

# User Interface Design and Development V

## (Usability Engineering and Usability Inspection Methods)

Nan Tu PhD

Spring, 2006

Tsinghua University

Department of Industrial Engineering

# Usability Definitions (ISO)

- Usability is the "effectiveness, efficiency and satisfaction with which a specified set of users can achieve a specified set of tasks in a particular environment."

# Usability Criteria

- **Learnability:** ease of learning for **novice** users.
- **Efficiency:** steady-state performance of **expert** users.
- **Memorability:** ease of using system intermittently for casual users.
- **Errors:** error rate for minor and catastrophic errors.
- **Subjective Satisfaction:** how pleasant system is to use.

# Usability Engineering Lifecycle

1. Know the User
2. Usability Benchmarking
3. Goal-Oriented Interaction Design
4. Iterative Design:
  - (a) Prototyping
  - (b) Usability Evaluation (Inspection and Testing)
5. Follow-up Studies

# Know the User

- Qualitative research: observation and interviews.
- Draw up a user profile for each (potential) class of user, based on behavioral and demographic variables.
- Identify user goals and attitudes.
- Analyze workflow and context of work.
- Draw up a set of typical user scenarios.

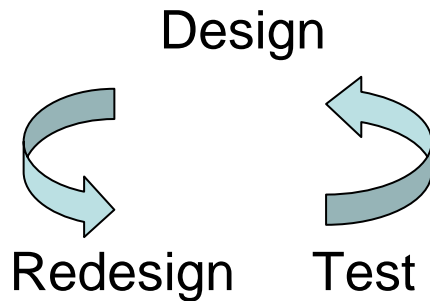
# Usability Benchmarking

- Analyze competing products or interfaces heuristically and empirically.
- Set measurable usability targets for your own interface.

# Interaction Design

- Goal-oriented initial design of interface
  - A very complex step, may be covered in the later lectures

# Iterative Design



- **Building Prototypes**
  - Verbal description.
  - Paper prototype.
  - Working prototype.
  - Implementation of final design.
- **Usability Evaluation**
  - Usability Inspection: Inspection of interface design using heuristics and guidelines (no user tests).
  - Usability Testing: Empirical testing of interface design with real users.

# Classifying Users

- Personal Attributes
  - Age
  - Gender
  - Educational Level
  - Computer Experience, etc
- Product Usage (Task, Goal) Attributes
  - Goal or task
  - User Experience

# Ethnographic Interview

- A combination of immersive observation and directed interview techniques.
  - Observe the user using their current tools in their normal environment.
  - Interviewer assumes the role of an apprentice learning from the master craftsman (user).
  - Alternate between observation of work and discussion of its structure and details.

# Usability Benchmarking

- How usable is the competition?
- How much better should your interface be?
- What is your likely return on investment?

# Competitive Analysis

- Determine the current state of the art and decide how far to raise the bar.
- Analyze competing products or interfaces heuristically or empirically.
- “Intelligent borrowing” of ideas from other systems.

# Set Usability Targets

- Decide in advance on usability metrics and desired level of measurable usability. i.e:
  - The current system exhibits 4.5 errors per hour on average for an experienced user. The target for the new version is less than 3 errors per hour.
  - From competitive analysis, on the main competing web site, novice users take 8 mins. and 21 secs. on average to book a flight. The target for our new web site is 6 mins.

# Return of Investment

- Estimate return on investment (ROI) by performing a financial impact analysis.
- Compare potential savings based on loaded cost of users to the estimated cost of the usability effort.

# User Interface Quality Criteria

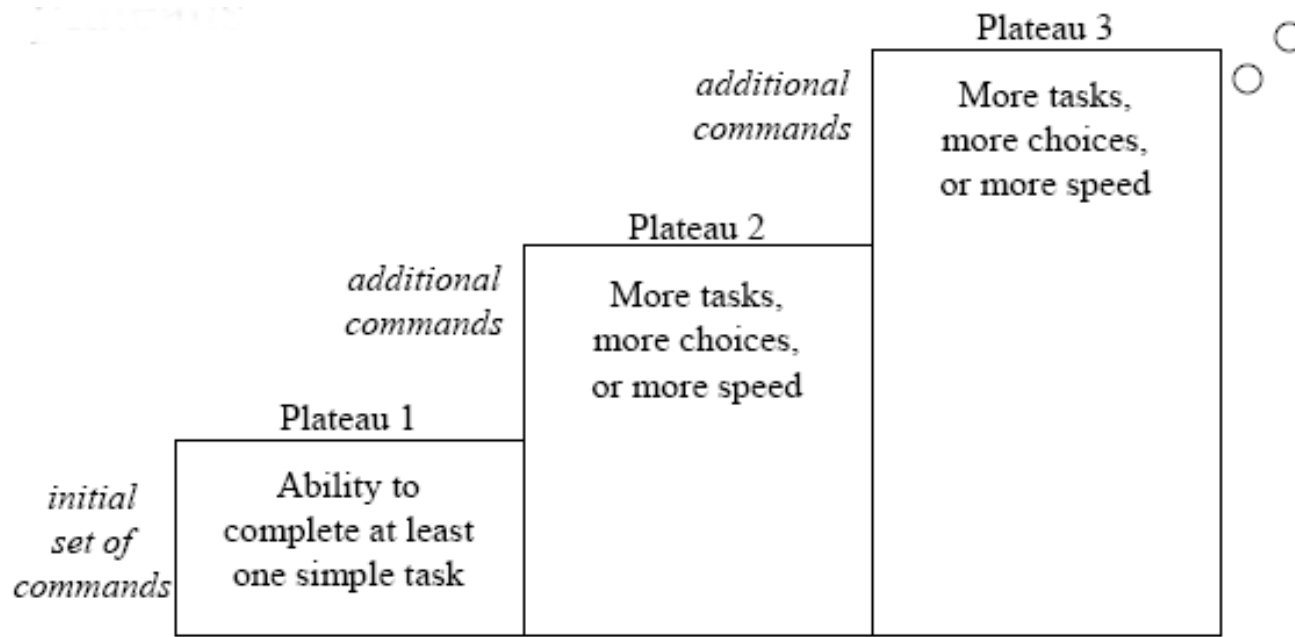
- Consistency
- Feedback
- Efficiency
- Flexibility
- Clearly marked Exits
- Wording in users' language
- Task orientation
- Control
- Recovery and Forgiveness
- Minimize Memory Load
- Transparency
- Esthetics and Emotional
- Effect

