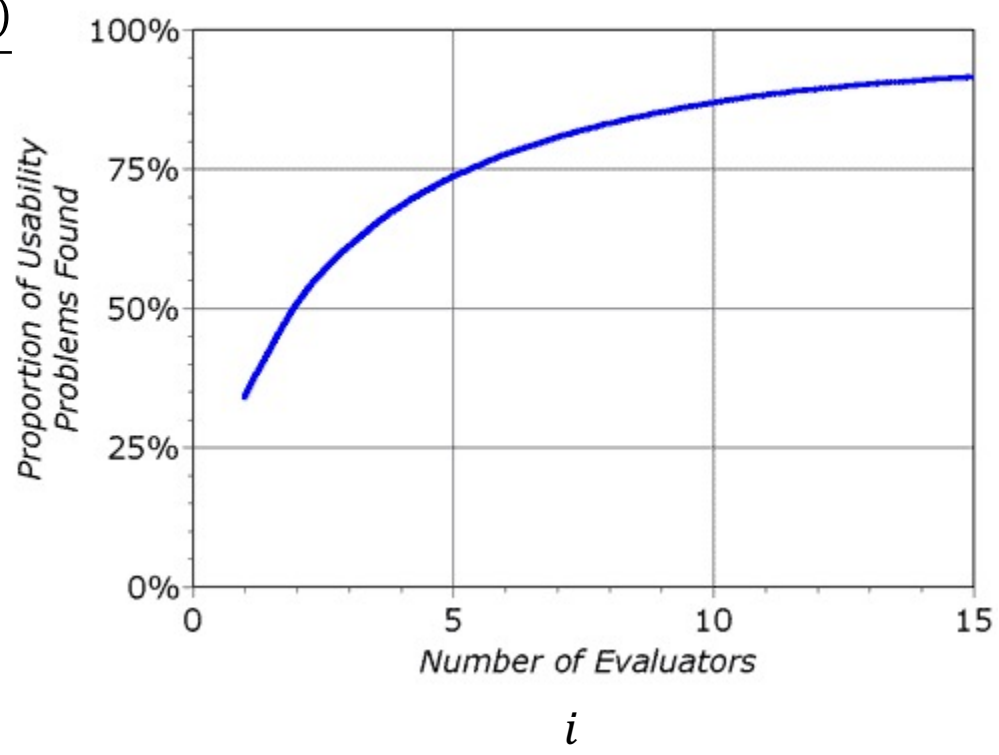
i

- $PF(i) = N\left(1 - (1 - l)^i\right)$
- $PF(i)$: 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?

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

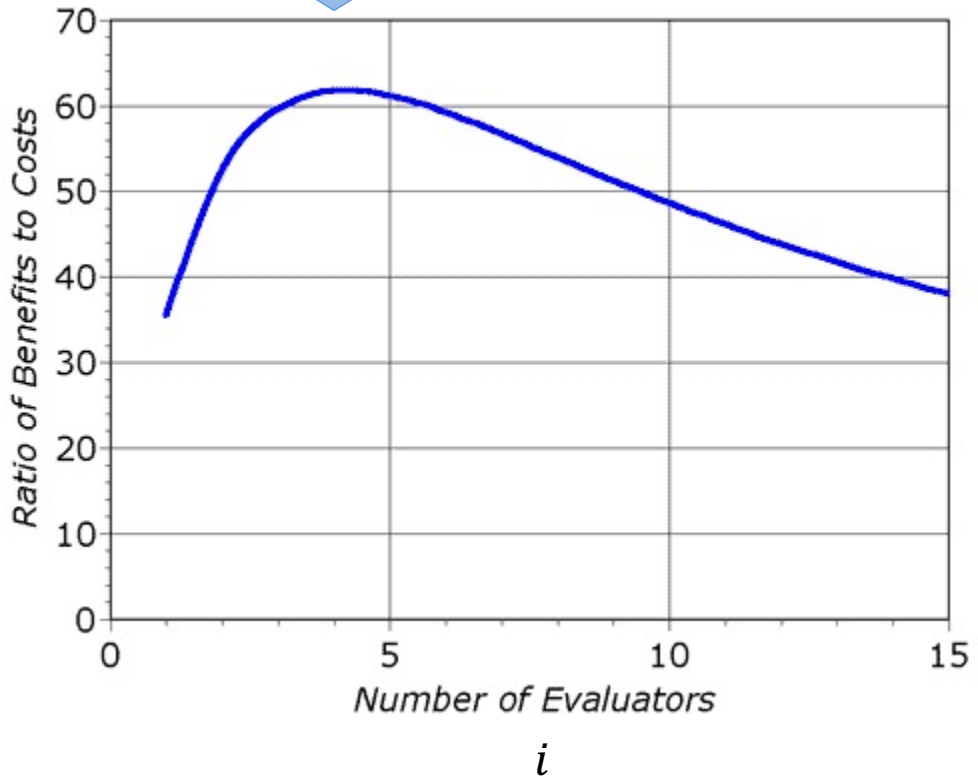
$\frac{PF(i)}{N}$



i

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

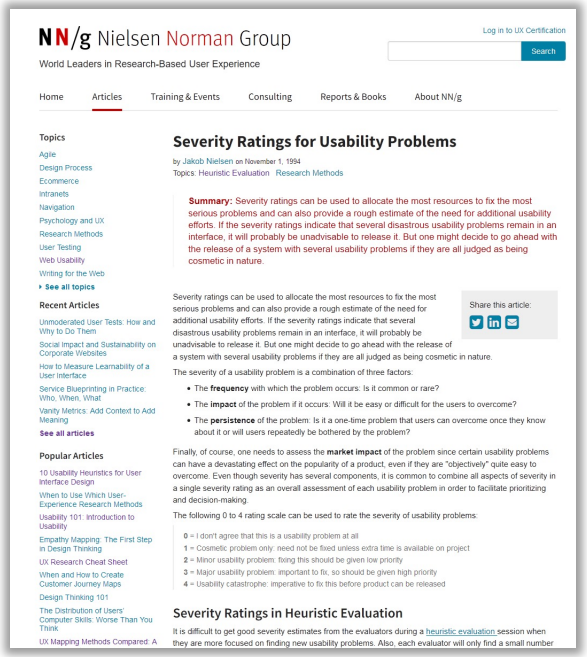
3-5



i

Severity Rating

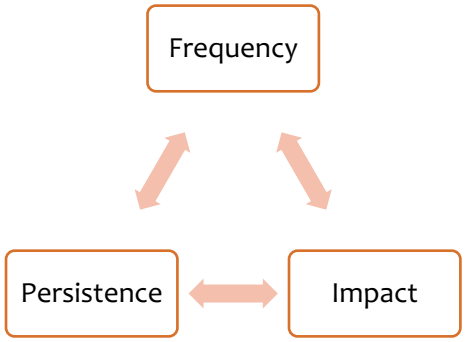
- We need to allocate the most resources to fix the most serious problems
- We need to understand if additional usability efforts are required
- **Severity** is a combination of:
  - **Frequency** with which the problem occurs: common or rare?
  - **Impact** of the problem if it occurs: easy to overcome or difficult?
  - **Persistence**, is it one-time or will it occur many times to users?
- Define a *combined severity rating*
  - Individually, for each evaluator



https://www.nngroup.com/articles/how-to-rate-the-severity-of-usability-problems/

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



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
  - Discussion: how can we fix it?
  - 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 sw, 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* some

User Testing

- Need to develop sw, and prepare the set-up
- More accurate (by definition!)
  - Actual users and tasks

