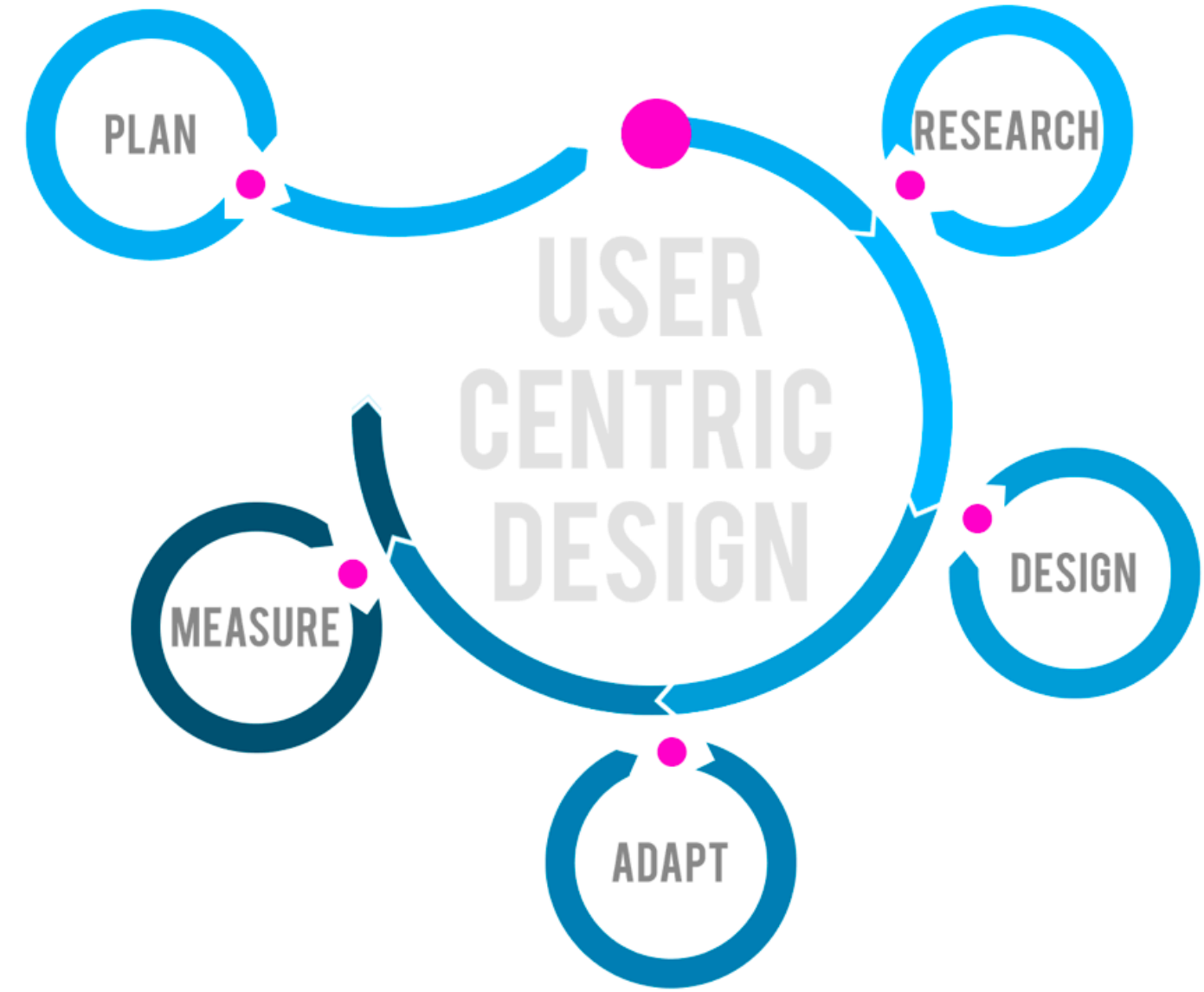
SEG 2105 - LECTURE 07

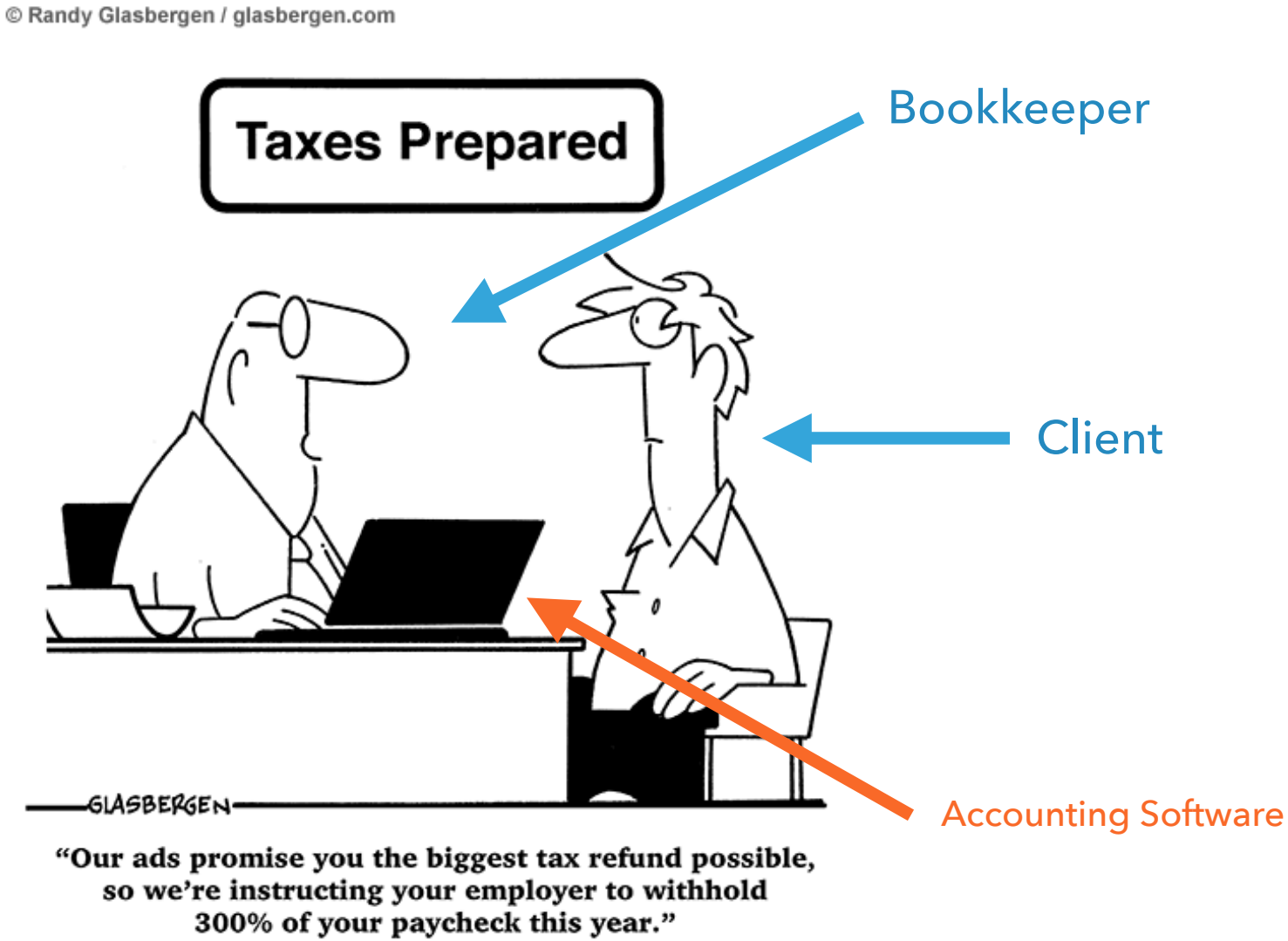
Focusing on Users and Their Tasks


PLAN
RESEARCH
USER
CENTRIC
DESIGN
MEASURE
DESIGN
ADAPT

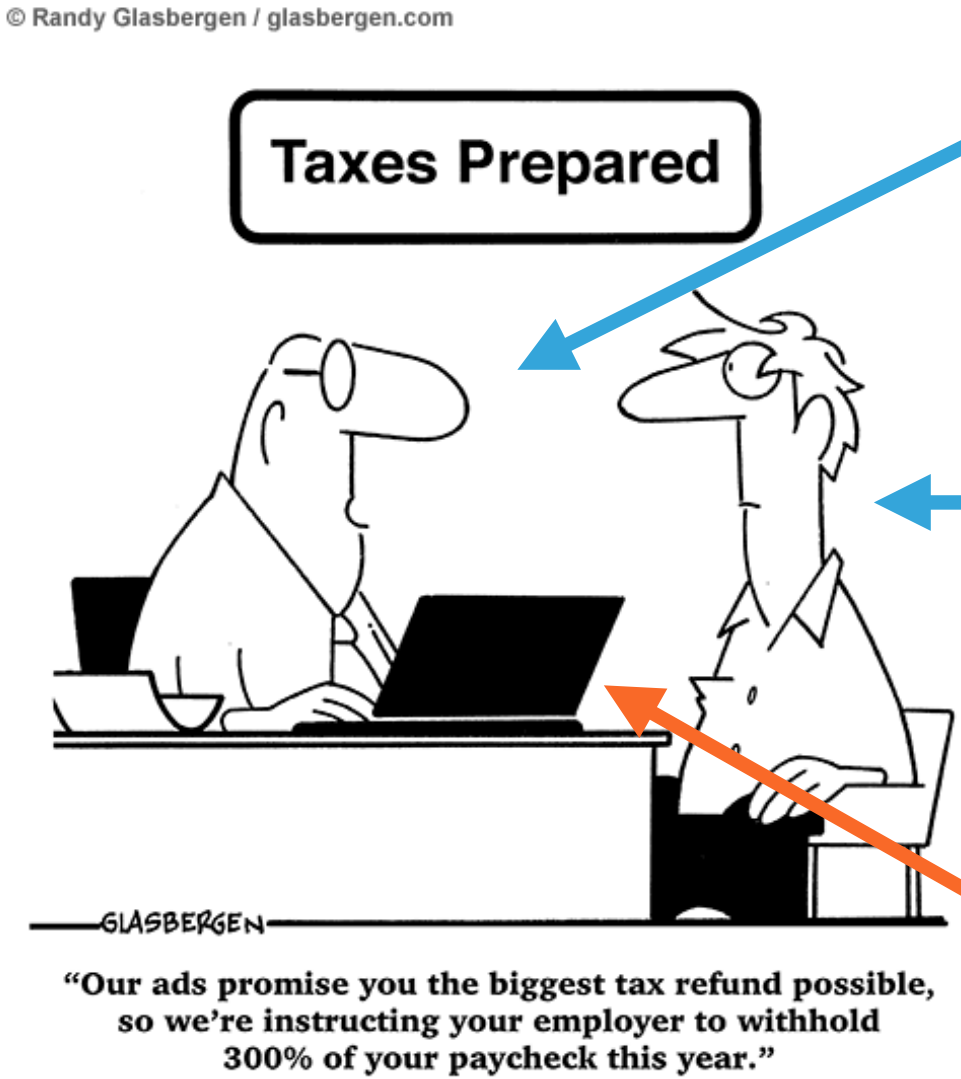
FOCUS ON THE NEEDS OF THE USER

- Know them
- Understand them
- Design based on their tasks
- Involve them
- Good Usability / User Experience
- Elicit feedback on prototypes, help and user manuals

STAKEHOLDERS ARE MORE THAN JUST USERS



STAKEHOLDERS ARE MORE THAN JUST USERS



Bookkeeper

Client

Accounting Software

Client's employer

Tax authority (CRA)

Auditors (also CRA)

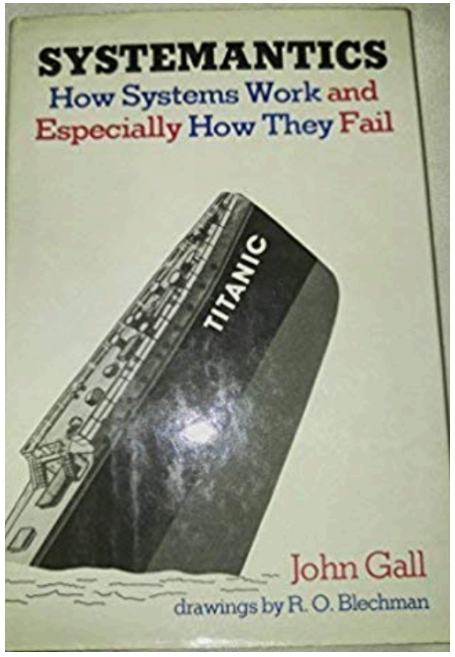
Tax Lawyers

Accountants (Bookkeepers boss)

Why focus on the user?