# Usability Criteria

- **Classical Criteria:**
  - Time to learn
  - Speed of performance
  - Rate of user errors
  - Retention of skills
  - Subjective satisfaction
- **Additional Criteria (User Experience):**
  - Attractivity
  - Enjoyability
  - Trust

# Time to Learn

- Length of time it takes to learn how to use an interface
- With complicated interfaces, learning happens in “plateaus”



# Speed of Performance

- Speed of user interface, NOT software
- Number of characters to type, buttons to press, mouse-clicks, mouse movements, ...
- Speed of performance often directly conflicts with time to learn
  - Faster systems are often harder to learn
  - Unix vs. Windows

# Rate of User Errors

- A UI can be structured so as to make user mistakes more or less likely
- Affected by factors such as:
  - Consistency
  - Instructions
  - Logical arrangement of screens
- Importance depends on criticality of software

# Retention of Skills

- We quickly forget how to use some user interfaces, but remember others for life
  - Airplanes vs. bicycles
- Affected by how closely the syntax of the operations match our understanding
- If learning is very very fast, retention may be less important

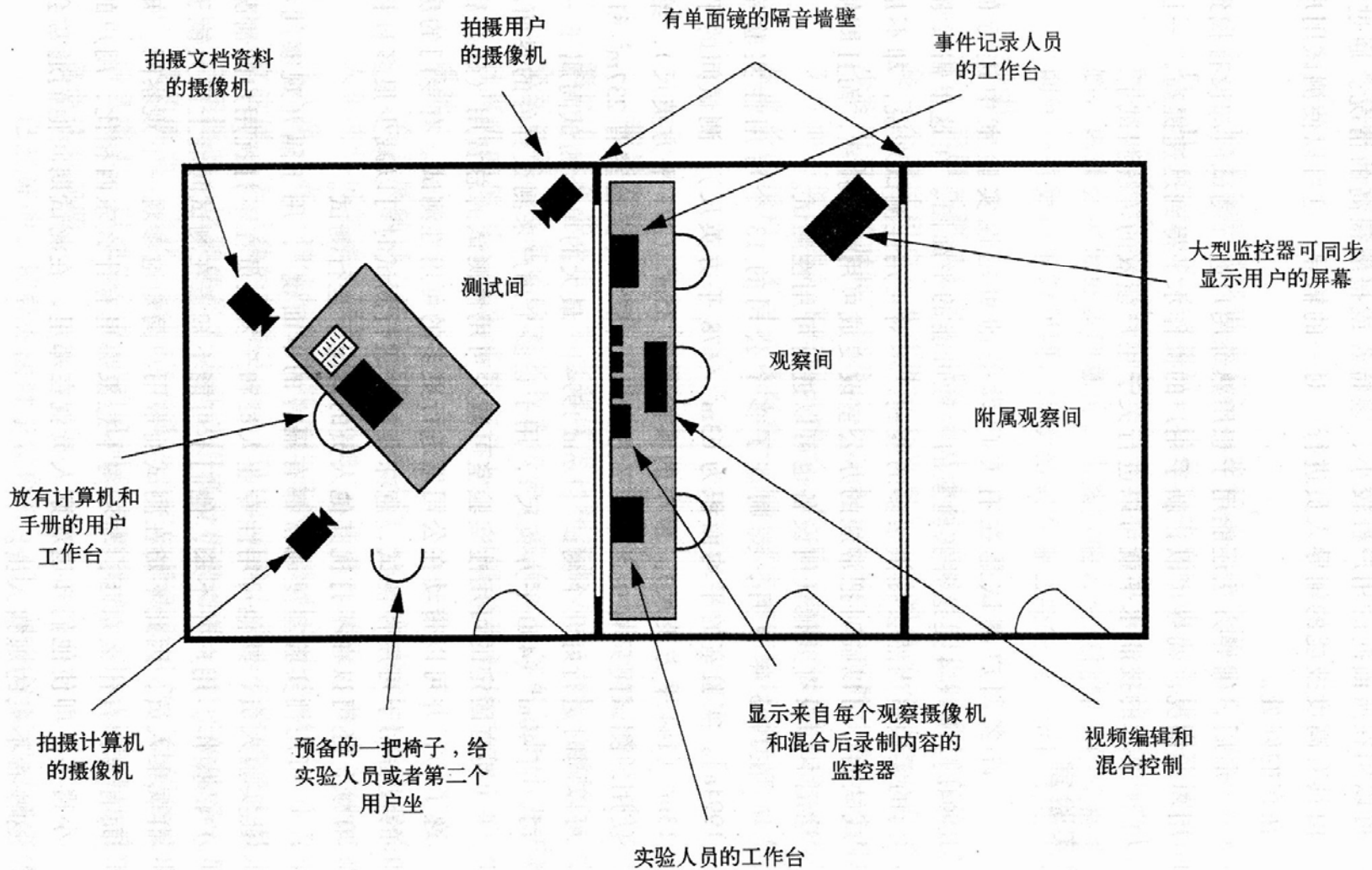
# Subjective Satisfaction

- How comfortable the users are with the software
- The other criteria are very analytical, objective, and measurable
- Subjective satisfaction captures other issues that are more specific to individual taste and background
- A little harder to measure

# Usability Inspection Methods

- **Heuristic Evaluations**
  - Small set of evaluators examines interface and judges its compliance with recognized usability principles.
- **Cognitive Walkthrough**
  - Task-oriented walkthrough based on formal cognitive model of user behavior. (analysis Learnability)
- **Action Analysis**
  - Quantitative analysis of actions to predict time required for tasks, based on time estimates for typical interface actions. (analysis efficiency)

# Experiment – Usability Lab



# Preparing for a Test

- Select your participants
  - understand background of intended users
  - use a questionnaire to get the people you need
  - don't use friends or family
- Prepare scenarios that are
  - typical of the product during actual use
  - make prototype support these (small, yet broad)
- Practice running the computer to avoid “bugs”

# Conducting a Test

- Three or Four testers (preferable)
  - greeter - puts users at ease & gets data
  - facilitator - only team member who speaks
    - gives instructions & encourages thoughts, opinions
  - computer - knows application logic & controls it
    - always simulates the response, w/o explanation
  - observers) - take notes & recommendations
- Typical session is approximately 1 hour
  - preparation, the test, debriefing