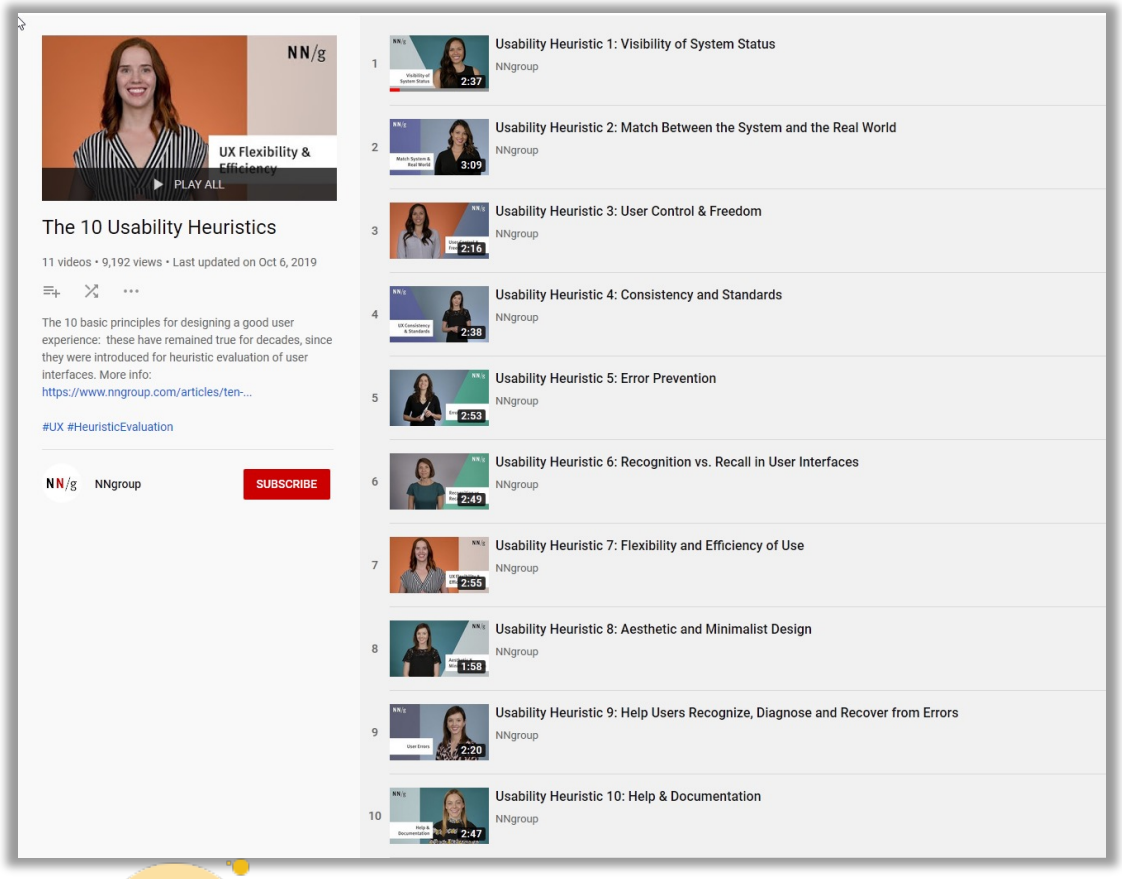
- Alternate the methods!
  - Find different problems
  - Don't waste participants

https://www.nngroup.com/articles/usability-problems-found-by-heuristic-evaluation/