Increase	Decrease
Efficiency of use	Training / Support
Prioritization of work	Time to Learn
Iterative development	Develop only needed features
Attractiveness of system	Cost for changes

> A complex system that works is invariably found to have evolved from a simple system that worked. A complex system designed from scratch never works and cannot be patched up to make it work. You have to start over, beginning with a working simple system.
>
> —
>
> *John Gall*



https://signalvnoise.com/posts/1414-a-complex-system-that-works-is-invariably

*When faced with multiple options, choose the path that **makes future change easier**.*

https://pragdave.me/blog/2014/03/04/time-to-kill-agile.html

User Characteristics

GOALS

PATTERNS OF USE

DEMOGRAPHICS

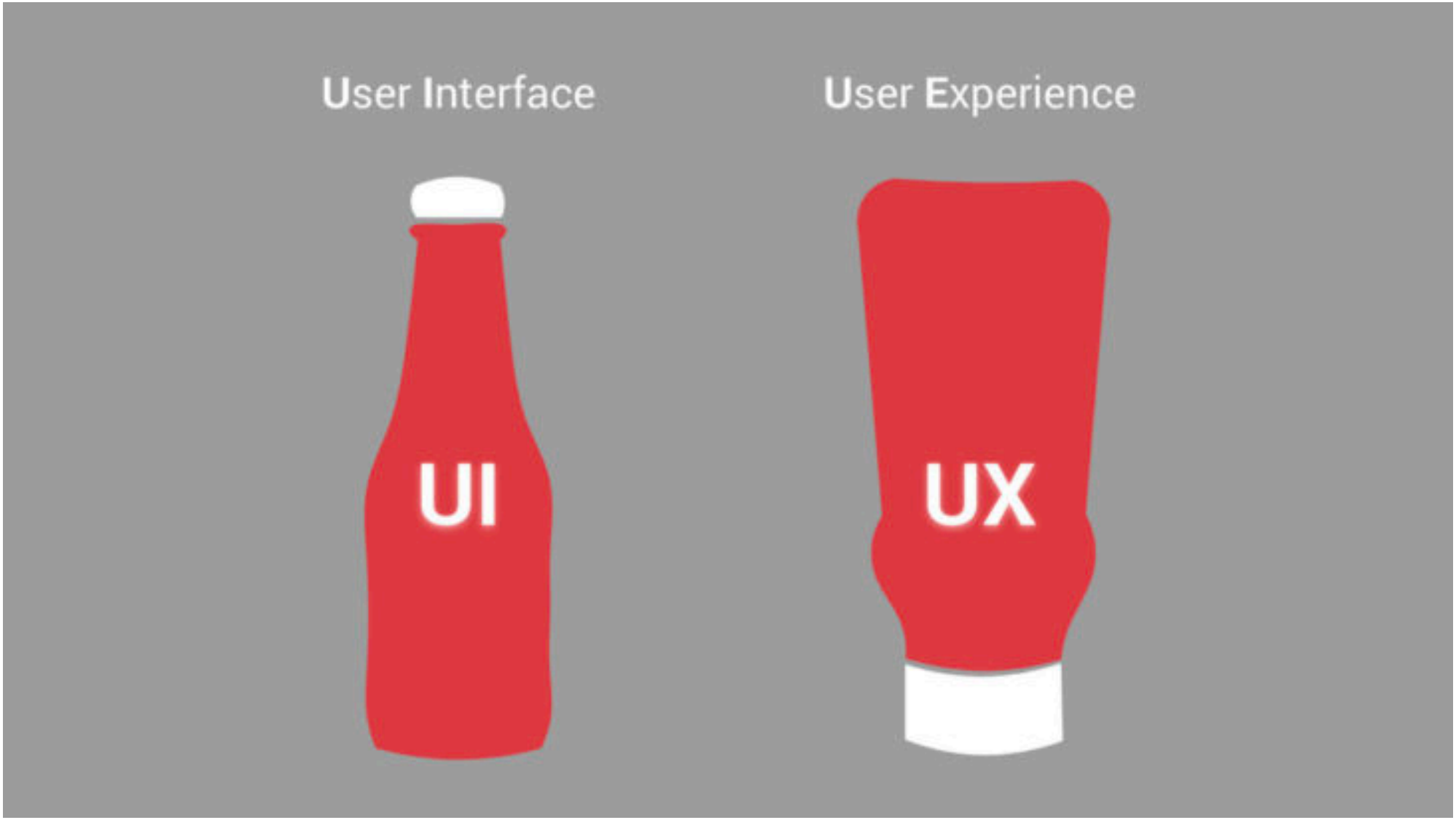
DOMAIN KNOWLEDGE

PHYSICAL ABILITIES

PSYCH TRAITS EMOTIONAL

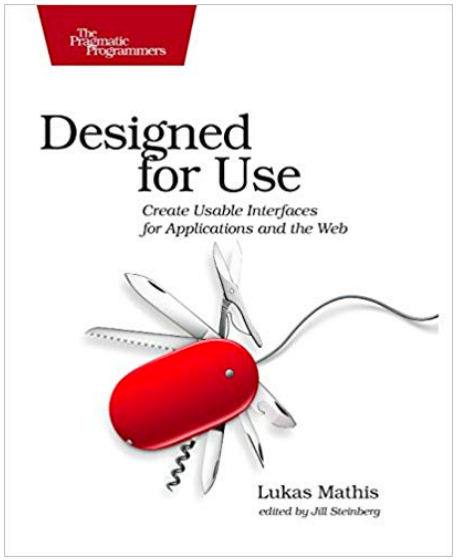
BASICS OF

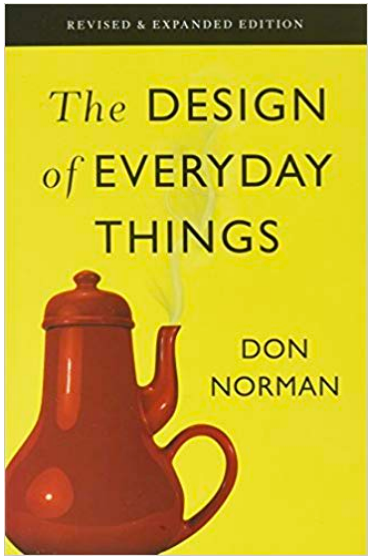
USER INTERFACE DESIGN

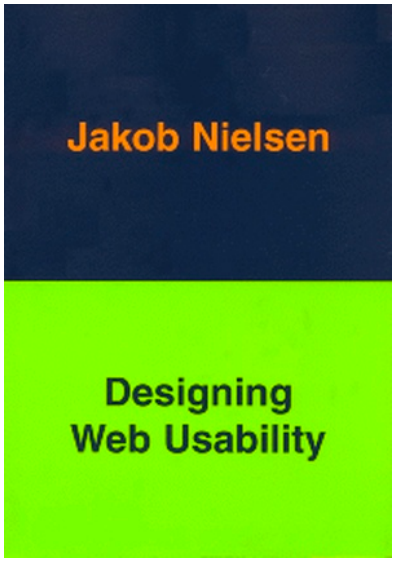


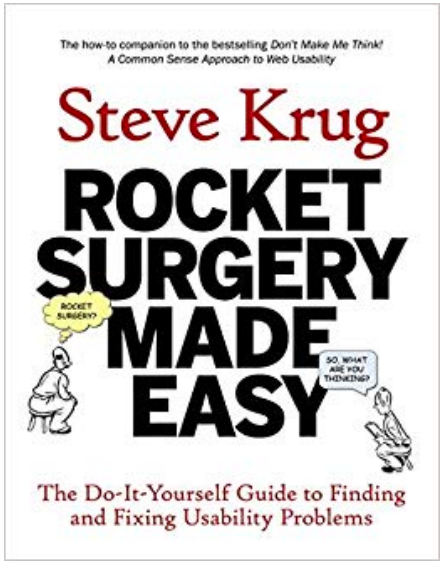
https://www.creative-tim.com/blog/web-design/how-to-develop-an-app-with-a-great-user-interface-and-experience/

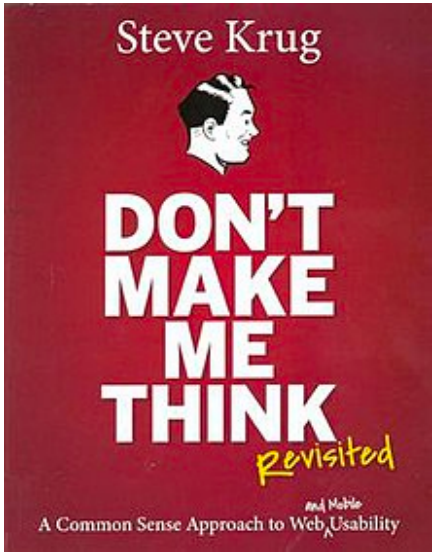
GOOD USABILITY IS A COURSE IN ITSELF

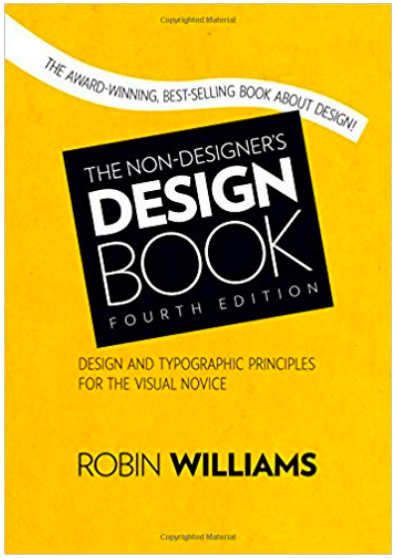


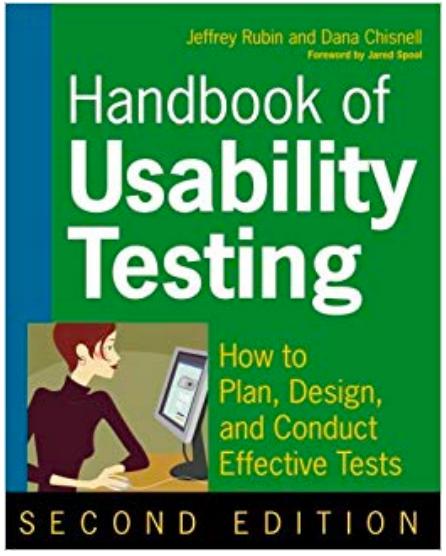


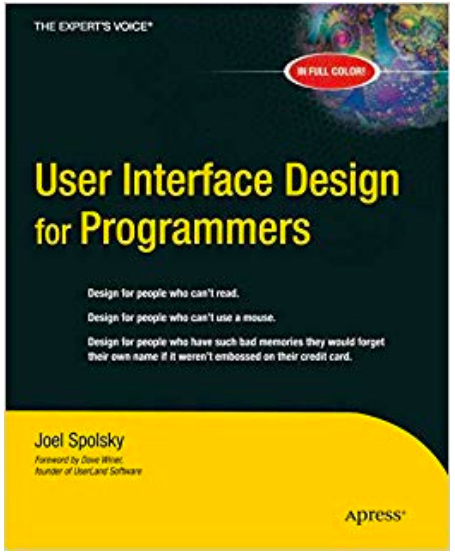












Usability (User Experience or UX)

