UI, UX and Design Thinking

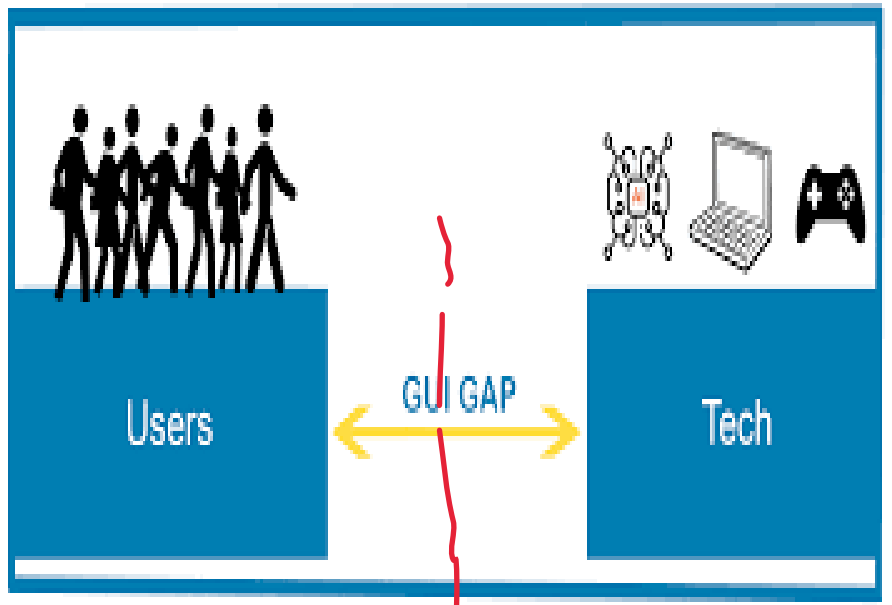
CSE608

Advanced Software Engineering

Dr Islam ElMaddah

Why UI?

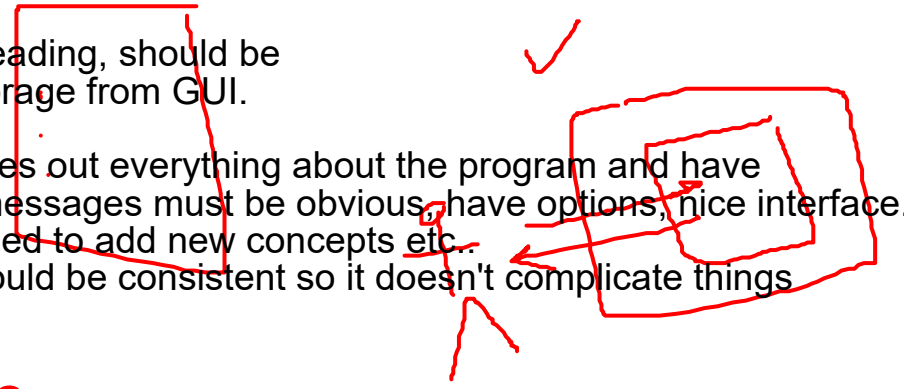
- Good UIs are critical to success
- UI programming is
  - easy (sophisticated algorithms not required)
  - straightforward (can immediately correct mistakes)
  - fun (results are immediately visible)
  - rational (apply simple rules)
- UI design is not graphic design



Cardinal axiom

The program shouldn't be missleading, should be straightforward otherwise its shorage from GUI.

The user is happy when he figures out everything about the program and have the necessary tools to control, messages must be obvious, have options, nice interface. Don't load user's memory, no need to add new concepts etc..
Every aspect of the program should be consistent so it doesn't complicate things



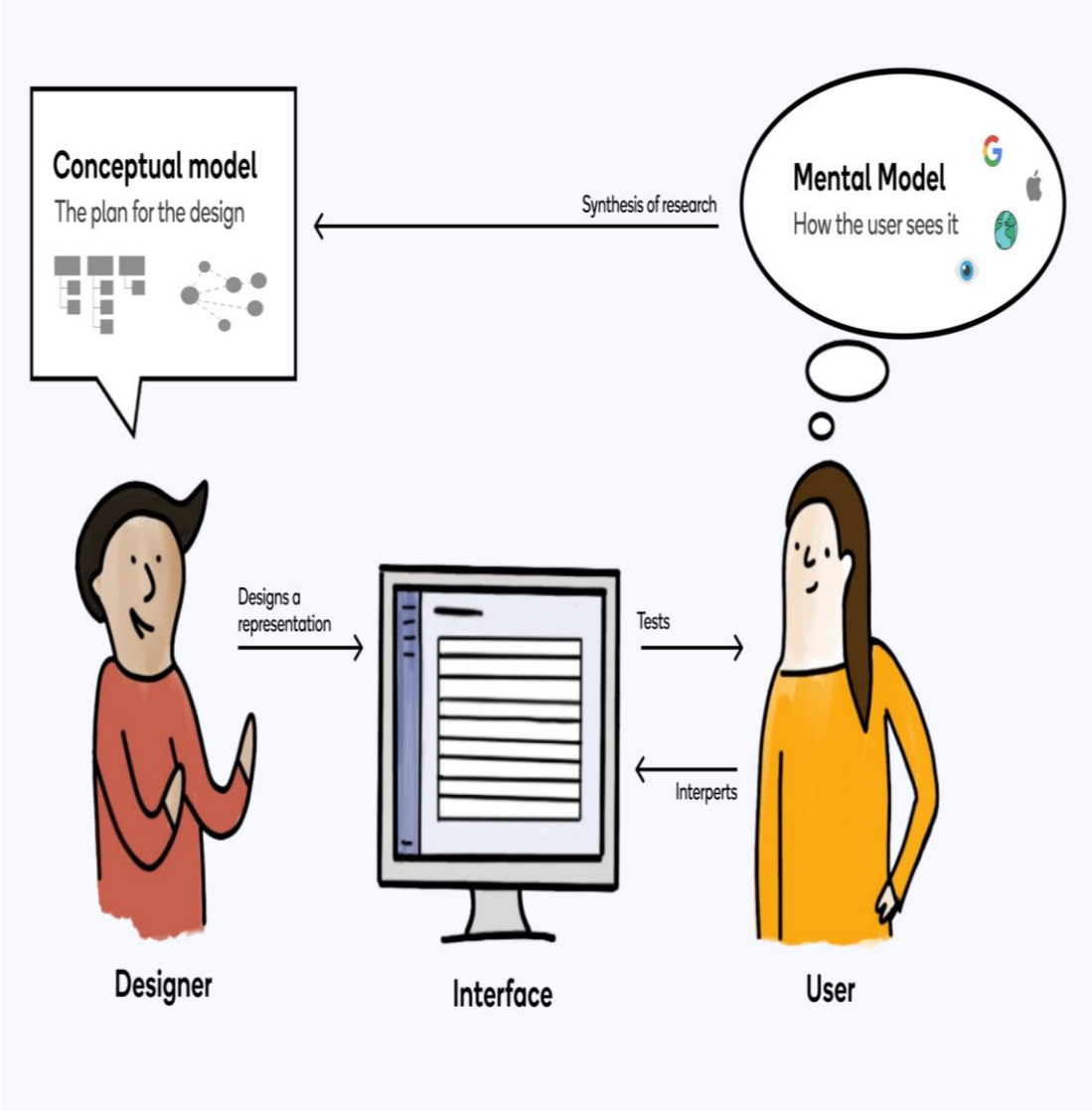
- "A user interface is well-designed when the program behaves exactly how the user thought it would." – Joel Spolsky
  - user is happy = user in control = S/W correctly interprets user's actions
  - loss of control → depression, frustration ("Learned Helplessness" [Seligman])
- All the other rules are just corollaries:

Golden rules: place user in control, reduce user's memory load, make interface consistent

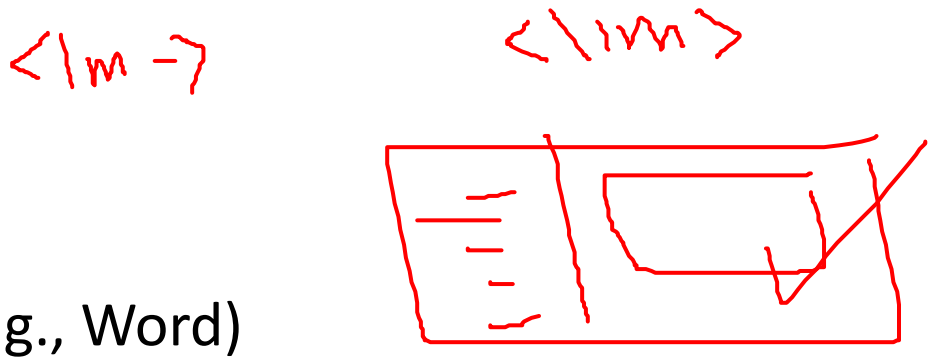
Designer must understand how the user interpret this program.
The program is like a common language between the user's and designers.
We should converse the people with their own language.
Our functionality can be innovative, but no need to be complex.
For metaphor, something from real life is trash can -> so user can understand its a delete operation.

User and program models

- User model: User's idea of what's happening
- Program model: Program's idea of what's happening (i.e., what's *actually* happening)
- Successful UI when program model corresponds to user model
  - Speak user's language
  - Follow real-world conventions, make information appear in natural and logical order
  - Use metaphors from real world



Example



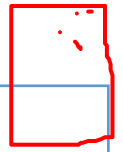
- Pictures in documents are
  - *embedded* in word processor (e.g., Word)
  - *not embedded* in HTML
- With WYSIWYG HTML editor (e.g., FrontPage), what do you do?
  - change user model (describe in manual, explain with popup dialog box)
  - change program model (make copy of picture in subfolder)

Before, we wrote code and then inspect it's output. now its WYSIWYG (like Word, HTML).
We try our best to preview the image of the user option with an example (like word font's)

50 Cent here means instead of ask all the users, just take a sample.
We can also use a sketches/prototypes and ask different people to know their opinion.

How do you get the user model?

Ask the users!