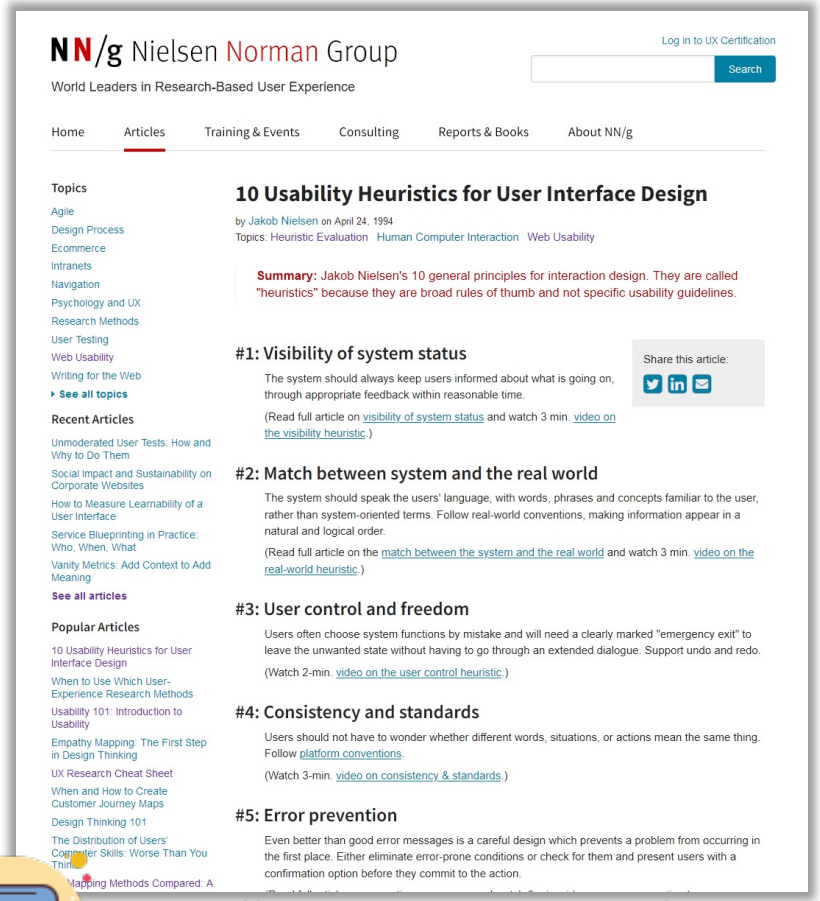
Nielsen's Usability Heuristics

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

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

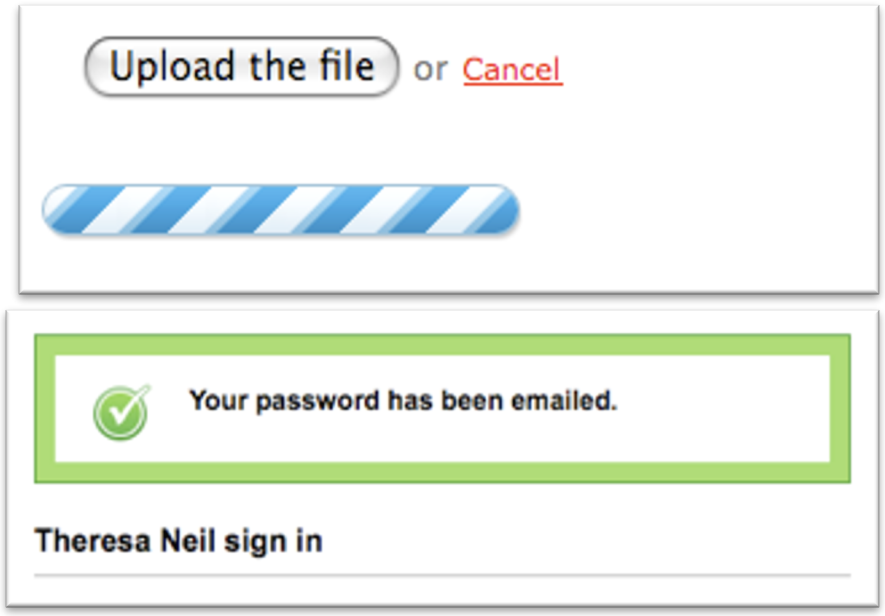
#1: Visibility of system status

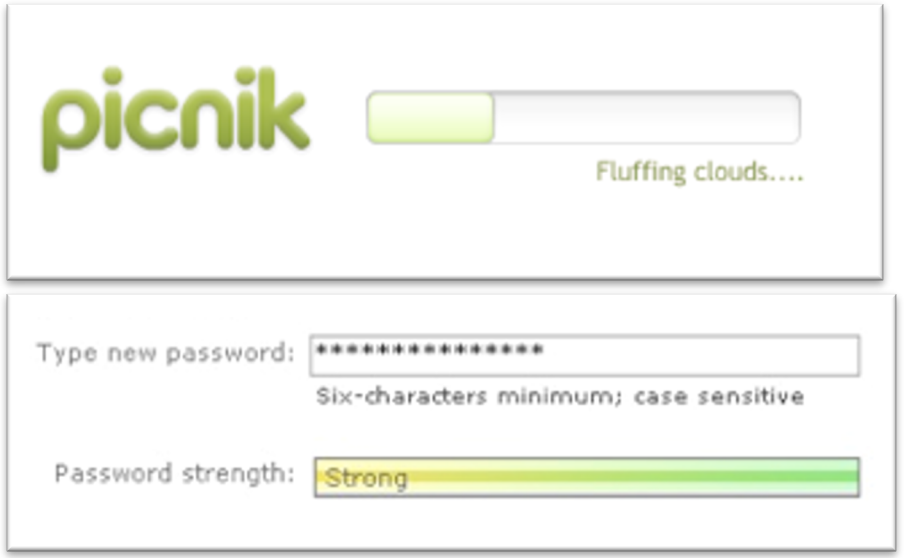
- The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

https://www.nngroup.com/articles/visibility-system-status/

#1: Visibility of system status

- The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.





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, …)

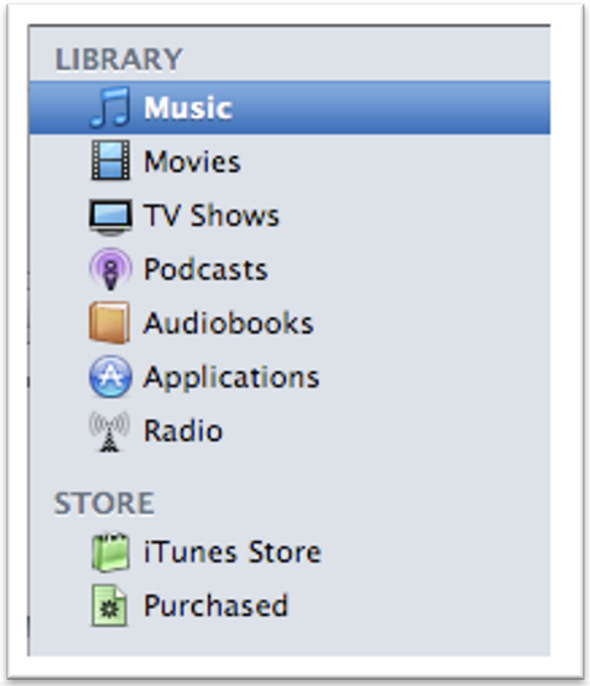
#2: Match between system and the real world

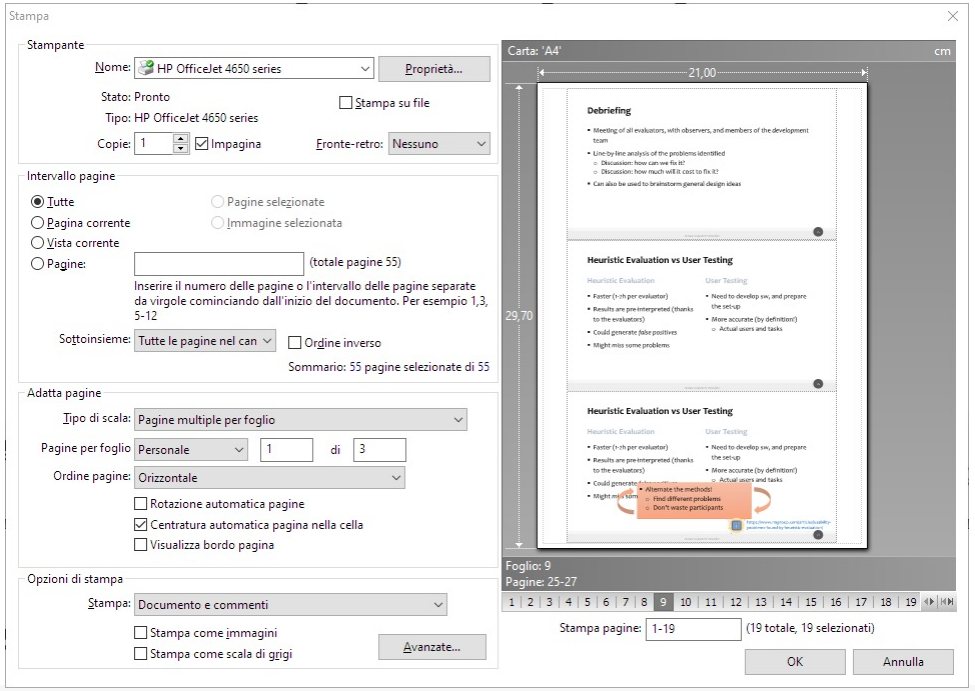
- The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
- Use familiar metaphors and language

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

#2: Match between system and the real world

- 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.





Exploit Familiarity

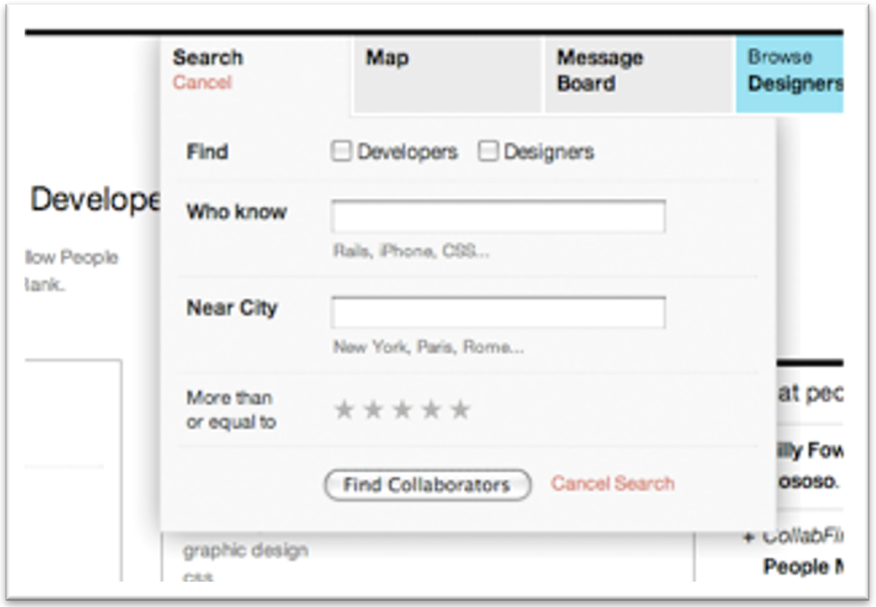
- Familiar Metaphors
  - Files, paper, folders, highlighters, …
- Familiar Language
  - Avoid jargon, acronyms, etc. that could be unknown to your users
- Familiar Categories
- Familiar Choices
  - E.g., explain the meaning of the error message (what happened, what are the consequences, what are the 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

- 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.

