Overview and Heuristic Evaluation

SWE 632
Fall 2023

© Thomas LaToza

In class exercise

- Write down at least 3 characteristics that makes something **usable**

Characteristics of usability

Characteristics of usability

- ease of use
- productivity
- efficiency
- effectiveness
- learnability
- retainability
- user satisfaction

Usable or unusable?

A teapot

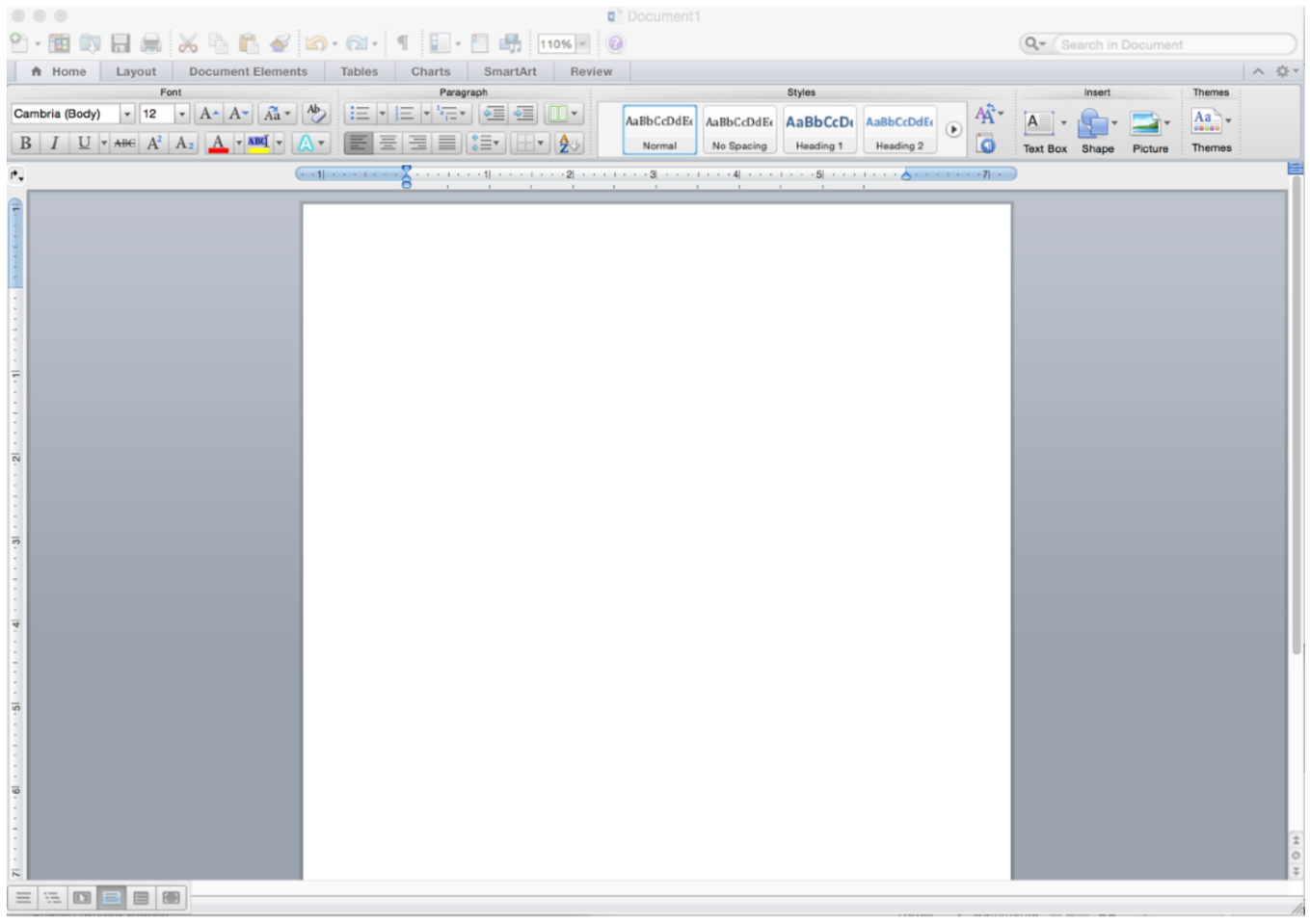
From Don Norman, Emotional Design

Usable or unusable?

A door

Usable or unusable?

A word processor



Usability

- A property of the relationship between
    - humans with goal-driven tasks
    - an artifact
- The speed and success with which the goals can be accomplished (task **performance**)

Needfinding

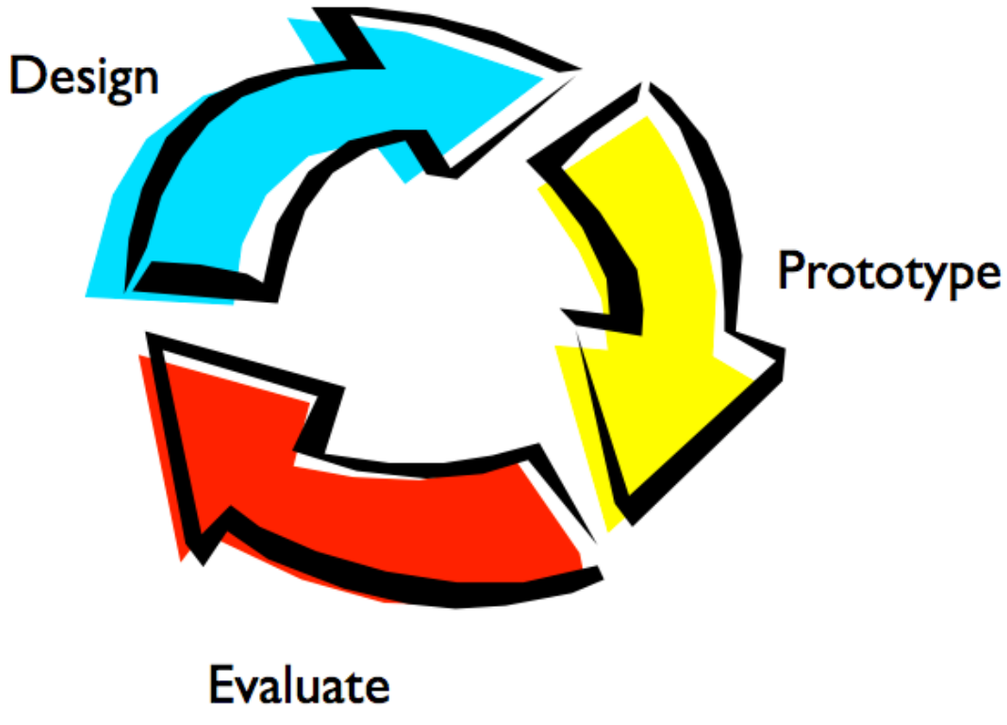
- Given an existing artifact and humans doing a set of tasks, determine **goals** and identify **usability issues** that decrease task performance

User-centered design

- Given humans with goals and tasks, design an **artifact** that helps to accomplish these tasks

Iterative **User-centered design**

- Given humans with goals and tasks, **re**design an **existing** artifact that helps to accomplish these tasks **faster** and **more successfully**