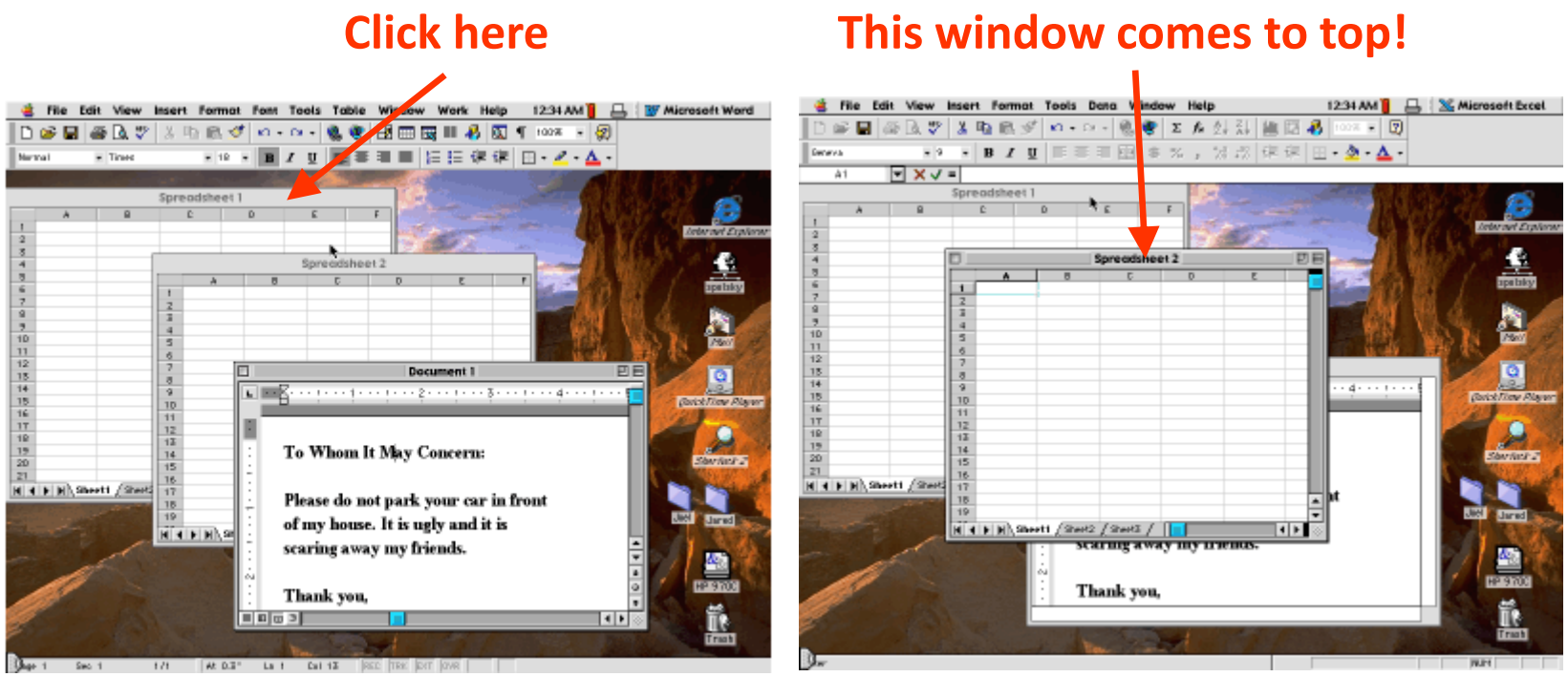
The 50-cent usability test

- Usually 5-6 people is enough, will start to see consensus
- Don’t need formal usability lab, or “people off the street”
- Just sketch or prototype and ask your neighbor

We don't want to overcomplicate for users experience.
The flow must be smooth. Like multiwindow example, when clicking on 1
It should be highlighted,
instead of some other thing like different places for windows etc..

User models are simple

- If your program model is nontrivial, it's probably wrong



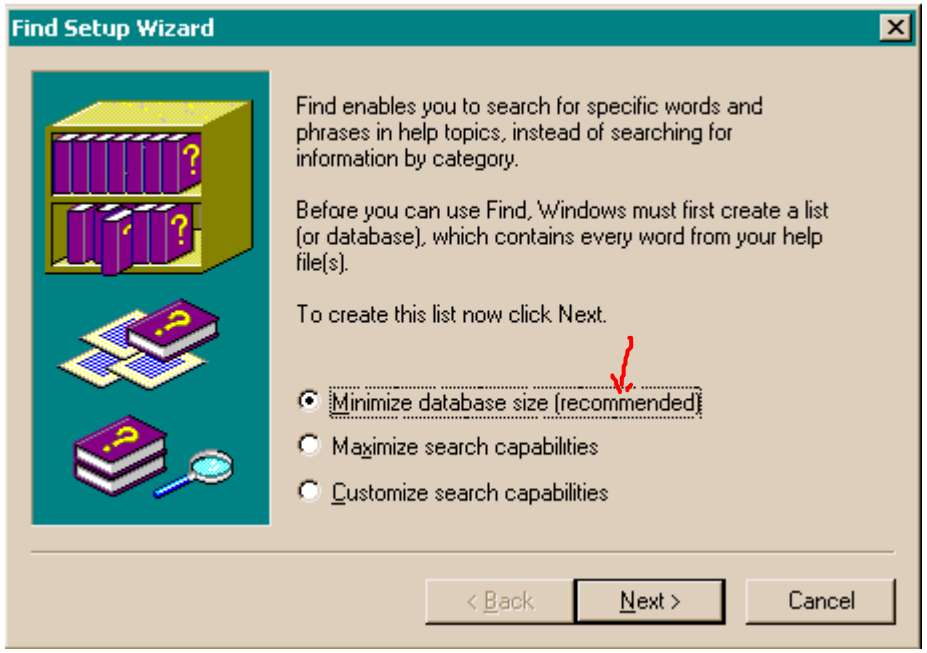
("invisible sheets" in Excel)

Choices

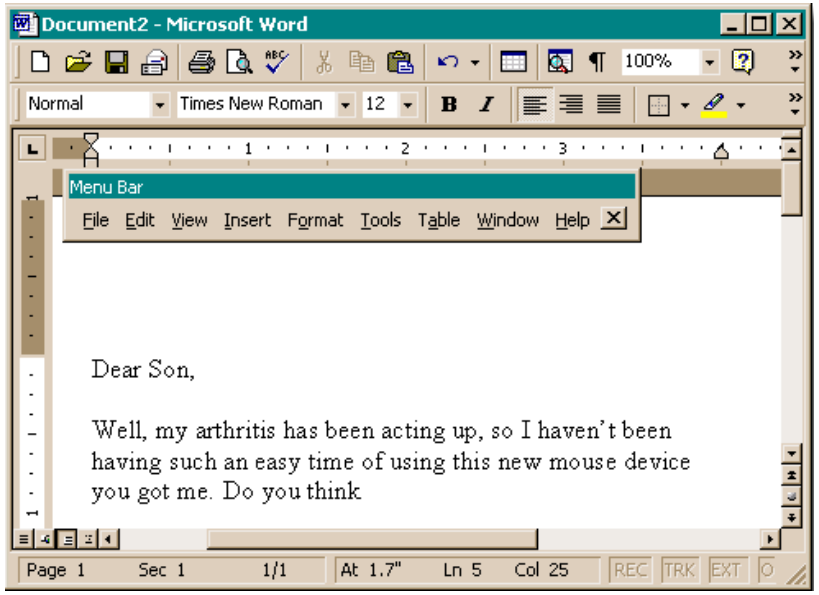
We must ease the different options for users, like add some specifications/details to every option.
Like this option is recommended, this option may increase the speed etc.. because it is useful for the user instead of this option uses model a, this model b etc..

- “Every time you provide an option, you're asking the user to make a decision.” – Joel Spolsky

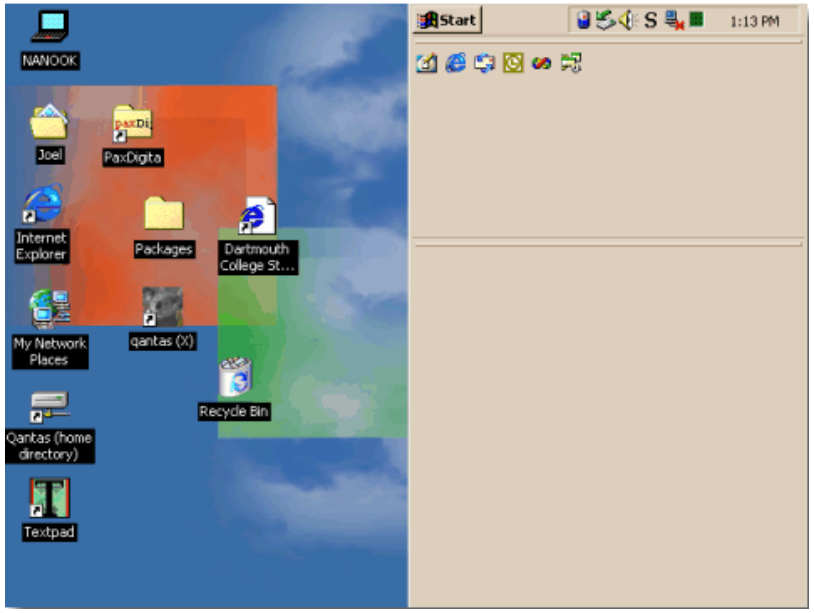
This is “unequivocally the most moronic ‘wizard’ dialog in the history of the Windows operating system. This dialog is so stupid that it deserves some kind of award. A whole new *category* of award.”



Too much freedom is dangerous



floating menu bar



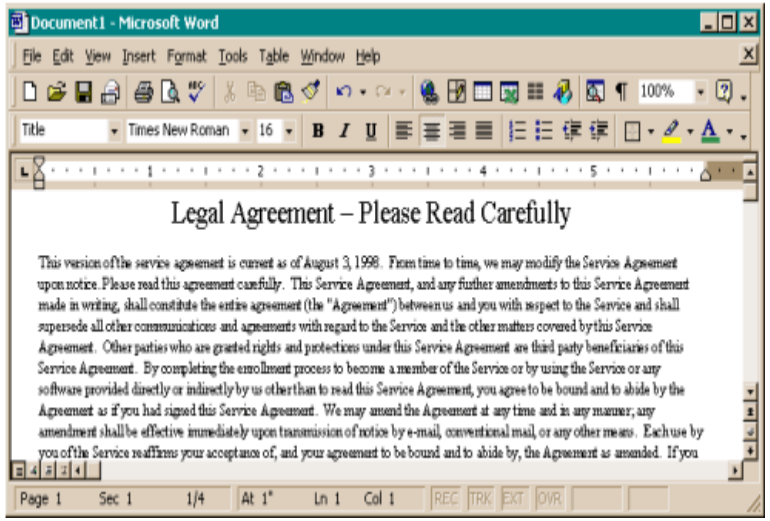
huge system tray

The more exteme personalization the user the more dangerous it can be and opposing to your design
Like the system tray, we can limit the width of the system tray like the above etc..

How many users want these?

Metaphors

vs.



Also desktop, folders, paintbrush, ...

Every metaphor from the above is like a standard, because it's understandable.

