CS 4350: Fundamentals of Software Engineering
CS 5500: Foundations of Software Engineering

Lesson 6.1 Requirements and User-Centered Design

Jon Bell, John Boyland, Mitch Wand

Khoury College of Computer Sciences

Outline of This Lesson

1. How do know what software should be built?
2. What does it mean for software to be usable?
3. How can we tell if we are building a usable product?

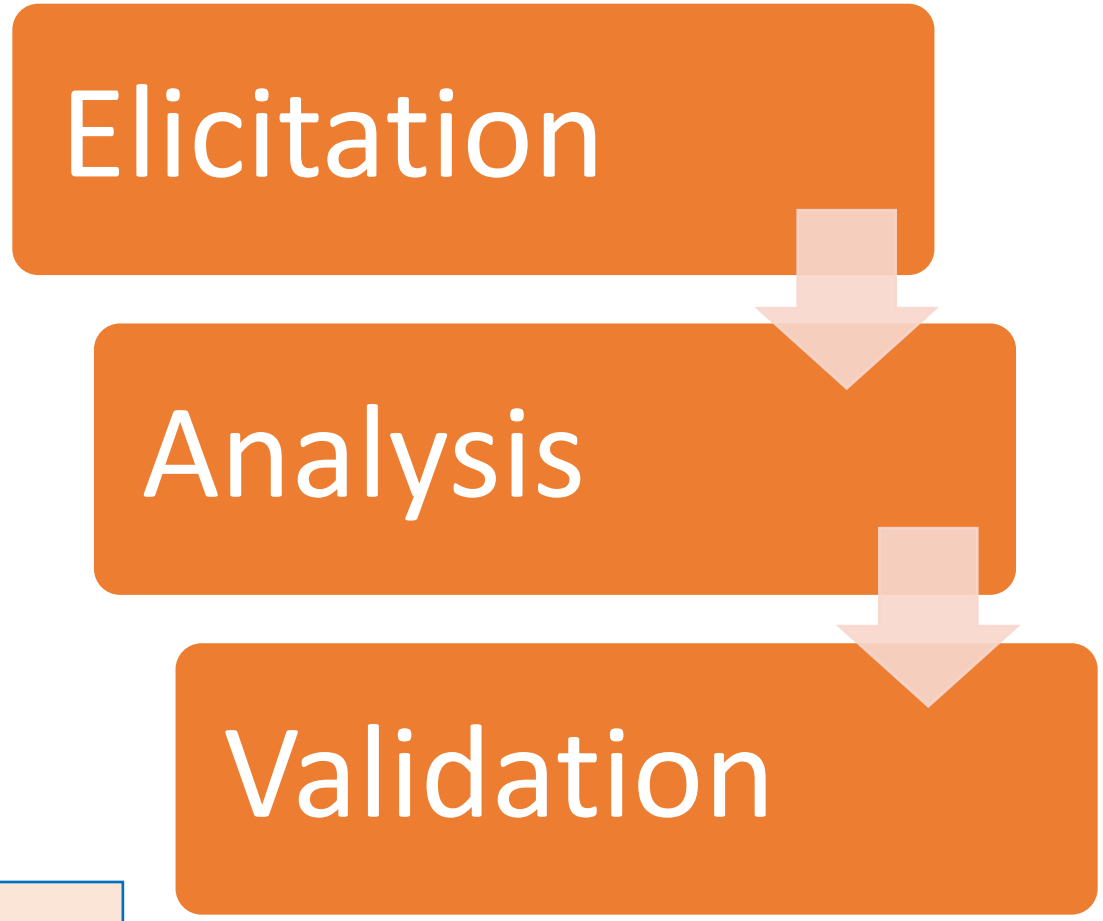
Image courtesy of Carnegie Mellon University
Human-Computer Interaction Institute

Learning Objectives for this Lesson

- By the end of this lesson, you should be able to:
  - List the three main steps to determine requirements;
  - Describe the major aspects of usability;
  - Articulate the process of user-centered design;
  - Explain several heuristics for good user interaction.

Requirements

- A three-step process:
  1. Elicitation:
     Discover requirements from prospective customers.
  2. Analysis:
     Understand and prioritize requirements elicited.
  3. Validation:
     Bring back requirements to customer for confirmation.



And Repeat!