SIMPLICITY

FAMILIARITY

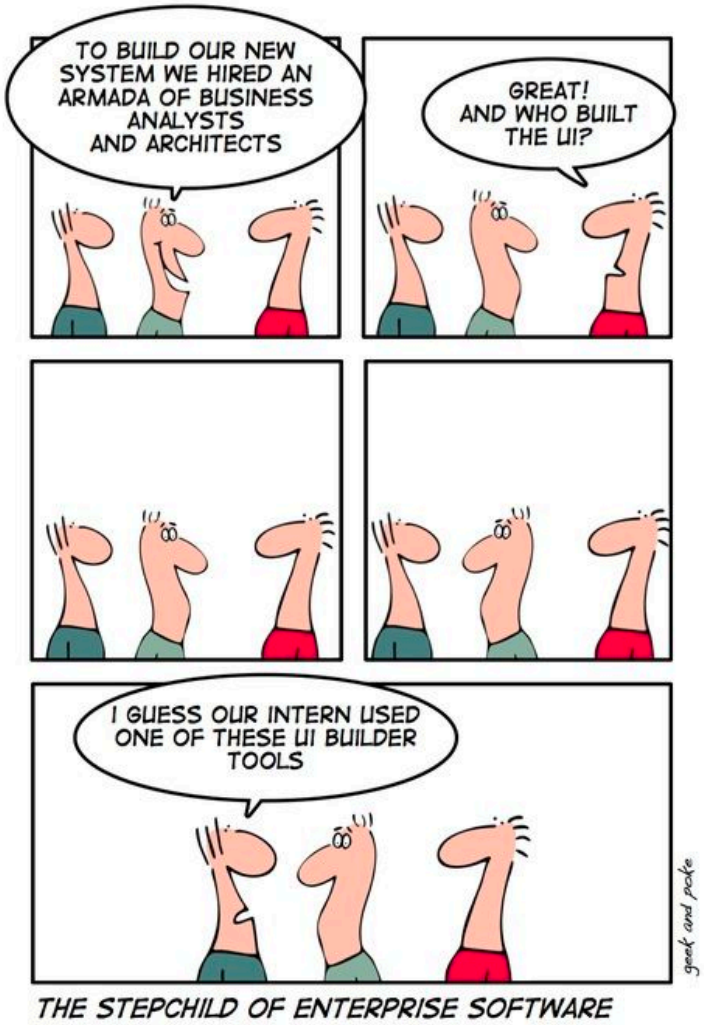
RECOGNITION

FLEXIBILITY

AFFORDANCE

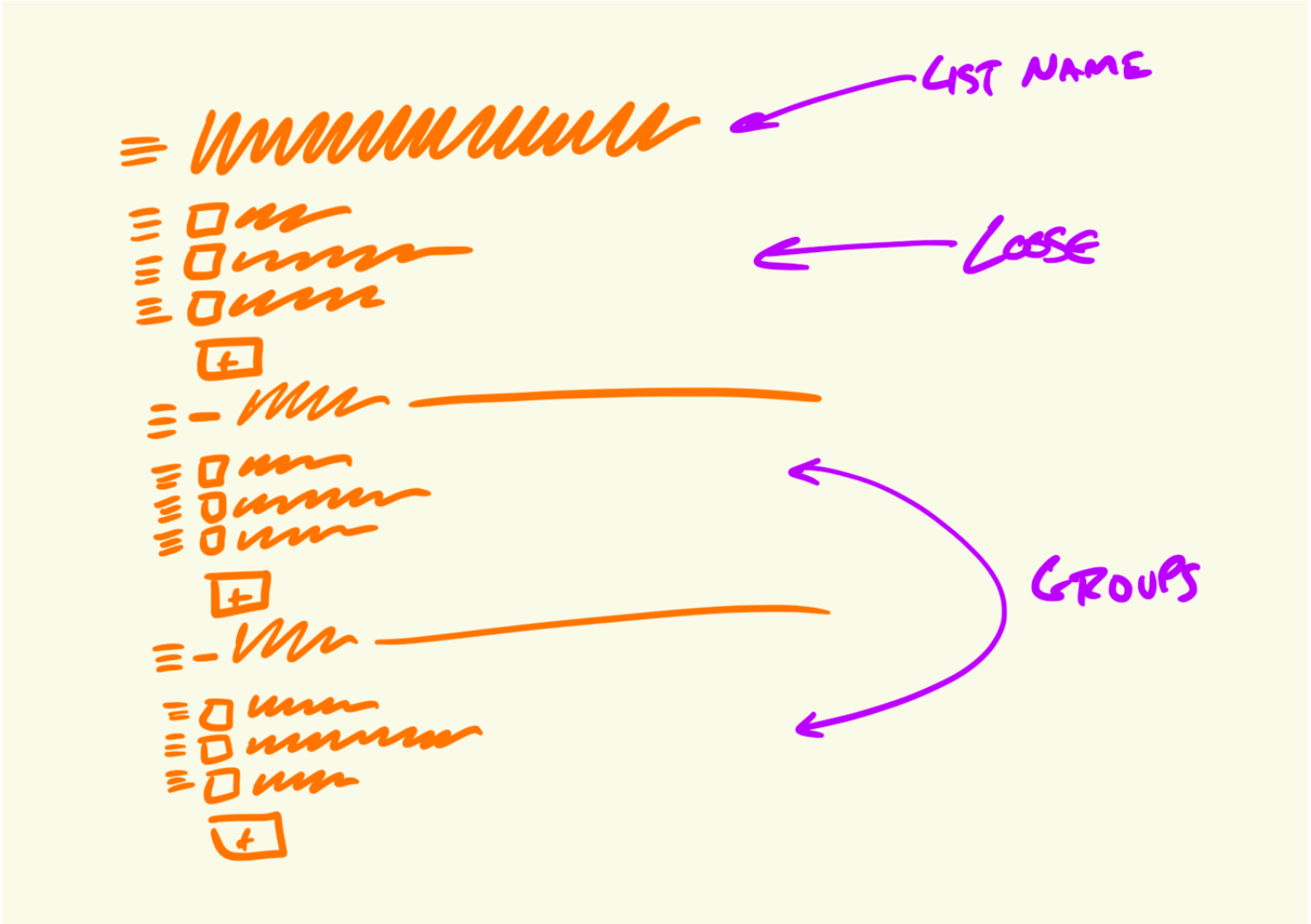
SAFETY

EXPLICIT PLANNING FOR UI DESIGN



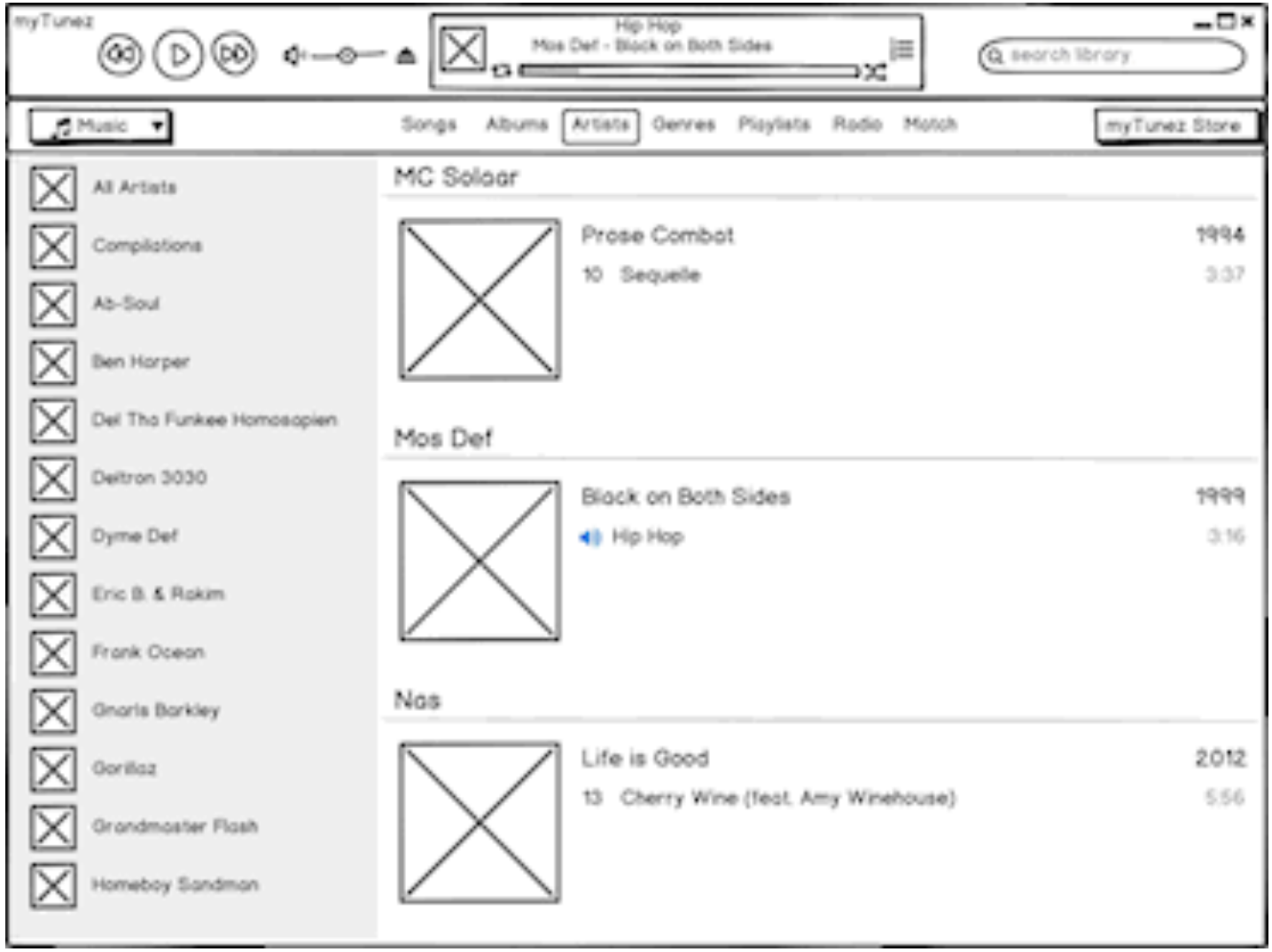
- UI Design as part of other SE activities
- Use Case Analysis
- UI Prototyping
- Iterative Approach

PAPER PROTOTYPING (FAT MARKER SKETCHES)



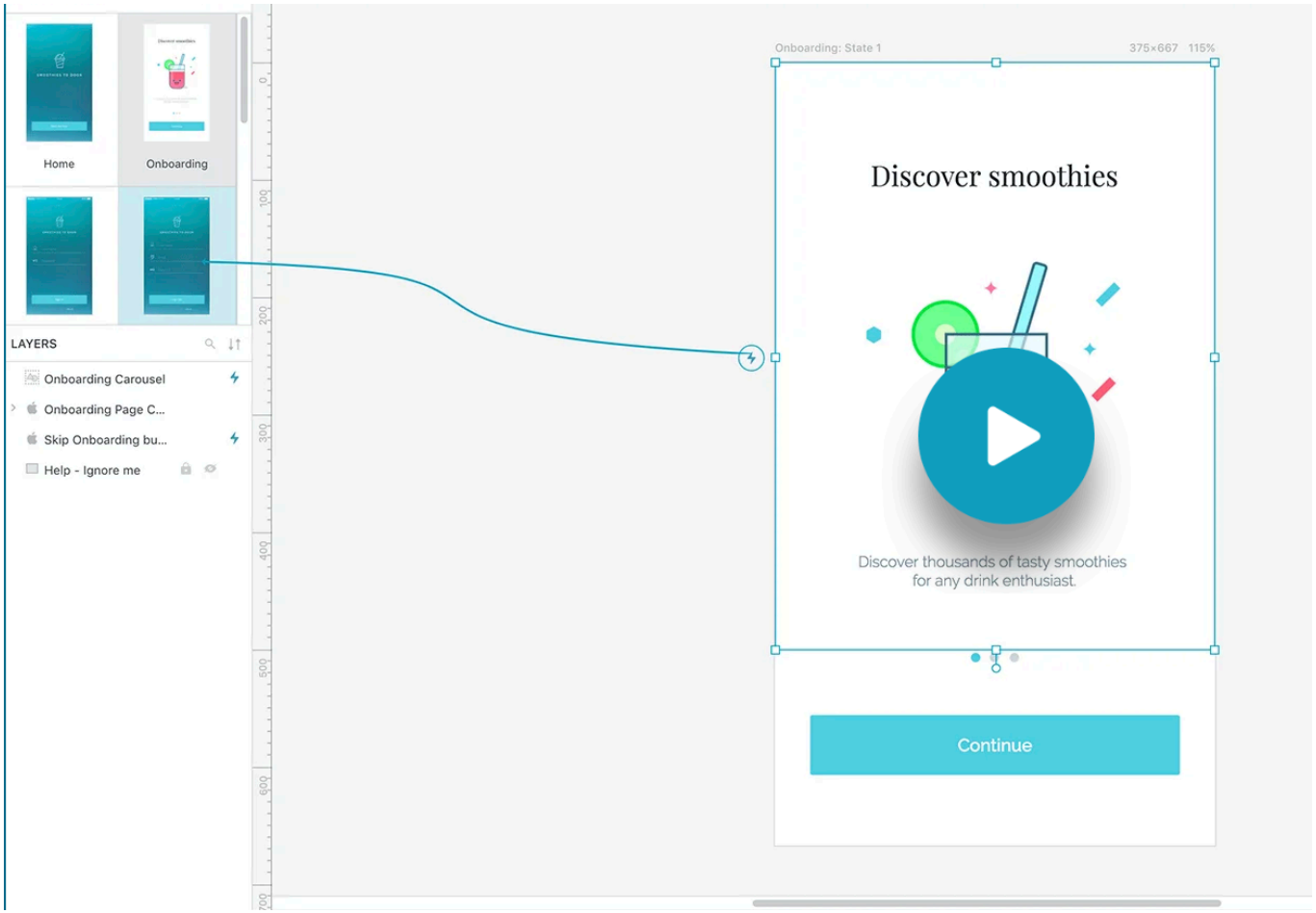
https://en.wikipedia.org/wiki/Paper_prototyping
https://basecamp.com/shapeup/

MOCK UPS / WIRE FRAMES



https://balsamiq.com/wireframes/

INTERACTIVE WALK THROUGHS



https://proto.io/ https://app.moqups.com/

fidelity noun

Save Word

fi·del·i·ty | \ fə-ˈde-lə-tē , fī-\

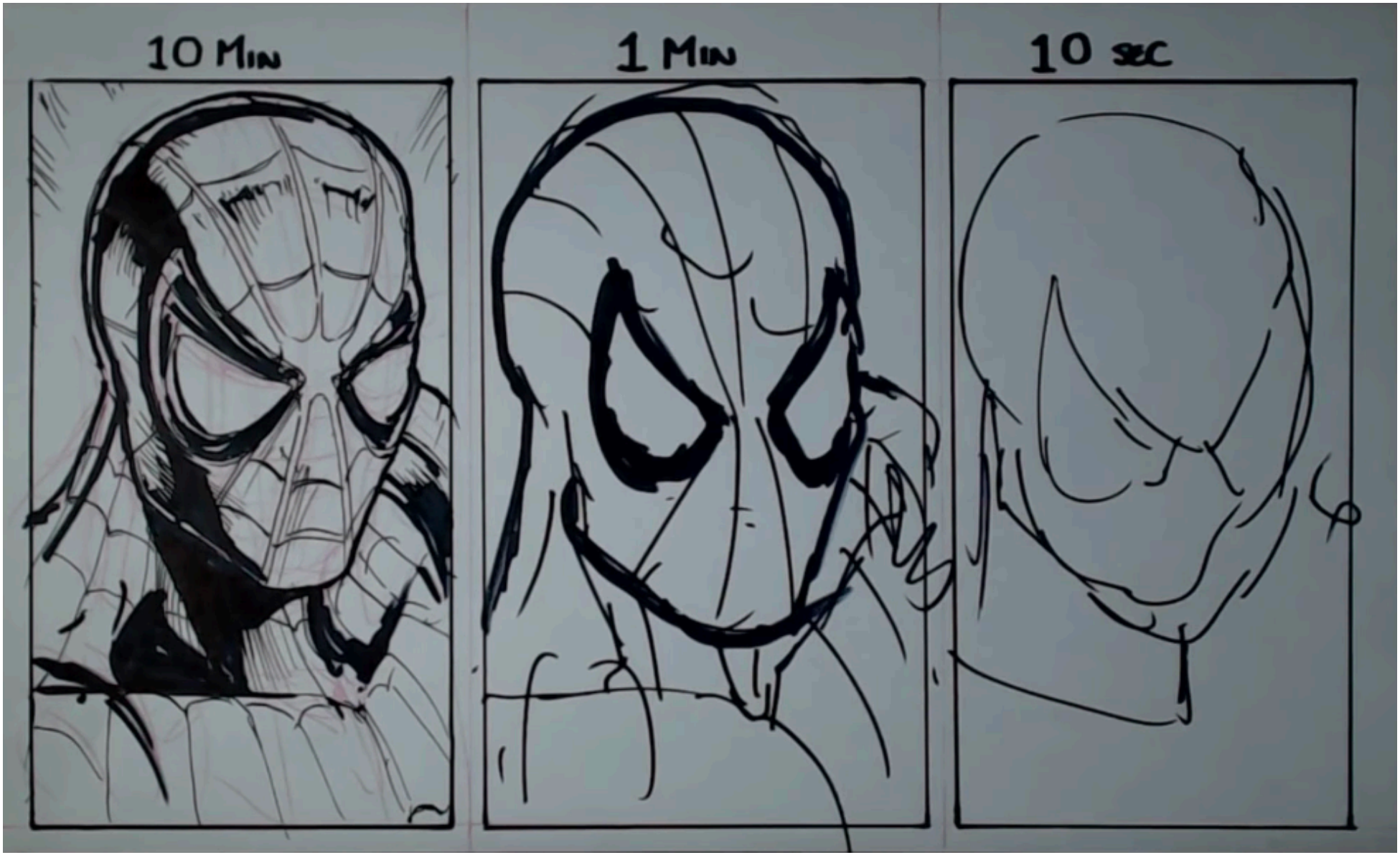
plural **fidelities**

Definition of *fidelity*

1 **a** : ~~the quality or state of being faithful~~
// ~~his *fidelity* to his wife~~

b : accuracy in details : EXACTNESS
// The movie's director insisted on total *fidelity* to the book.

2 : the degree to which an electronic device (such as a record player, radio, or television) accurately reproduces its effect (such as sound or picture)


10 Min
1 Min
10 sec

Utility

Does the system provide the **raw capabilities** to allow the user to achieve their goal?

Usability

Does the system allow the user to **learn and to use** the raw capabilities **easily**?

Aspects of Usability (the "ilities")

LEARNABILITY

EFFICIENCY

ACCEPTABILITY

ERROR HANDLING

...