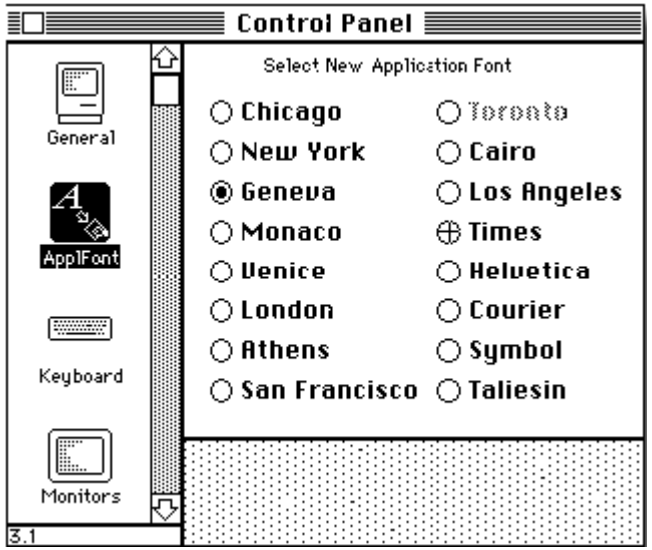
Affordance in design refers to the properties of an object that indicate how it can be used, suggesting its functionality to users through its shape, size, and features

Affordance

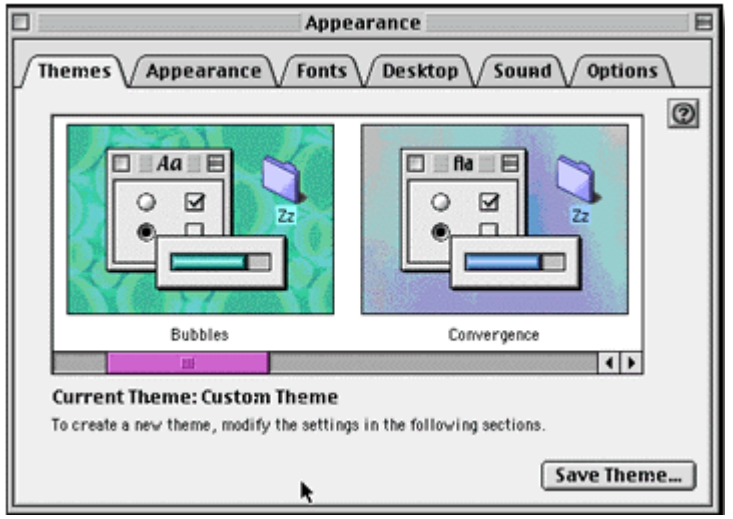
afford – to make available or provide naturally

(door with metal plate *affords* pushing)

The left design has low affordance (30% usability) because it relies on text alone, making it less intuitive for users to understand and use quickly.
The right design has high affordance (100% usability) as it uses visual cues like icons and graphics, making the functionality more obvious and easier to navigate.

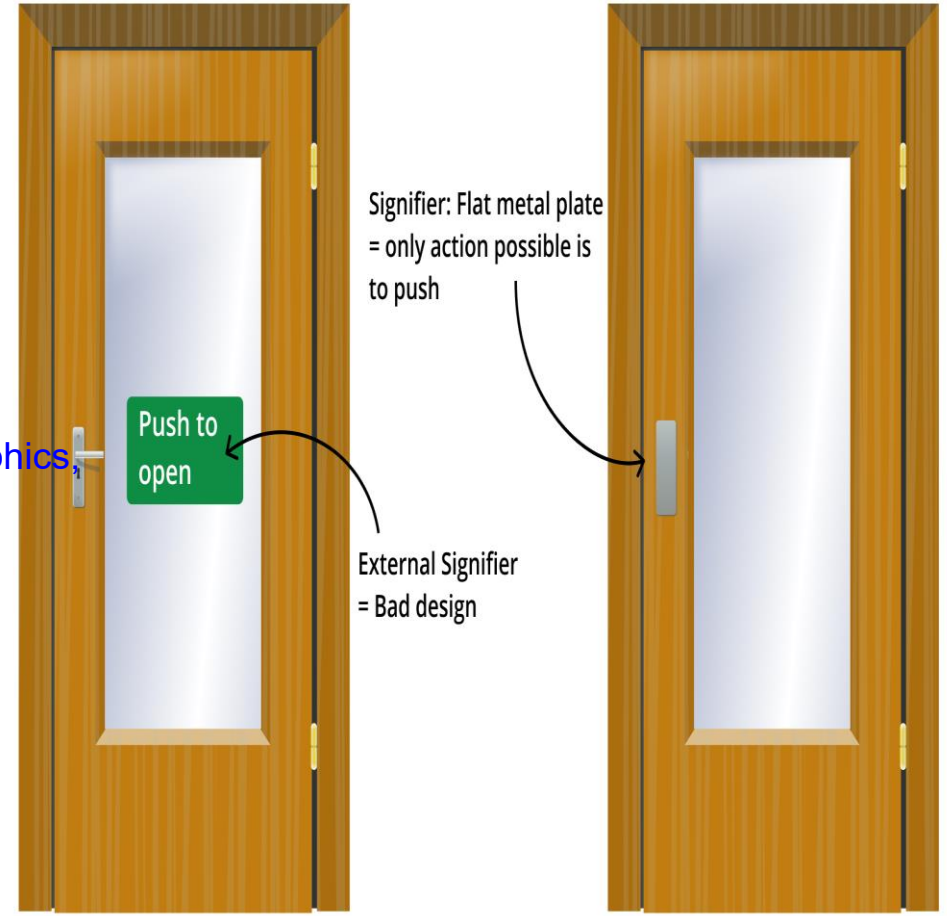


vs.



(30% usability)

(100% usability)



We should add normal controls which job is dedicated,
Like the above example, no need to add more controls for user

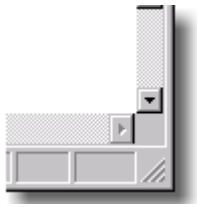
The most important thing, is not to get confused where to grab/click/drag.
It should be straightforward.
As affordance interpreted, we shouldn't put controls unless it's obviously stating what its doing

Affordance (cont.)

Where to grab?

Where to click?

What to drag?



Consistency, not creativity

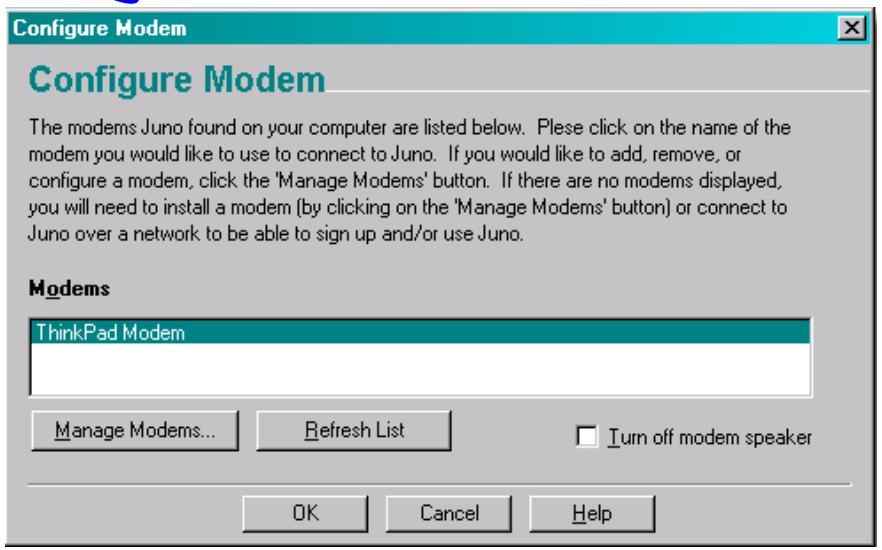
- "A *foolish* consistency is the hobgoblin of little minds" – Emerson
- Application should be consistent with itself *and* with other programs
- Examples: FrontPage, Visio
- Beware of creativity:
  - Less like user model
  - More work to implement
  - Do not leverage future/hidden features
  - "Just because Microsoft does it, doesn't mean it's right"
  - Examples: Tab from name to password, Netscape's reimplementation of common controls

Consistency is better in UI than creativity. Consistency is very important to avoid confusion.
No need to go in the against direction of some common implementations.
No need to increase the usage of hidden features etc..

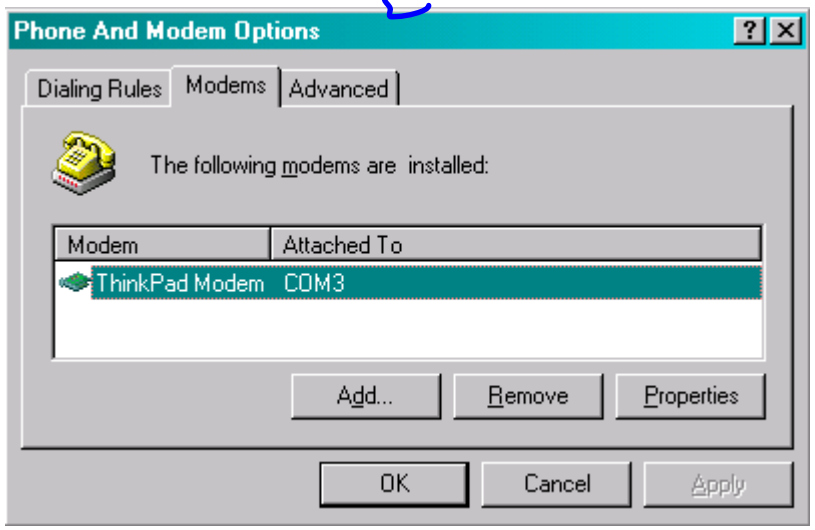
No need to make the user always search the manual, we should ofcourse have a manual but users should also know how to handle things without going to the manual

Make explanations brief

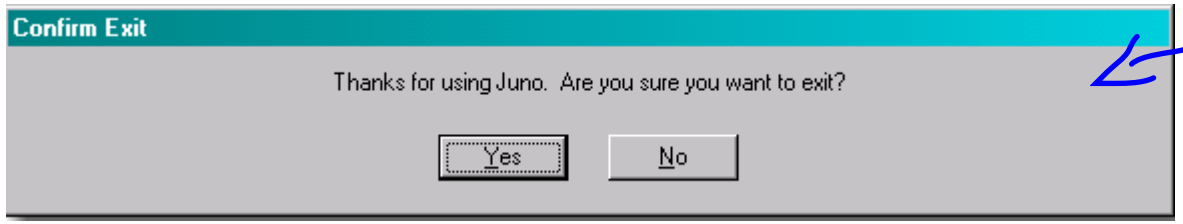
- "Users don't read the manual" – Spolsky
  - May not have the manual (on airplane, demo version)
  - Too busy / distracted / impatient
- "Users don't read anything" – Spolsky
  - advanced too busy
  - novice hope defaults are ok
  - in-between try to read but get confused



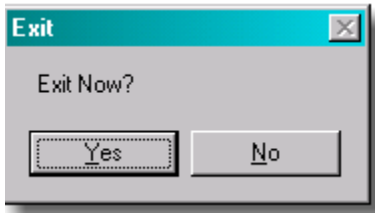
vs.



Many users are intimidated by computers



vs.



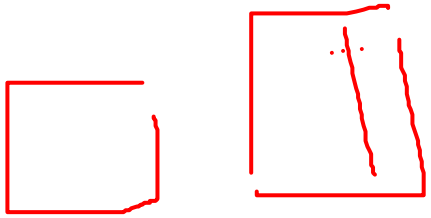
vs.

(no dialog)

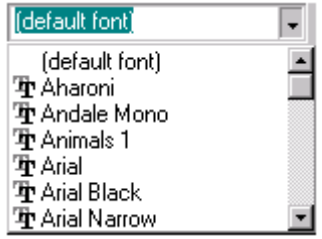
Which is better for an intimidated user?

We must also consider where will the user utilize this, we must handle different interfaces/devices.