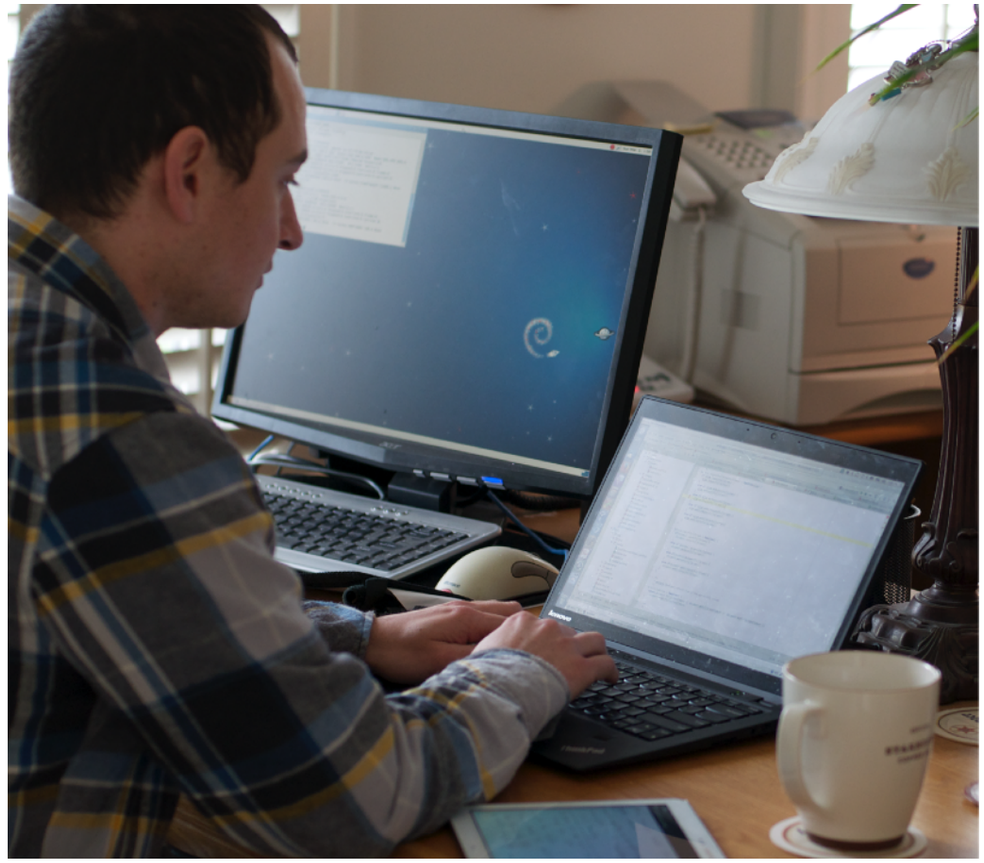
Usability evaluation

- Given humans with goals and tasks and a new artifact, identify usability issues that decrease task performance

Empirical: Usability evaluation study

- Given humans with goals and tasks and an artifact, **observe humans to** identify usability issues that decrease task performance

- (ground truth)



Analytical: Usability principles

- Given humans with goals and tasks and an artifact, **assess for conformance to UI principles to** identify usability issues that decrease task performance
- (lightweight approximation of ground truth)

Why study usability?

"The results show that in today's applications, an average of **48% of the code** is devoted to the user interface portion.
The average time spent on the user interface portion is 45% during the design phase, 50% during the implementation phase, and 37% during the maintenance phase."
– Myers & Rosson, CHI'92

Why study usability?

Adapted from Maneesh Agrawala & Bjoern Hartmann

Life-Threatening Errors

- 1995 American Airlines jet **crashed** into canyon wall, killing all aboard
- On approach to Rozo airport in Colombia
- Pilot skipped some of the approach procedures
- Pilot typed in “R” and system **completed** full name of airport to Romeo
- Guidance system executed turn at low altitude to head for Romeo airport
- 9 seconds later plane struck canyon wall
- Is the pilot to blame?
- http://en.wikipedia.org/wiki/American_Airlines_Flight_965

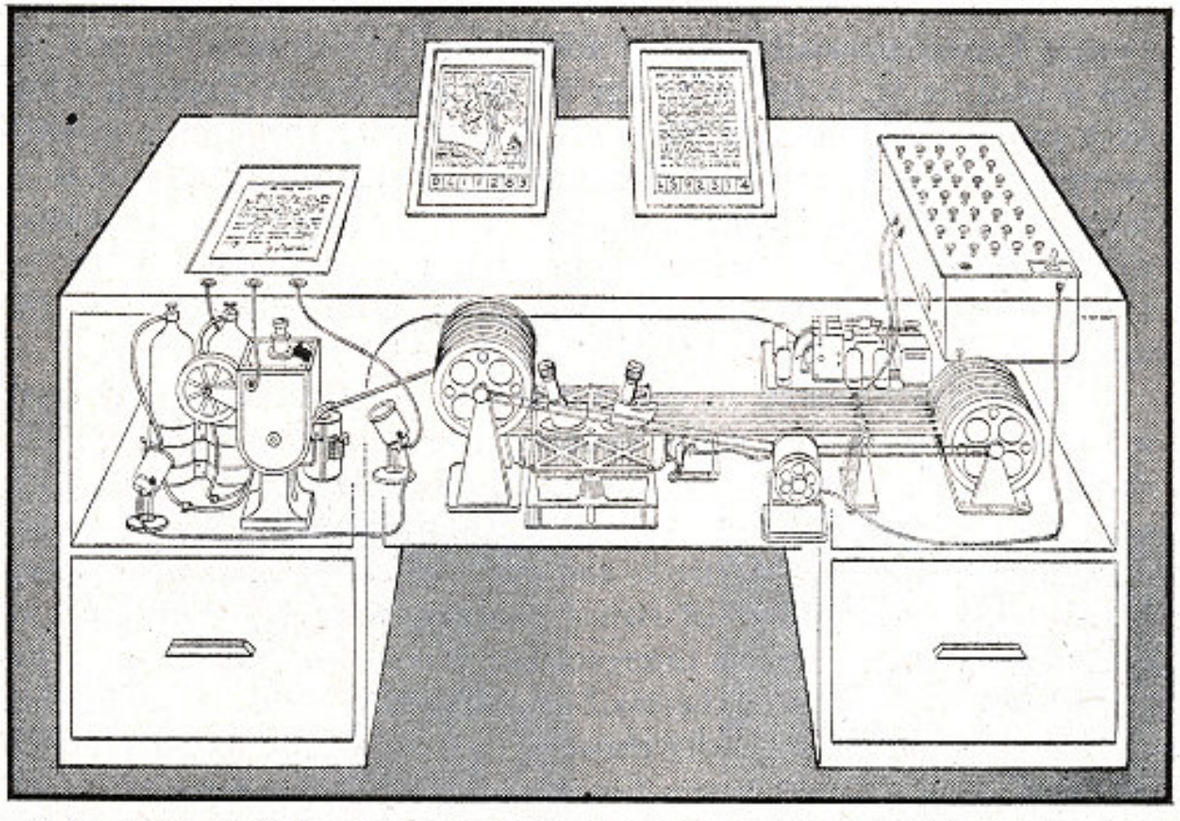
What usability is not

- Not “dummy proofing”
- Not being “user-friendly”
- Not just “usability testing”
- Not just making software pretty

As we may think

- Vannevar Bush, The Atlantic, July 1945

- Described the Memex and predicted hypertext, personal computers, the Internet, the WWW, speech recognition, online encyclopedias



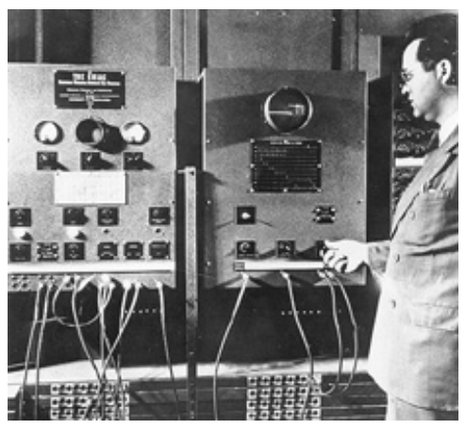
MEMEX in the form of a desk would instantly bring files and material on any subject to the operator's fingertips. Slanting translucent viewing screens magnify supermicrofilm filed by code numbers. At left is a mechanism which automatically photographs longhand notes, pictures and letters, then files them in the desk for future reference.