# Three Steps in Conducting a Test

- Greet
- Test
- Debrief

# Three Steps in Conducting a Test

## Step I

- Greet
  - Small talks to make the participants feel easy
  - get forms filled, (the forms should include the participants information
    - Personal
    - Computer background
    - Product usage background
  - assure confidentiality, etc.

# Three Steps in Conducting a Test

## Step II

- Test
  - facilitator hands **written** tasks to the user
    - must be clear & detailed
  - facilitator keeps getting “output” from participant
    - “What are you thinking right now?”,  
“Think aloud” (**Encourage Think Aloud**)
  - observe -> no “a-ha”, laugh, etc.

# Three Steps in Conducting a Test

## Step III

- Debrief
  - fill out post-evaluation questionnaire
  - ask questions about parts you saw problems on
  - gather impressions (usually subjective rating and open end text questions)
  - give thanks
  - pay

# Evaluating Results

- Sort & prioritize observations
  - what was important?
  - lots of problems in the same area?
- Create a written report on findings
  - gives agenda for meeting on design changes
- Make changes & **iterate**

# Recruiting for the User Testing

- Criteria
  - Real user
  - Relative real user
- Number
  - Depends on experiment design, power and the difference to be tested

# Recruiting

- Screening
  - Purpose
    - Homogeneous
  - Example

Name: \_\_\_\_\_

Home Phone: \_\_\_\_\_

Work Phone: \_\_\_\_\_

Age:

15-20     21-30     31-40     41-50     51 or above

Sex:  Male     Female

Right handed     Left handed

**Please answer the following questions about your computer experience:**

1. Do you use an IBM or compatible personal computer?

Yes

No

If you answered "no," please disregard the remaining parts of the questionnaire.

2. What kind(s) of programs have you worked with? Check all that apply.

Word Processing

Spreadsheets

Graphics

Other(s) specify \_\_\_\_\_

# Experiment - Recruiting

1. Name:↵
2. Email Address:↵
3. Phone Number:↵
4. Status:            Undergraduate            Graduate↵
5. Gender:            Male    Female↵
6. Age:↵
7. Major:↵
- ↵
8. How many years have you been using a computer? ↵
  - a. Less than 1 year↵
  - b. 1-2 years↵
  - c. 3-5 years↵
  - d. More than 5 years↵
- ↵
9. How often do you use computer?↵
  - a. Almost everyday↵
  - b. Several times a week↵
  - c. Several times a month↵
  - d. Almost never↵
- ↵
10. How long have you been using the Web browser?↵
  - a. Less than 1 month↵
  - b. 1-6 months↵
  - c. 6-12 months↵
  - d. More than 1 year.↵

# Experiment - Task Design

- Criteria
  - Includes most of important steps
  - Represents typical tasks real user do
  - Reflects usability requirements concerned
  - Simplified

# Pilot Test

- Purpose
  - Determine sample size
  - Find out any problems with tasks, instructions, User Interface models, etc.