Users can't control the mouse well



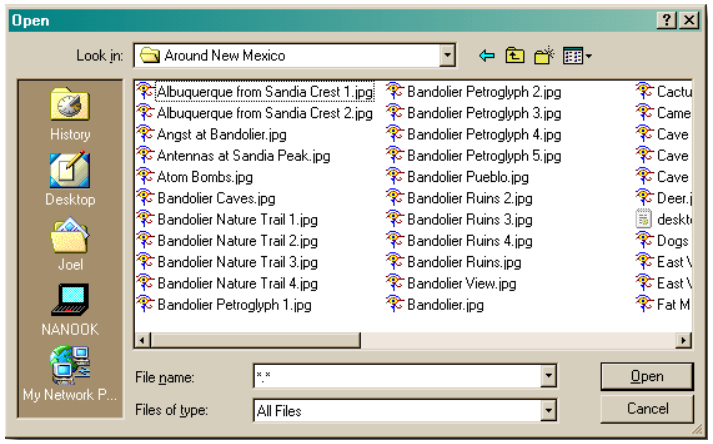
- What's the problem?
  - sub-optimal pointing devices
  - bad conditions (dirty, old, or cheap mouse; crowded desk)
  - medical disabilities (young, old, arthritis, ...)
  - in a hurry
- "Mile-high menu bar"
  - Macintosh: slam mouse to top, get menu
  - Windows: ½ by ¼-inch target
- Easiest places to point: four corners
  - (Windows 95 start menu blunder: 2 pixels from corner)
- Programmers generally stick to 0, 1, or n
  - They want to avoid magic numbers (Why can you only open 20 windows?)
  - But all n>1 are not equally likely
    (window close to edge should snap in place)

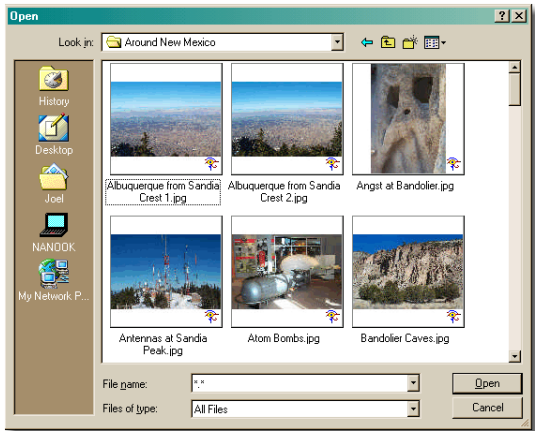


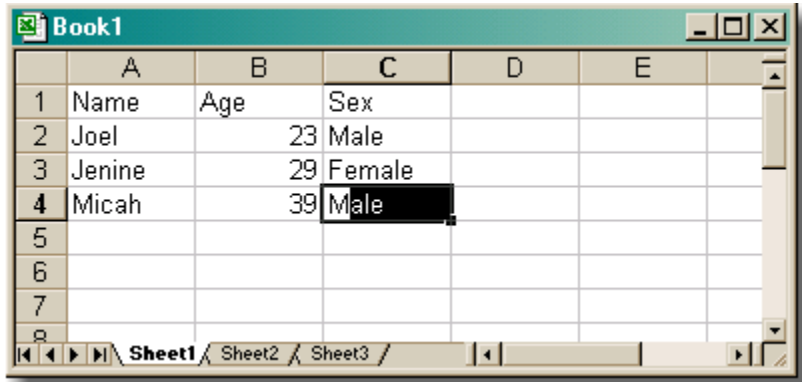
We must ease experience for user, like adding previews for different files, complete suggestions like excel, most recent change.

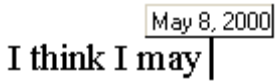
Don’t tax the user’s memory

- Make objects, actions, and options visible
- User should not have to remember (too much) information



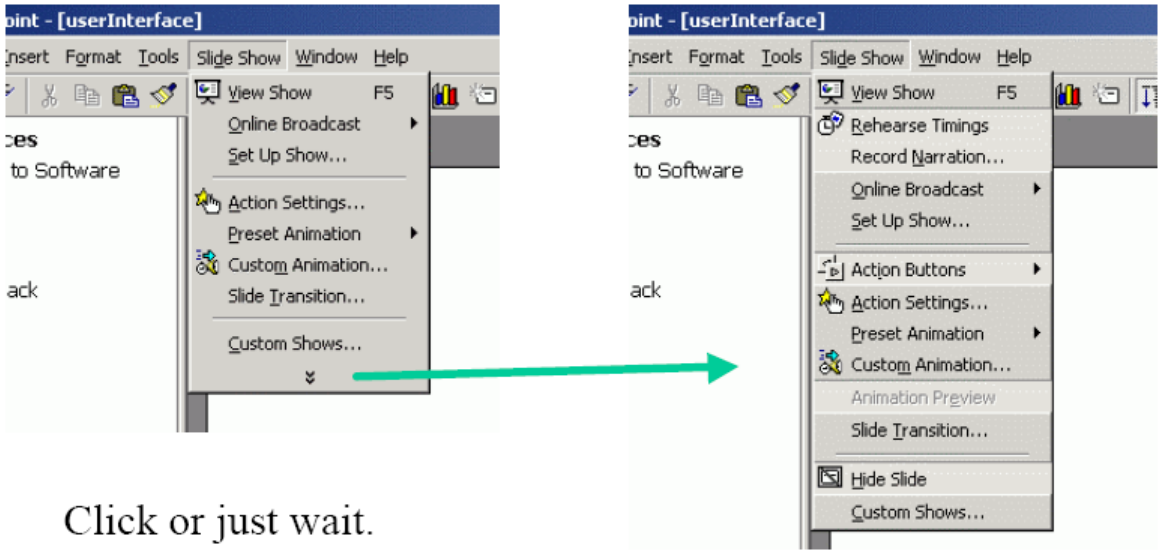






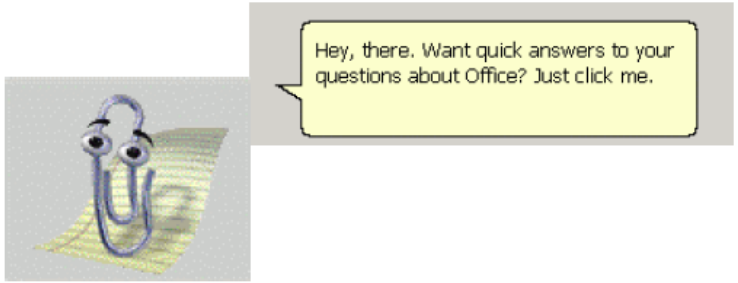
Some bad designs

adaptive menu



Click or just wait.

office "assistant"



What principle is being violated?

We must know which distribution of people are we targetting, we would like also to target the average.

The bell curve

- Users lie on a bell curve
  - 98% can use a TV
  - 70% can use Windows
  - 15% can use Linux
  - 1% can program
- Users are not dolts
- But, the easier you make the program, the more people can use it (10% more usable → 50% more users)

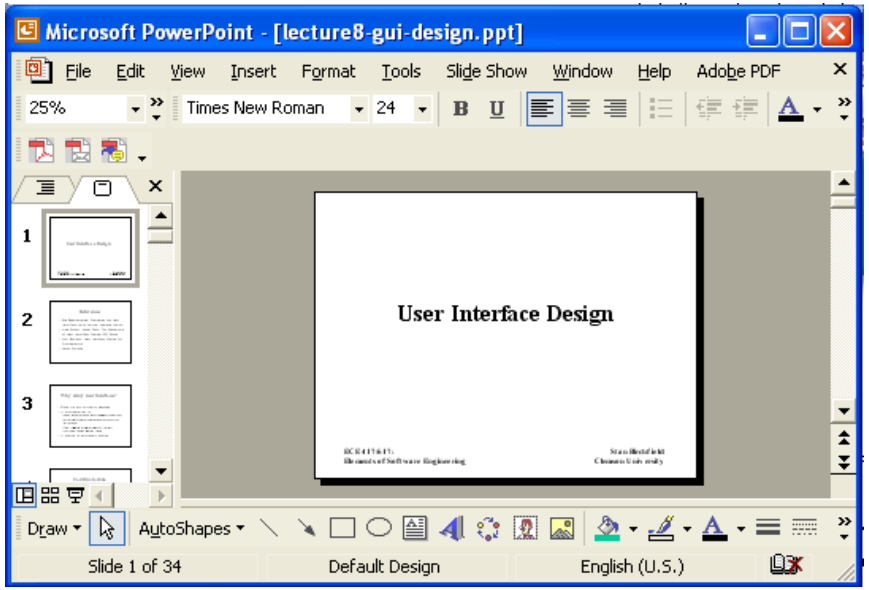
We must consider our users, what do they want, what activities they will do so we can design tools/features according to them.

Activity-based UI

- Two ways of designing UI:
  - What features should be there?
    - Greeting card example: add text, add picture, get predesigned card from library, send by email, print
  - What activities will users do?
    - Greeting card example: birthday greeting, party invitation, anniversary greeting
      (leads to unexpected features: remind to send next year)
- Example:
  - Excel was designed for financial number-crunching, but many use it for lists
  - Improv was to be “killer app” for NeXT
    - great for complicated multi-dimensional financial models
    - painful for lists

Open-ended vs. sequential operation

- History of UI goes back-and-forth b/w
  - user-in-control (command-line, Word, ...)
  - sequential steps (wizards, ...)

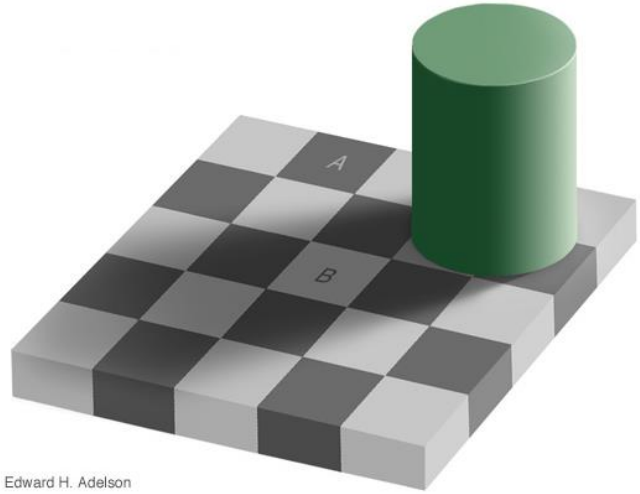


vs.