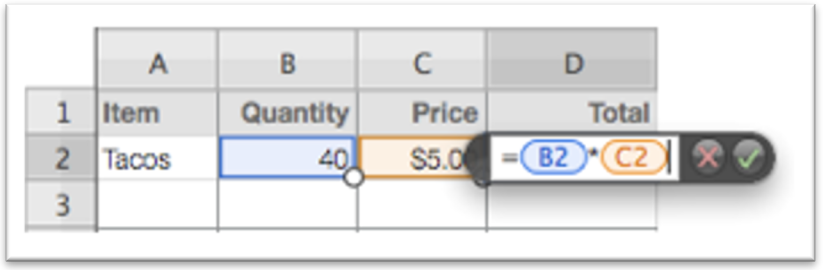


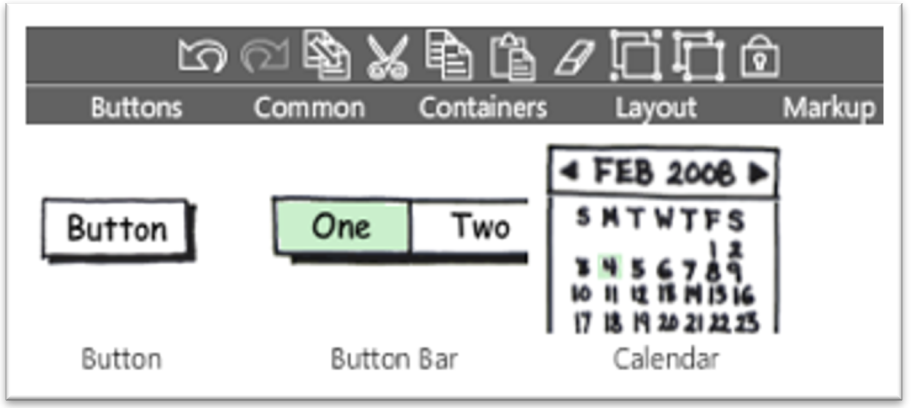
Suggestions

- 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

#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.