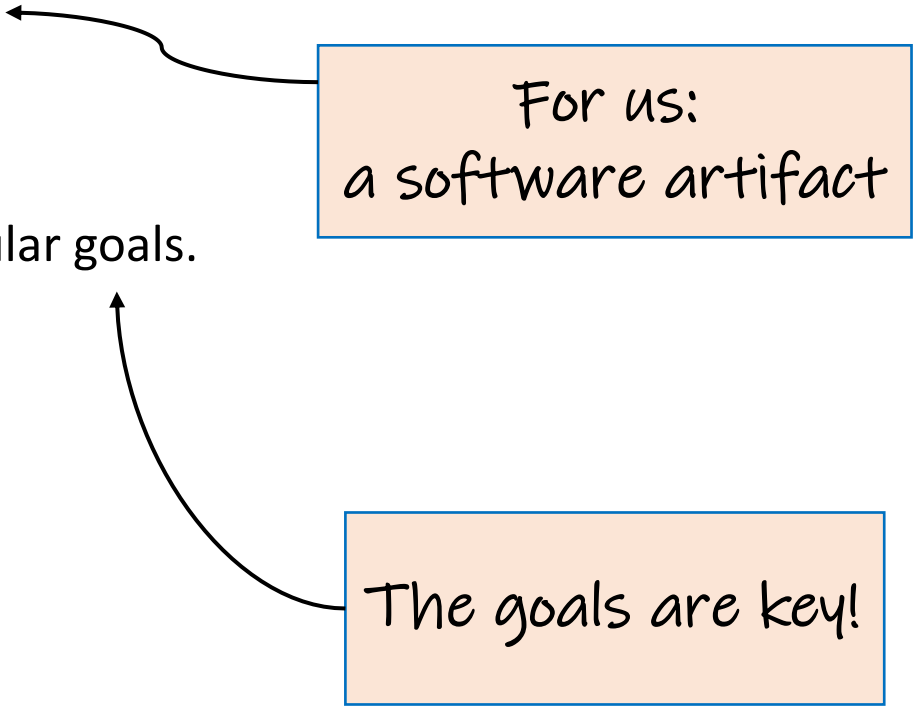
Goal: Build the Right Product

- If the product doesn't do what the users want…
  - … we've wasted time and money.
- If the product is not usable by the users…
  - … we will need to invest time/money to make it usable.
- Users are often not sure exactly what they want,
  - … so we need to iterate the requirements process.
- We shift development "to the left" (closer to user)
  - We correct mistakes
    - Before design, or else
    - Before coding, or else
    - Before debugging, or else
    - Before deployment.

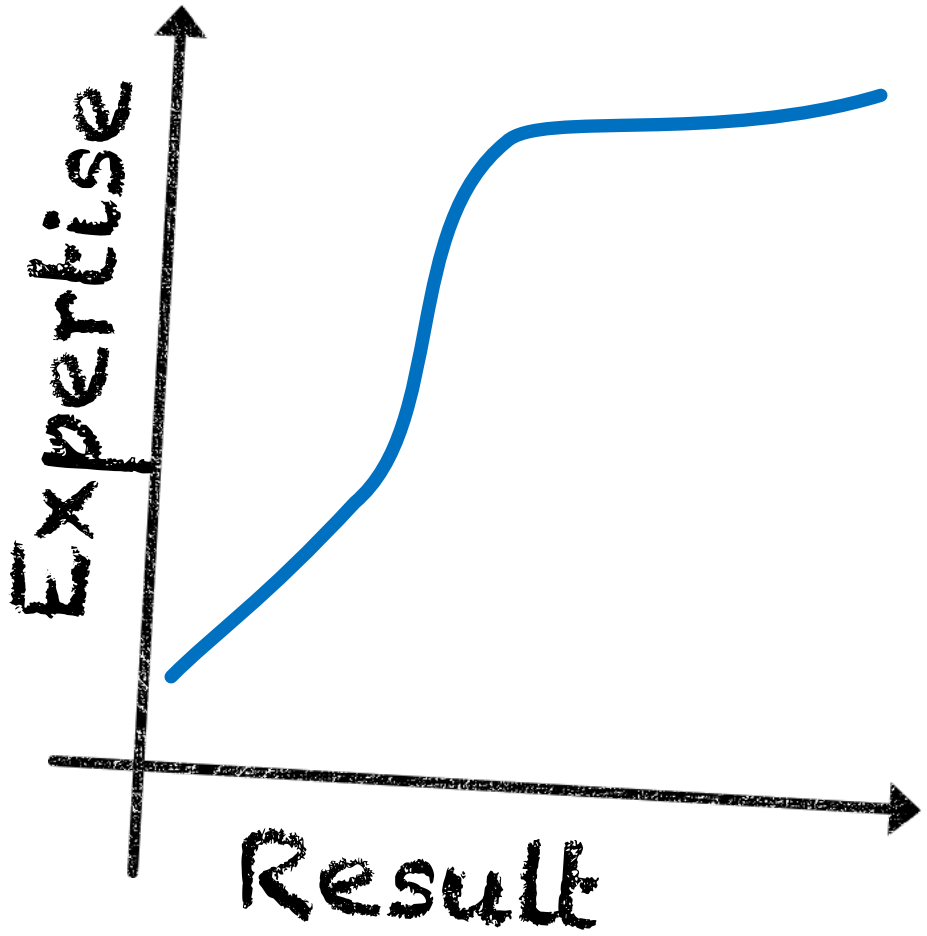
The earlier,
The better!

“Usability”: a Definition

- *Usability* is …
- … a measure of how …
  - … an artifact …
  - … impacts …
  - … a human …
    - … with particular goals.