Visual perception differs from one to another, so we can interpret with common ground between them

Visual perception

color constancy

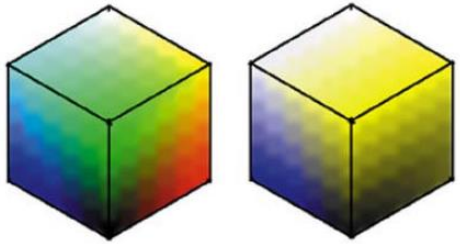


font spacing:

323 Fillmore Street

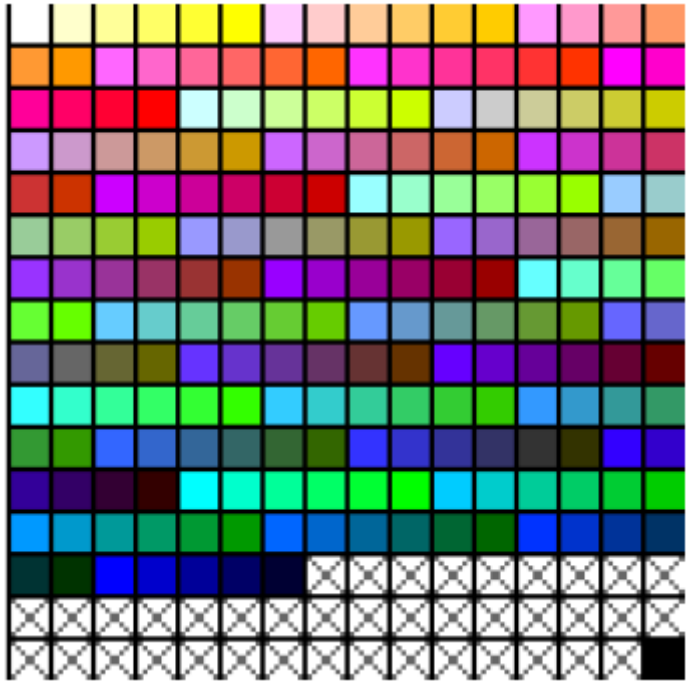
323 Fillmore Street

color-blind:
8% of men,
0.5% of women

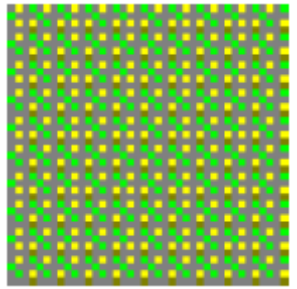


[from Michael Black]

Web-safe colors



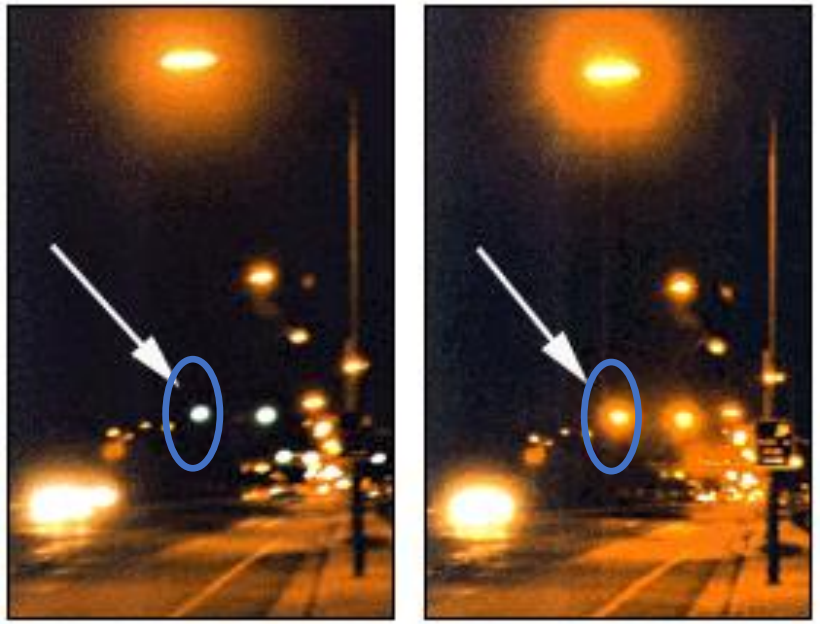
216 can be reproduced on all displays (including 8-bit)



dithering may produce other colors

Dangers of color

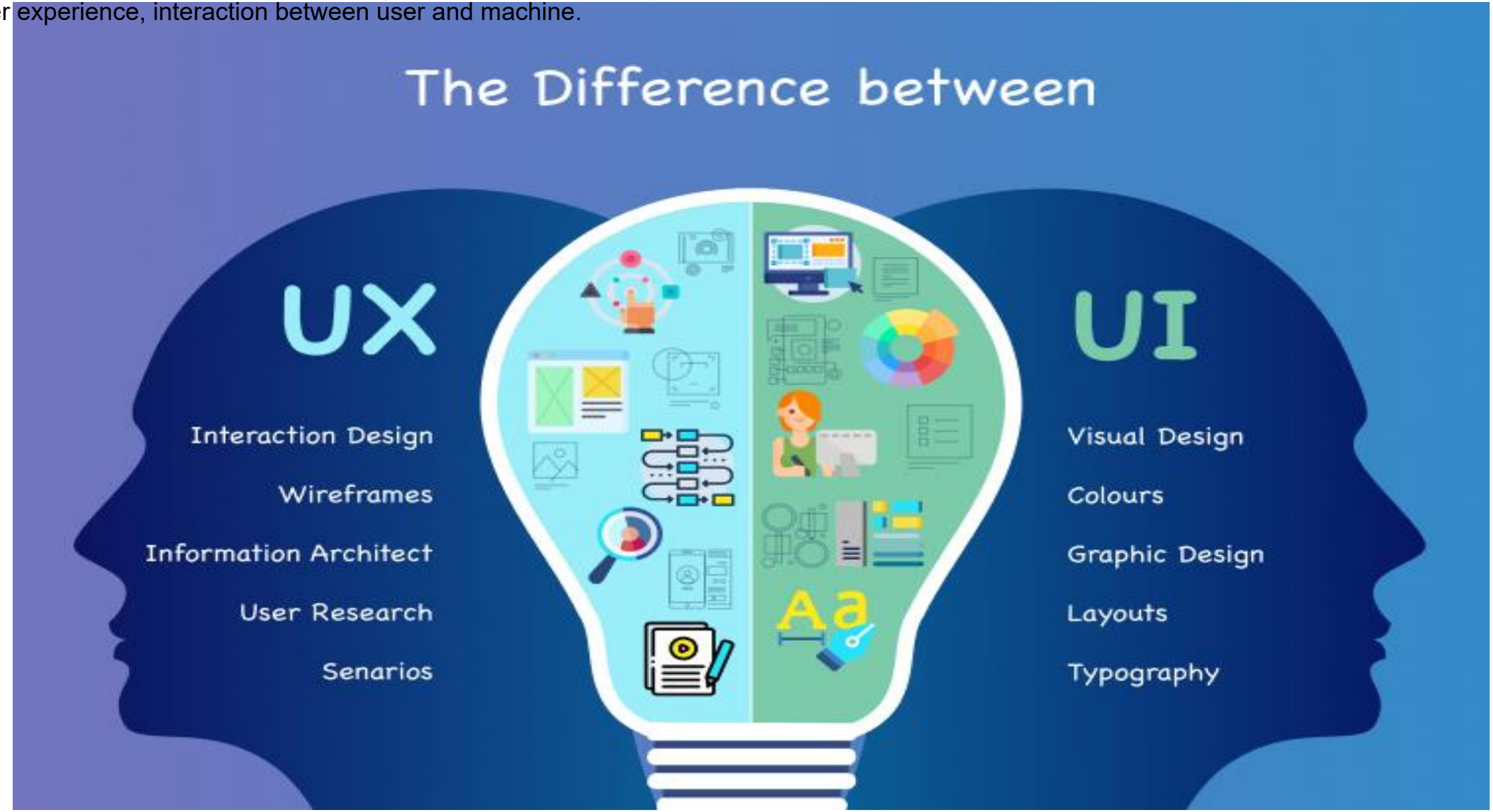
Driving at night in San Jose, where the street lights are yellow



traffic light is green

traffic light is yellow

All of the above was about UI, affordance, golden rules, consistency, control.
UX: User experience, interaction between user and machine.



We need to know which distribution of people will use our program etc..

WHAT DOES A UX DESIGNER DO?

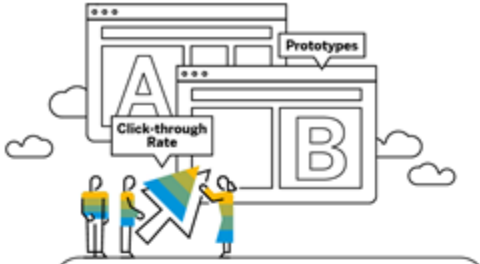
- Competitor analysis
- Customer analysis and user research
- Product structure and strategy
- Content development
- Prototyping and wireframing
- Testing and iteration
- Coordination with UI designer(s)
- Coordination with developer(s)
- Analysis and iteration

User research method cards

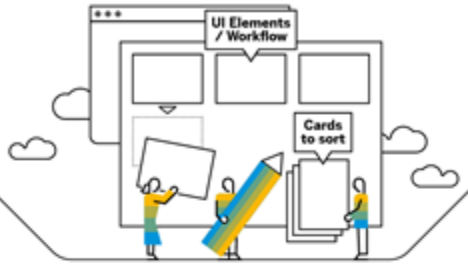
We must know what our users interpret or not, we can also validate our use case fast so we know that the feature has a purpose or not -> control/judge our sw early on.
Usability testing: test our ease of use for our program.
Usability benchmarking: reference for different features etc..
Surveying/Questioniare/Feedback: Analysis to extract good information.