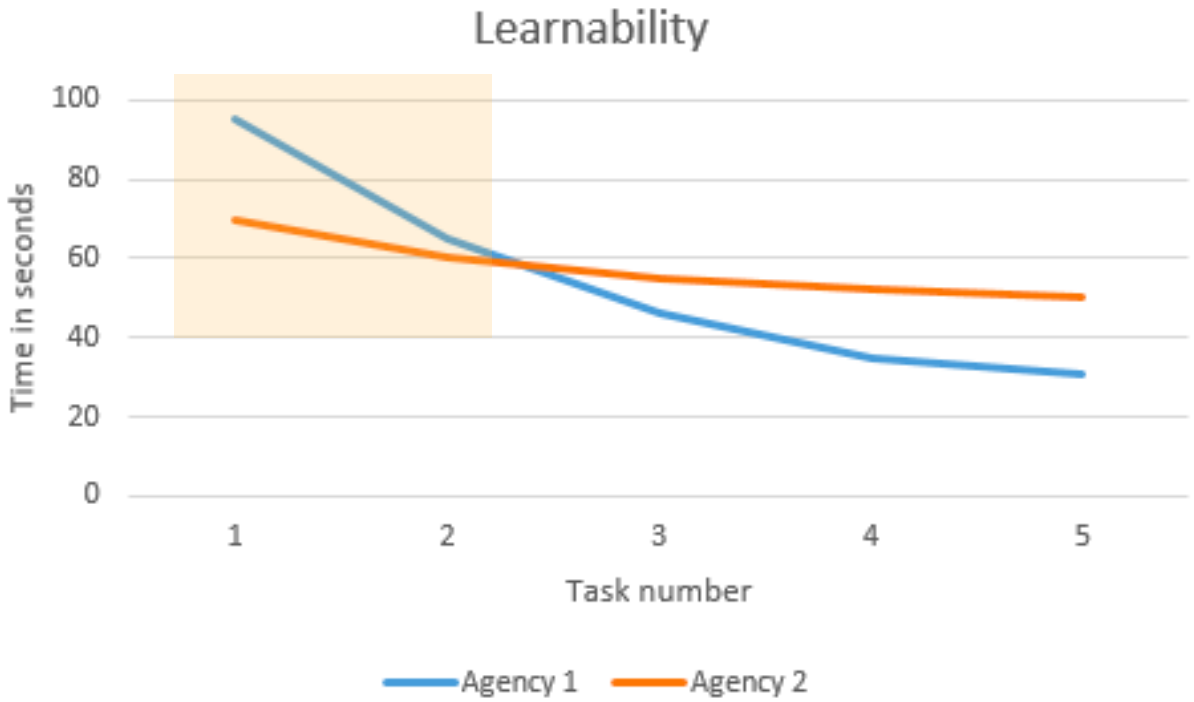
...

Usability Subcharacteristics	Objectives
Learnability	To Learn
Understandability	To Understand
Operability	To Operate / Control
Attractiveness	To Be Attractive
Usability Compliance	To Adhere

https://usabilitygeek.com/the-difference-and-relationship-between-usability-and-learnability/

LEARNABILITY

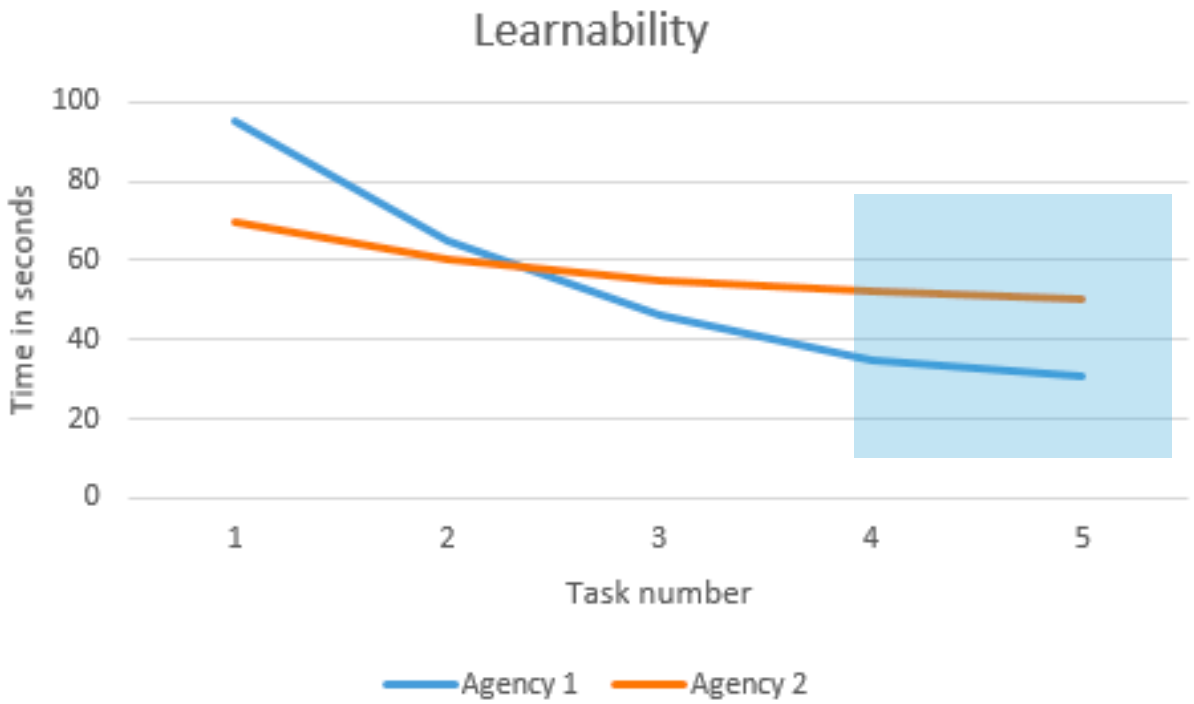
- The **speed** with which a new user can become **proficient** with the system.



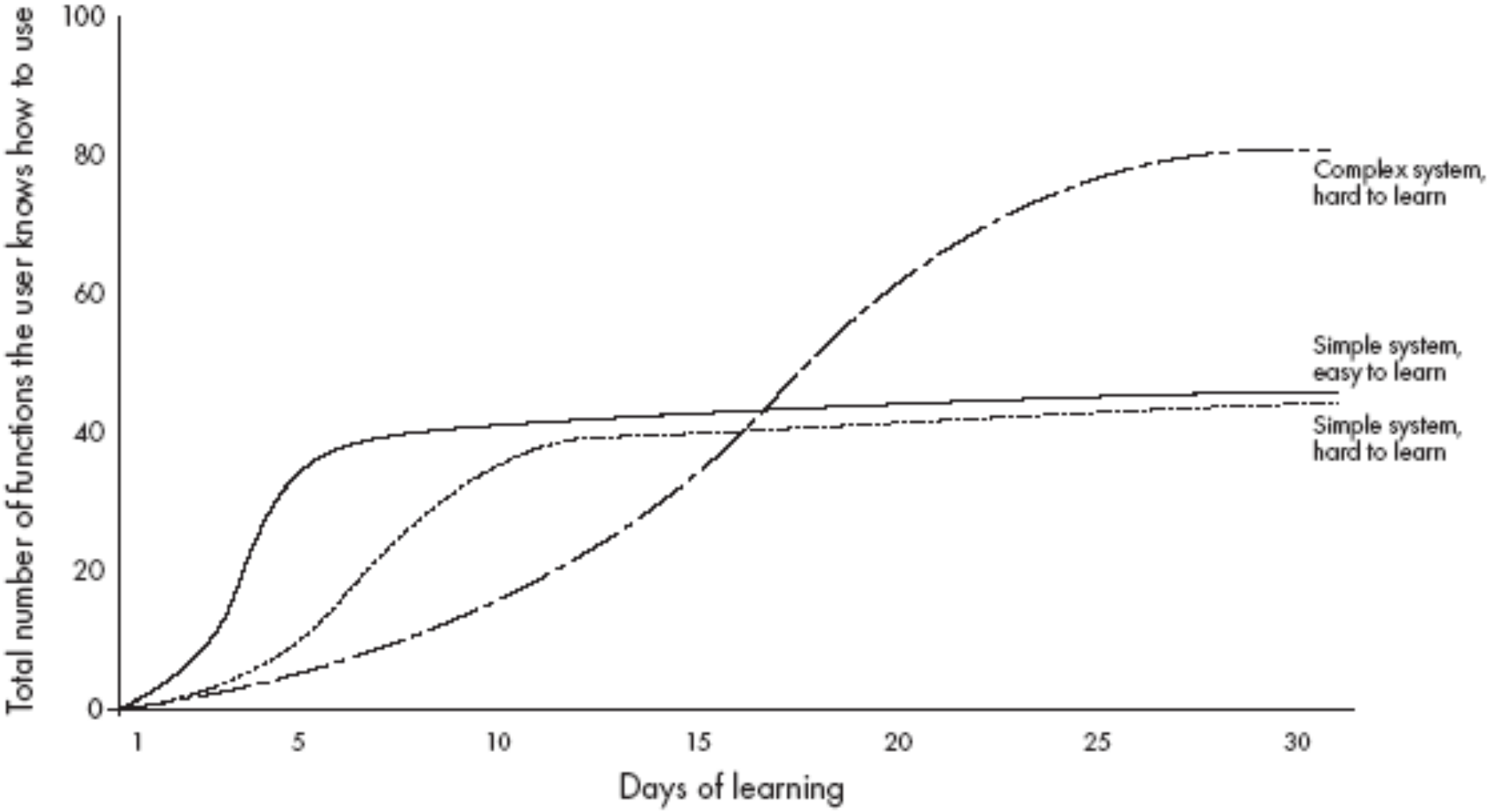
https://ux.stackexchange.com/questions/26554/how-to-measure-learnability

EFFICIENCY

- How **speed** with which an expert user can **do their work**.

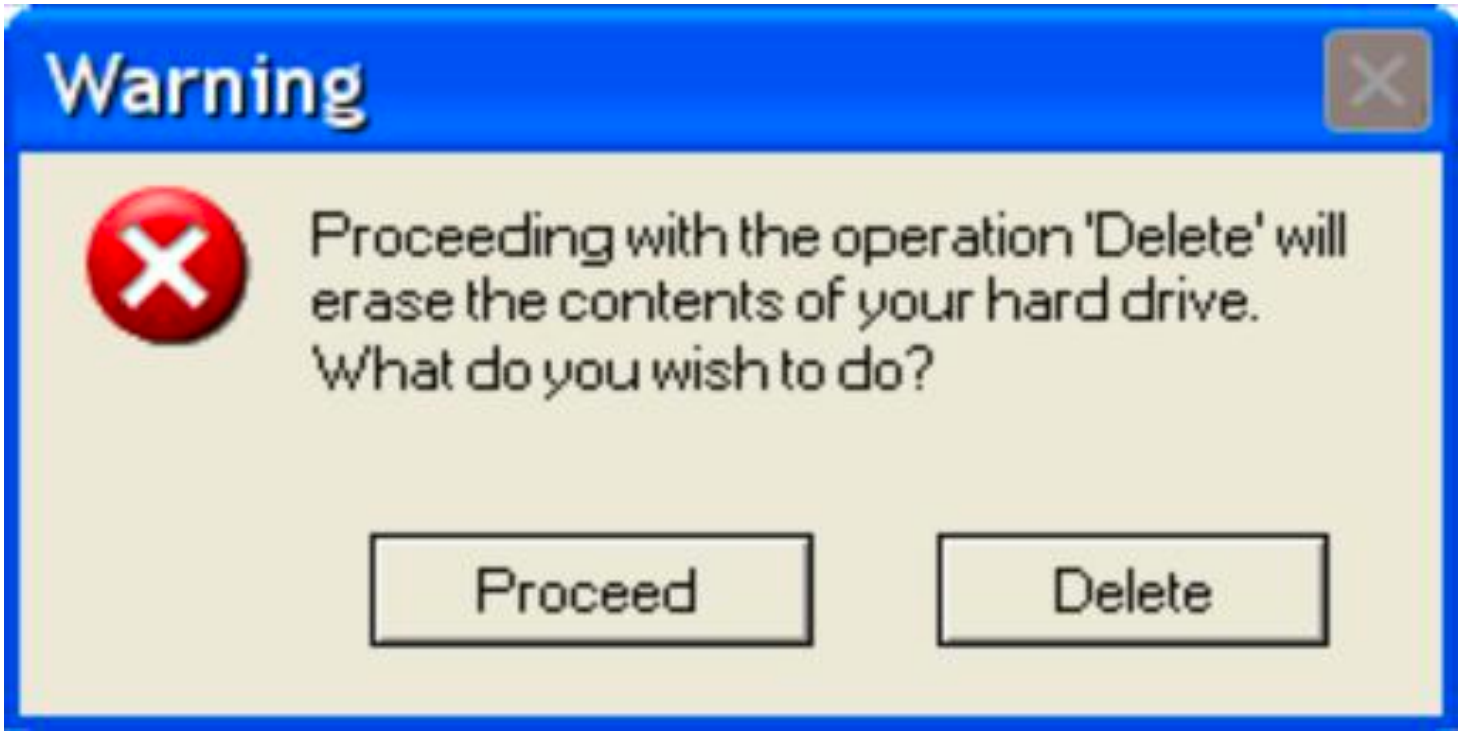


https://ux.stackexchange.com/questions/26554/how-to-measure-learnability


Total number of functions the user knows how to use
100
80
60
40
20
0
1
5
10
15
20
25
30
Days of learning
Complex system, hard to learn
Simple system, easy to learn
Simple system, hard to learn

ERROR HANDLING (SAFETY / SECURITY)

- The extent to which it prevents the user from making errors, detects errors, and helps to correct errors.



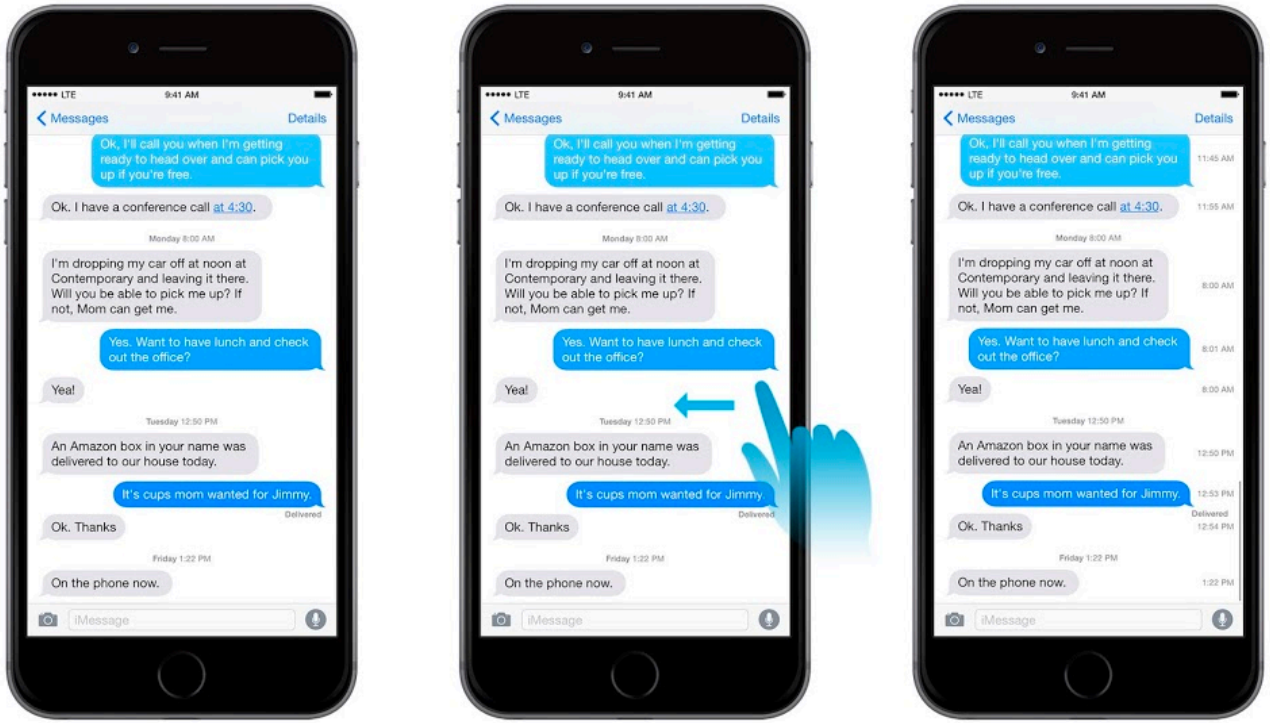
http://www.funcage.com/blog/wp-content/uploads/2011/05/Confusing-Computer-Error.jpg

ACCEPTABILITY

- The extent to which users like the system.