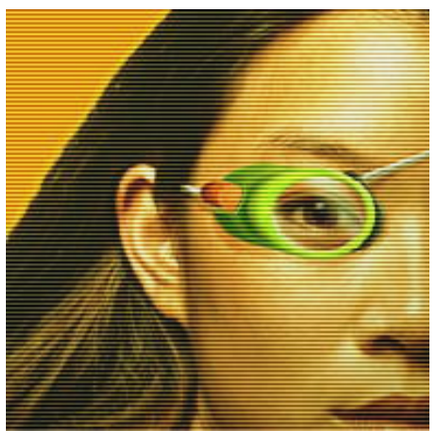
The user is NOT like me

- Understanding user needs, tasks, goals

Human-Computer Interaction

"a discipline concerned with the design, evaluation, and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them." ACM SIGCHI Curriculum Development Group Report, 1992





This course

- Comprehensive introduction to usability and human-computer interaction (HCI)
- Basic cognition, user-centered design, usability evaluations, principles for UI design

Instructor

- Thomas LaToza
  Associate Professor, Computer Science
  Office: ENGR 4431

- Research interests: software engineering, human-computer interaction, crowdsourcing

- Studies how developers interact with code and designs new ways to program.

- Have conducted dozens of user studies, including interviews, surveys, think-aloud usability studies, controlled experiments, field deployments

com › crowdcoding › commands › DesignDocCommand

DesignDocCommand.java

```java
package com.crowdcoding.commands;

import com.crowdcoding.entities.artifacts.DesignDoc;
import com.crowdcoding.servlets.ThreadContext;


public abstract class DesignDocCommand extends Command {
    protected long DesignDocId;

    // This function is called when a new DesignDoc must be created.
    public static DesignDocCommand create(String title, String description, boolean isApiArtifa
        return null;
    }

    private DesignDocCommand(Long DesignDocId) {
        this.DesignDocId = DesignDocId;
        queueCommand(this);
    }

    // All constructors for DesignDocCommand MUST call queueCommand and the end of
    // the constructor to add the
    // command to the queue.
    private static void queueCommand(Command command) {
        ThreadContext threadContext = ThreadContext.get();
        threadContext.addCommand(command);
    }

    public void execute(final String projectId) {
        if (DesignDocId != 0) {
            DesignDoc designDoc = DesignDoc.find(DesignDocId);

            if (designDoc == null)
                System.out
                        .println("error Cannot execute DesignDocCommand. Could not find DesignD
                                + DesignDocId);
            else {
                execute(designDoc, projectId);
            }
        } else
            execute(null, projectId);

    }

    public abstract void execute(DesignDoc DesignDoc, String projectId);
```

DesignDocCommand

Ant Build | Database | Maven Projects | Google Cloud Storage | ActiveDocs

Table of Content | All Rules | Violated Rules | Generate Rules

Rules applicable for File:

CrowdCode-master/CrowdCoding/src/com/crowdcoding/commands/DesignDocCommand.java

(view the rule and all snippets)

All Microtask commands must be handled by Command subclasses

IF a method is a static method on Command THEN it should implement its behavior by constructing a new Command subclass instance. The Command class contains a number of static methods. Each method creates a specific type of Command by invoking the constructor of the corresponding subclass.

Microtask | Command | Sharding

Examples 0 out of 54 Violated 1 out of 1

Commands must implement execute (view the rule and all snippets)

IF a class is a subclass of Command THEN it must implement execute. Commands represent an action that will be taken on an Artifact. In order for this action to be invoked, each subclass of Command must implement an execute method. This method should not be directly invoked by clients, but should be used by the Command execution engine.

Microtask | Command | Sharding

Examples 0 out of 53 Violated 0 out of 0

(view the rule and all snippets)

Artifacts should be marked as a data region with an @Entity annotation

IF an object is an artifact subclass THEN it needs to be an entity. To signal that instances of a class constitute a separate data region, the class should have the @Entity annotation. All

Graduate Teaching Assistant

- Divesh Upreti
- Office hours: TBA

Resources

- Course website - Syllabus, Schedule
- Piazza - Announcements, Assignments, Discussion, Questions
- Blackboard - grades

HWs (a.k.a. “project”)

- Build a (really simple) web app
- Use usability concepts and methods to identify usability issues
- Iteratively improve your app to address usability issues
- HW1: build app
- Remaining HWs: find usability issues and fix
    - Note: you'll be focused on improving usability **not** building parts of the app you didn't build before

HW0

- Due next Tuesday before class
- Form a group of 1, 2, or 3
- Pick an app to build
- Describe what you propose to build in ~1 paragraph

Examples of project apps

- Restaurant online ordering site
- Class schedule creator
- Sports statistics browser
- Fashion storefront
- Word guessing game

Policy on code reuse

- Can borrow code from online sources as much or as little as you'd like.
- **You must document** instances of code that you reuse.

Late HW assignments

- HWs will often involve peer evaluations
- Can submit up to 24 hours late, lose 10%
- **HW submissions more than 24 hours late will receive a 0**

Tech talks

- 14 minute overview of a front-end web technology or UX tool
- Groups of 4
- Use piazza to find a partner, reserve topics
- Only 1 group can cover a technology
- Signup by start of class next Tuesday

SWE Subject Pool

- To gain experience in user studies, you will participate in 6 hours of user studies, split across 2 or 3 studies
- All studies will be remote, with times on evenings & weekends
- Will have mechanism for signing up for studies - details on Piazza

In class activities

- Each class will include an extended in-class activity in small groups
- Practice methods on small examples
- Will generate a small hand-written deliverable
- Graded
    - Satisfactory: put forth a good effort in accomplishing the activity's goals (10/10)
    - Needs improvement: substantially misunderstood the activity or did not make meaningful progress (5/10)
    - Not present: did not submit deliverable from activity (0/10)
- To accommodate planned or unplanned absences, three lowest scores (including absences) dropped
- Turn in at the end of class

Exams

- Midterm exam and comprehensive final
- Includes both in class lectures and material from assigned readings
- Mix of multiple choice, short essay
- In-class, closed book

Grades

- In-Class Activities: 10%
- Tech talk: 5%
- HWs and project presentation: 30%
- SWE Subject Pool participation: 10%
- Mid-term exam: 20%
- Final exam: 25%

Heuristic evaluation

- “**Discount** usability engineering methods”… Jakob Nielsen
- Involves a small team of evaluators to evaluate an interface based on recognized usability principles
- Heuristics–”rules of thumb”