360° Analysis

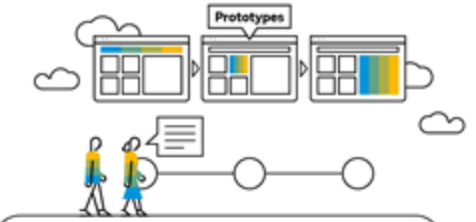
A/B Testing



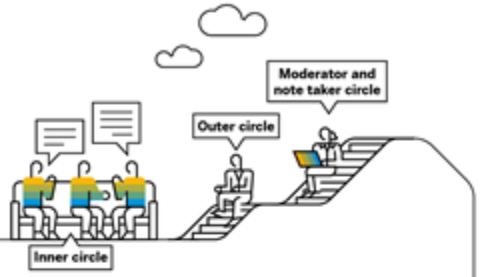
Card Sorting



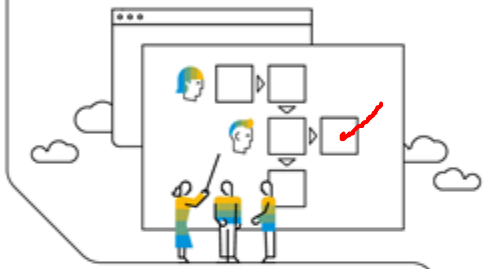
Cognitive Walkthrough



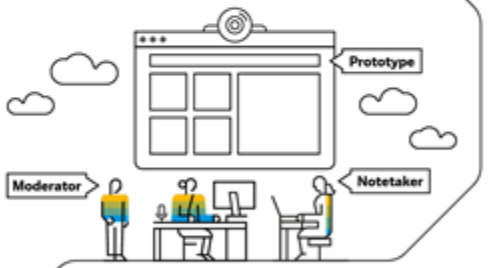
Fish Bowl



Use Case Validation



Usability Testing



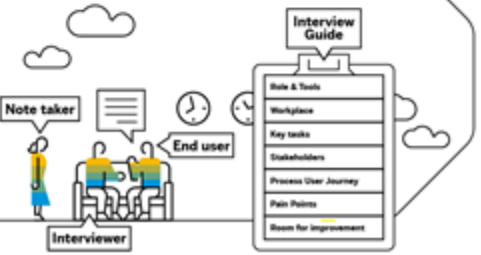
Usability Benchmarking

Tree Test

Survey & Questionnaire

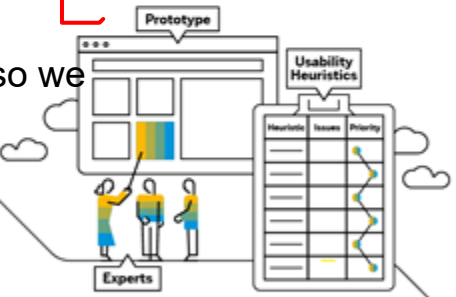
Shadowing

Interviews Field Research



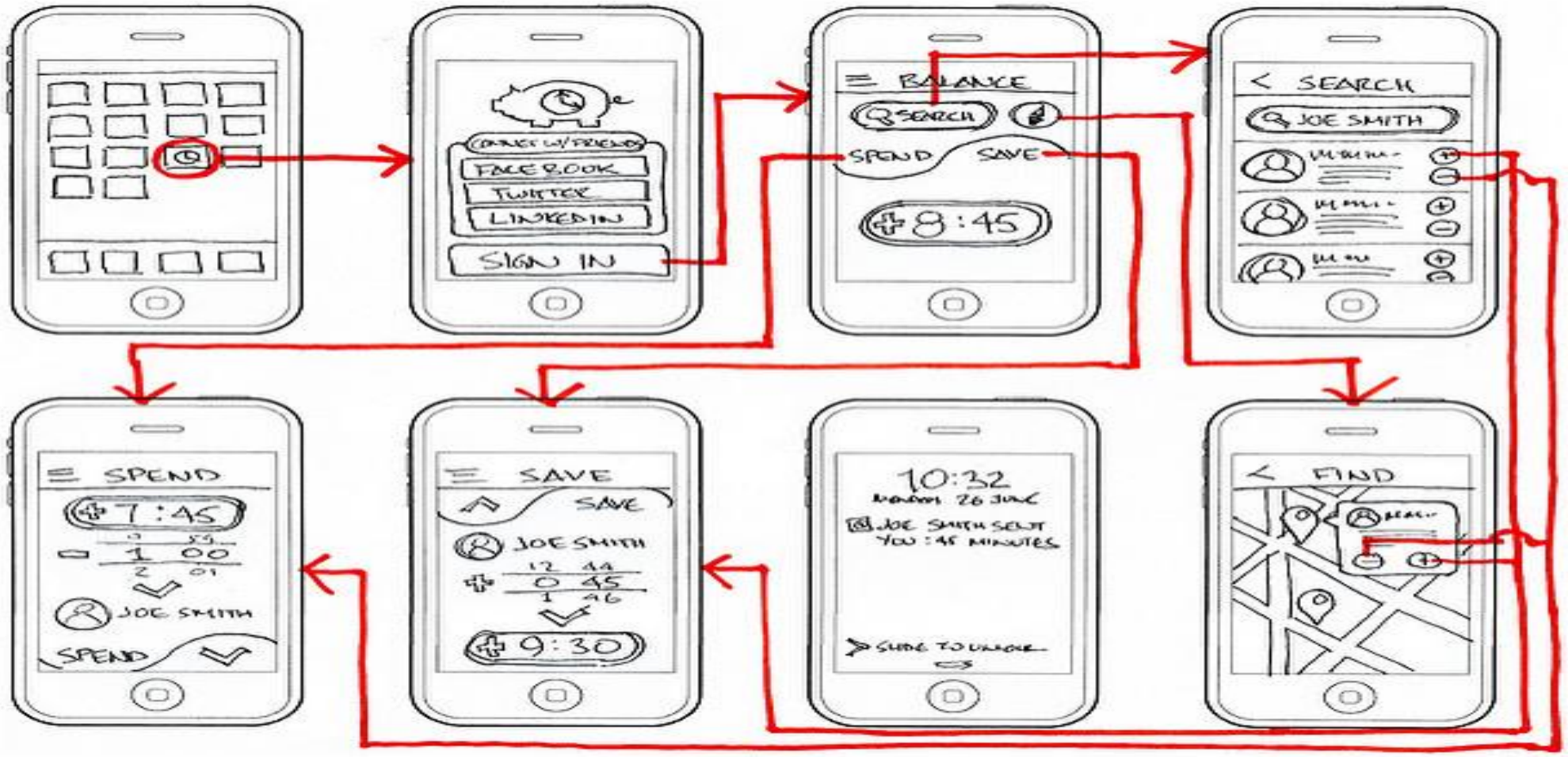
Focus Groups

Heuristic Evaluation



wireframe: a first design for the screens that might be available, so we can plan a scenario like doing X we should passthrough screen A.
This first design we can edit, remove or add or creating a shortcut etc..

wireframe



Beyond WIMP

- WIMP (windows, icons, menus, pointers)
- WYSIWYG is WYSIAYG
- Importance of language
  - grouping, conditionals, referring to objects not immediately visible or future
  - support novice and power-user
  - provide concrete and abstract ways of manipulation
  - keyboard shortcuts / macros
- Shared control
  - Delegation of routine or complex tasks to computer

Design Thinking

A new idea to get requirements, instead of asking someone what features he want,
We can ask sympathy with him, knowing what makes him happy, what ease his work
so we can focus on this points and figure out solutions in our program for this user.
We go to the client to solve his problems, like considering him a part of a program not just an input or not
after this ideas, we can create prototypes, and test this idea, if it fails -> remove it
empathize with people to know their problem -> converting to solutions -> prototypes -> test
So we can extract several features from just having a conversation with the client.

is a design methodology that provides a solution-based approach to solving problems. It's extremely useful in tackling complex problems that are ill-defined or unknown,

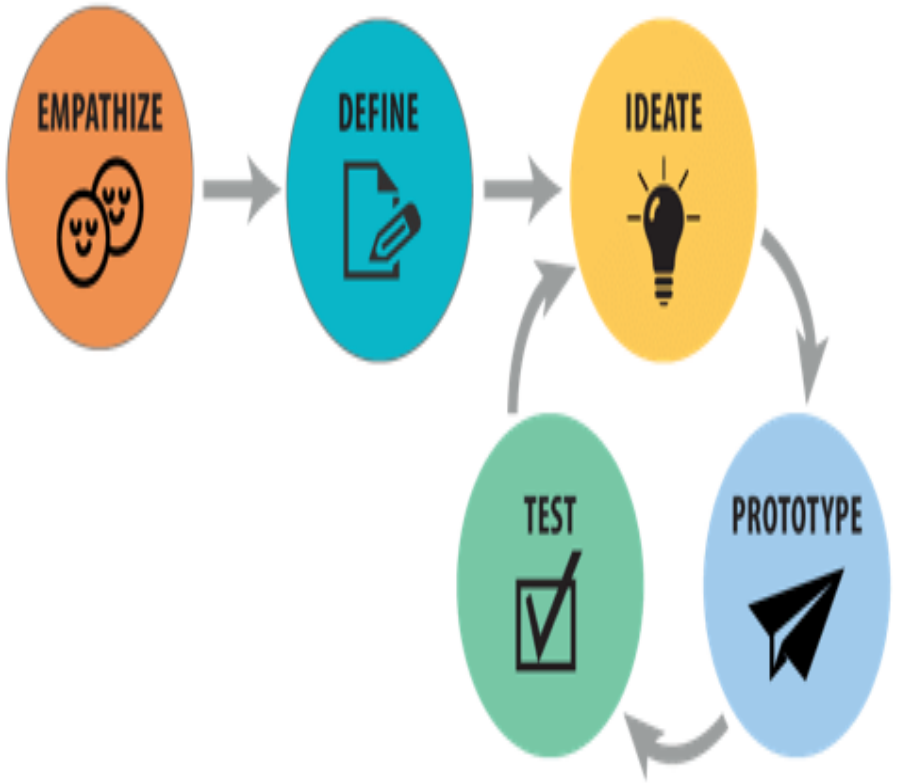
- by understanding the human needs involved, by re-framing the problem in human-centric ways,
- by creating many ideas in brainstorming sessions
- by adopting a hands-on approach in prototyping and testing.

Empathize with customer -> define his problem -> conver it to ideas -> creating prototypes -> testing them

Empathize

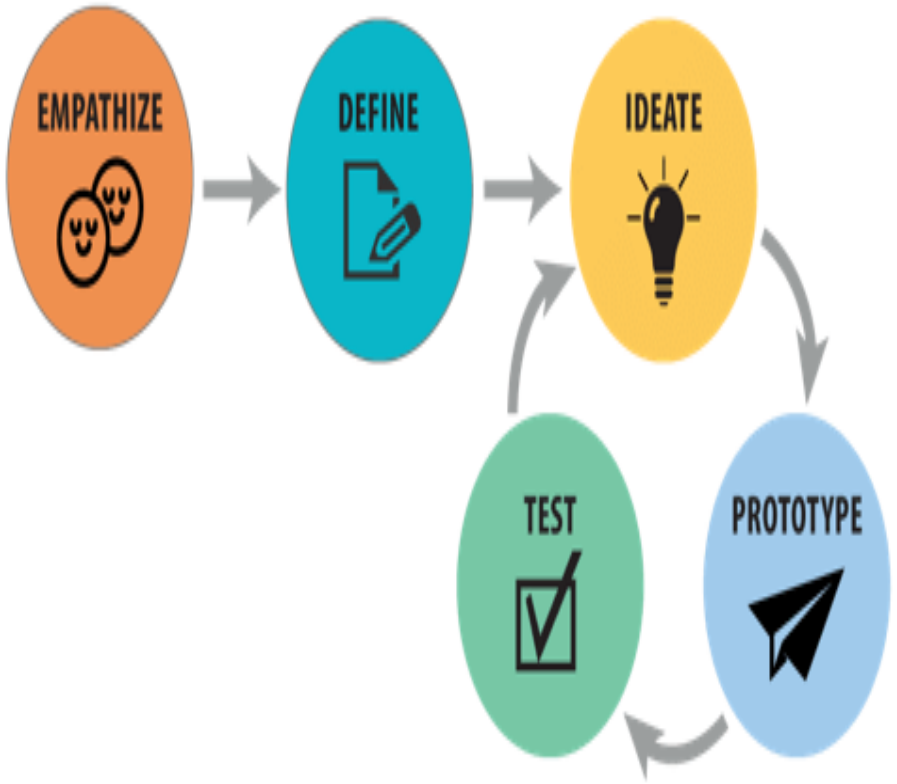
demands gaining an empathic understanding of the problem you're trying to solve, typically through some form of user research.

Empathy is crucial because it allows you to set aside your own assumptions about the world in order to gain insight into users and their perspectives. This stage involves entering the realm of the users and, as far as possible, "becoming" them so as to begin work on custom-designing a solution.