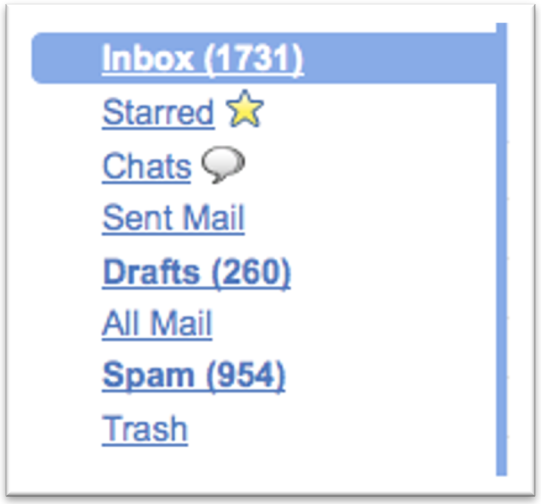


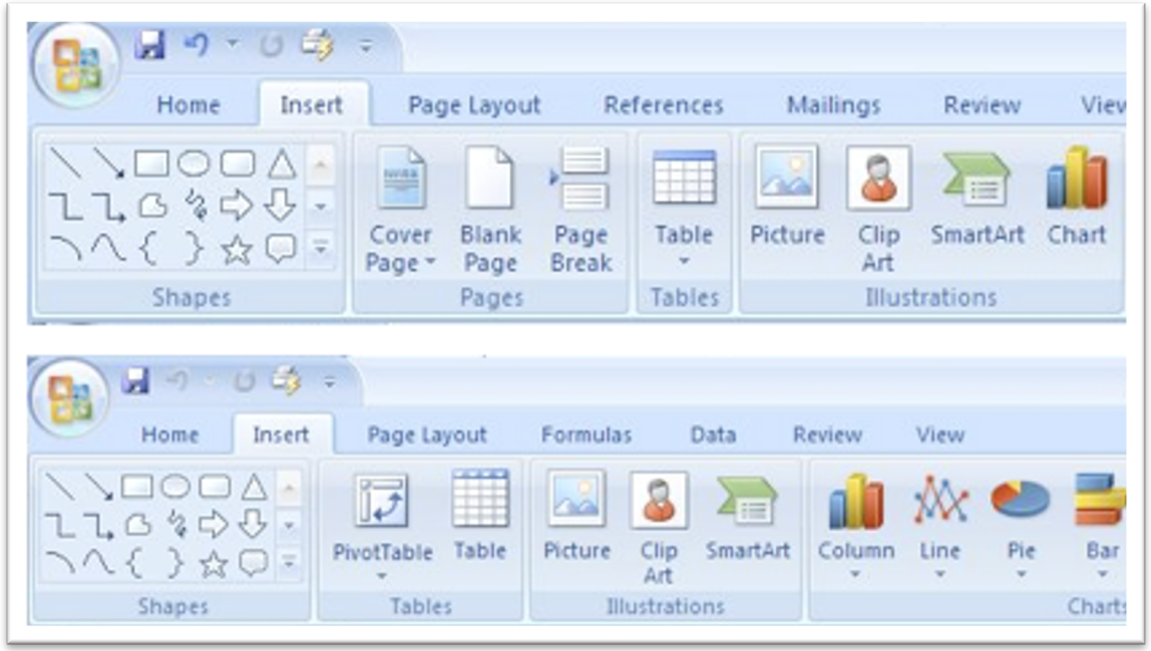
#4: Consistency and standards

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

#4: Consistency and standards

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

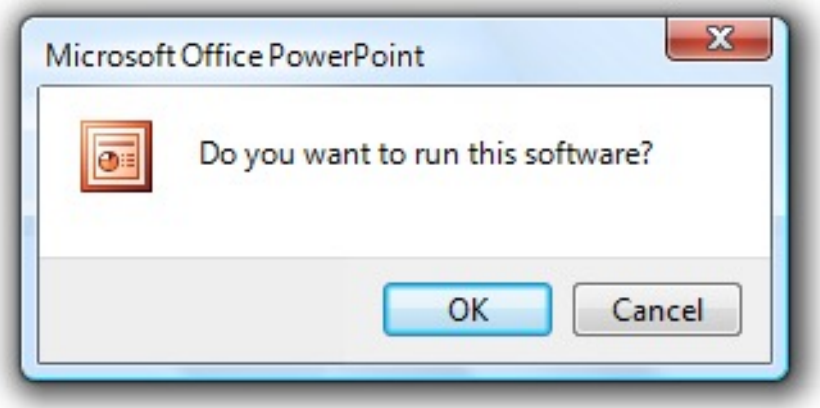




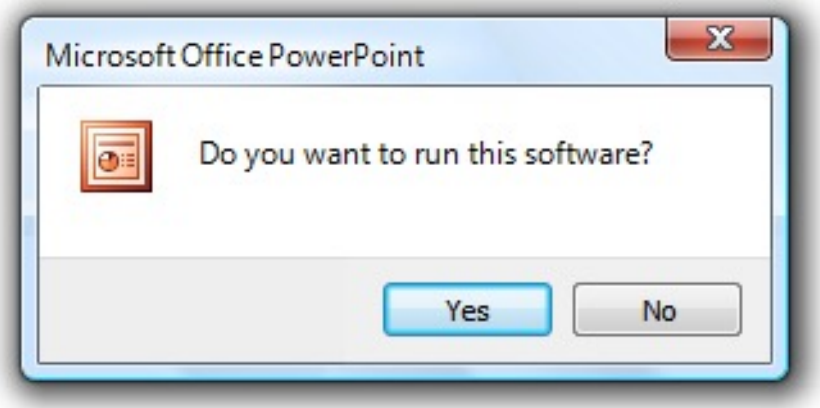
Suggestions

- 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

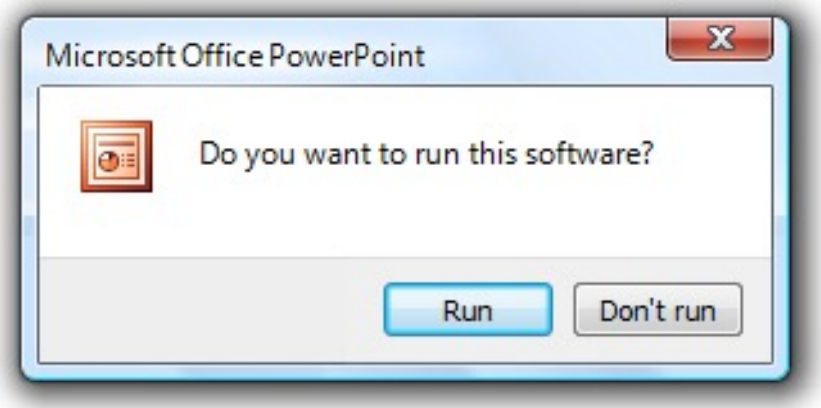
Examples



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.

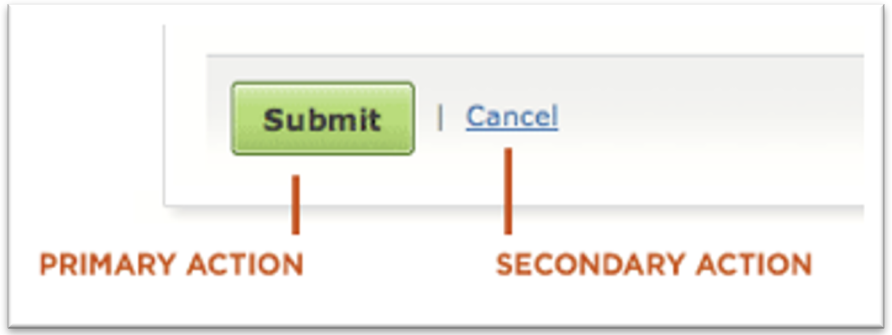
https://www.nngroup.com/articles/slips/

Suggestions

- Preventing data loss
- Prevent clutter
- Prevent confusing flow
- Prevent bad input
- Prevent unnecessary constraints (e.g., provide defaults for missing data)

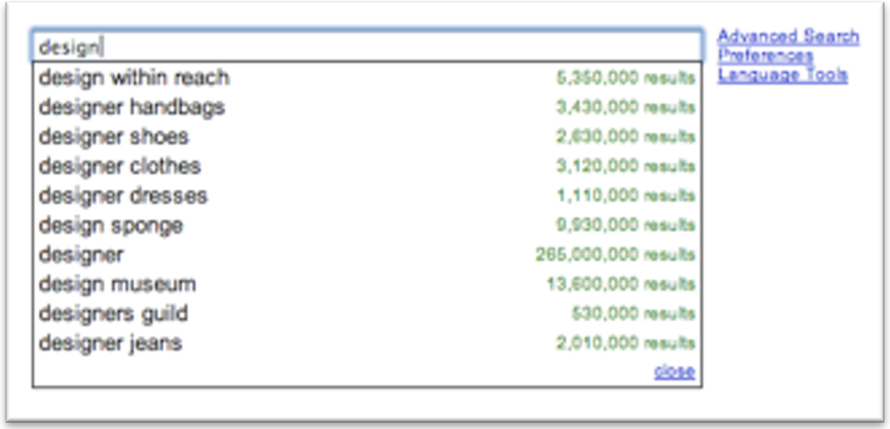
#5: Error prevention

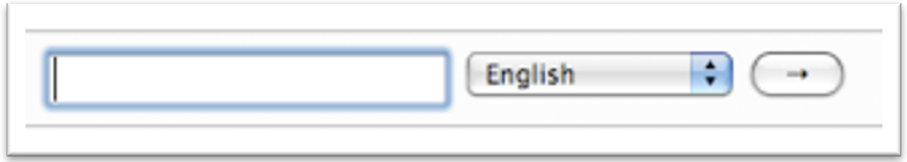
- Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.



#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.





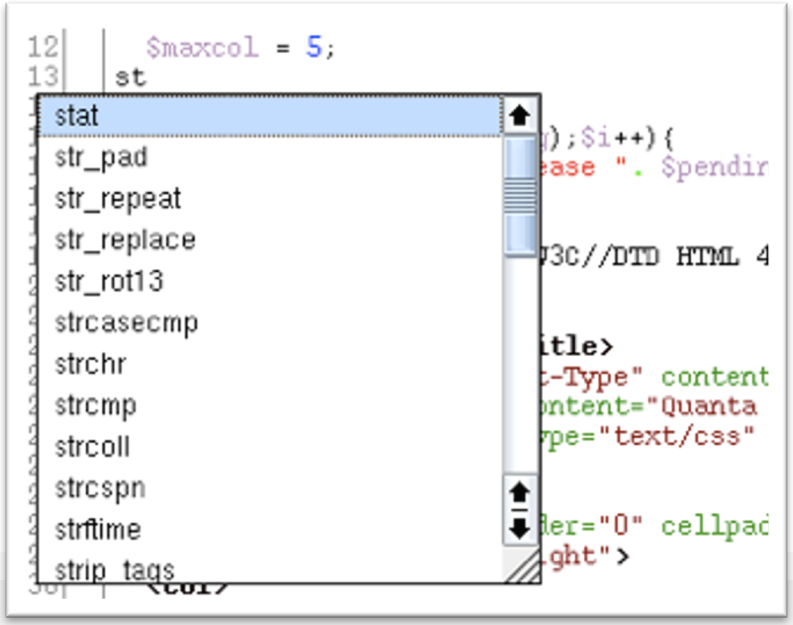
#6: Recognition rather than recall

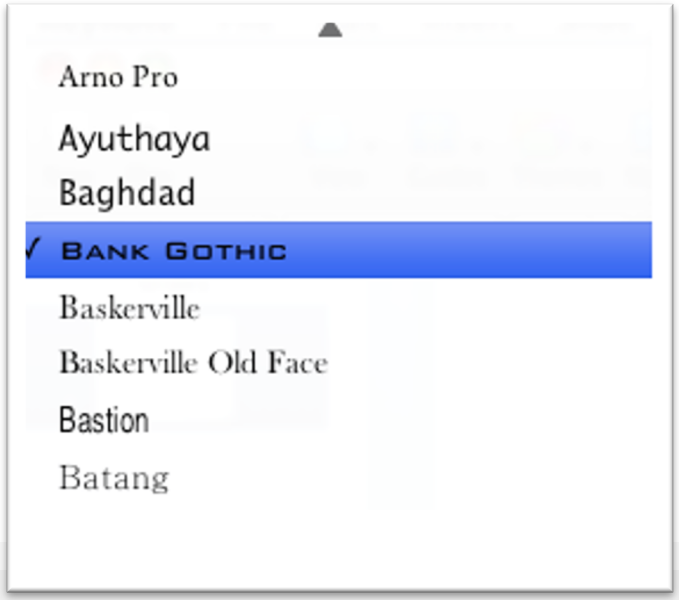
- Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