Adapted from slides by Bonnie John and Jennifer Mankoff

Heuristics

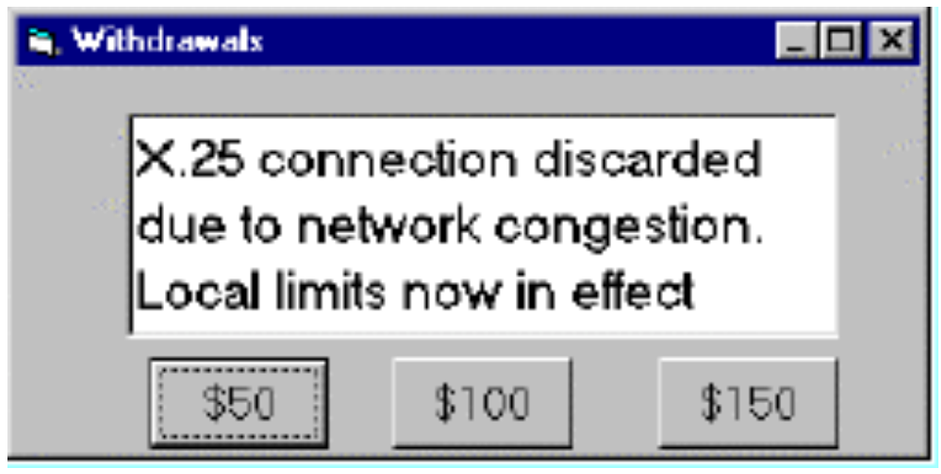
1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition vs. recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

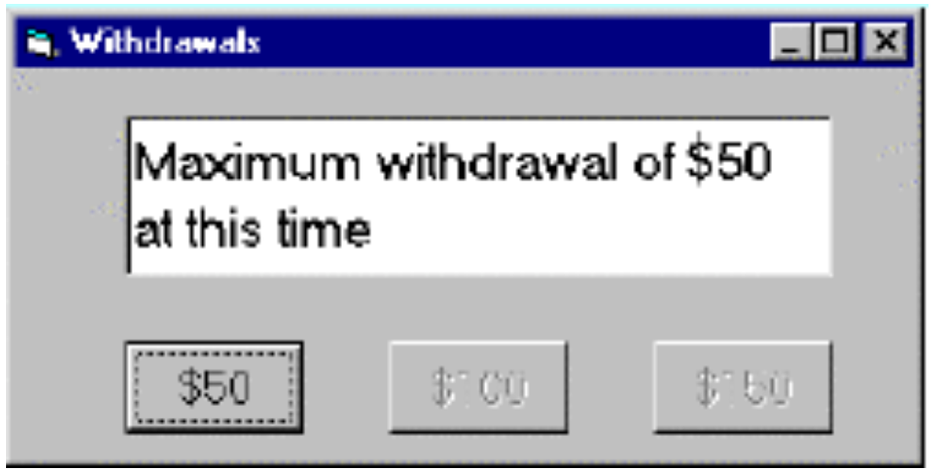
H1: Visibility of System Status

- What input has been received--Does the interface above say what the search input was?
- What processing it is currently doing--Does it say what it is currently doing?
- What the results of processing are--Does it give the results of processing?
- Feedback allows user to monitor progress towards solution of their task, allows the closure of tasks and reduces user anxiety (Lavery et al)

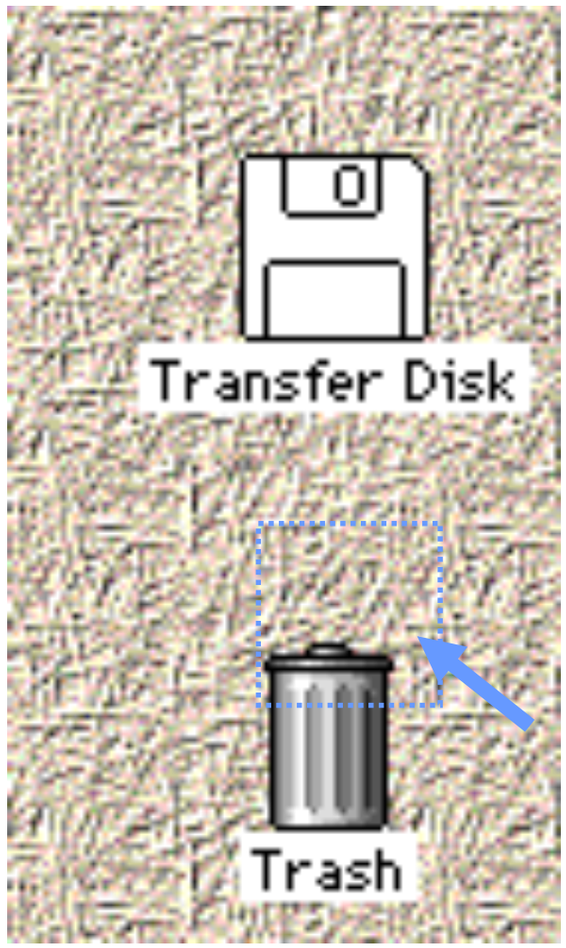
H2: Match between system and the real world