Define

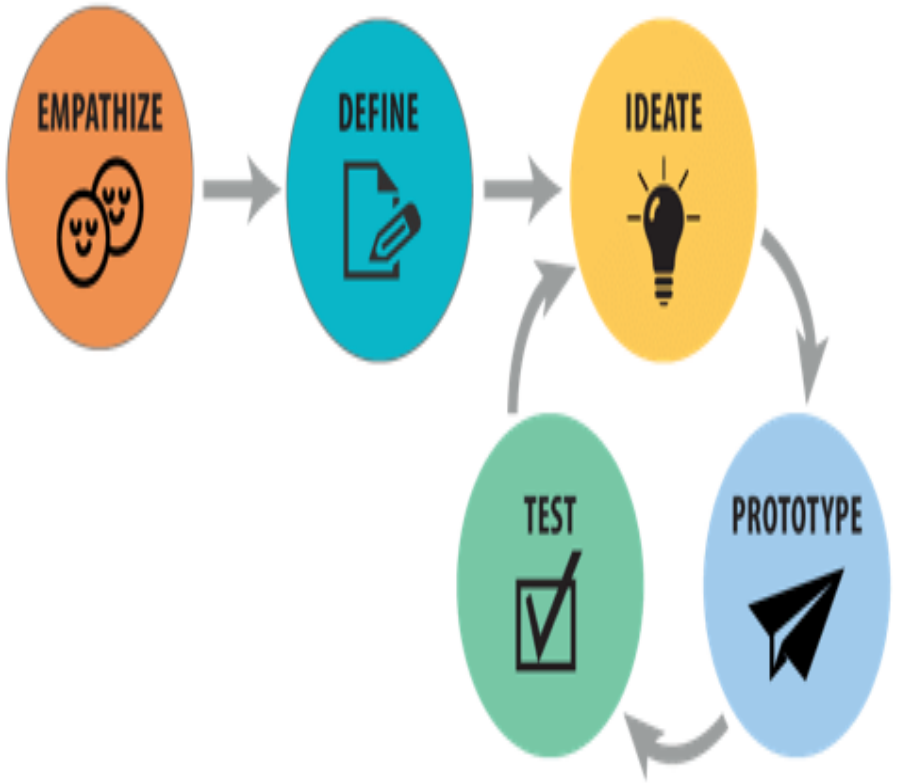
During the Define stage of Design Thinking, you put together the information you have created and gathered during the Empathize stage. You analyze your observations and synthesize them in order to define the core problems you and your team have identified so far. This is where you ensure that what you are addressing sits in sharp relief before you, its properties known in full.



Ideate

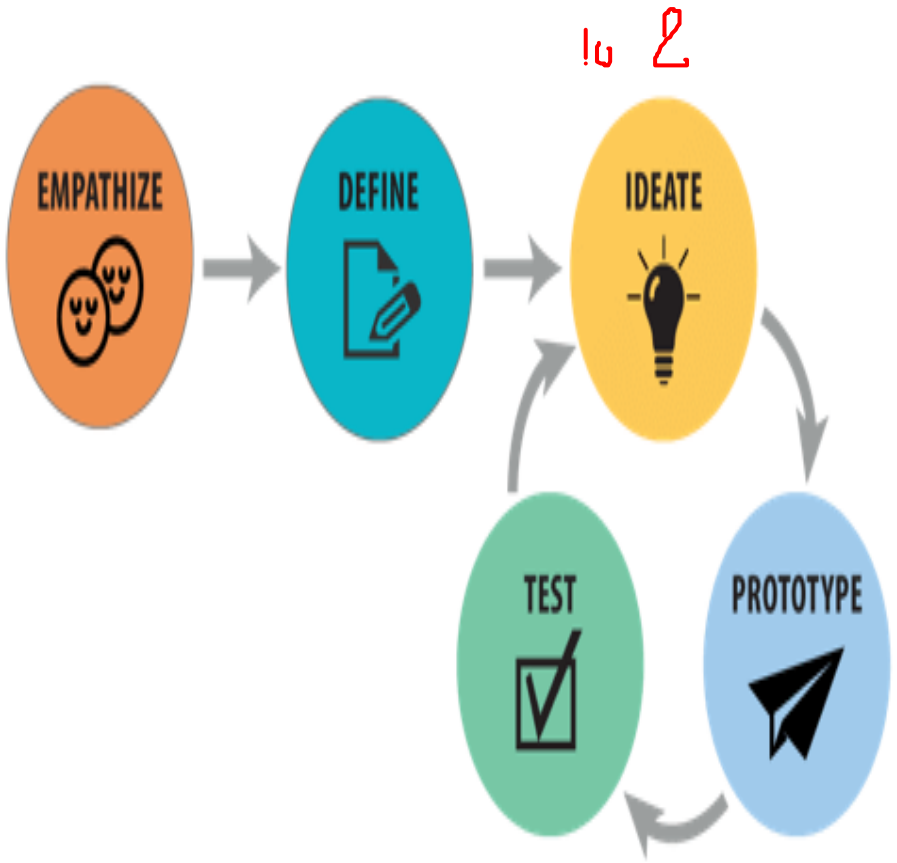
The process's third stage finds you ready to start generating ideas.

With the knowledge you have gathered in the first two phases, you can start to "think outside the box" to identify new solutions to the problem statement you've created, and you can start to look for alternative ways of viewing the problem.



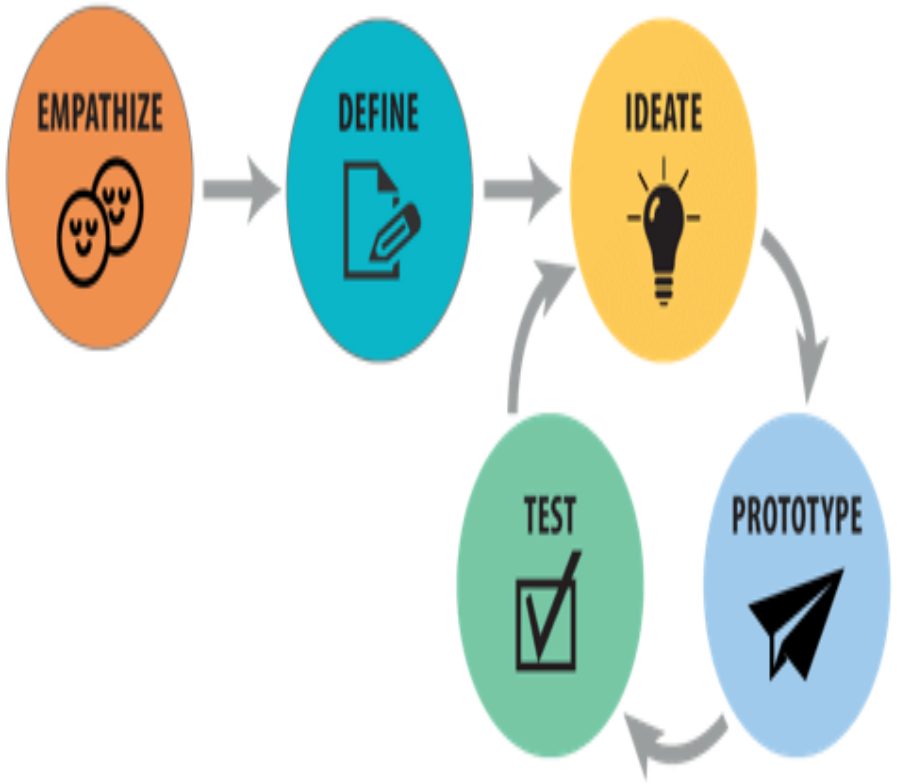
Prototype

- In the Prototype phase of Design Thinking, your design team produce a number of inexpensive, scaled-down versions of the product or specific features found within the product so you can investigate the problem solutions generated in the previous stage.



Test

- In the Test phase of Design Thinking, you rigorously test the completed product using the best solutions identified during the prototyping phase. This is the final stage; however, in an iterative process, the results generated during the testing phase are what you will often use to redefine one or more problems.



Perspectives on Design Thinking

- the importance of empathy. Empathy is about stepping out of your own role and try to understand the perspective of others. In Perspectives we will particularly ask the questions what the goals and responsibilities of the stakeholders are and which information they need to realise them.
- a fast iteration of Ideate, Prototype and Test. Using the Perspectives Software it is possible to both generate prototypes of the stakeholder's perspectives and validate them during workshops. This will improve the workshop's dynamics and the quality of the result. It will be interesting to see how Perspectives can be used in Design Thinking workshops.

Design Thinking Examples

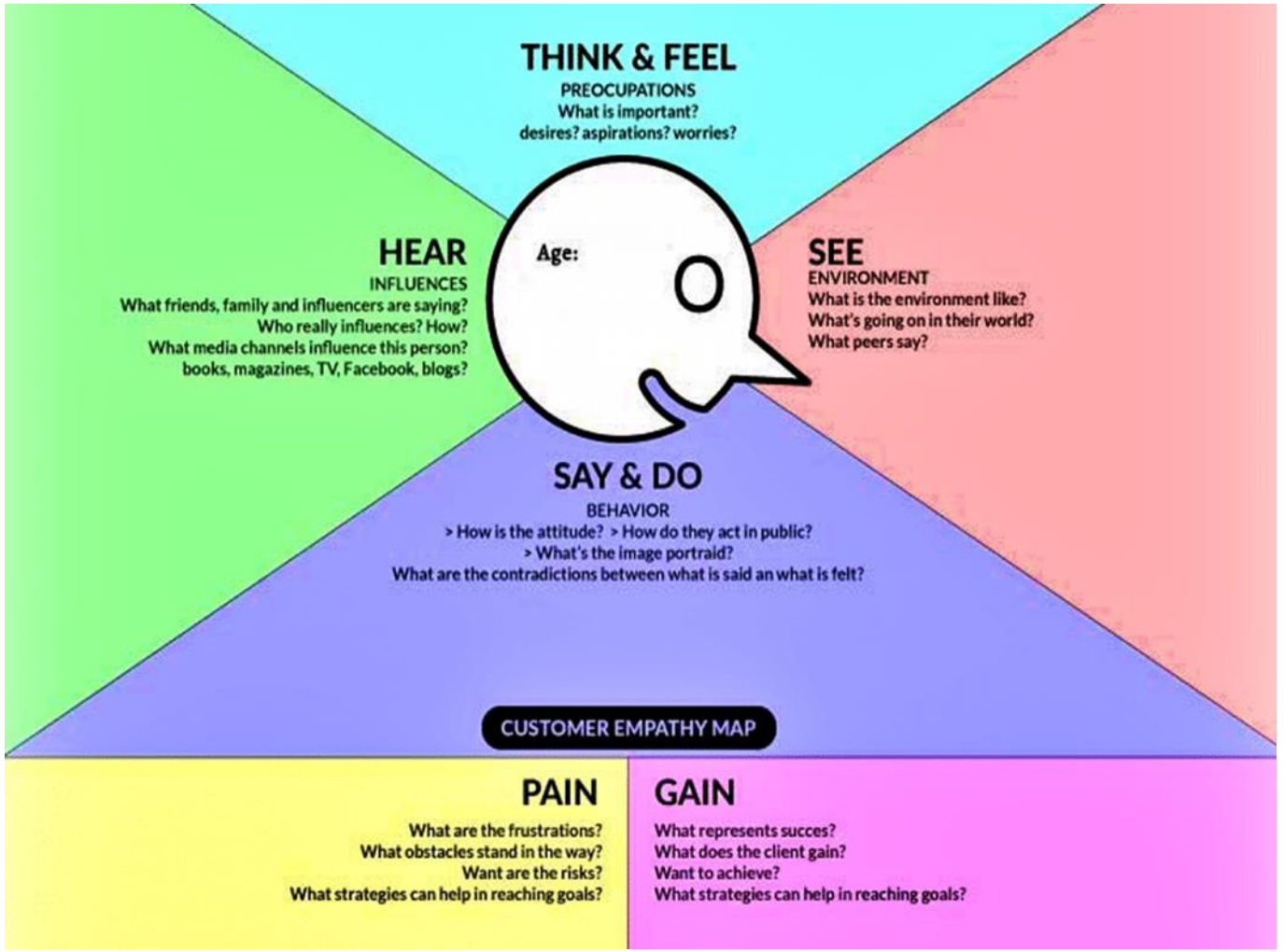
When people think of IBM, the first thing that comes into their mind is technology, business, and computers. As their former CEO Thomas Watson Jr. declared, “Good design is good business”, IBM has invested heavily in design thinking. They started holding empathy map sessions and kept users in mind while designing processes and products. Consequently, they have witnessed significant ROIs with this change in approach. They have also made it openly available.