https://www.nngroup.com/articles/recognition-and-recall/

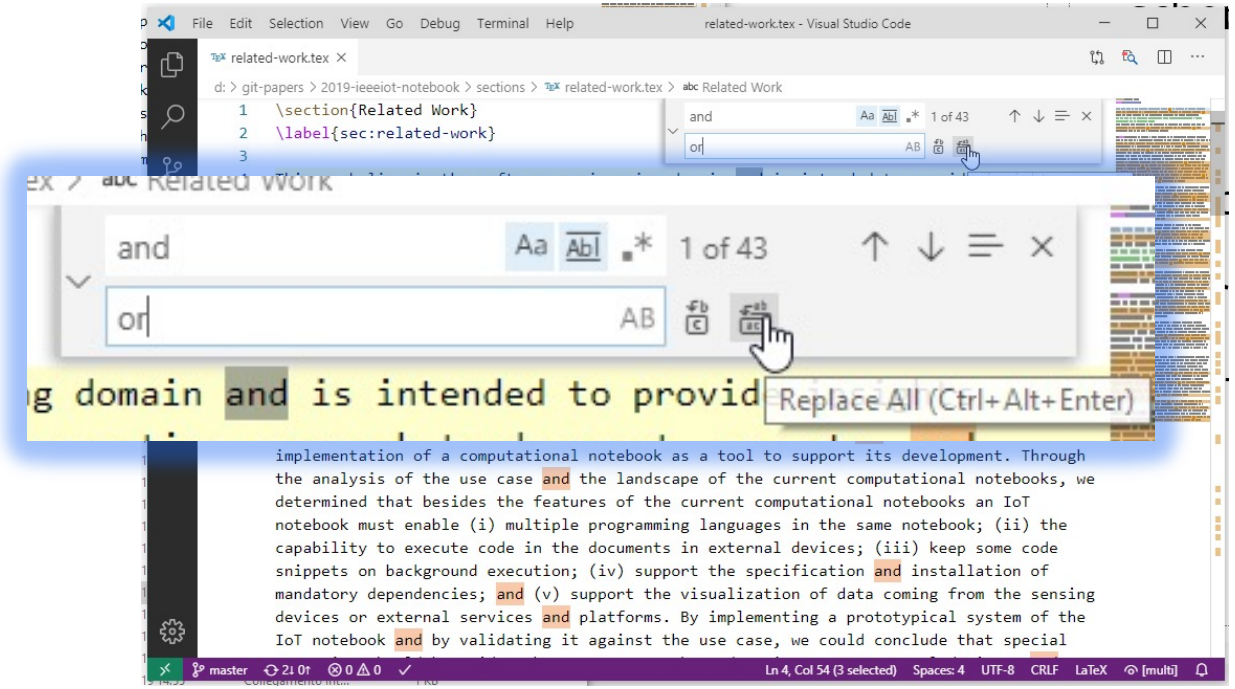
#6: Recognition rather than recall

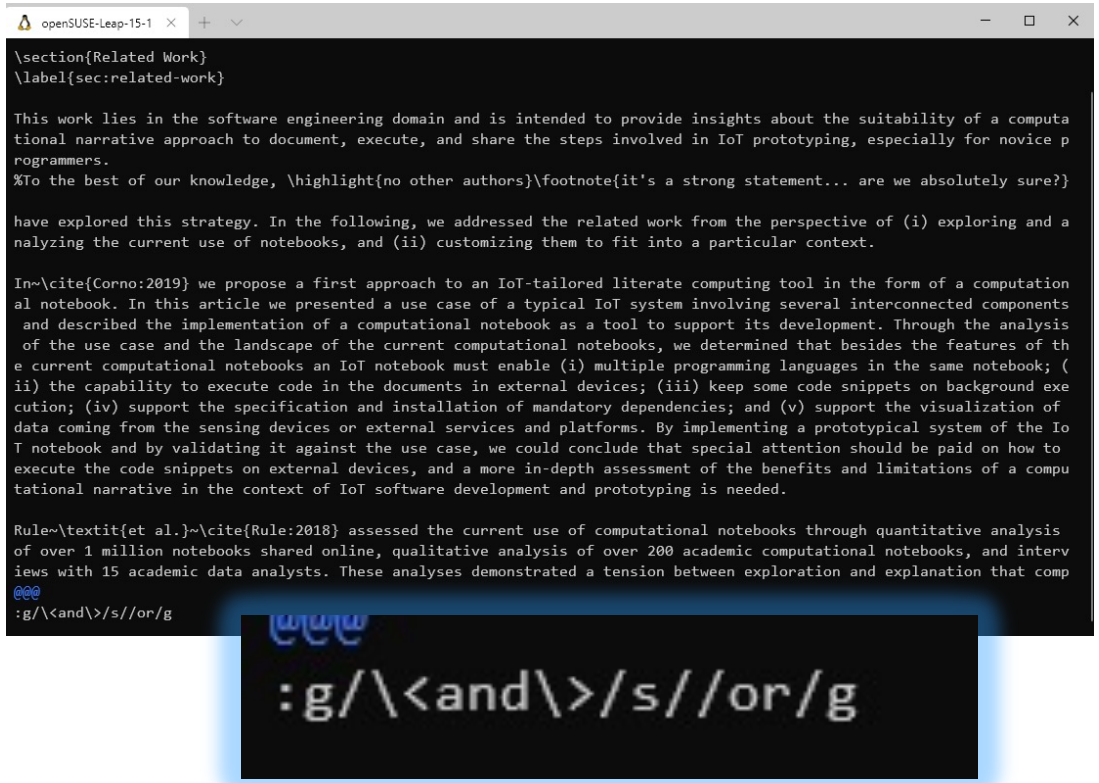
- Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.





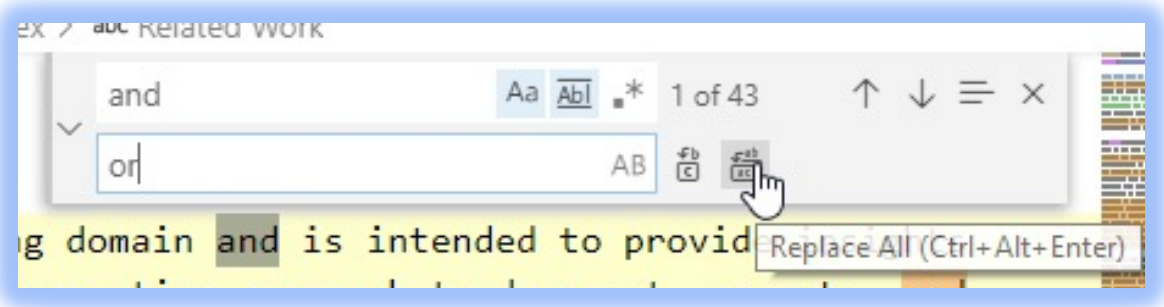
Example


File Edit Selection View Go Debug Terminal Help
related-work.tex - Visual Studio Code
related-work.tex
d: > git-papers > 2019-ieeeiot-notebook > sections > related-work.tex > Related Work
1 \section{Related Work}
2 \label{sec:related-work}
and
or
1 of 43
Replace All (Ctrl+Alt+Enter)
implementation of a computational notebook as a tool to support its development. Through
the analysis of the use case and the landscape of the current computational notebooks, we
determined that besides the features of the current computational notebooks an IoT
notebook must enable (i) multiple programming languages in the same notebook; (ii) the
capability to execute code in the documents in external devices; (iii) keep some code
snippets on background execution; (iv) support the specification and installation of
mandatory dependencies; and (v) support the visualization of data coming from the sensing
devices or external services and platforms. By implementing a prototypical system of the
IoT notebook and by validating it against the use case, we could conclude that special
master
Ln 4, Col 54 (3 selected)
Spaces: 4
UTF-8
CRLF
LaTeX