- Speak the users' language
- Follow real world conventions



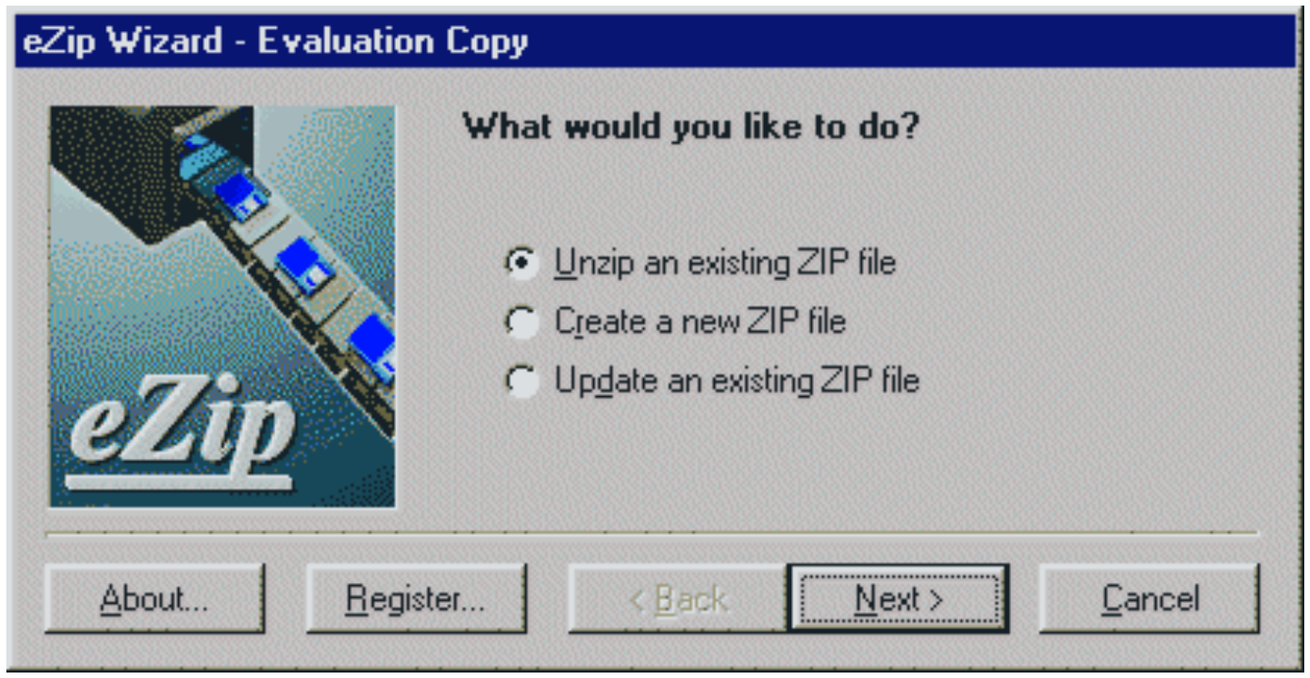


Mail
Read


Transfer Disk
Trash

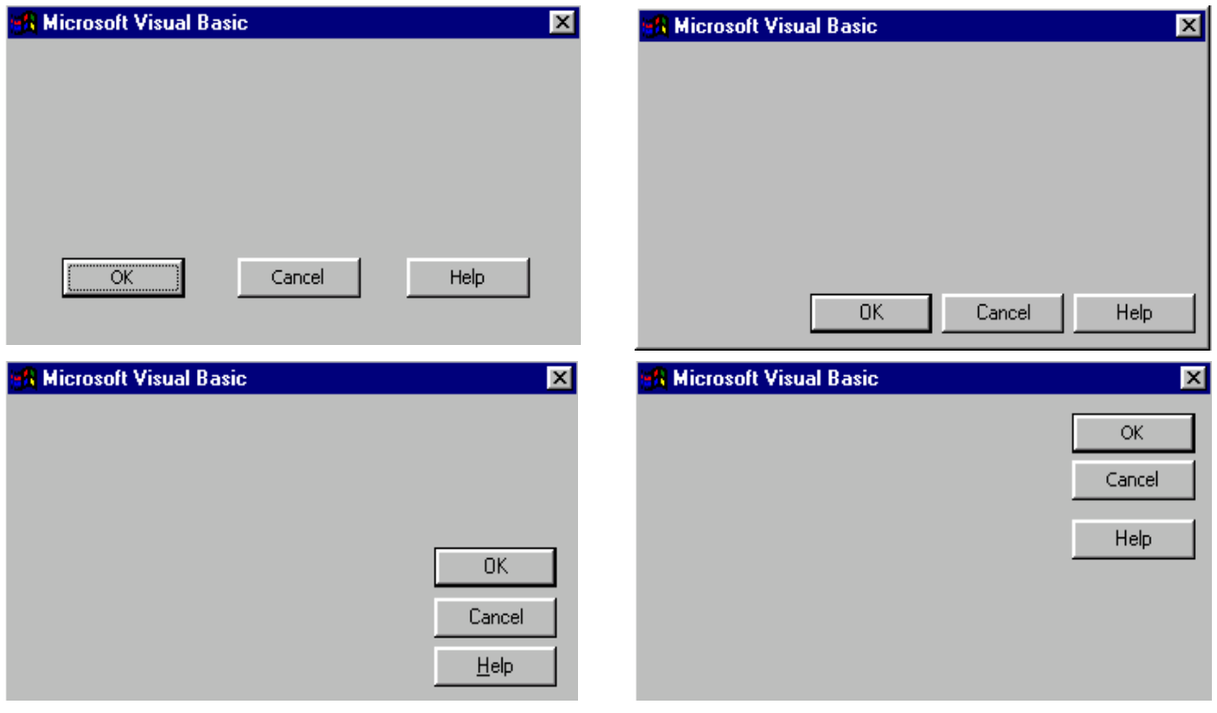
H3: User Control and Freedom

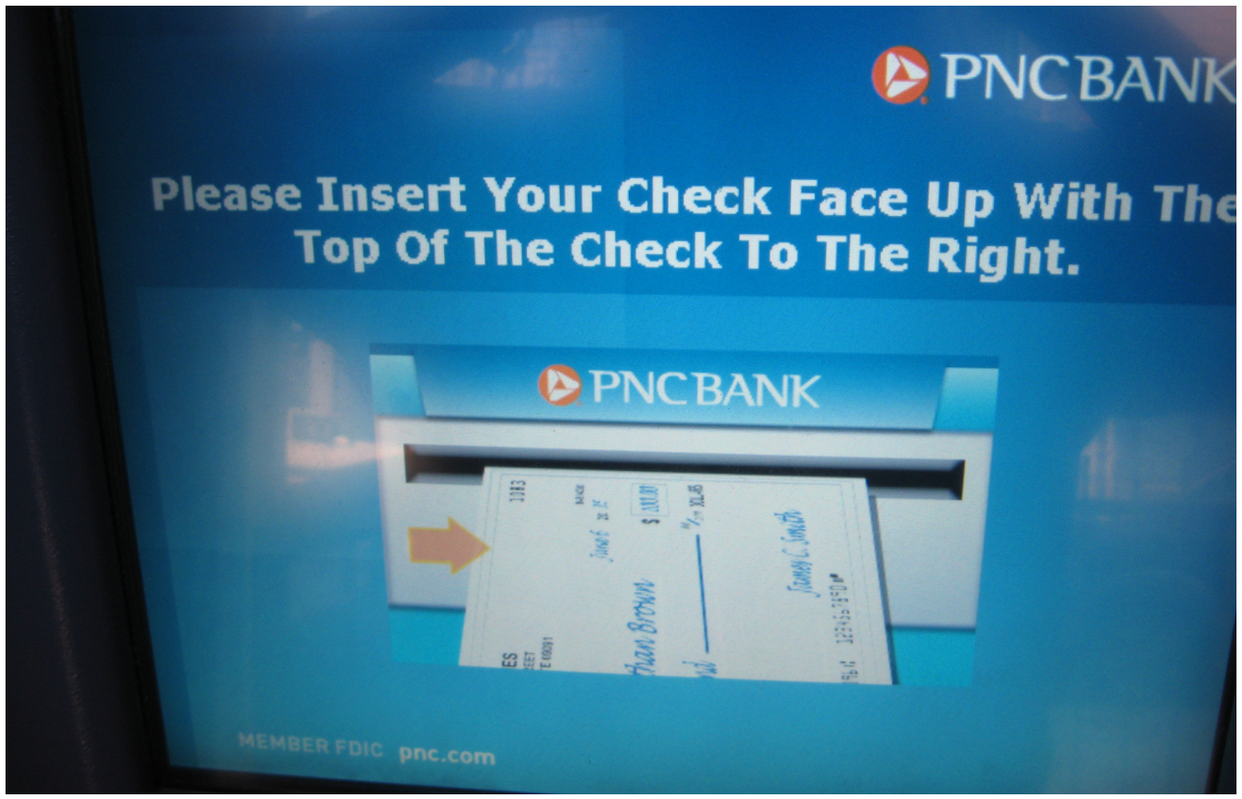
- “Exits” for mistaken choices, undo, redo
- Don’t force down fixed paths



H4: Consistency and Standards

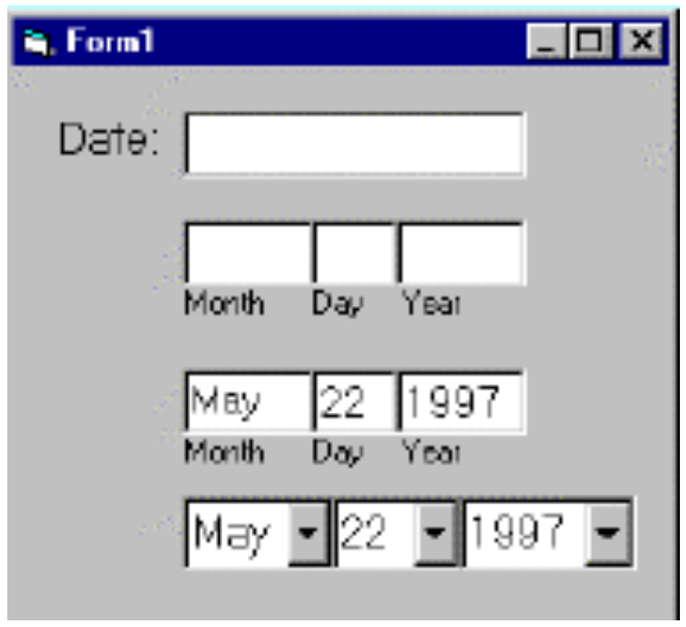
- Same words, situations, actions, should mean the same thing in **similar** situations; same things look the same, be located in the same place.
- Different things should be different