MEMORABILITY

- How easily can I remember how to accomplish a task



DON'T MAKE ME THINK

A COMMON SENSE APPROACH TO WEB USABILITY

by STEVE KRUG

"MANY PEOPLE WHO ENCOUNTER PROBLEMS WITH A SITE TEND TO BLAME THEMSELVES AND NOT THE SITE."

THE USABILITY LAWS

1. DON'T MAKE ME THINK!
2. IT DOESN'T MATTER HOW MANY TIMES I HAVE TO CLICK, AS LONG AS EACH CLICK IS A MINDLESS, UNAMBIGUOS CHOICE.
3. GET RID OF HALF THE WORDS ON EACH PAGE, THEN GET RID OF HALF OF WHATS LEFT.

3 FACTS OF WEB LIFE

1. WE DON'T READ PAGES. WE SCAN THEM.
2. WE DON'T MAKE OPTIMAL CHOICES. WE SATISFICE.
3. WE DON'T FIGURE OUT HOW THINGS WORK. WE MUDDLE THROUGH.

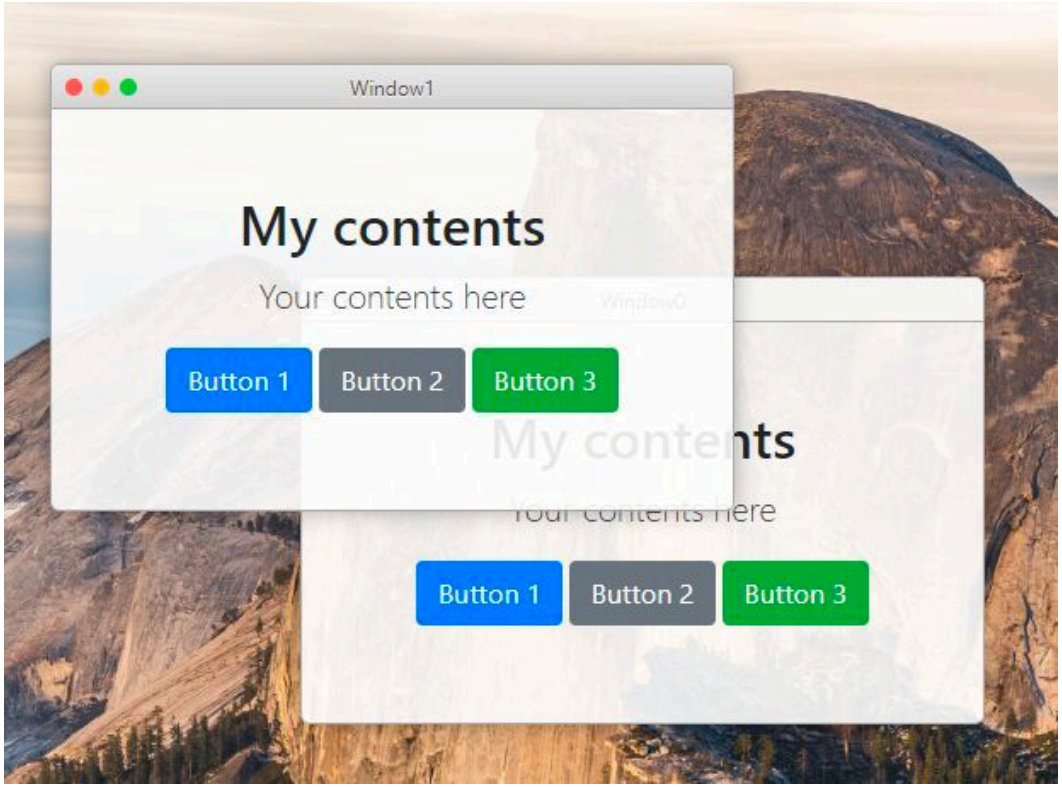
https://blog.prototypr.io/breaking-design-rules-the-3-click-rule-of-ux-a70404909a45

USER INTERFACE

TERMINOLOGY

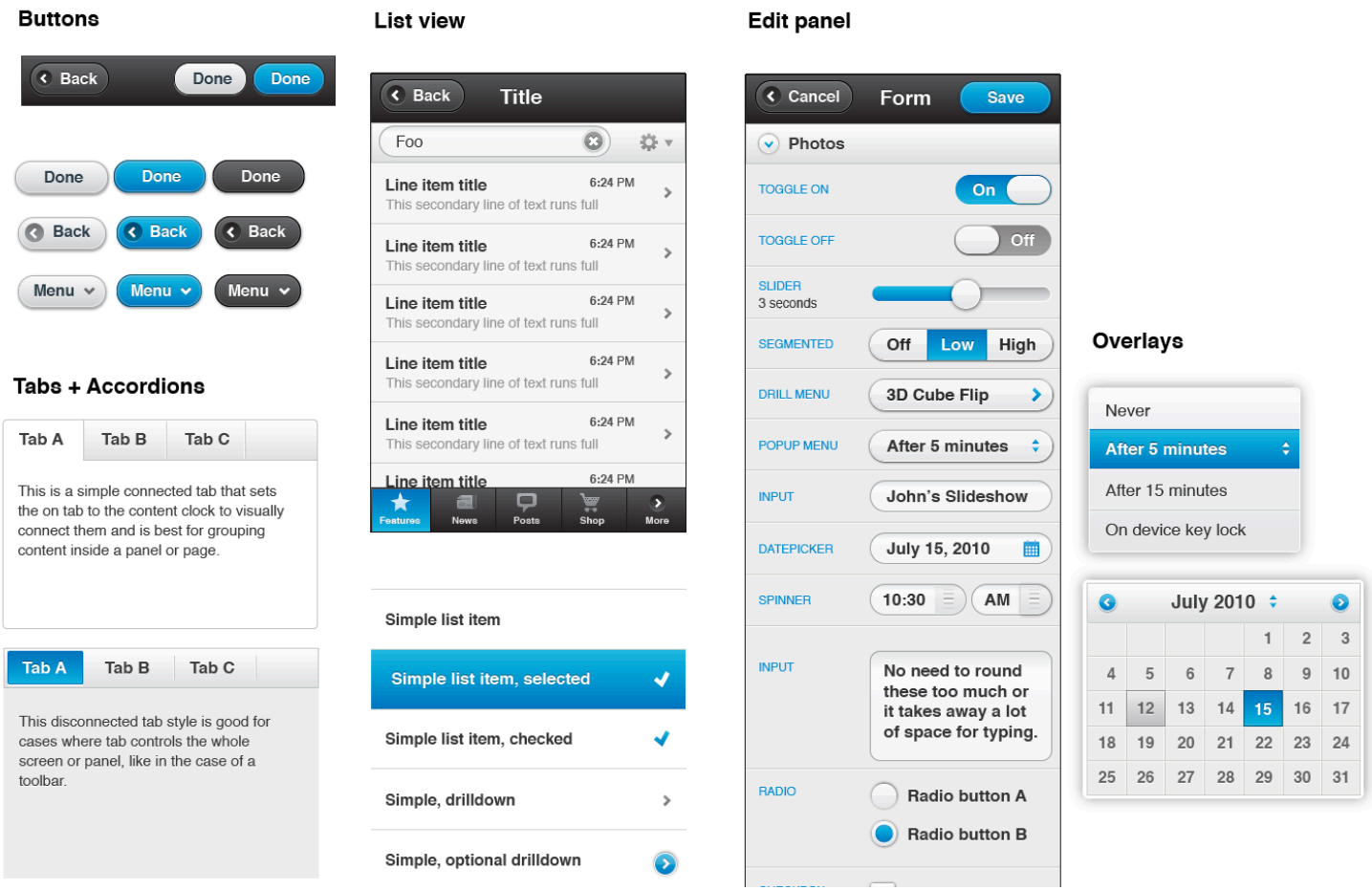
DIALOG

- A specific window with which a user can interact, but which is not the main UI window.



CONTROL OR WIDGET

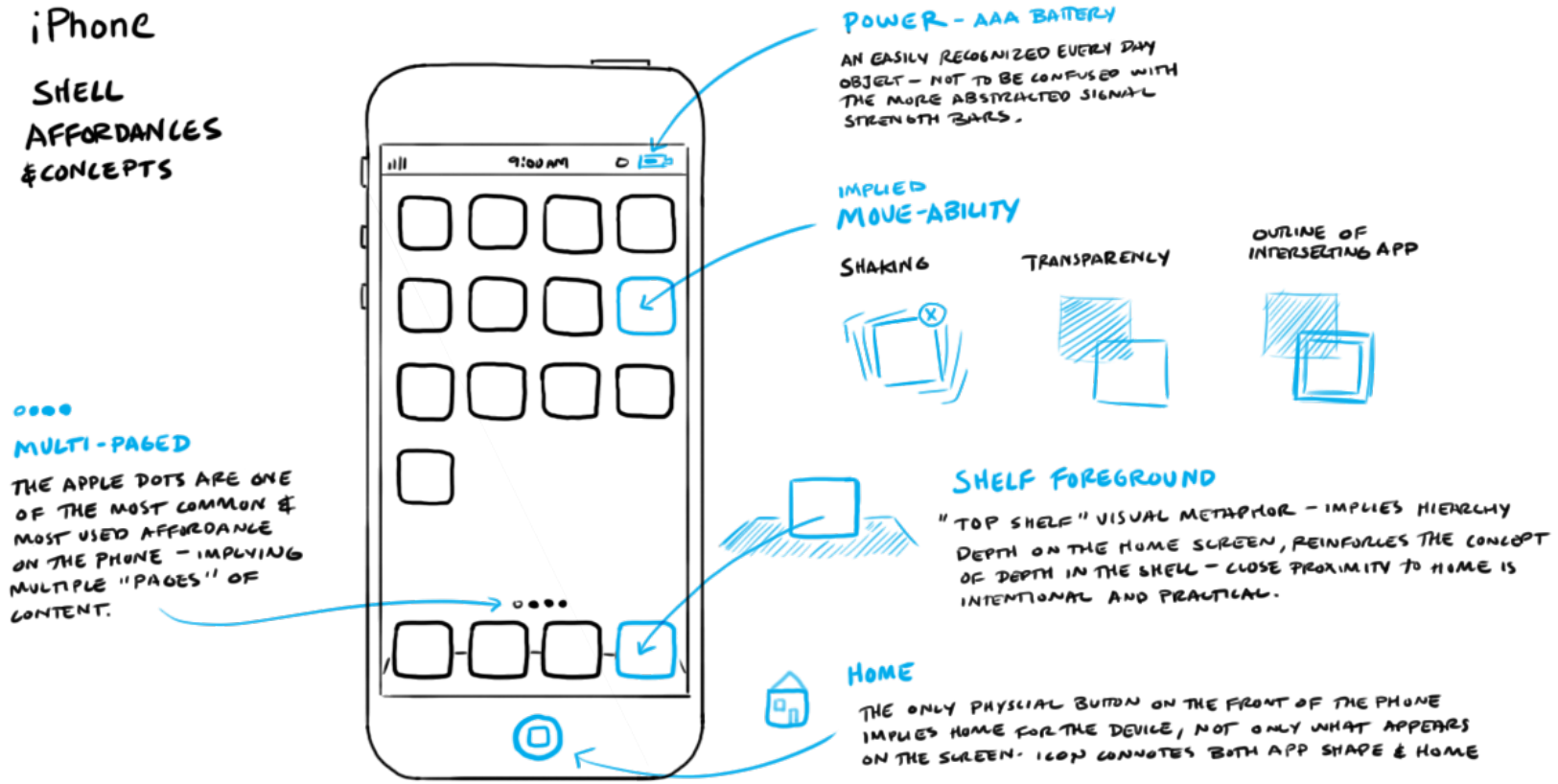
- Specific components of a user interface.



https://jqueryui.pbworks.com/w/page/27358432/Mobile

AFFORDANCE

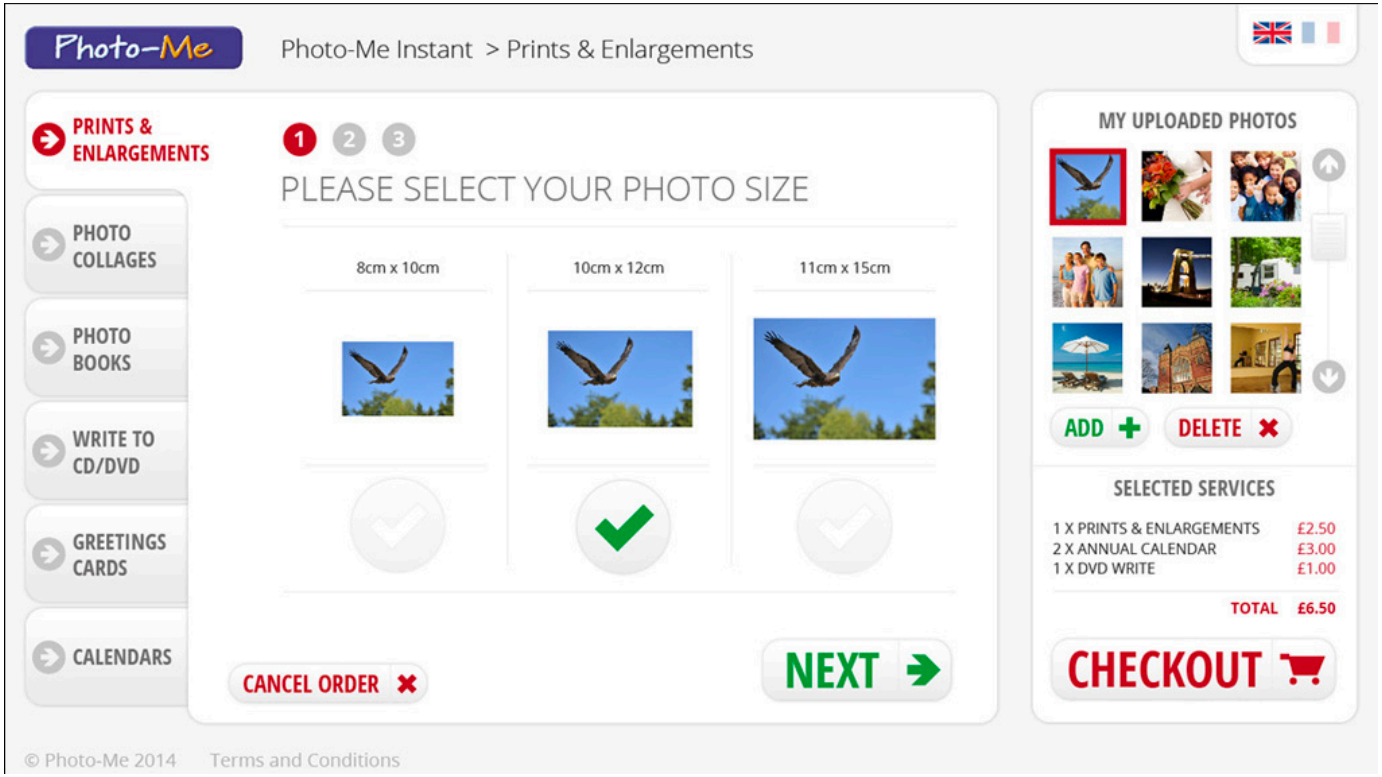
- The set of operations that the user can do at any given point in time.



https://mattink.com/post/66003190793/affordance-shell

STATE

- At any stage in the dialog, the system is displaying certain information in certain widgets, and has a certain affordance.