# User Testing Steps

- Introduction of experiment to subjects
- Fill out consent form
- Training if necessary
- Doing real experiment and collect data
- Fill out post experiment questionnaire
- Pay

# Data Analysis

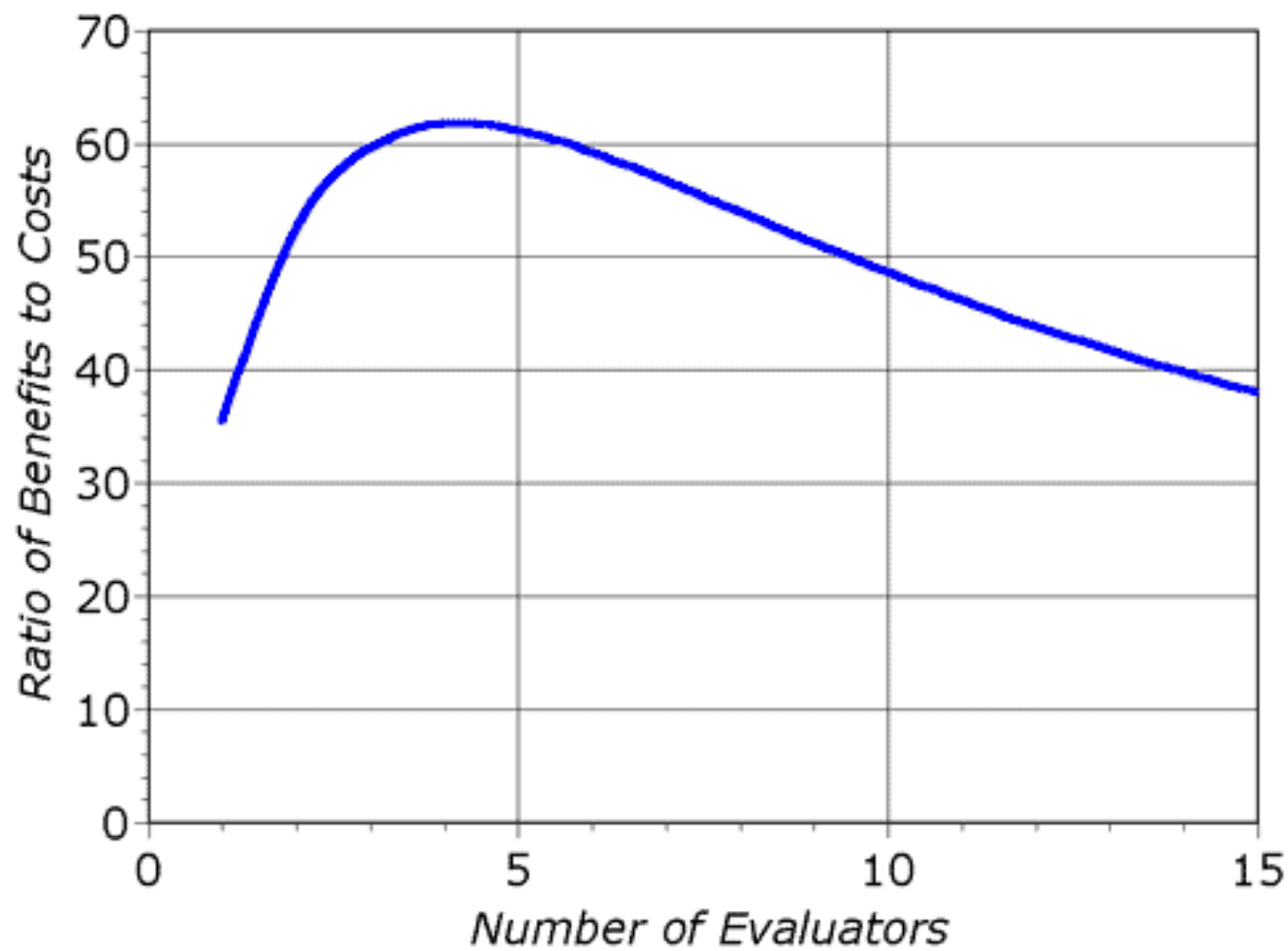
- Subject Homogeneous testing
  - What is this test?
  - Why this test?
  - How to test?
- Performance data analysis
  - Conduct correct test
  - Model checking
  - Conclusion