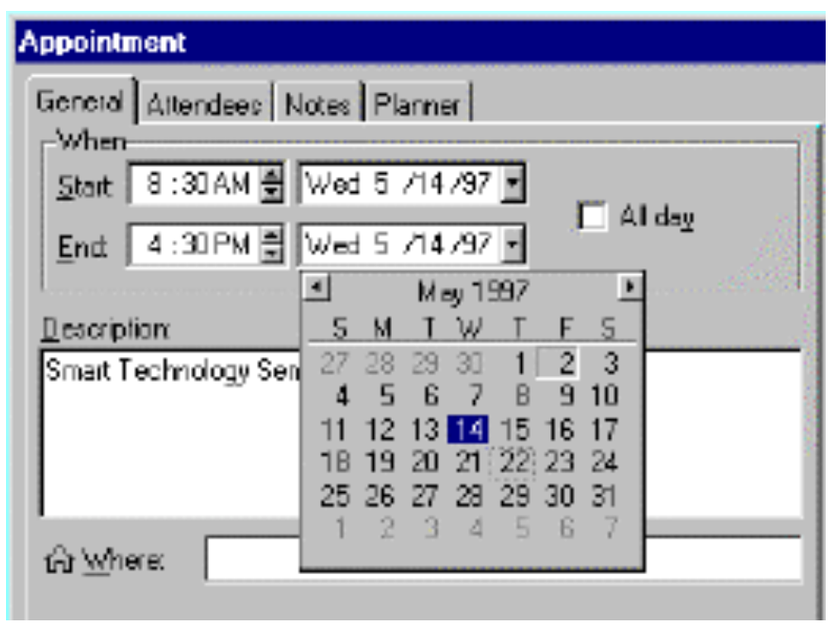



PNC BANK
Please Insert Your Check Face Up With The
Top Of The Check To The Right.
PNC BANK
MEMBER FDIC pnc.com

H5: Error prevention

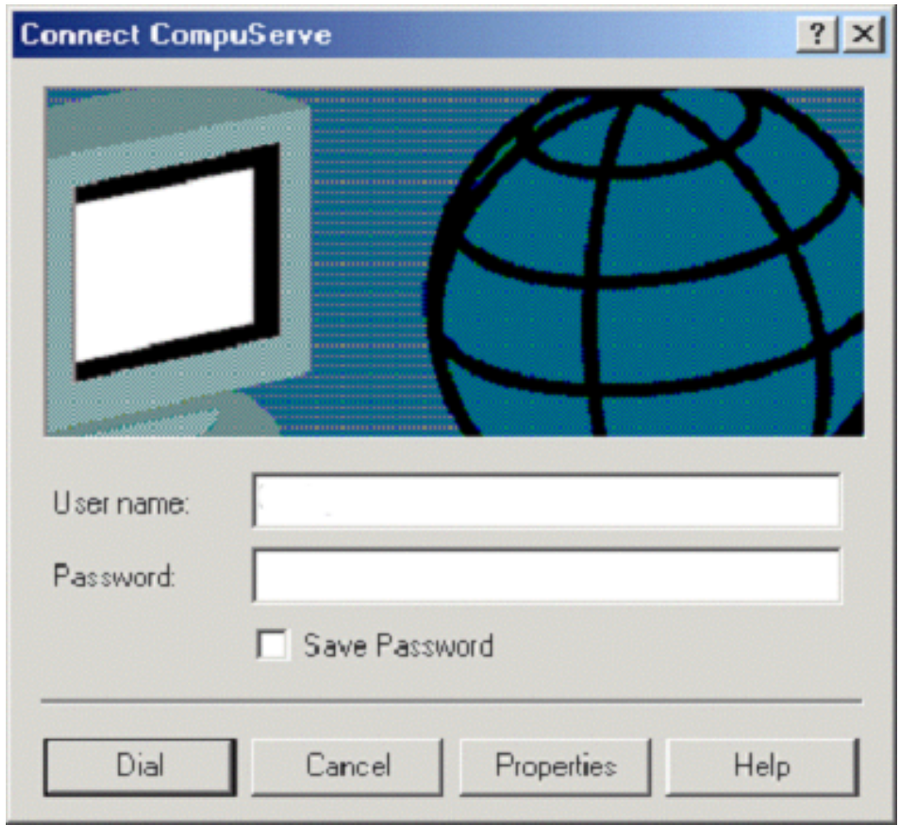
- Careful design which prevents a problem from occurring in the first place