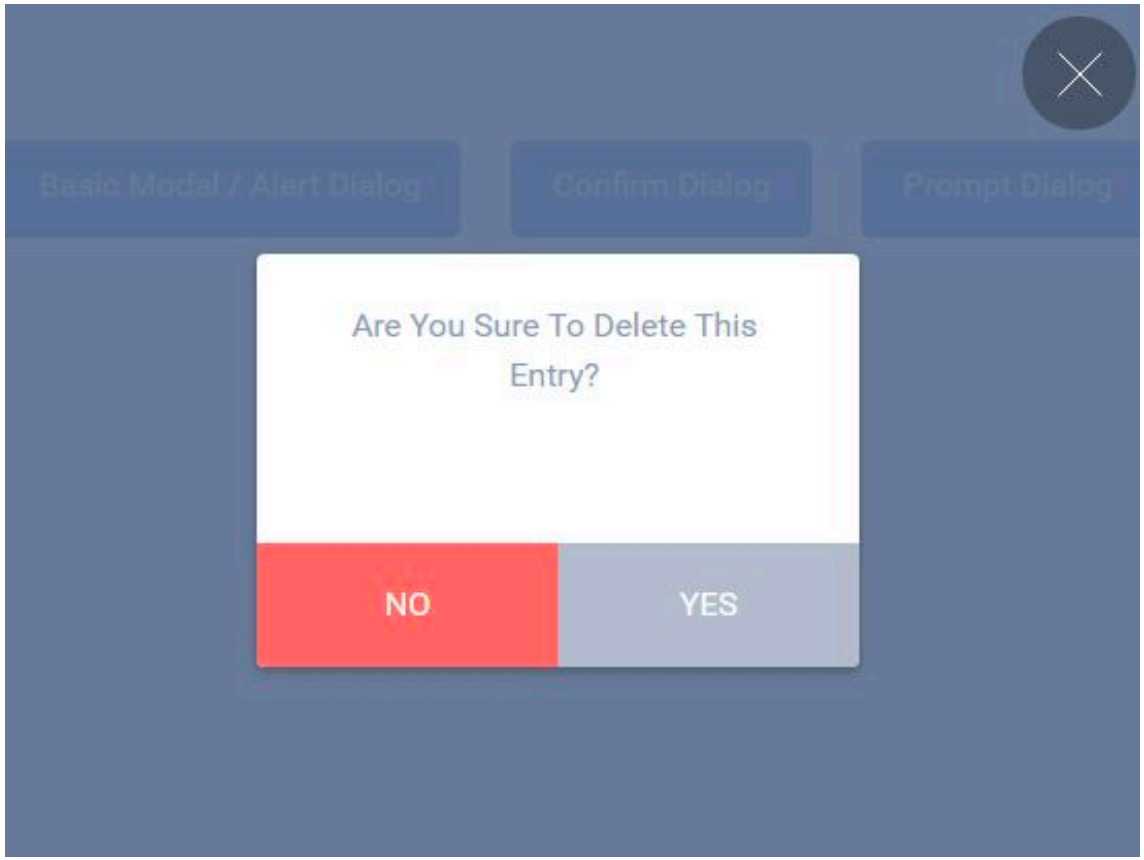


MODE

- A situation in which the UI restricts what the user can do.

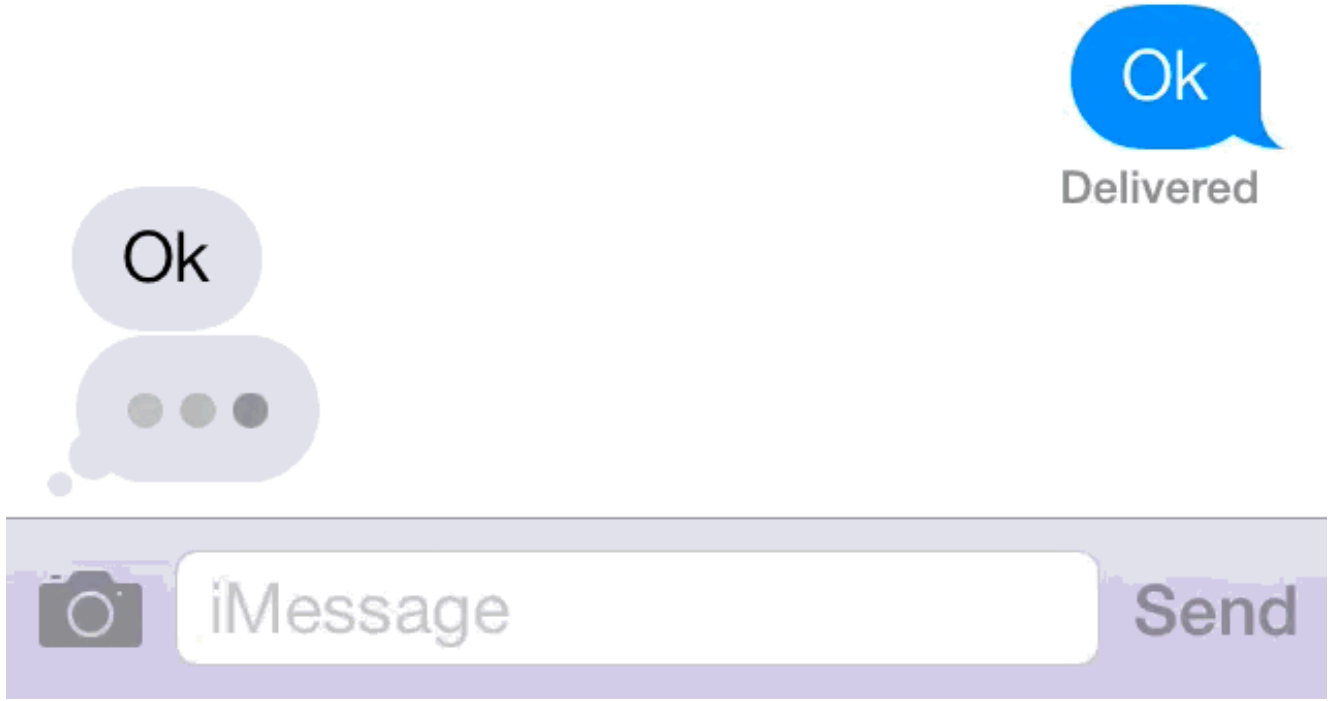
MODAL DIALOG

- A dialog in which the system is in a very restrictive mode.



FEEDBACK

- The response from the system whenever the user does something, is called feedback.



ENCODING TECHNIQUES

- Ways of encoding information so as to communicate it to the user.

USABILITY

PRINCIPLES

1. Do not rely only on usability guidelines

- Always test with users
- Usability guidelines have exceptions
- You can only be confident that a UI is good if you test it successfully with users.

2. Base UI designs on users' tasks

- Perform use case analysis to structure the UI

3. TASKS ARE AS SIMPLE AS POSSIBLE.

- The sequences of actions for a task as simple as possible
- Reduce reading and user manipulation
- Limit user navigation to do subsequent steps

4. User always knows what can and should do next

- Ensure that the user can see what commands are available and are not available.
- Make the most important commands stand out.

5. Provide great feedback

- Inform users of the **progress of operations** and of their location as they navigate.
- When something goes wrong explain the situation in adequate detail and **help the user to resolve the problem**

6. Allow for undos

- Ensure that the user can always get out, go back or undo an action.
- Ensure that all operations can be undone.
- Ensure it is easy to navigate back to where the user came from.

7. Make it seem fast enough

- Ensure that response time is adequate.
- Users are very sensitive to slow response time
- They compare your system to others.
- Keep response time less than a second for most operations.
- Warn users of longer delays and inform them of progress.

8. Use understandable encoding techniques.

- Choose encoding techniques with care.
- Use labels to ensure all encoding techniques are fully understood by users.

Some encoding techniques

- Text and fonts
- Icons
- Photographs
- Diagrams and abstract graphics
- Colours
- Grouping and bordering
- Spoken words
- Music
- Other sounds
- Animations and video
- Flashing

9. Ensure that the UI's appearance is uncluttered.

- Avoid displaying too much information.
- Organize the information effectively.

10. Consider the needs of different groups of users.

- Accommodate people from different locales and people with disabilities.
- Ensure that the system is usable by both beginners and experts.

11. Provide all necessary help.

- Organize help well.
- Integrate help with the application.
- Ensure that the help is accurate.

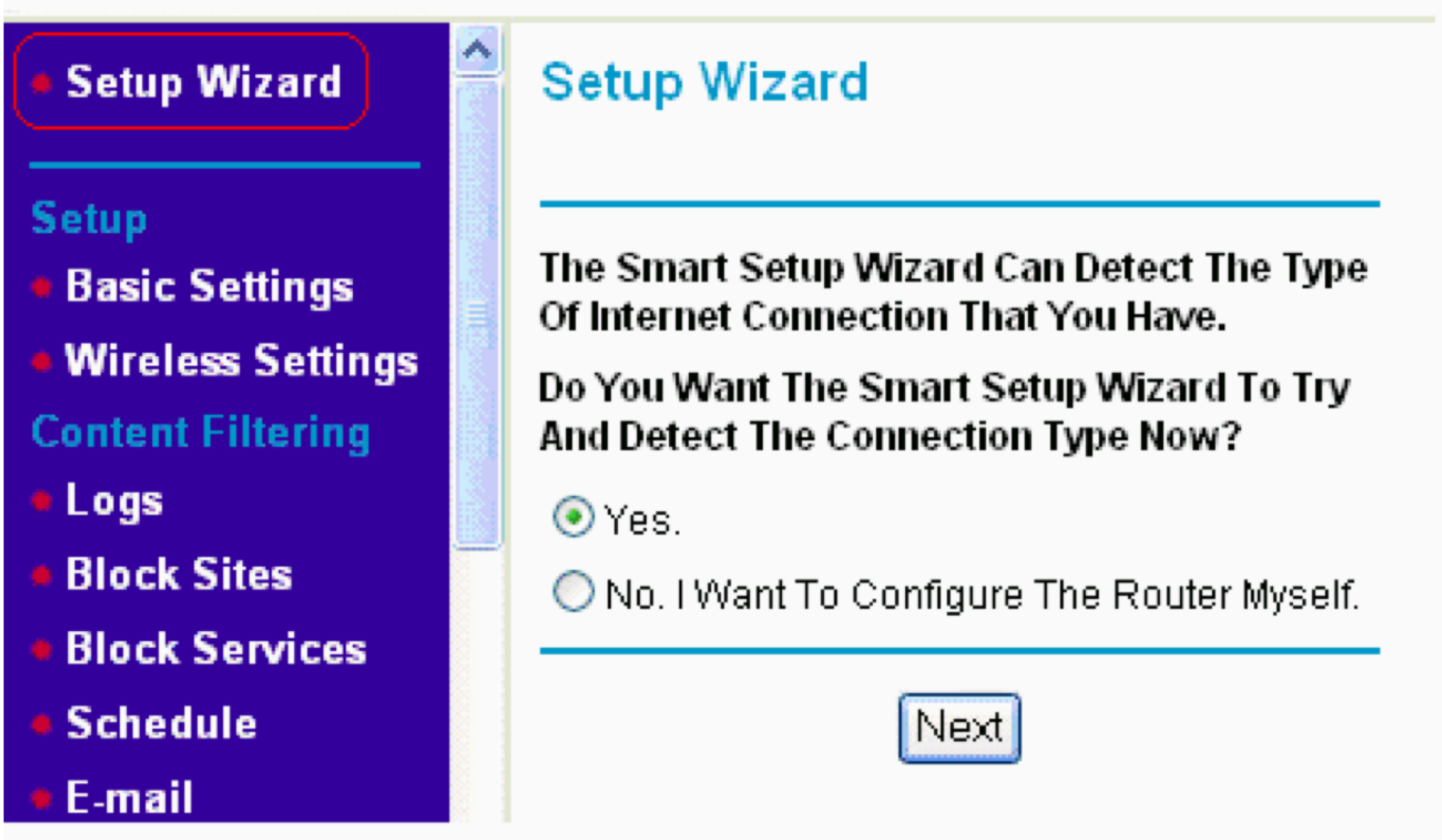
12. BE CONSISTENT.

- Use similar layouts and graphic designs throughout your application.
- Follow look-and-feel standards.
- Consider mimicking other applications.