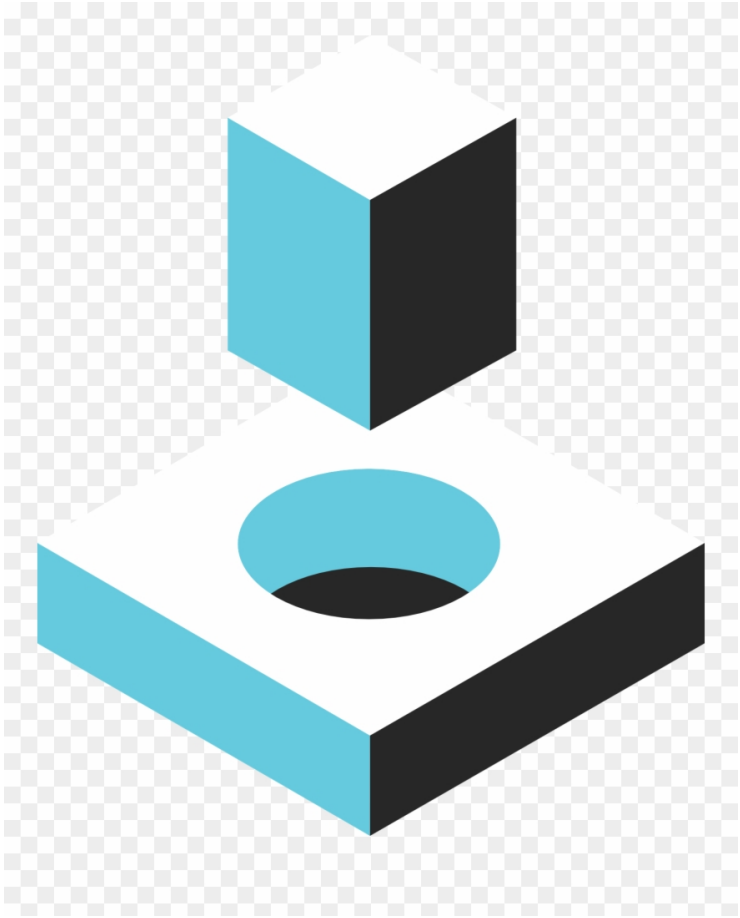
Usability (1 of 5): Learnability



- How easy is it to learn to use the artifact to accomplish a goal?
- A “steep” learning curve requires a lot of expertise before one can achieve results.

Usability (2 of 5): Effectiveness

- How often does the use lead to completion of a goal?
- Is the artifact “fit for purpose”?

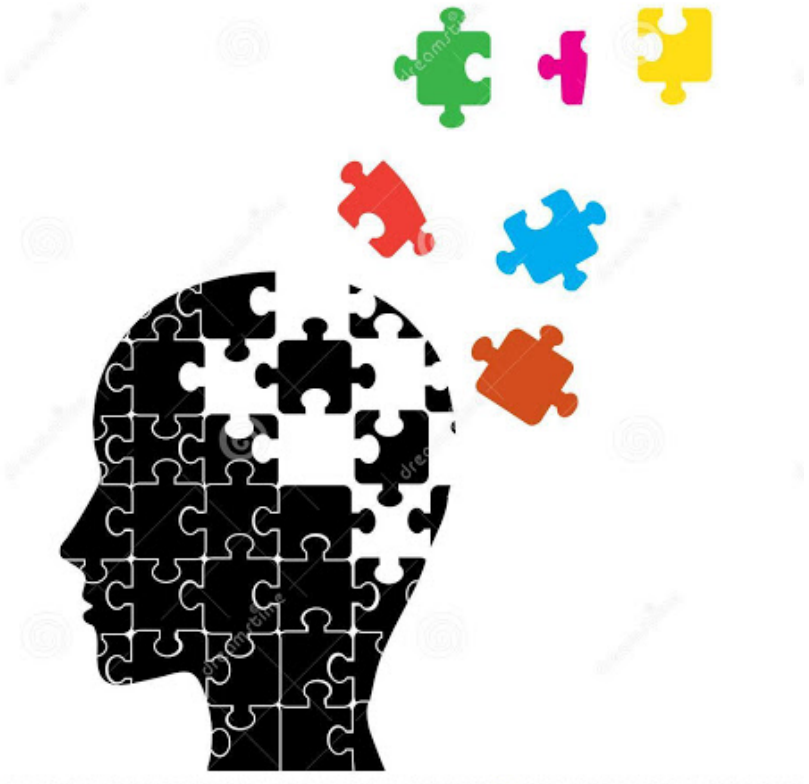


Usability (3 of 5): Productivity

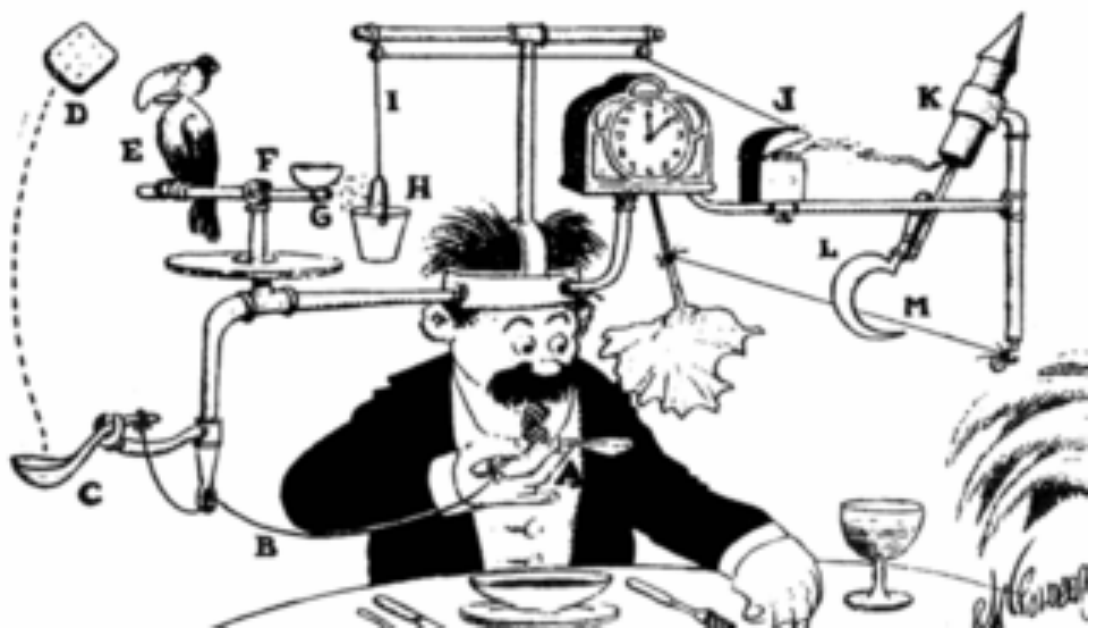
- How large a multiplier of human effort does this artifact give?
- Does it make hard things easy? (or the reverse!)