openSUSE-Leap-15-1
\section{Related Work}
\label{sec:related-work}
This work lies in the software engineering domain and is intended to provide insights about the suitability of a computational narrative approach to document, execute, and share the steps involved in IoT prototyping, especially for novice programmers.
%To the best of our knowledge, \highlight{no other authors}\footnote{it's a strong statement... are we absolutely sure?}
have explored this strategy. In the following, we addressed the related work from the perspective of (i) exploring and analyzing the current use of notebooks, and (ii) customizing them to fit into a particular context.
In~\cite{Corno:2019} we propose a first approach to an IoT-tailored literate computing tool in the form of a computational notebook. In this article we presented a use case of a typical IoT system involving several interconnected components and described the implementation of a computational notebook as a tool to support its development. Through the analysis of the use case and the landscape of the current computational notebooks, we determined that besides the features of the current computational notebooks an IoT notebook must enable (i) multiple programming languages in the same notebook; (ii) the capability to execute code in the documents in external devices; (iii) keep some code snippets on background execution; (iv) support the specification and installation of mandatory dependencies; and (v) support the visualization of data coming from the sensing devices or external services and platforms. By implementing a prototypical system of the IoT notebook and by validating it against the use case, we could conclude that special attention should be paid on how to execute the code snippets on external devices, and a more in-depth assessment of the benefits and limitations of a computational narrative in the context of IoT software development and prototyping is needed.
Rule~\textit{et al.}~\cite{Rule:2018} assessed the current use of computational notebooks through quantitative analysis of over 1 million notebooks shared online, qualitative analysis of over 200 academic computational notebooks, and interviews with 15 academic data analysts. These analyses demonstrated a tension between exploration and explanation that comp
@@@
:g/\<and\>/s//or/g

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
  - …

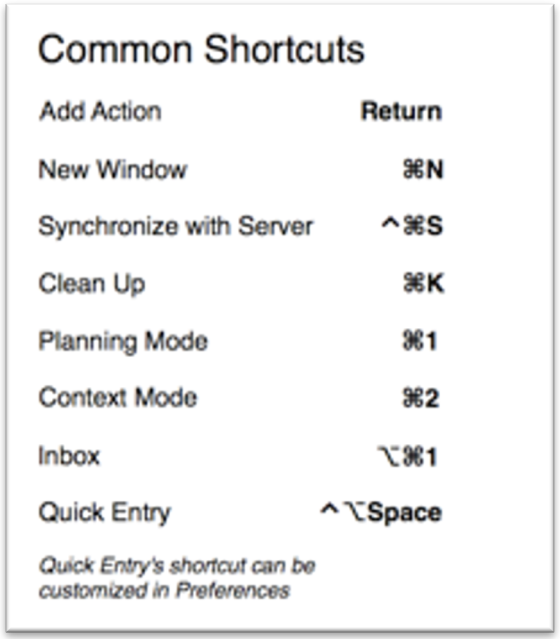


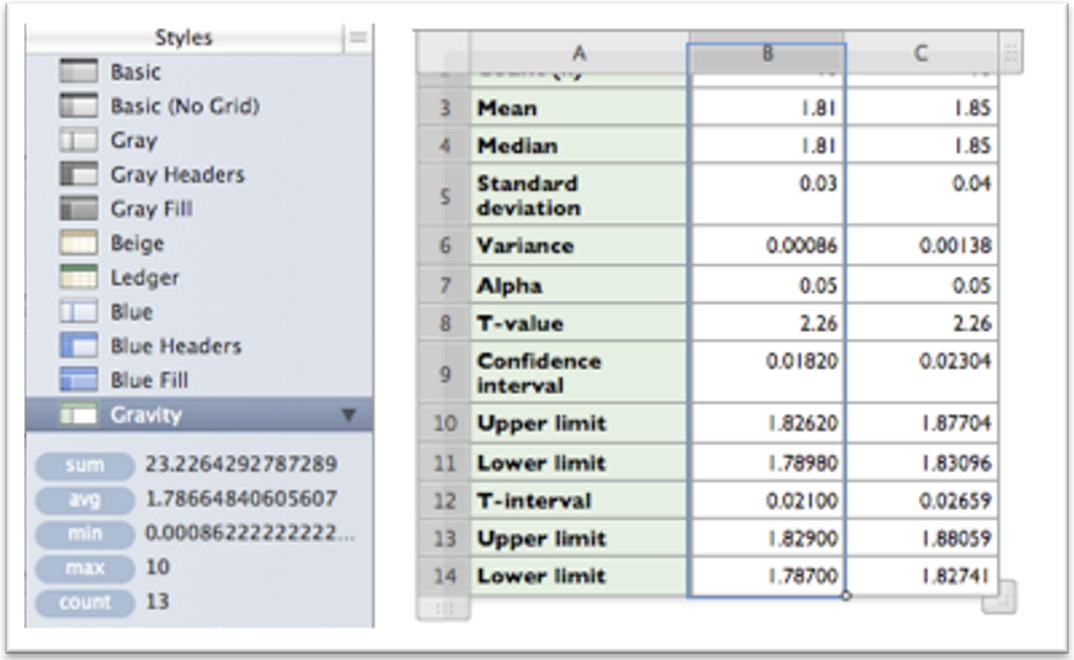
#7: Flexibility and efficiency of use

- 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

- 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.





Suggestions

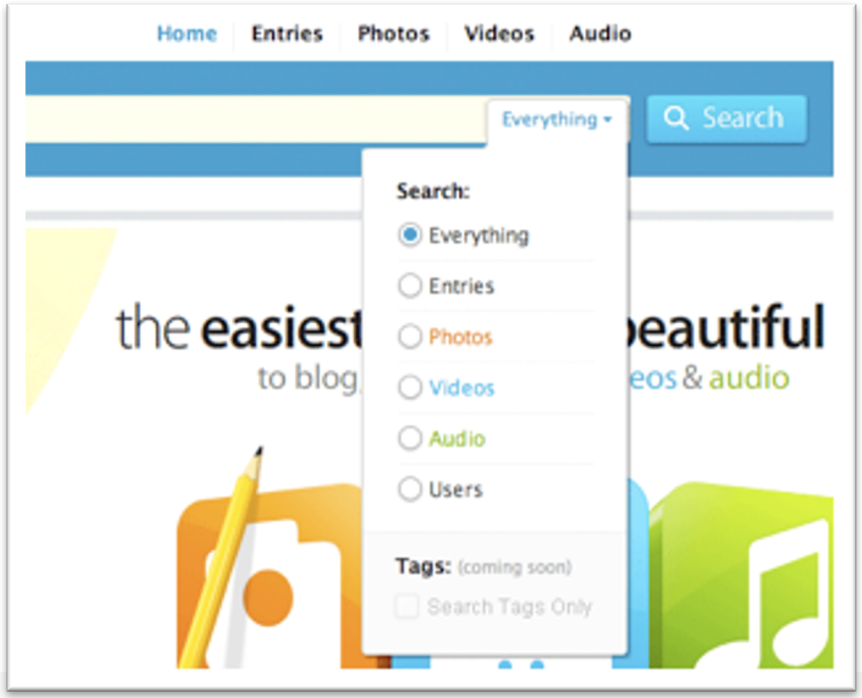
- Flexibility = Default + Options
  - E.g., present some popular choices, but let the user enter a custom one (train ticket machines)
- Exploit background information for providing more information
  - E.g., weather forecasts in a calendar interface
- Proactivity
  - E.g., “mark as spam” proposed to “unsubscribe”, too
- Recommendations
- Provide relevant information, only