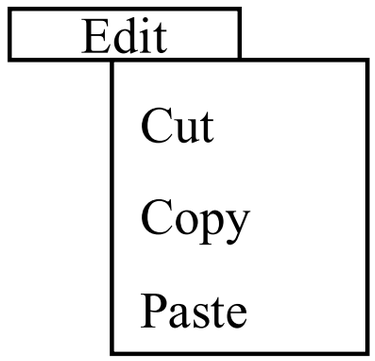
H6: Recognition rather than recall

- Make objects, actions and options visible or easily retrievable



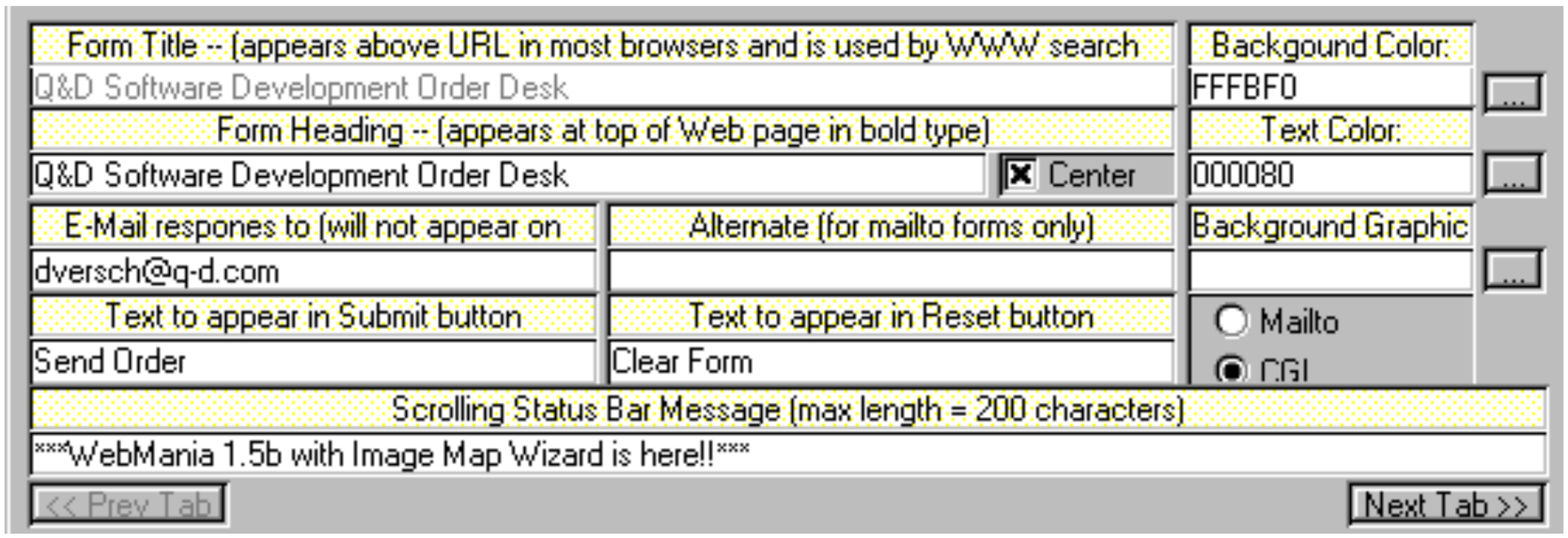
H7: Flexibility and Efficiency of Use

- 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

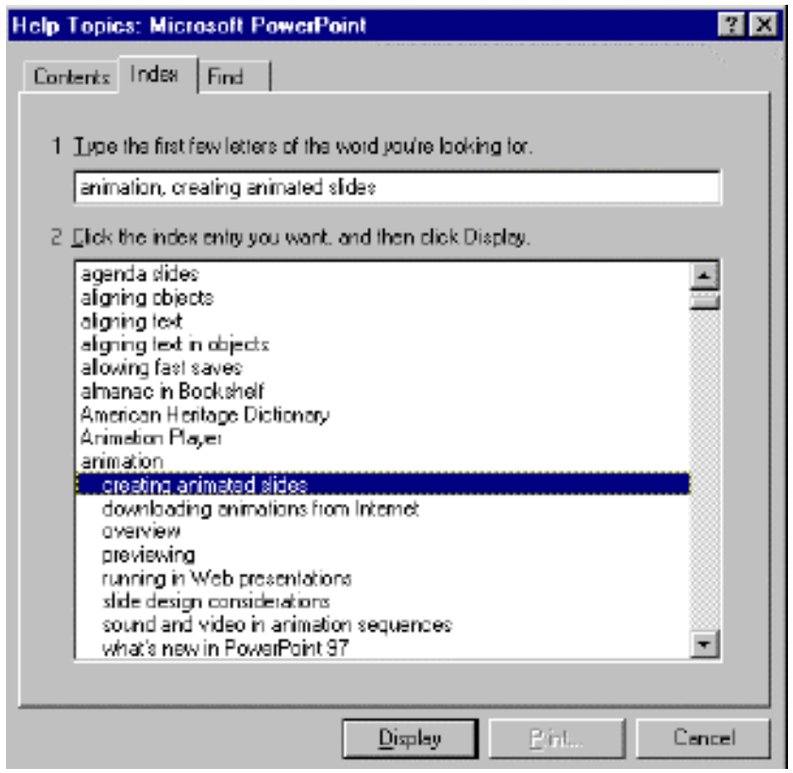


H9: Help users recognize, diagnose, and recover from errors

- Error messages in language user will understand
- Precisely indicate the problem
- Constructively suggest a solution