Usability (4 of 5): Retainability

- How long is the ability to use the artifact retained between uses?
- Inner consistency can help mitigate a steep learning curve.



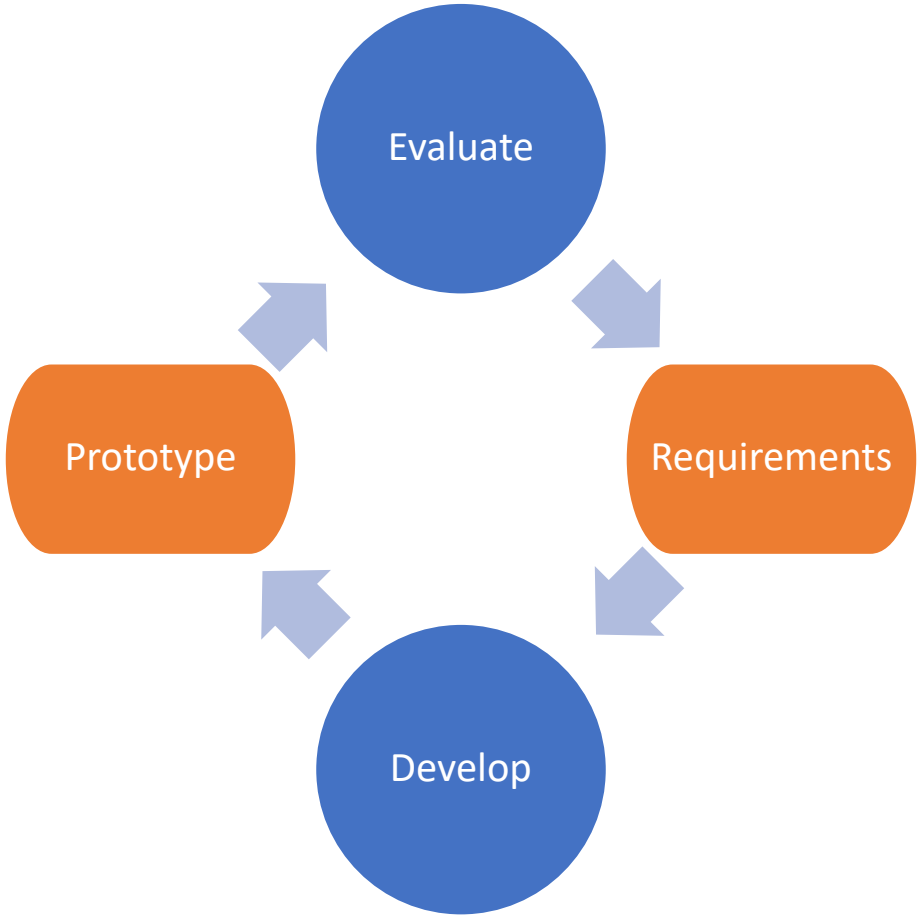
Usability (5 of 5): Satisfiability



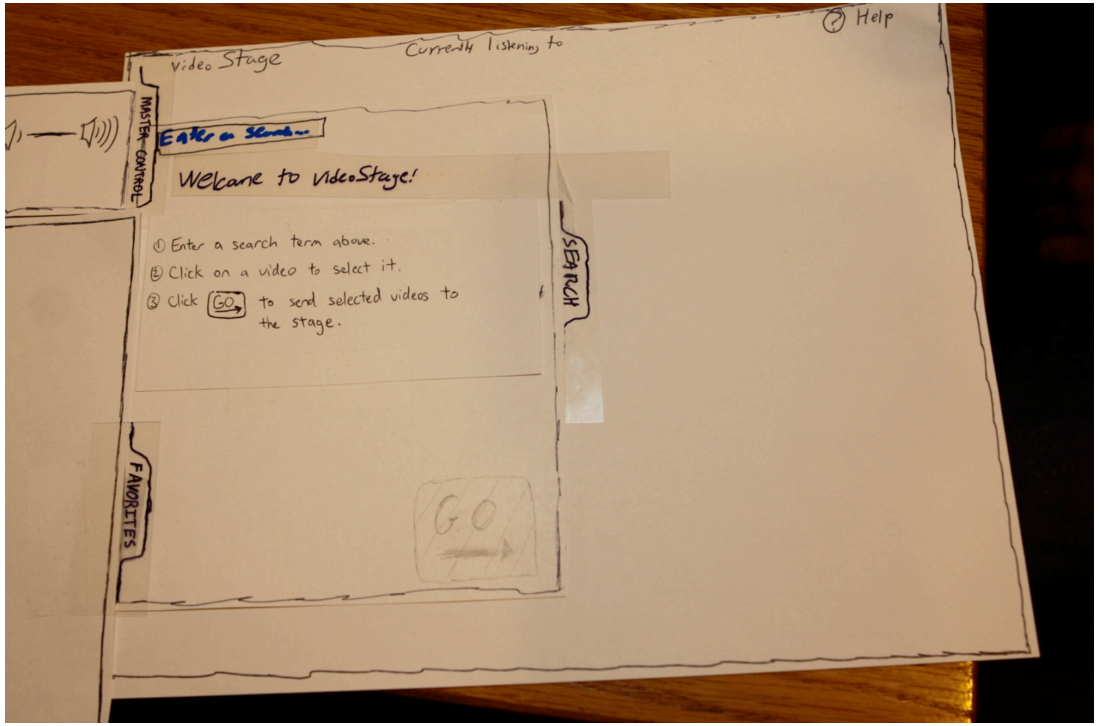
- How pleasant is the artifact to use?
- Is it elegant and simple?

User-Centered Design

- A system is evaluated from the user viewpoint.
  - Ideally by the users!
- Tension: when do we evaluate?
  - An incomplete product may not be usable;
  - If a product is complete, using evaluation has cost.
- Resolution: evaluate *prototype*!



Prototype (1 of 3): Paper Simulation



- Hand-drawn user interfaces:
  - on paper or card;
  - made on the spot.
- Developers animate:
  - Present to test user;
- Users act:
  - Indicate what they would do.

Prototype (2 of 3): Wizard-of-Oz

- Software has right “look”
  - But barely functional.
- Scripted interaction only
  - All responses are “canned.”
- Illusion is effective.

Prototype (3 of 3): Working Prototype

- The software system partly implemented:
  - User interface fully realized;
  - Functionality limited.