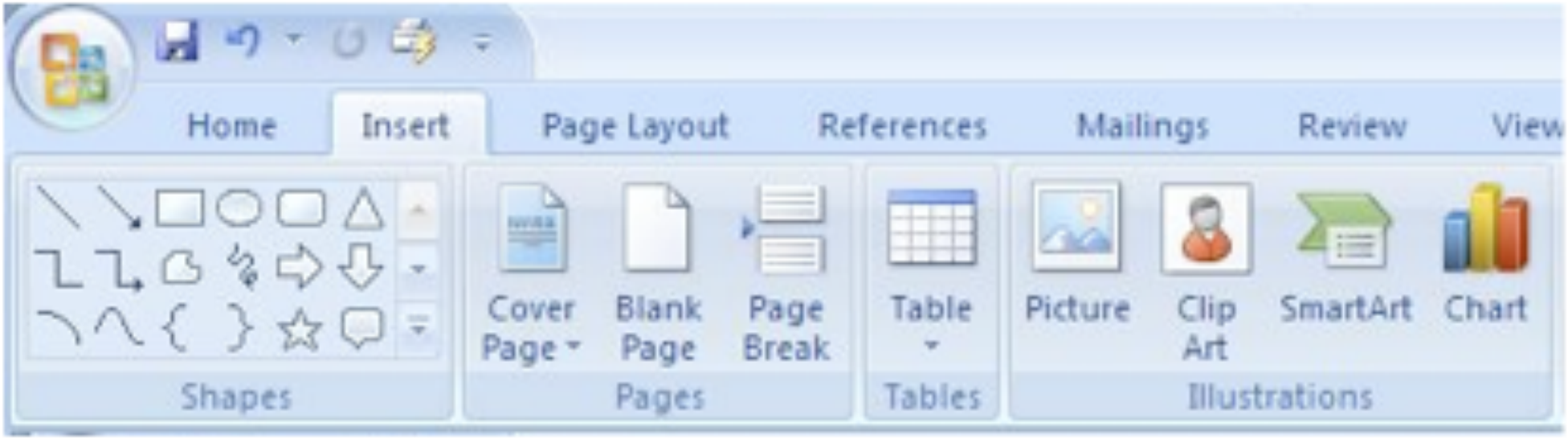
USER INTERFACE

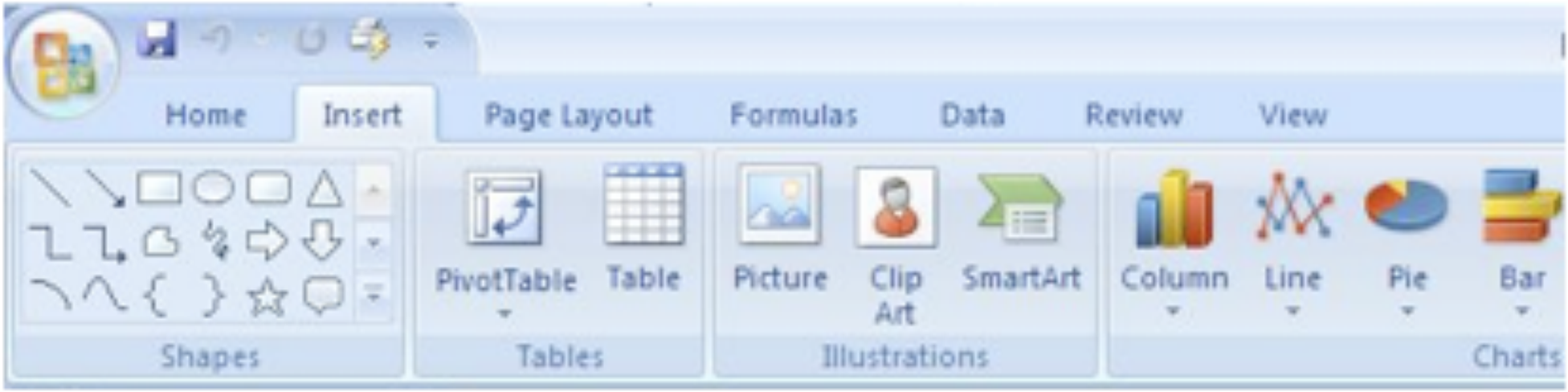
EXAMPLES

USER IN CONTROL

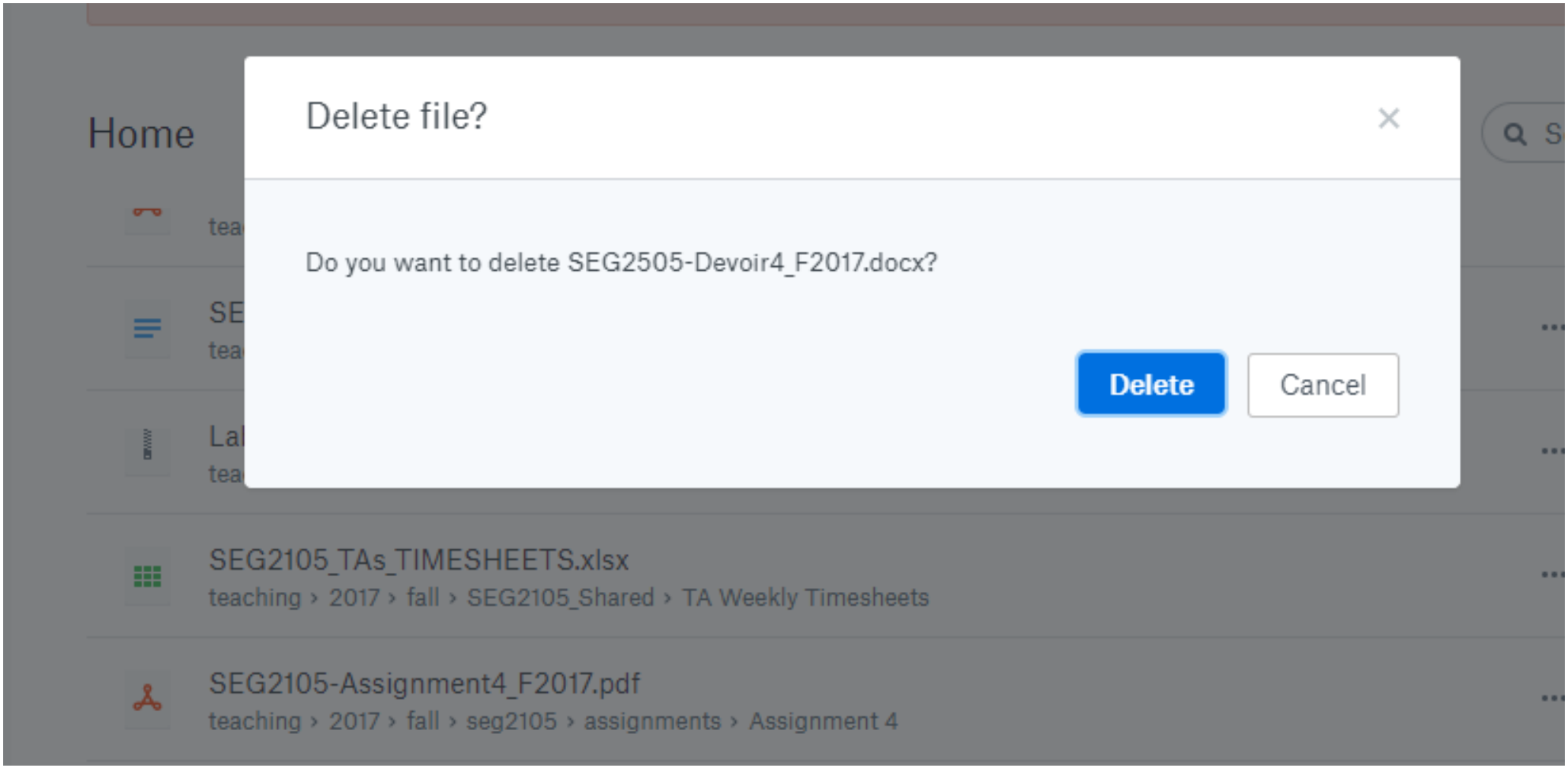


INCONSISTENCY

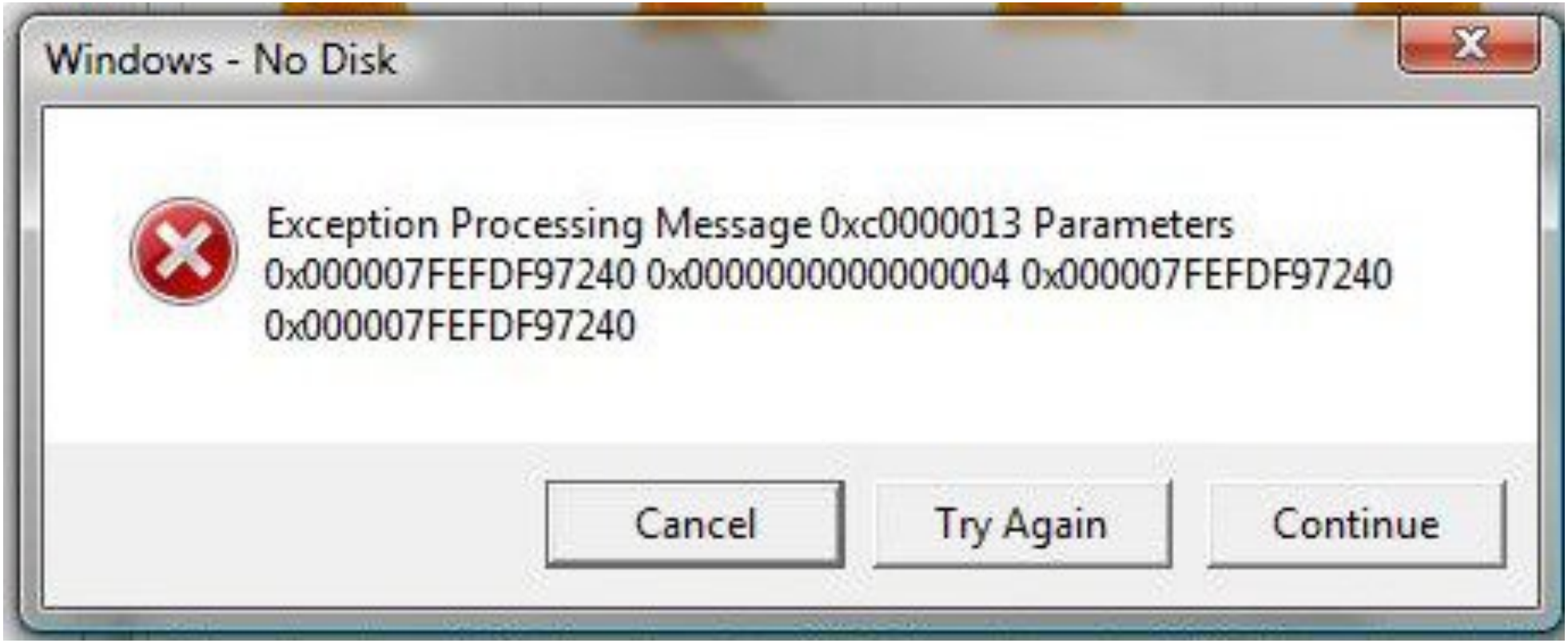




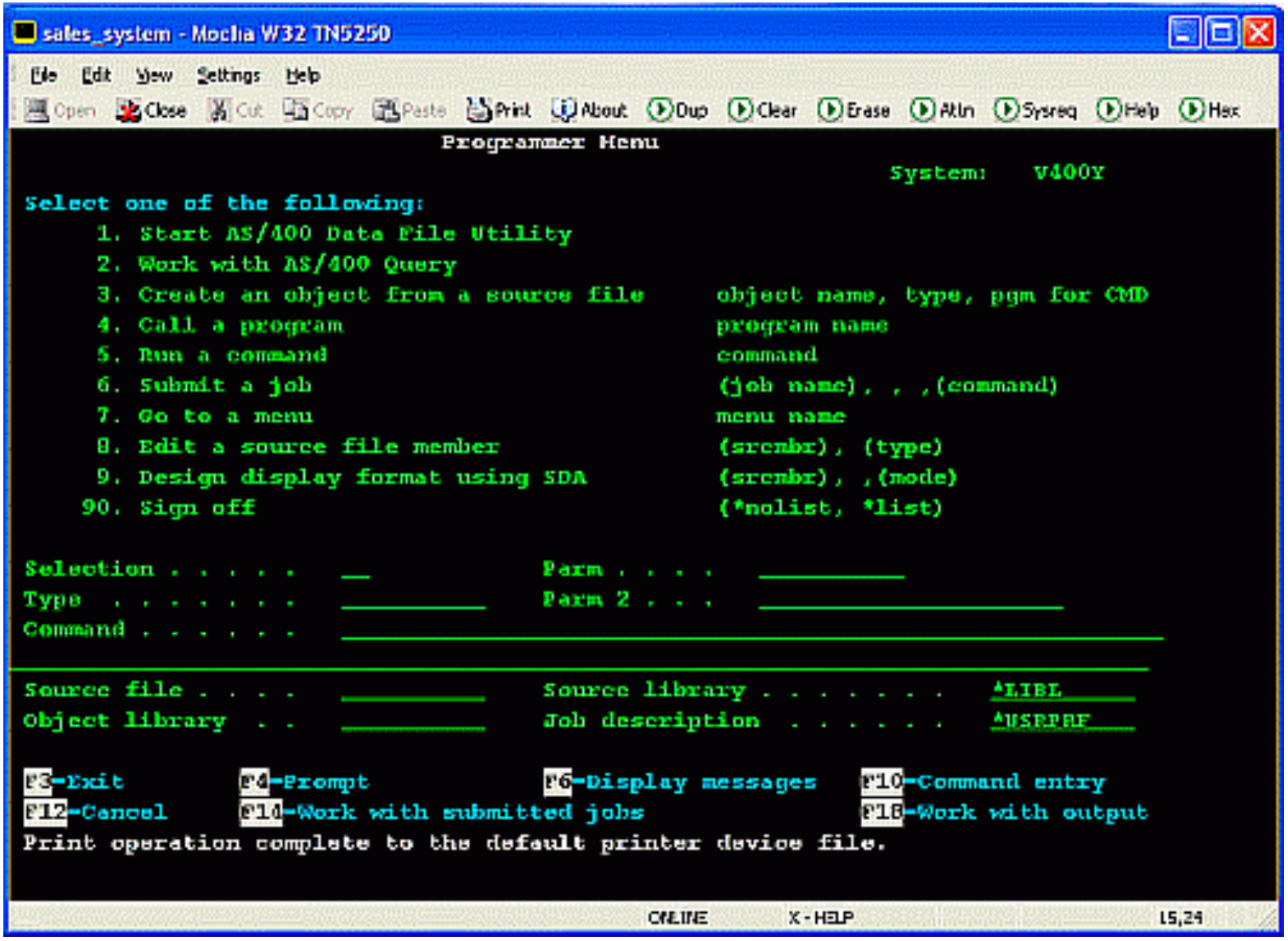
PREVENTING ERRORS



LAZY ERRORS

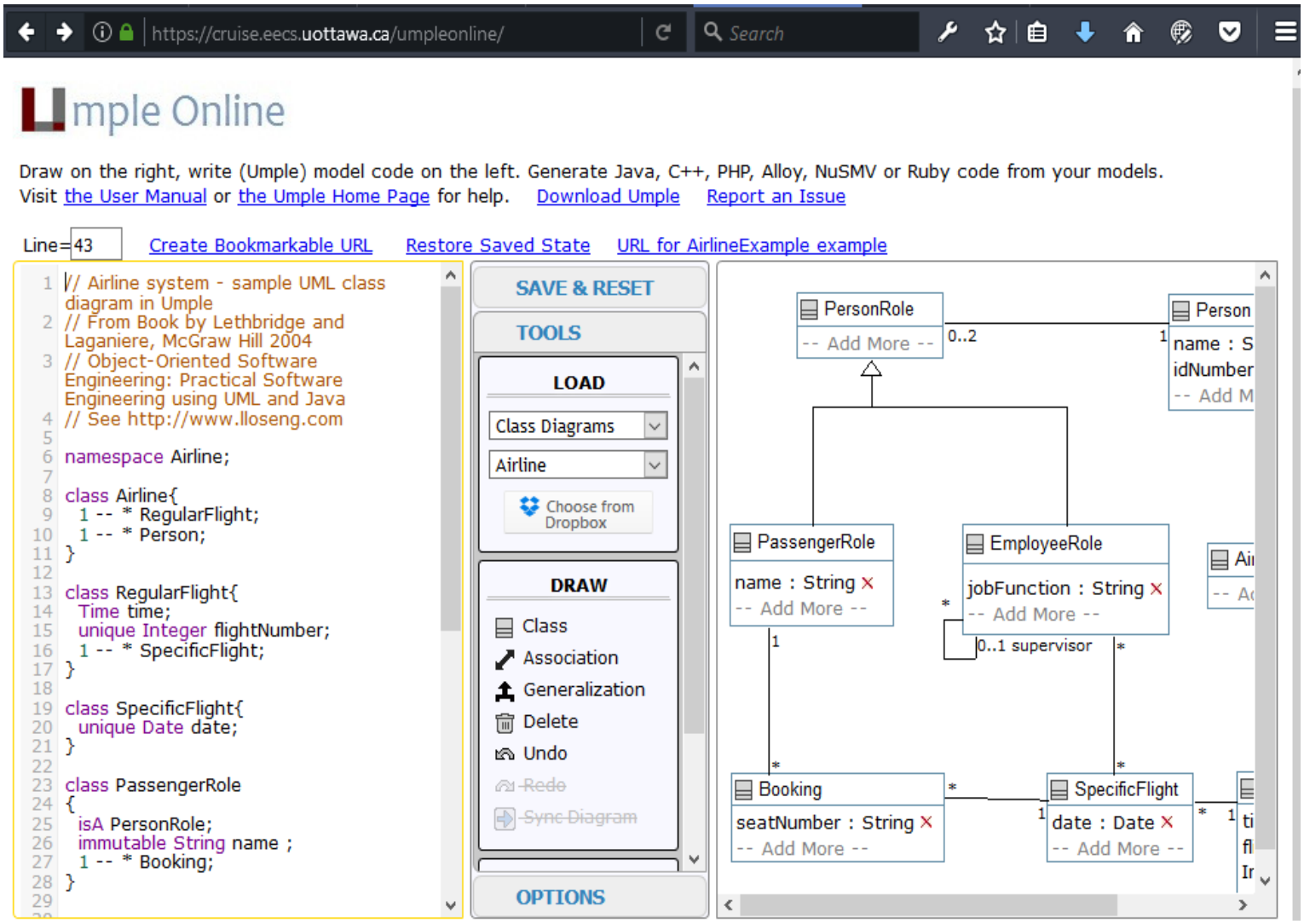

Windows - No Disk
Exception Processing Message 0xc0000013 Parameters
0x000007FEFDF97240 0x0000000000000004 0x000007FEFDF97240
0x000007FEFDF97240
Cancel
Try Again
Continue

BAD UI, GOOD UX



Good UI, with bad UX

GOOD OR BAD?


https://cruise.eecs.uottawa.ca/umpleonline/
Search
Umple Online
Draw on the right, write (Umple) model code on the left. Generate Java, C++, PHP, Alloy, NuSMV or Ruby code from your models.
Visit the User Manual or the Umple Home Page for help. Download Umple Report an Issue
Line=43 Create Bookmarkable URL Restore Saved State URL for AirlineExample example
1 // Airline system - sample UML class diagram in Umple
2 // From Book by Lethbridge and Laganiere, McGraw Hill 2004
3 // Object-Oriented Software Engineering: Practical Software Engineering using UML and Java
4 // See http://www.lloseng.com
5
6 namespace Airline;
7
8 class Airline{
9 1 -- * RegularFlight;
10 1 -- * Person;
11 }
12
13 class RegularFlight{
14 Time time;
15 unique Integer flightNumber;
16 1 -- * SpecificFlight;
17 }
18
19 class SpecificFlight{
20 unique Date date;
21 }
22
23 class PassengerRole
24 {
25 isA PersonRole;
26 immutable String name ;
27 1 -- * Booking;
28 }
29
SAVE & RESET
TOOLS
LOAD
Class Diagrams
Airline
Choose from Dropbox
DRAW
Class
Association
Generalization
Delete
Undo
Redo
Sync Diagram
OPTIONS
PersonRole
-- Add More --
0..2
1
Person
name : S
idNumber