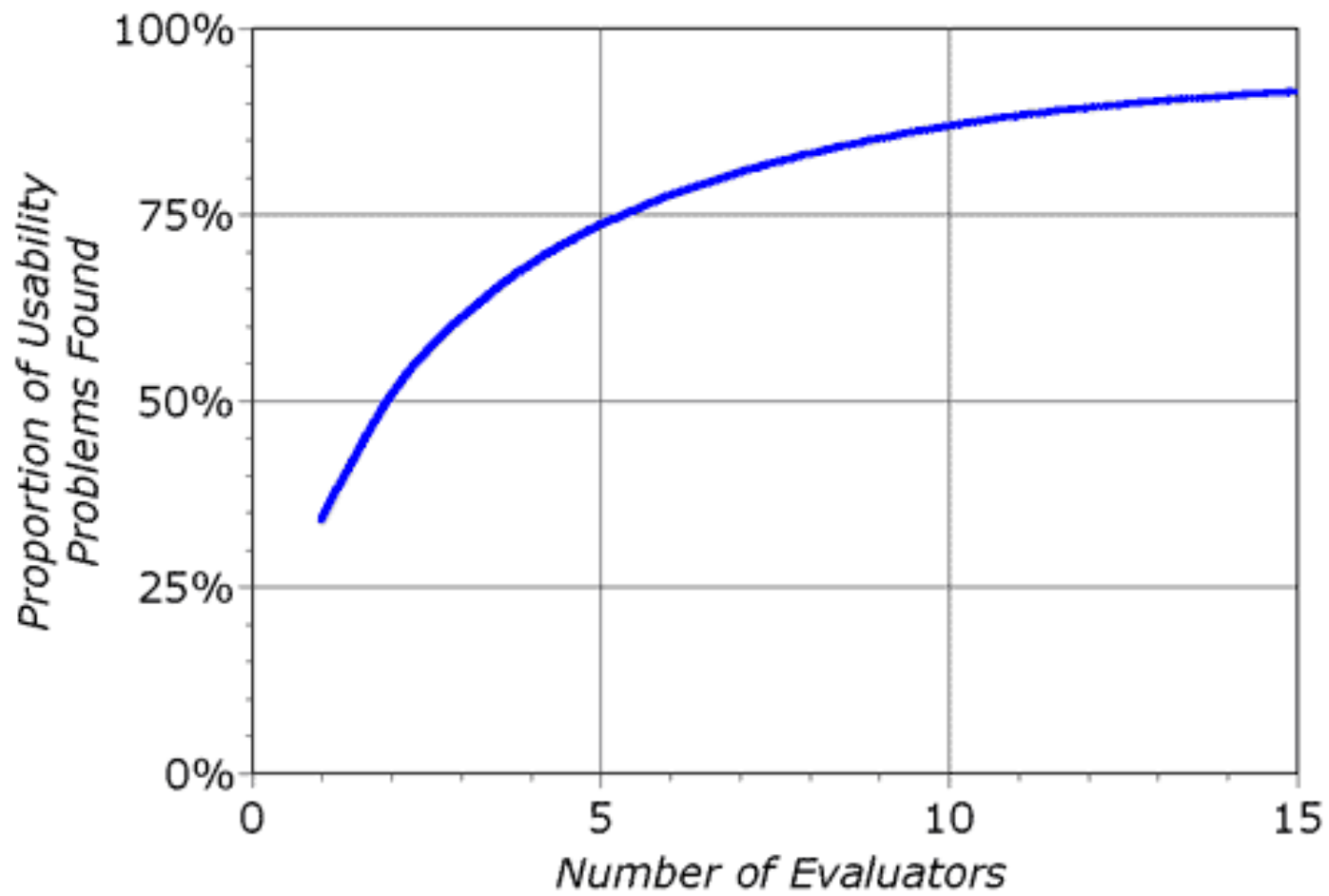
# Heuristic Evaluation

- A usability engineering method
- Part of the iterative design process
- A small set (3 – 5 )of evaluators
- Steps
  - Inspect the interface alone
  - Aggregate their findings
  - Ensure independent and unbiased evaluators

# Discount Usability Engineering

- Proposed by Dr. Jacob Nielsen (1994)
- Not perfect or “best” methods, but “good” is better than nothing.
- Early stage design
- **Quick and cheap** approach
- Find most of the problems with limited resource
- **Usually 3 – 5 users**





# Discount Usability Engineering

- **Cheap**
  - No special lab or equipment needed
  - The more careful you are, the better it gets
- **Fast**
  - Standard usability testing may take weeks
  - Discount usability testing takes days or hours
- **Easy to use and learn**
  - Can be taught in 2 – 4 hours

# Three Techniques

- Scenarios
- Simplified thinking aloud
- Heuristic evaluation

# Heuristic Evaluation – 10 Basic Rules

- **Visibility of system status**  
The system should keep users informed about what is going on through appropriate feedback within reasonable time.
- **Match between system and the real world**  
The system should speak the users' language, with words, phrases and concepts familiar to the user.
- **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.

# Heuristic Evaluation – 10 Basic Rules

- **Consistency and standards**  
Users should not have to wonder whether different words, situations, or actions mean the same thing.  
Follow platform conventions.
- **Error prevention**  
Even better than good error messages is a careful design which prevents a problem from occurring in the first place.
- **Recognition rather than recall**  
Make objects, actions, and options visible. The user should not have to remember information from one part to the next. Instructions should be visible or easily retrievable whenever appropriate.

# Heuristic Evaluation – 10 Basic Rules

- **Flexibility and efficiency of use**  
Have accelerators that can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
- **Aesthetic and minimalist design**  
Dialogues should not contain information which is irrelevant or rarely needed. Extra units of information compete with the relevant units of information and diminishes their relative visibility.

# Heuristic Evaluation – 10 Basic Rules

- **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.
- **Help and documentation**

It is preferable that documentation is not necessary but if it is, documentation may be necessary to provide help. It should be easy to search, focused on the user's task, list concrete steps.

# Limits on Response Times

- 0.1 sec.: is the limit so that the system appears to react instantaneously.
  - Important for direct manipulation, virtual world navigation.
- 1 sec.: is the limit so that the user's flow of thought stays uninterrupted.
  - Display a busy cursor if things will take longer than 1 sec.
- 10 secs.: is the limit for keeping the user's attention on the task at hand.
  - Display a progress indicator if things will take longer than 10 secs.