- Particularly for feature requests:
  - New feature can get quick-and-dirty implementation
  - Quickly get feedback if the right feature is implemented.
- Comparison with TDD:
  - In TDD: feature request is realized in a test;
  - In UCD: feature request is realized in a user-interface.

In both cases, we delay implementation until more understanding gained: Move decisions closer to customers.

Forms of User Evaluation

- Empirical evaluation
  - “How many tasks accomplished in N minutes?”
- Qualitative evaluation
  - Observers find patterns in interaction;
  - Users give feedback after use.
- ”Dogfooding” (internal evaluation)
  - Developers use product as soon as feasible.
- Heuristic evaluation
  - Evaluate against best practices.

Best Practice Heuristics (Nielsen)

- "*Discount* ($) usability engineering methods"
  - Pioneered by Jakob Nielsen in the 1990s
- Involves a small team of evaluators to evaluate an interface based on recognized usability principles
- Heuristics–"rules of thumb"

Much cheaper than an evaluation with "real" users!

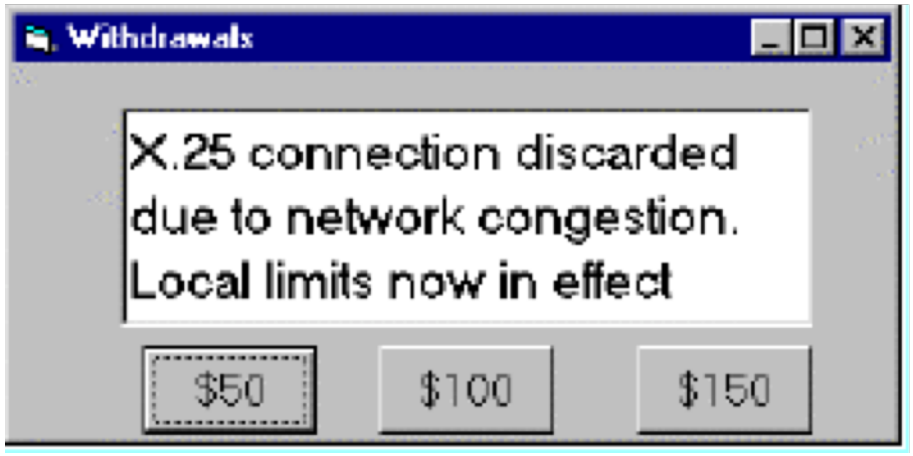
(Adapted from slides by Bonnie John and Jennifer Mankoff)