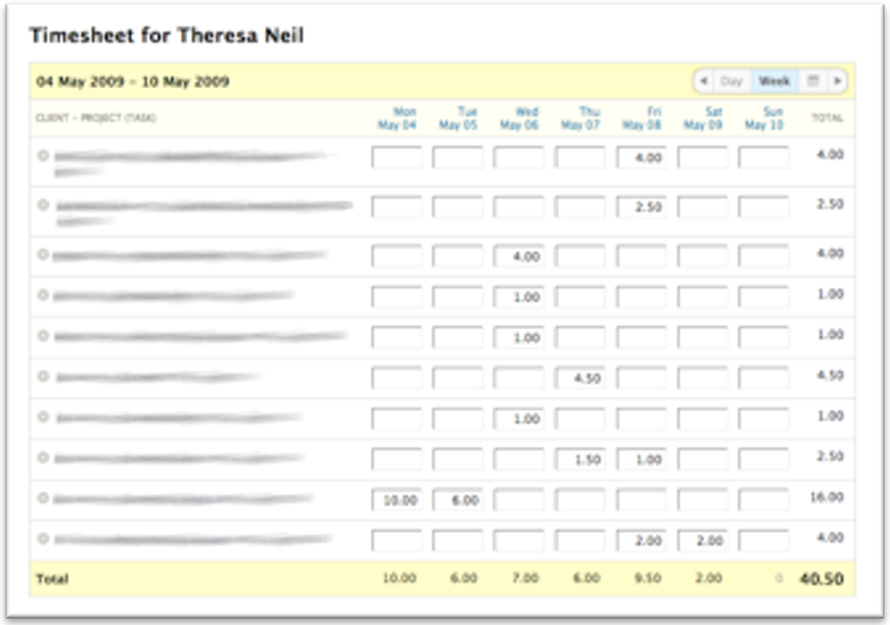
#8: Aesthetic and minimalist design

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

#8: Aesthetic and minimalist design

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





Suggestions

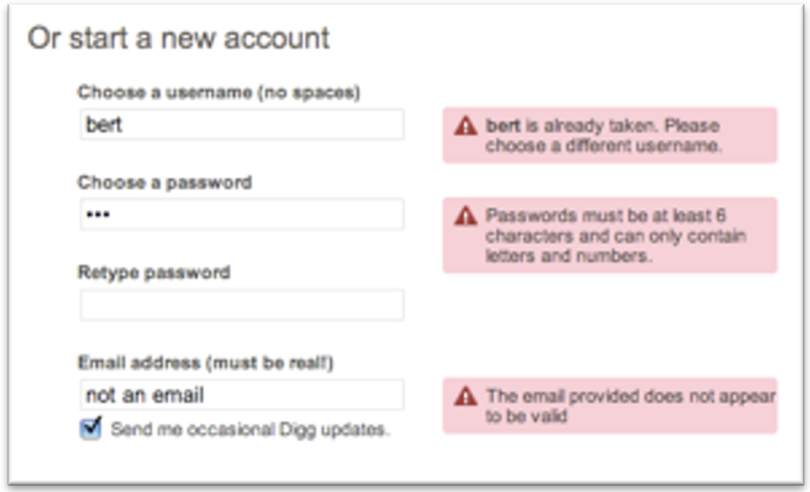
- Key information must be “above the fold”
  - Especially on low-resolution devices
- Keep high signal-to-noise ratio
  - Colors, fonts, backgrounds, animations, …
  - Borders, dividers, …
- Minimalistic login experience
- Accept redundant ways of entering information
- Prune features that are outside the “core” functionality

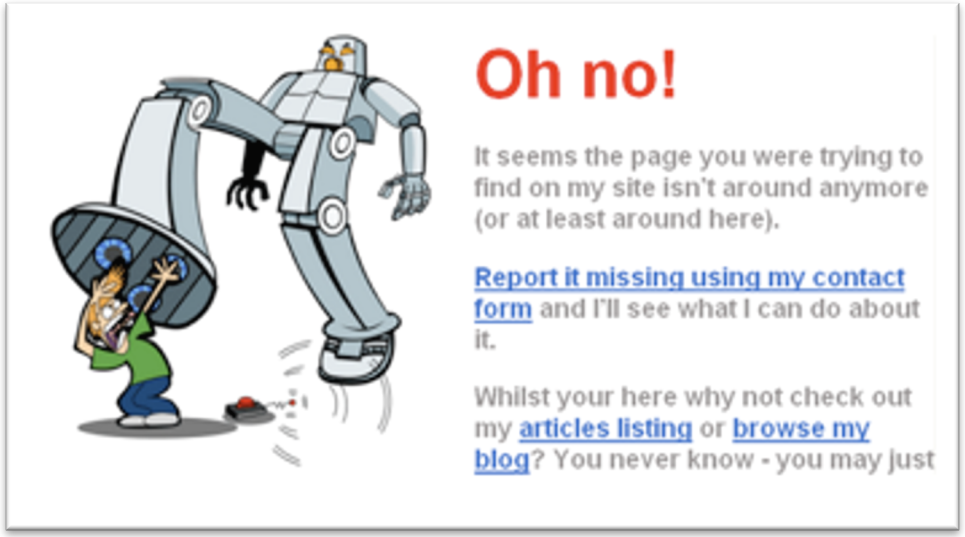
#9: Help users recognize, diagnose, and recover from errors

- 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

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





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

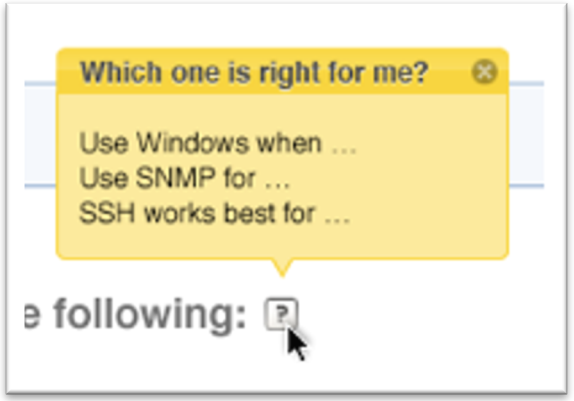
- Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

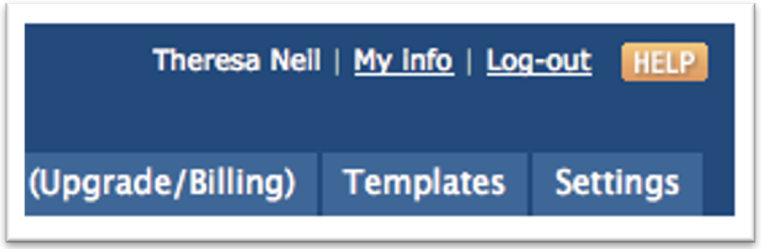
#10: Help and documentation

- 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

- 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.





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)

References

- Alan Dix, Janet Finlay, Gregory Abowd, Russell Beale: Human Computer Interaction, 3rd Edition
  - 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
  - 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

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/