# How to Perform Heuristic Evaluations?

- At least two passes for each evaluator
  - first to get feel for flow and scope of system
  - second to focus on specific elements
- Assistance from implementors / domain experts
  - If system is walk-up-and-use or evaluators are domain experts, then no assistance needed
  - Otherwise might supply evaluators with scenarios and have implementors standing by

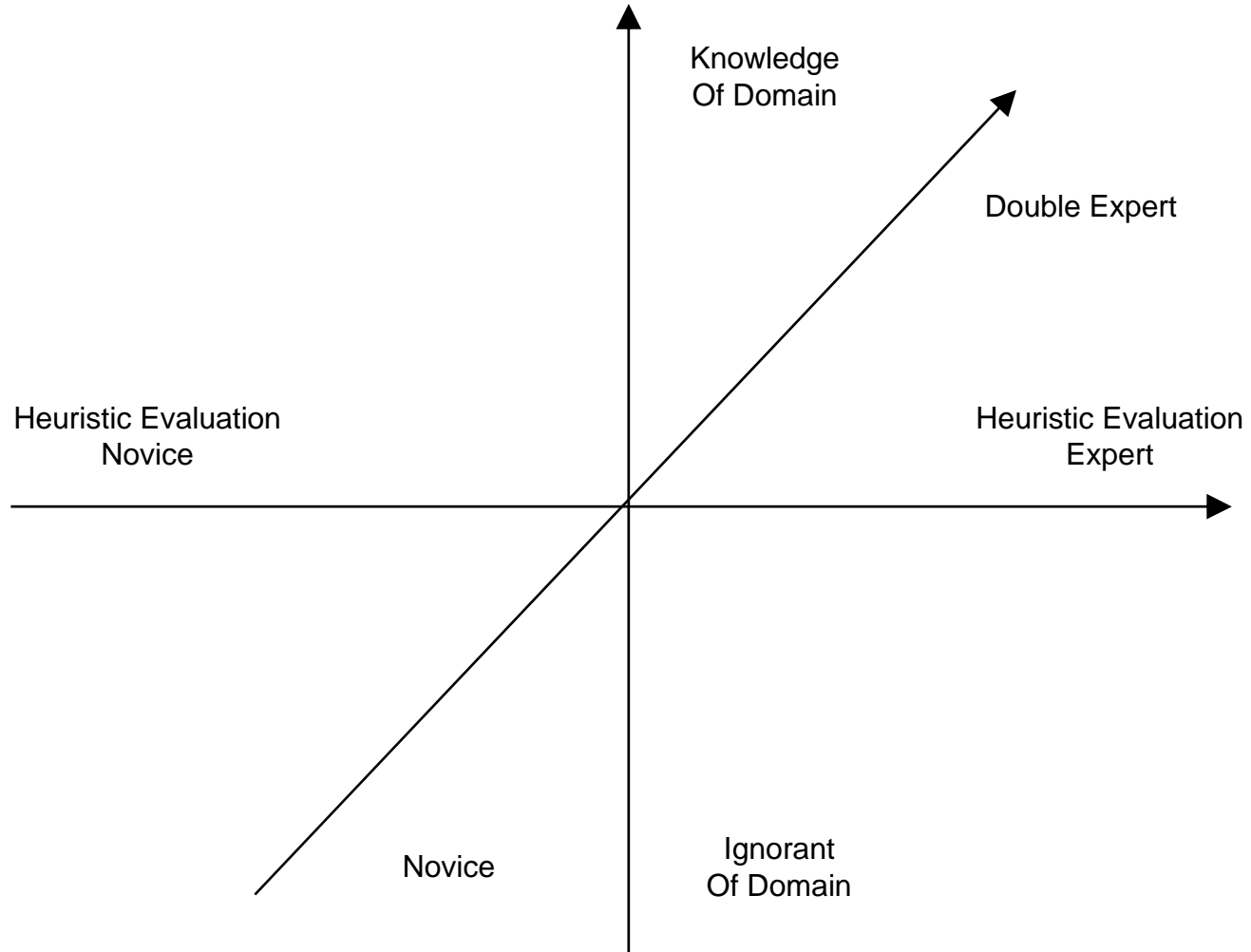
# Performing a Heuristic Evaluation (I)

- Provide evaluators with checklist of usability heuristics.
- Optionally provide evaluators with some domain-specific training.
- Each evaluator works alone (1–2 hours).
- Interface examined in two passes: first pass focuses on general flow, second on particular elements in detail.
- Notes taken either by evaluator or evaluation manager.
- Make list of potential problems and list of positive findings.

# Performing a Heuristic Evaluation (II)

- Independent findings are then aggregated into one large list (by evaluation manager). This is best done with a spreadsheet.
- The large list of potential problems is distributed to each evaluator.
- Each evaluator now assigns severity ratings individually to each problem in the large list (unseen by the other evaluators).
- The individual severity ratings are averaged to obtain the final severity rating for each problem.
- The long list is sorted in decreasing order of average severity. Now you know why a spreadsheet comes in handy.
- Group debriefing session to suggest possible redesigns.

# Type of Evaluators



# Type of Users

- Novice Evaluators: have no formal knowledge of usability.
- Regular Evaluators: have usability experience, but no domain knowledge.
- Double Experts: have usability and domain expertise.