Uber

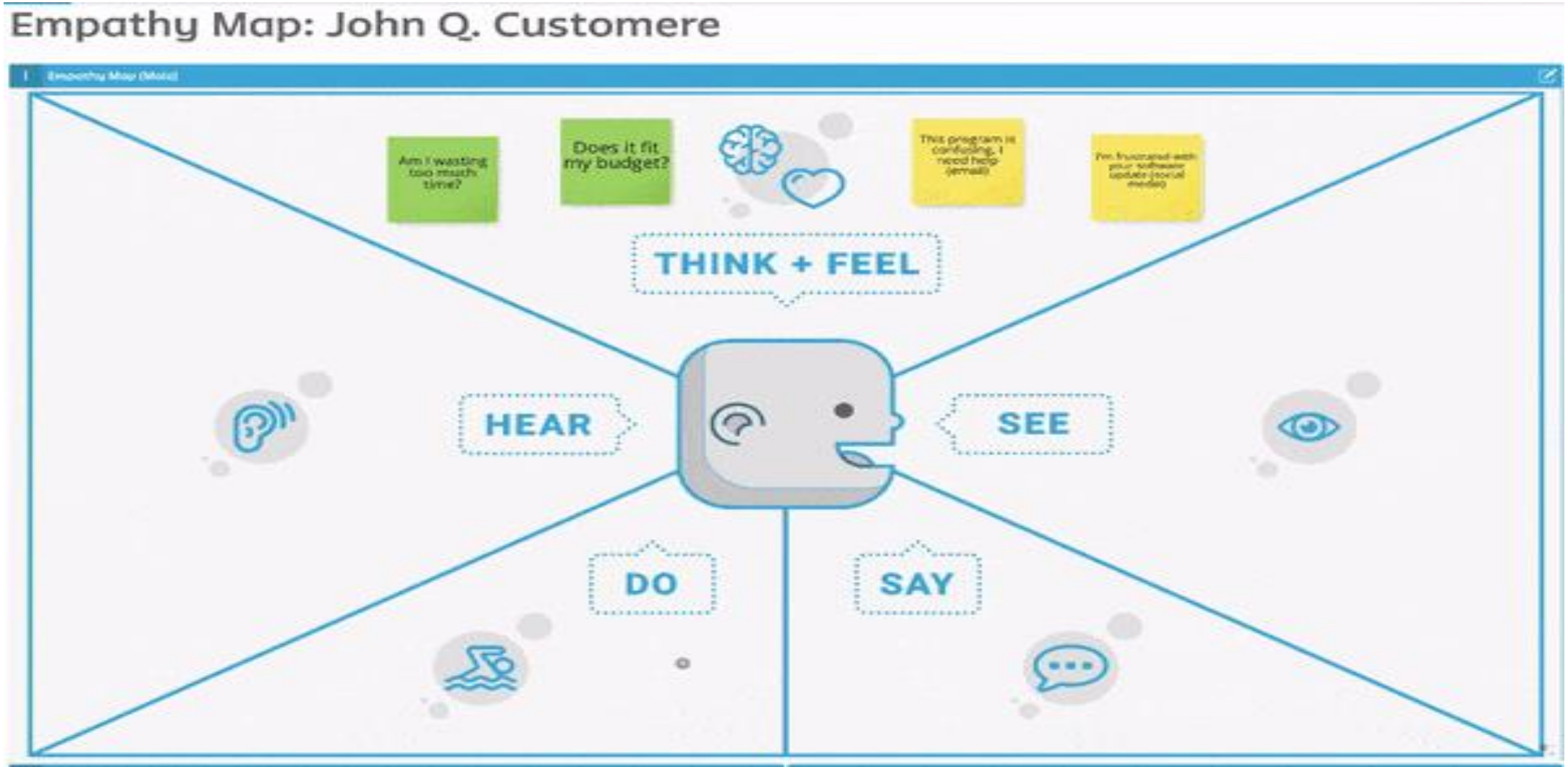
Uber is another famous design thinking example. With the help of design thinking and a user-focused approach, it eliminated simple problems that had been plaguing customers in the past. It introduced features such as cashless payments, another great design thinking process example, to make transactions straightforward and reduce the chances of fraudulent activities. By providing the power to give ratings for both drivers and users, it increased the incentive for good behavior. Simple design tweaks, aided by a substantial user understanding, helped Uber pivot itself to the behemoth it has become today. It is one of the best design thinking problem statement examples.

Customer Empathy Map

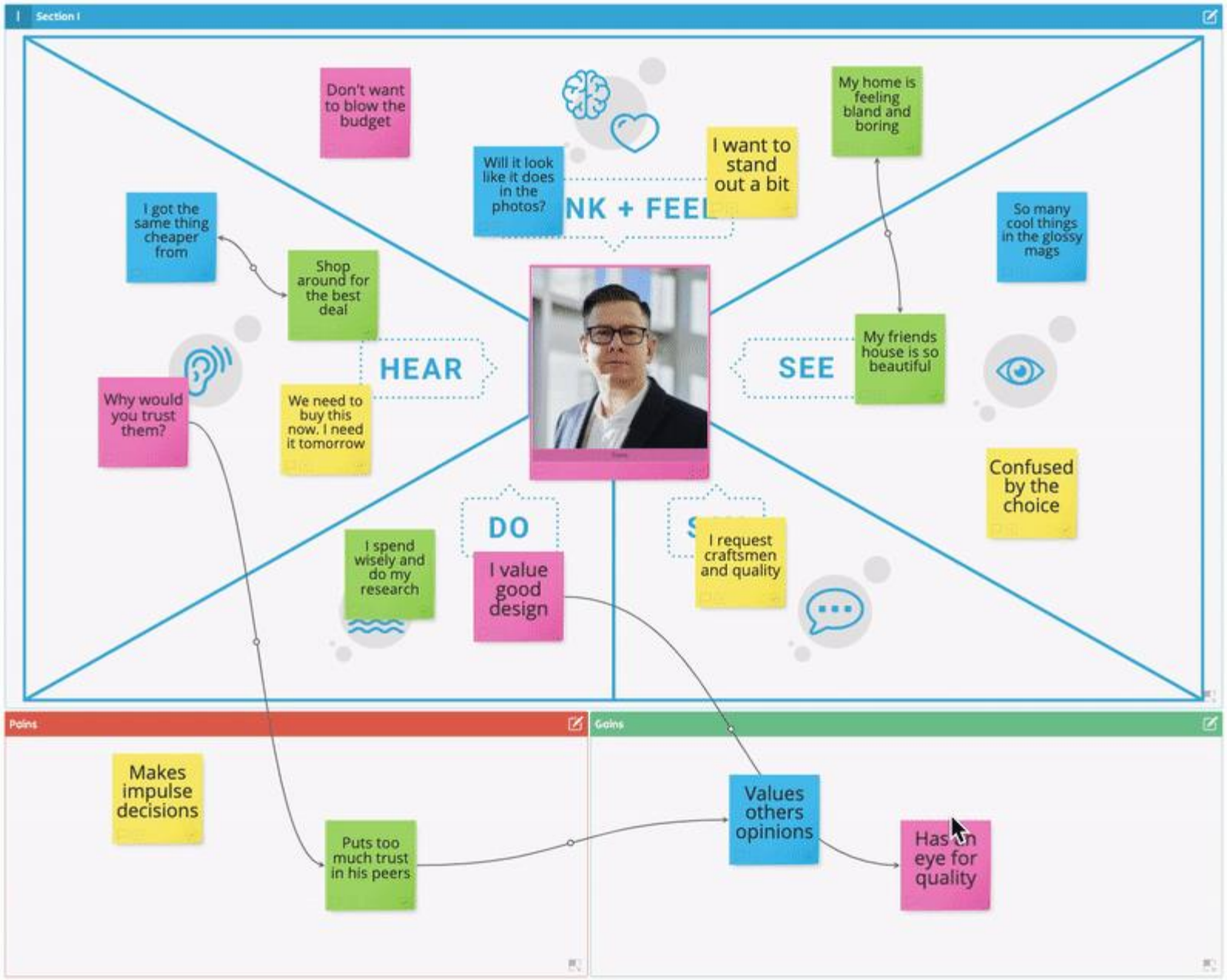
Empathy Mapping is a visual tool that is designed to show detailed insights into how customers feel about your product or service, and if they think it's sustainable for them.



Empathy Mapping



Empathy Mapping



get started with this template

- an Empathy Map looks like a basic chart containing four different sections, including *‘Say’, ‘Think’, ‘Do’,* and *‘Feel’*. These four sections represent what your customers are saying, thinking, doing , and most importantly, feeling.
- The Stormboard template includes the sections Say, Do, Hear, Think + Feel, See, and also includes Pains and Gains sections for extra information on your customers.

1

Say
write down anything the customer might say out loud to other people about a problem they are trying to solve, a product they are looking for, or things they might mention in general conversation. This can include things mentioned in an email, over the phone, face to face, or on social media.

2

Do
relates to the actions a customer takes. What are they doing, or what have they done? What might your customers do in the future? How will their actions affect the product or service that you provide and vice versa?

3

Hear
includes everything that your customer hears others saying. It's a great way to identify the community that they live in.

4

Think + Feel
get creative with how you empathize with your customer. Your customer won't use your product if it doesn't fit their budget, or if it's too difficult to learn. Do you offer different tiers of price points, or are there user guides available? Dig deep to think of ways your customers can love what you offer. Also note down what the customer is most likely feeling (including both their current emotional state and/or what they feel towards your product or service).

5

See
Document what a customer sees in their immediate environment. This also includes what customers are reading or watching, and what they see others doing.

6

Pains
brainstorm all of the pains that your customer might have that your product or service would solve.

7

Gains
to brainstorm what a customer would gain by using your product or service. How would it make their life better?

Once everything is documented, you have a better view of your customers and how they might feel about using your product or service. Then, you can dive into making changes and adjustments if needed.

Customer Journey Map - Alternative

Visualize all of the places, or touchpoints, that your customer comes in contact with your company, product, or service both online and off with the Customer Journey Map Template.

To make it easy for you to understand what motivates your customers to ultimately make a purchase, the template is set up with four rows that help you identify what your customers are Doing, Thinking, and Feeling, plus any Opportunities that are there for you and your team. The columns in this template can be customized to suit your needs.