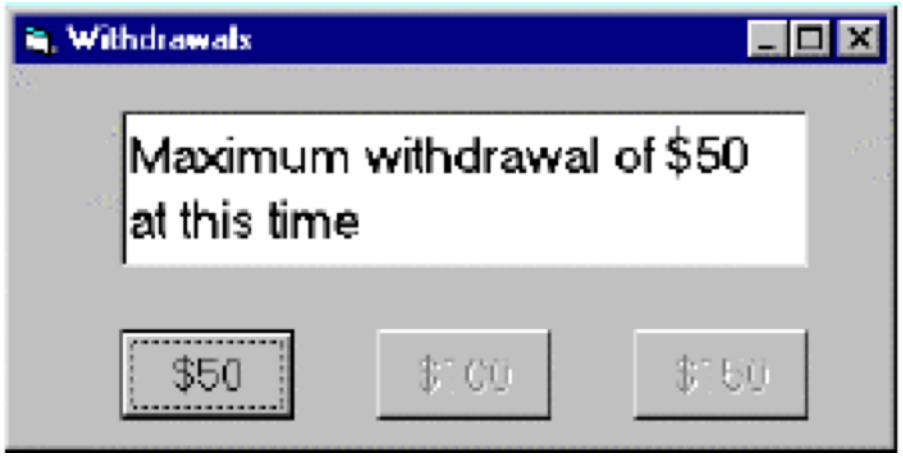
H1: Visibility of System Status

- Interface should show:
  - What input has been received;
  - What processing is currently happening;
  - What results have already been completed.
- This feedback allows
  - user to monitor progress towards solution of their task;
  - allows the closure of tasks; and
  - reduces user anxiety (Lavery et al).
- Great podcast with interview with Brad Myers, creator/popularizer of progress bar in his 1985 PhD thesis (99 Percent Invisible 9/3/19)

H2: Match Between System and Real World

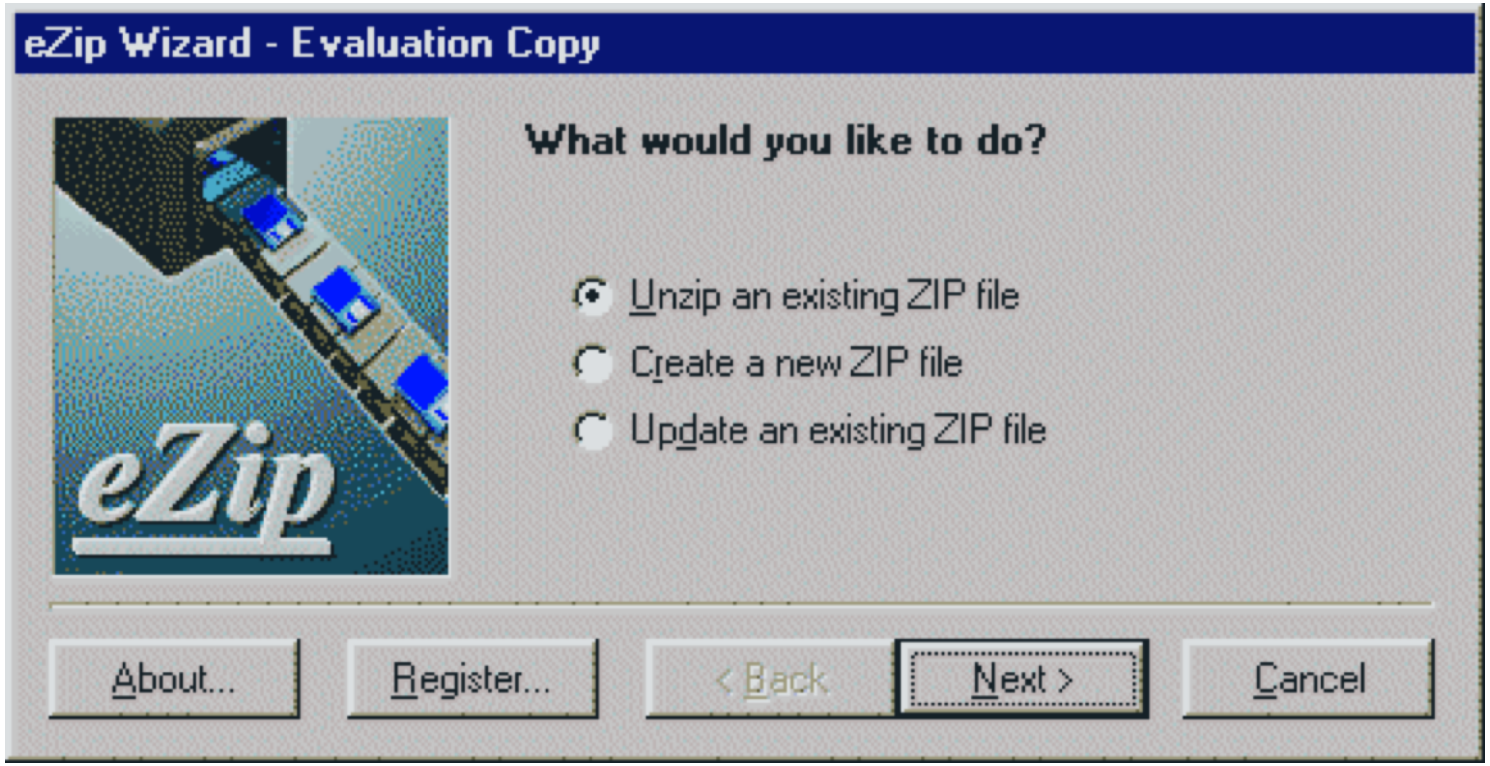
- Speak the users' language.
- Follow real world conventions.
- Don't use internal jargon ("X.25 connection discarded")
- "Gray out" illegal options.