# Pros and Cons of Heuristic Evaluations

- ++ cheap
- + intuitive
- + usable early in development process
- + finds many problems
- + finds both major and minor problems
- may miss domain-specific problems

# Guideline Checking

- The interface is judged according to its conformance with a detailed checklist of guidelines.
- Whereas heuristic evaluation employs 10 broad principles, guideline checking often involves dozens (or hundreds) of individual items on a checklist.

# Guideline Checking Example

Homepage (URL)	Date	Nr.			
Tester	Size	kB Score %			
Nr	Topic	Recommended Design	Strength	Points	You
1	Download time	50 kB (<10 sec for your customer)	***	3	
2	Window title	Start with Company Name	***	3	
3	Title tag line	What about	***	3	
4	Readable URL	Hackable URL	**	2	
5	Error page	Catch errors/dead links, to search	**	2	
6	Meta tags	For search engines	***	3	
7	Alt Information	Images, accessibility, Lynx	**	2	
8	Page width	770 pixel (620-1024)	**	2	
9	• Liquid vs. frozen layout	Liquid	**	2	
10	• Page length	<2 pages (1000-1600 px)	**	2	
11	Frames	No	***	3	
12	Logo placement	Upper left	***	3	
13	• Logo size	80x88 Pixel	**	2	
14	Search	Yes, in a box, always	***	3	
15	• Search placement	Upper part, right or left corner	***	3	
16	• Search box colour	White	***	3	
17	• Search button	Call it "Search" or "Go"	**	2	
18	• Width of search box	>=26 characters (30 best)	**	2	
19	• Type of search	Simple search (Link to advanced)	**	2	
20	Navigation	4 types: left, tabs, top, categories	**	2	
21	• Footer navigation links	Max. 7 links, single line	*	1	
22	• Sitemap link	Name "Site Map"	**	2	
23	Routing page	No	**	2	
24	Splash page	No	***	3	
25	Sign-In	"Account" or "Sign In"	*	1	
26	About the company	Always include it	***	3	
27	• About link	Call it "About <company>"	**	2	
28	• Contact information	Call it "Contact us"	**	2	
29	• Privacy policy	If you collect data	***	3	
30	• Name of privacy link	Call it "Privacy Policy"	**	2	
31	Job opening	Call it "Jobs" if you have it	**	2	
32	Help	If it is a complex site	*	1	
33	• Help placement	Upper right	**	2	
34	Auto-playing music	No	***	3	
35	Animation	No	**	2	
36	Graphics/illustration	5-15%	*	1	
37	Advertising	<= 3 ads	**	2	
38	Body text colour	Black	**	2	
39	• Body text size	12 points	*	1	
40	• Body text size frozen	No	***	3	
41	• Body text typeface	Sans-serif	*	1	
42	• Background colour	White	**	2	
43	• Link colour (unvisited)	Blue	**	2	
44	• Link colour (visited)	Purple	*	1	
45	• Link colour different	Yes (not light grey)	***	3	
46	• Link underlining	Yes (except in navigation bar)	**	2	
<b>Score of URL:</b>				<b>100</b>	

\* Default Recommendation  
 \*\* Strong Recommendation  
 \*\*\* Essential Recommendation

# Heuristic Evaluation – Severity Ratings

- Severity ratings can help prioritize the fixing of usability problems.
- After evaluation sessions, a complete aggregate list of usability problems is given/sent to each.
- Working independently, evaluators assign severity rating [on scale of 0–4] to each problem.
- Severity rating of single evaluator is unreliable, mean of 3–5 evaluators is satisfactory.

# Heuristic Evaluation – Severity Ratings

## Ratings

**0** = 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

# Severity Ratings and Fix Priority

<b>Score</b>	<b>Severity</b>	<b>Fix Priority</b>
<b>4</b>	Catastrophic problem	imperative
<b>3</b>	Major problem	high
<b>2</b>	Minor problem	low
<b>1</b>	Cosmetic problem only	low
<b>0</b>	Not a problem at all	

# Cognitive Walkthrough

- Identify user population.
- Define suite of representative tasks.
- Describe or implement interface or prototype.
- Specify correct action sequences for each task.

# Cognitive Walkthrough

- For each action in solution path, construct credible “success” or “failure” story about why user would or would not select correct action.
  - Will the user be trying to achieve the right effect?
  - Will the user know that the correct action is available?
  - Will the user know that the correct action will achieve the desired effect?
  - If the correct action is taken, will the user see that things are going ok?

# Cognitive Walkthrough Example

- Forwarding calls on a campus telephone system, from the perspective of a first time user.
- a) Users: New faculty, staff, guests, and visitors. For this evaluation assume that the user is a new university professor.
- b) Task: Cancel current forwarding and forward calls instead to a colleague with the extension 1234.
- c) Interface: Standard-size, touch-tone phone on desk. Overlay template includes the following information:
  - FWD \*2
  - CNCL #2
  - SEND ALL \*3

# Cognitive Walkthrough Example

- d) Correct Action Sequence: The seven correct actions for accomplishing this task are:
  - 1. Pick up the receiver.
    - Phone: dial tone
  - 2. Press #2. {command to cancel forwarding}  
Phone: bip bip bip
  - 3. Hang up the receiver.
  - 4. Pick up the receiver.
    - Phone: dial tone
  - 5. Press \*2.
    - Phone: dial tone
  - 6. Press 1234.
    - Phone: bip bip bip
  - 7. Hang up the receiver.