-- Add M
PassengerRole
name : String
-- Add More --
EmployeeRole
jobFunction : String
-- Add More --
0..1 supervisor
Booking
seatNumber : String
-- Add More --
SpecificFlight
date : Date
-- Add More --

EVALUATING

USER INTERFACES

Heuristic evaluation

1. Pick a use case
2. For each UI component during that use case
   - Study, and identify defects
3. For each defect
   - Describe it
   - Ideas on fixing it

EVALUATION BY OBSERVATION OF USERS

- Select representational users
- Select a use case
- Write instructions about each scenarios
- Arrange evaluation sessions
  - Explain the purpose
  - Record session (is possible)
  - Talk with user during the task
- De-brief after each task
- Take notes of difficulties
- Recommend changes

DIFFICULTIES AND RISKS

USER INTERFACE DESIGNS

Users differ widely

- Account for differences among users when you design the system.
- Design it for internationalization.
- When you perform usability studies, try the system with many different types of users.

UI Tech changes quickly

- Stick to simpler UI frameworks
- Avoid fancy and unusual UI designs

UI Implementation can take majority of work

- Make UI design an integral part
- Allocate time for many iterations of
  - prototyping, and
  - evaluation

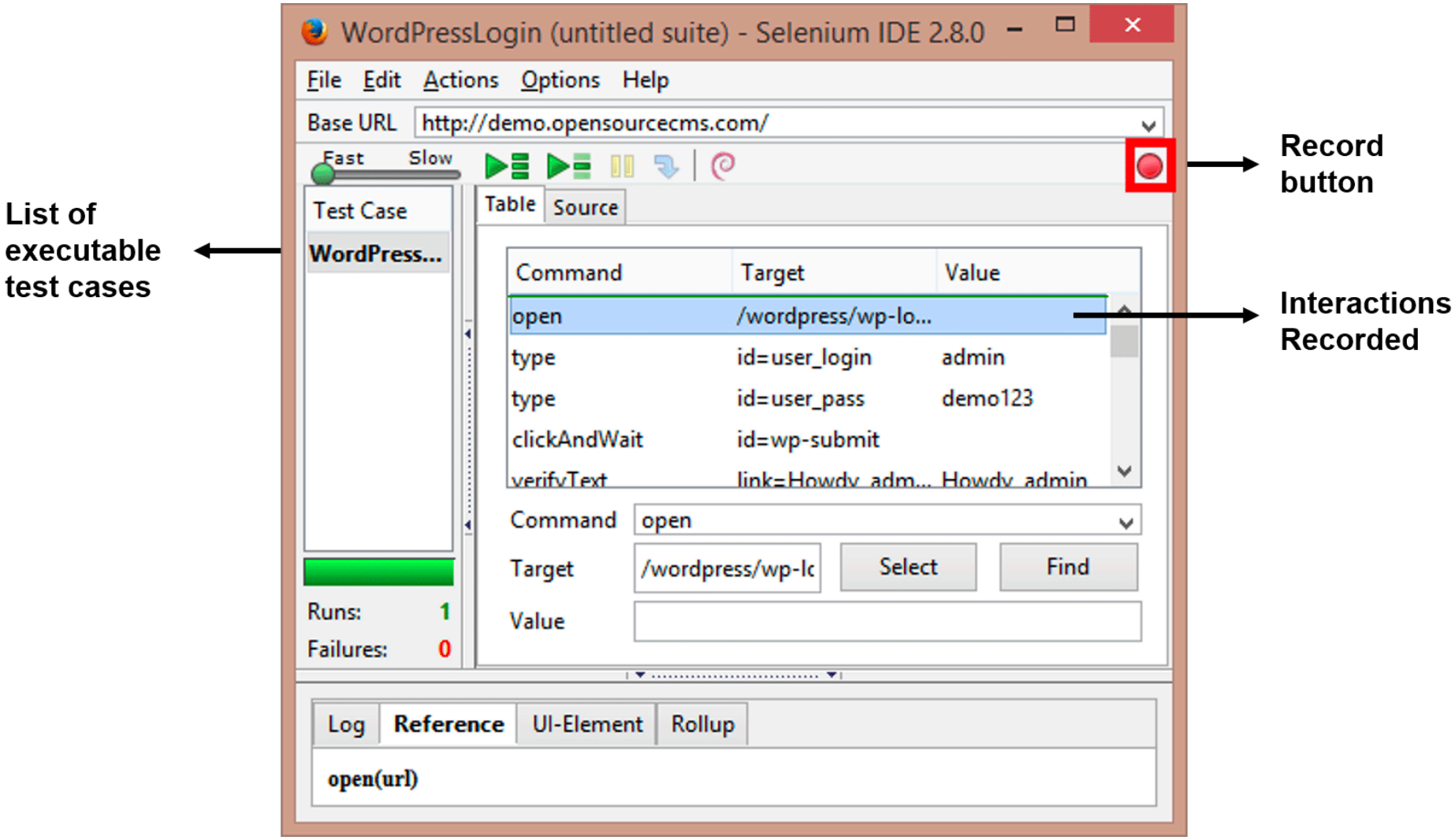
Developers underestimate weaknesses in a GUI

- PEBKAC is developer arrogance
- Ensure training in UI development.
- Always test with users.
- Study the UIs of other software.

UI TESTING

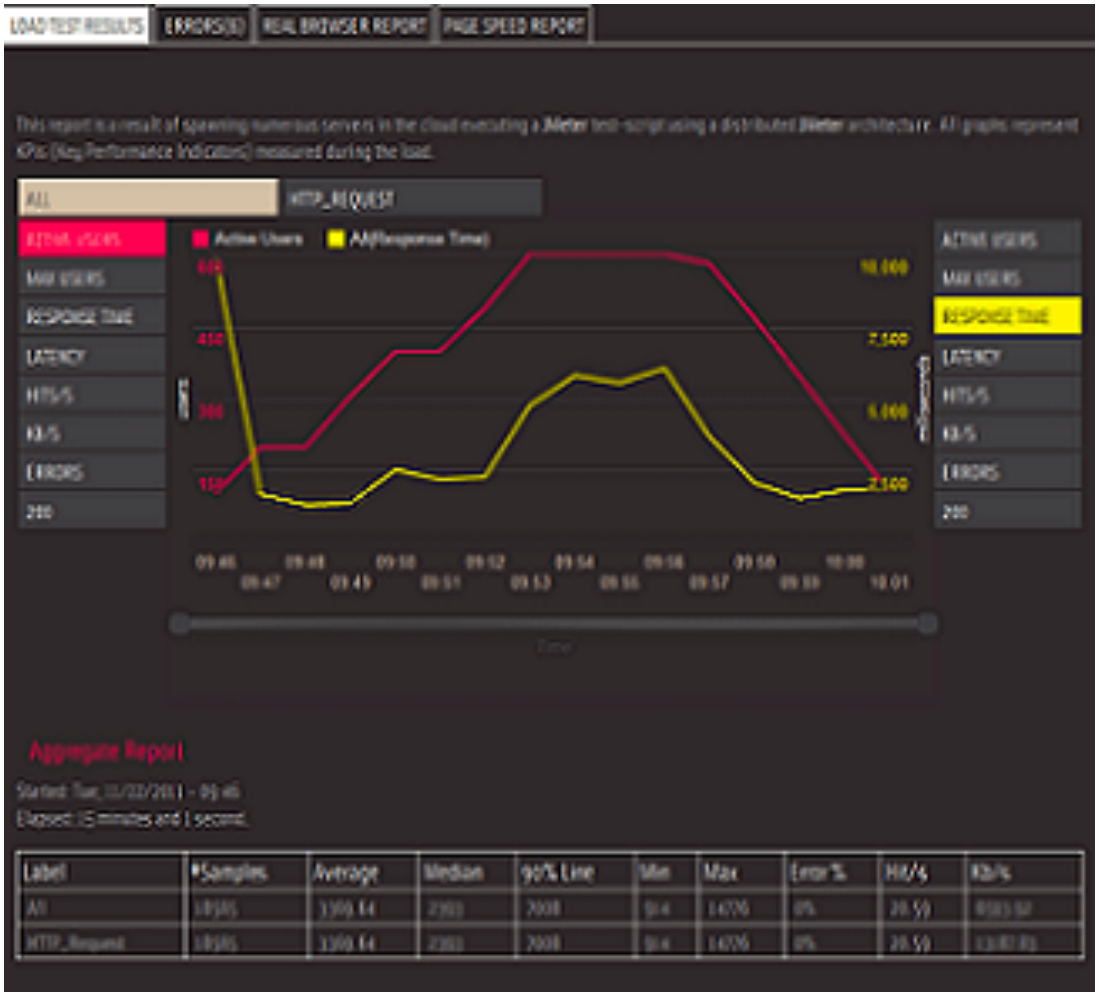
TOOLS

SELENIUM BROWSER AUTOMATION



https://www.edureka.co/blog/what-is-selenium/

BLAZEMETER + JMETER



https://www.techmaish.com/jmeter-cloud-testing-with-blazemeter/

https://www.blazemeter.com/