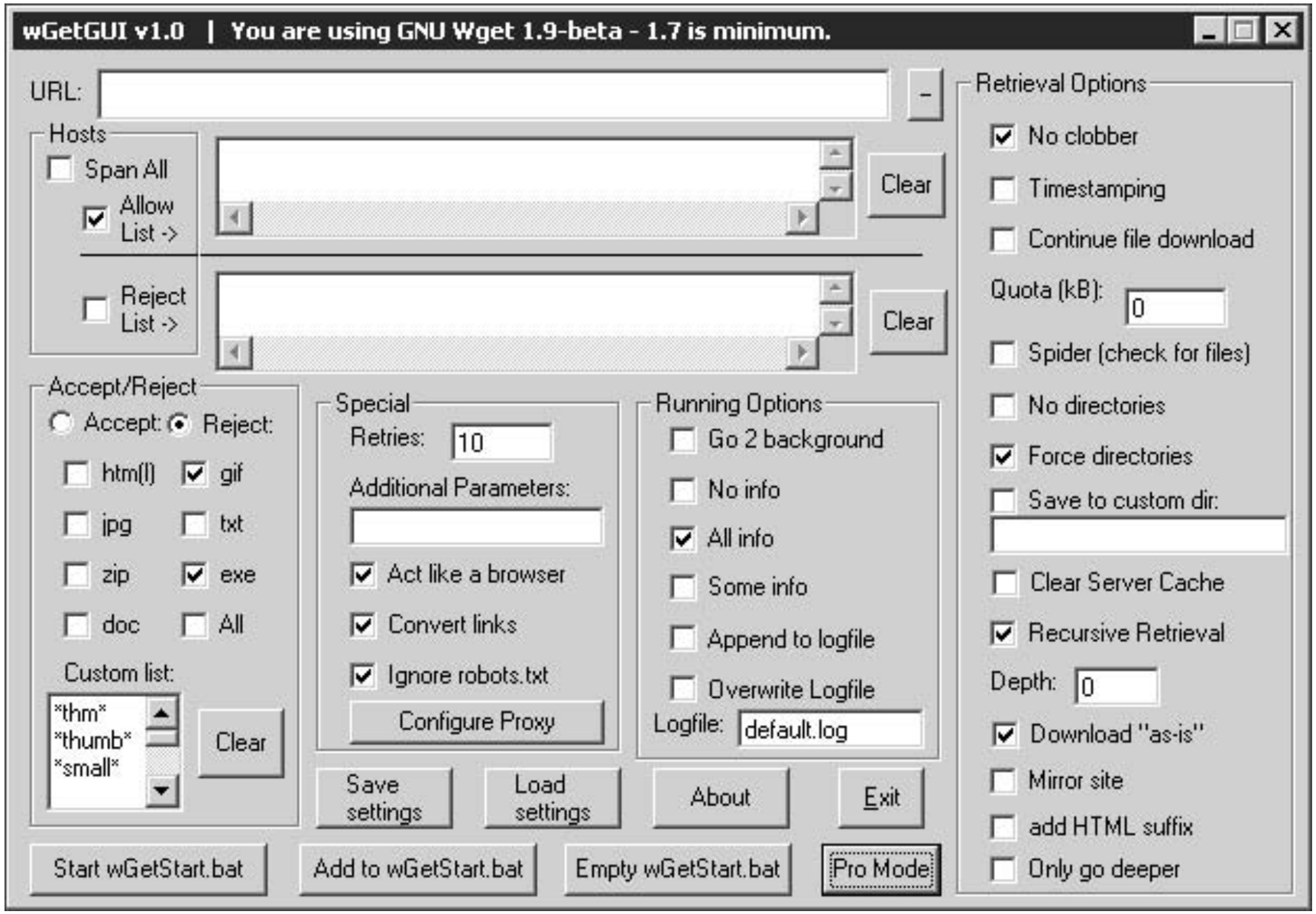
H10: Help and documentation

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



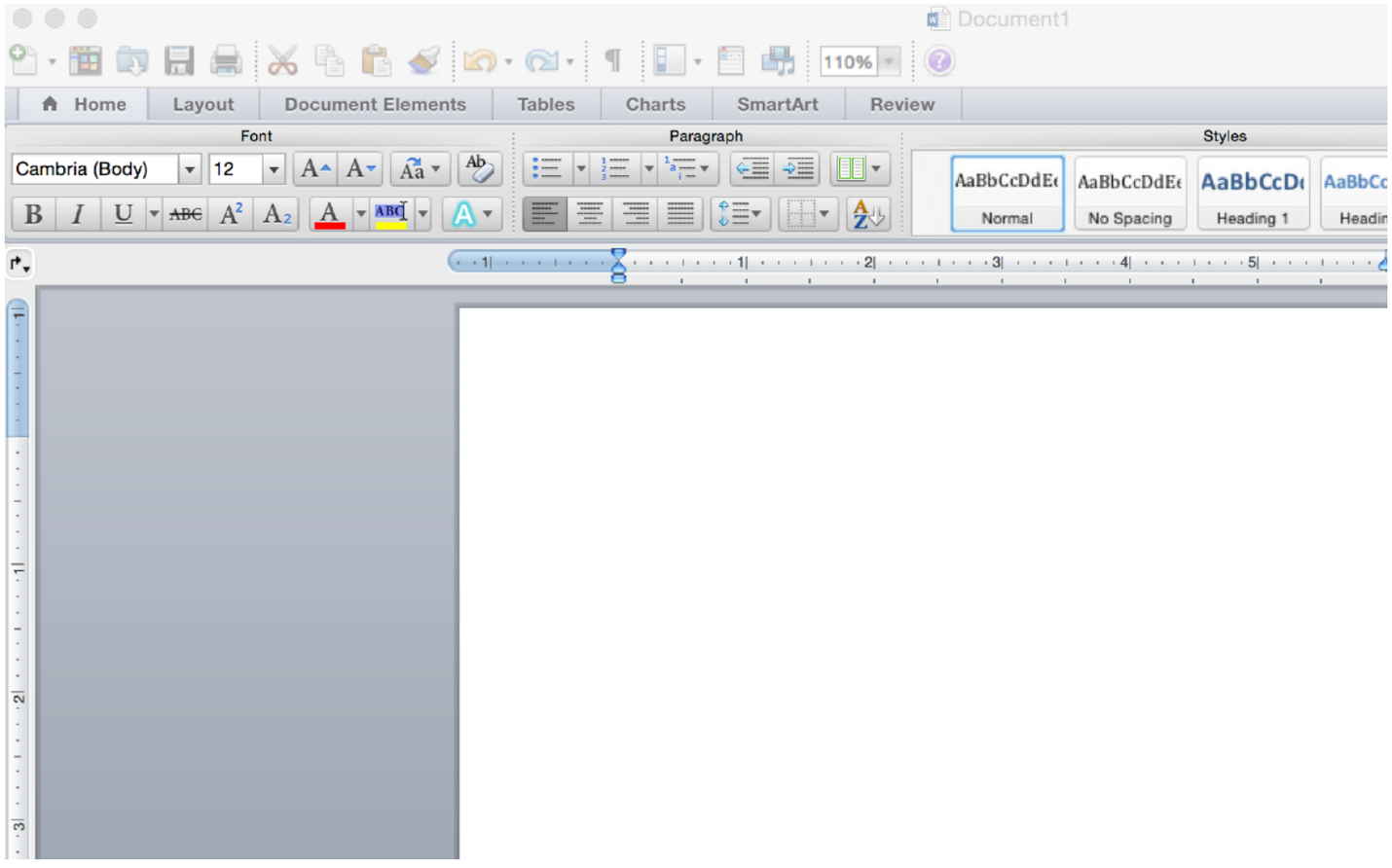
Example

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 vs. 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



Example

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 vs. 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



Using heuristic evaluation

- Can be used informally to identify issues in a website
- Can be used as a more formal usability inspection method
- Evaluators each first separately identify issues
- Issues then combined from each evaluator

Heuristic evaluation in groups

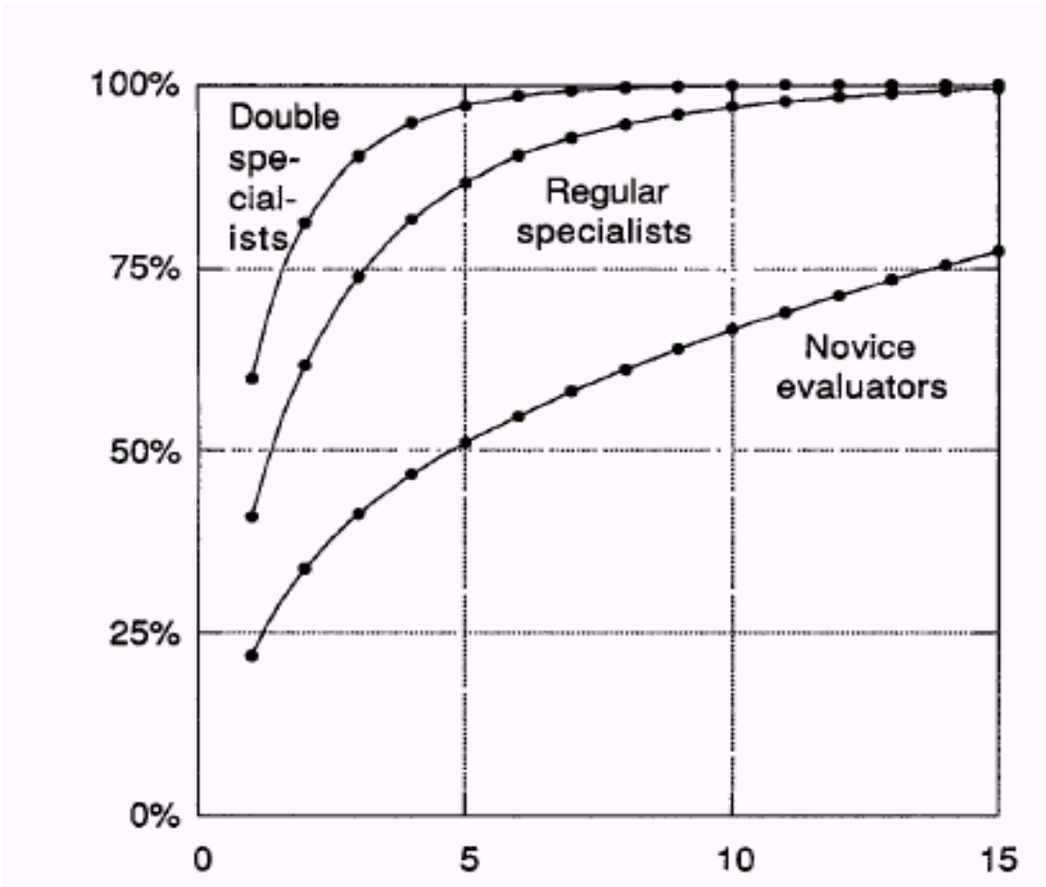


Figure 2 *Average proportion of usability problems found as a function of number of evaluators in a group performing the heuristic evaluation.*

Advantages of HE

- “Discount usability engineering” - Intimidation low
- Don’t need to identify tasks, activities
- Can identify some fairly obvious fixes
- Can expose problems user testing doesn’t expose
- Provides a language for justifying usability recommendations

Disadvantages of HE

- Un-validated
- Do not employ real users
- Can be error prone
- Better to use usability experts
- Problems unconnected with tasks
- Heuristics may be hard to apply to new technology

Ways to use HE

- Early in design process to catch major issues
- When time or resources are not available for empirical usability evaluation

In class activity

- Form groups of 3 or 4
- Together select an application or website (e.g., Excel, Google Maps, Mason CS website)
- Work individually to identify at least 6 usability issues, reflecting at least 4 different heuristics
- For each issue, identify the heuristic, identify the functionality in the application, and summarize how the heuristic is violated in a few sentences

Heuristics - Find 6 Issues

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 vs. 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