H3: User Control and Freedom

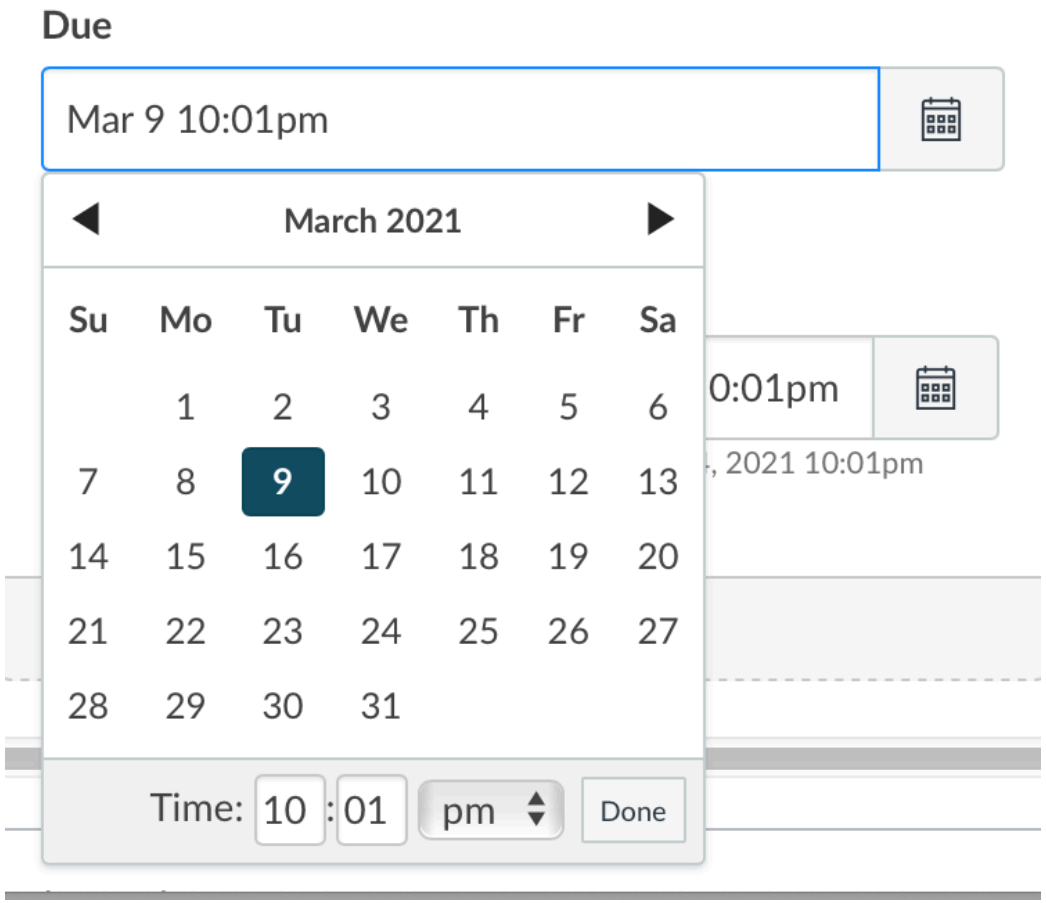
- “Exits” for mistaken choices: undo, redo, cancel
- Don’t force down fixed paths.



H4: Consistency and Standards

- Same words, situations, actions, should mean the same thing in similar situations;
- Same things look the same and be located in the same place.
- Text consistent with figures. ⟶
- Different things should be different.

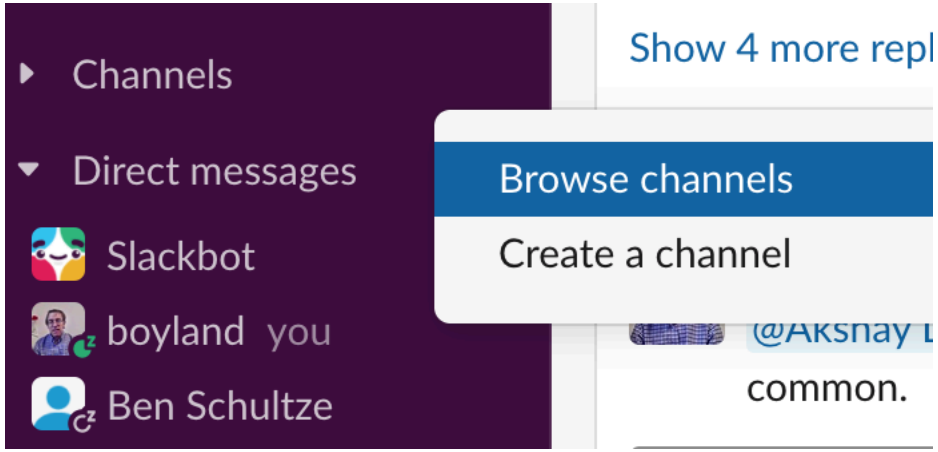
H5: Error Prevention



- Careful design can prevent a problem from occurring in the first place.
- It's easier to point to a date on the calendar than to type it in the correct format.

H6: Recognition rather than Recall

- Make objects, actions and options visible or easily retrievable.
- It’s easier to pick out the channel we want to add than to enter the correct name.

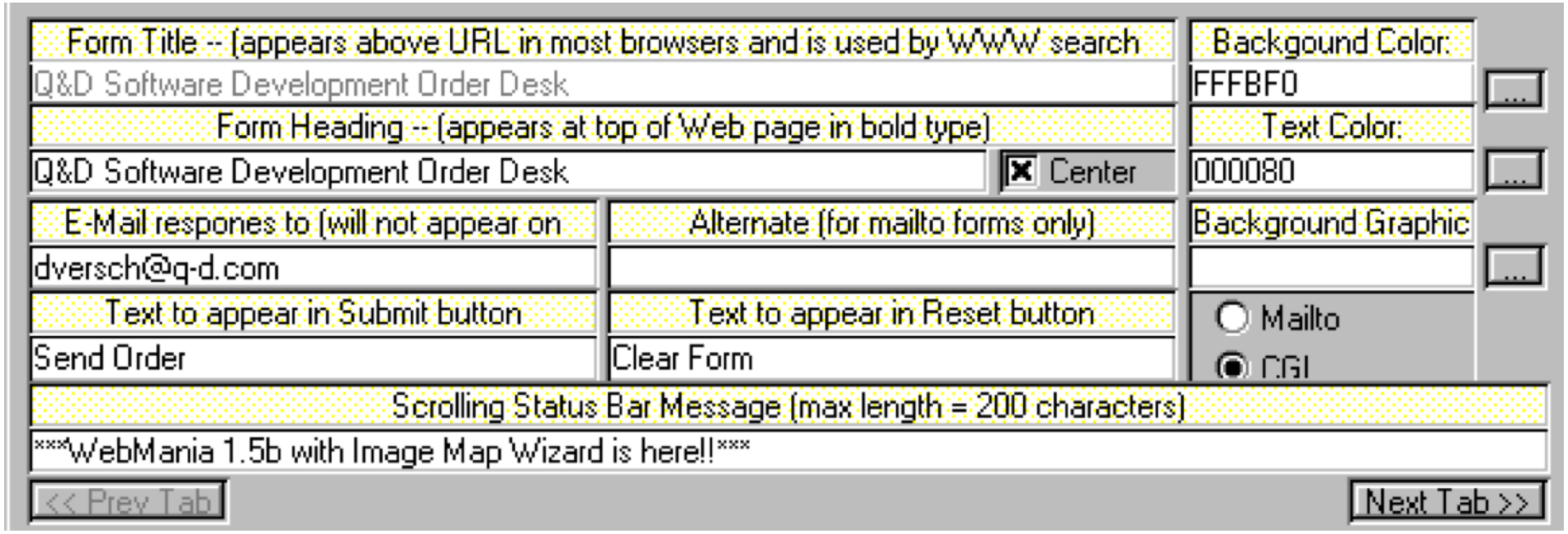


H7: Flexibility and Efficiency of Use

- Accelerators for experts (e.g., gestures, kb shortcuts)
- Allow users to tailor frequent actions (e.g., macros)

H8: Aesthetic and Minimalist Design

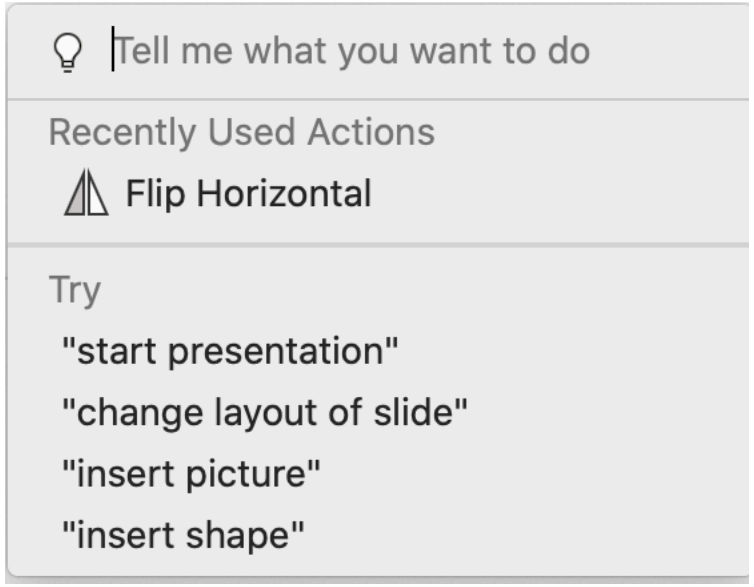
- Interfaces should not contain irrelevant or rarely needed information.



H9: Help users recognize, diagnose, and recover from errors

- Use standards to convey errors;
- Error messages should be in language user will understand;
- Precisely indicate the problem;
- Constructively suggest a solution.

H10: Help and Documentation



- Should be
  - Easy to search;
  - Focused on the user's task;
  - List concrete steps to carry out;
  - Always available.

Review: Learning Objectives for this Lesson

- You should now be able to:
  - List the three main steps to determine requirements;
  - Describe the major aspects of usability;
  - Articulate the process of user-centered design;
  - Explain several heuristics for good user interaction.

Looking forward…

- In the next part of Lesson 6, we describe React, a user-interface architecture.