# Example Walkthrough Steps

- 1. Pick up the receiver.
  - Phone: dial tone
  - Success story:
  - Seems ok based on prior experience with phones.
- 2. Press #2.
  - Phone: bip bip bip
  - Failure story: (see Next Page)

## 2. Press #2. Phone: bip bip bip Failure story:

- Will the user be trying to achieve the right effect?
  - How does the user even know that forwarding is in effect?
- Will the user know that the correct action is available?
  - Probably yes, if forwarding is active, one must be able to cancel it. CNCL is visible on the template.
- Will the user know that the correct action will achieve the desired effect?
  - Might not recognize CNCL as the control to cancel forwarding. Might think that just pressing '2' is sufficient, instead of '#2'. Might try to press the buttons simultaneously, rather than sequentially.
- If the correct action is taken, will the user see that things are going ok?
  - How do first-time users know they have succeeded? After some experience, they will recognize the bips as confirmation, but will they at first?

### 3. Hang up the receiver Failure story:

- Will the user be trying to achieve the right effect?
  - Big trouble. How do you know you have to hang up before reestablishing forwarding?

# Pros and Cons of Cognitive Walkthrough

- ++ finds task-oriented problems
- + helps define users' goals and assumptions
- + usable early in development process
- time-consuming
- some training required
- needs task definition methodology
- applies only to ease of learning problems

# Action Analysis

- Quantitative analysis of actions to predict time skilled user requires to complete tasks, based on time estimates for typical interface actions. Focuses on performance of skilled user (efficiency).
- Two flavors (levels of detail):
  - a) Formal or “Keystroke-Level”
  - b) Informal or “Back-of-the-Envelope”

# Keystroke-Level Analysis

- Developed from GOMS (Goals, Operators, Methods, Selection) modeling.
- Extremely detailed, may often predict task duration to within 20%, but very tedious to carry out.
- Used to estimate performance of high-use systems (e.g. telephone operator workstations).

# Keystroke-Level Analysis

- Break down tasks hierarchically into subtasks until reach fraction of second level of detail.
- Use average values for action times (determined through extensive published research) to predict expected performance for particular task.
- The analysts do not measure action times themselves, but refer to published tables.

# Average Times for typical keystroke-level actions, in seconds.

	<i>Action</i>	<i>Time</i>
<i>Physical Movements</i>	One keystroke	0.28
	Point with mouse	1.5
	Move hand to mouse or function key	0.3
<i>Visual Perception</i>	Respond to brief light	0.1
	Recognise 6-letter word	0.34
	Move eyes to new location on screen	0.23
<i>Mental Actions</i>	Retrieve one item from long-term memory	1.2
	Learn one step of a procedure	25
	Execute a mental step	0.075
	Choose among methods	1.2

# Back-of-the-Envelope Action Analysis

- List actions required to complete a task (as before), but in much less detail – at level user:
  - “Select Save from the File menu”
  - “Edit the file name”
  - “Confirm by pressing OK”
- At this level of analysis, every action takes at least 2 to 3 seconds (videotape a few tasks if you do not believe it takes this long!).
- Allows quick estimation of expected performance of interface for particular task.

# Formal Usability Studies

- To determine time requirements for task completion
- To compare two designs on measurable aspects
  - time required
  - number of errors
  - effectiveness for achieving very specific tasks
- Require Experiment Design

# Performance Measures

- Time to complete specific task(s).
- Number of tasks completed within given time.
- Number of errors.
- Ratio successful interactions : errors.
- Time spent recovering from errors.
- Number of commands/features used.
- Number of features user can remember after test.
- How often help system used.
- Time spent using help.
- Ratio positive : negative user comments.
- Number of times user sidetracked from real task.

# Experiment Design

- Experiment design involves determining how many experiments to run and which attributes to vary in each experiment
- Goal: isolate which aspects of the interface really make a difference
- Refer to an experiment design book for more details

# Between Groups Experiment

- Two equally-sized groups of test users.
- Randomly assign users to two groups.
- Identical tasks for both groups.
- Group 1 uses only system A, group 2 only system B.

+ no problems with learning effect

- large individual variation in user skills (std. dev.  
‡ 50%)

# Within-Group Experiment

- One group of test users.
  - Randomly assign users to two equally-sized pools.
  - Users perform equivalent tasks on both systems.
  - Pool 1 uses system A first, pool 2 system B first.
- + automatically controls for individual variability  
- transfer of skill between systems (learning effect)

# Example Designs

<b>Between-Groups</b>	
<i>System A</i>	<i>System B</i>
John	Dave
James	Mariel
Mary	Ann
Stuart	Phil
Keith	Tony
Gary	Gordon
Jeff	Ted
Bill	Edward
...	...
Charles	Thomas
Celine	Doug

<b>Within-Groups</b>	
<i>Participant</i>	<i>Sequence</i>
Elisabeth	A, B
Sven	A, B
Amanda	A, B
Claudia	A, B
Terry	A, B
Nigel	A, B
Barry	A, B
...	...
Ben	B, A
Michael	B, A
Richard	B, A

# Statistical Analysis

- Is there a statistically significant difference between system A and B? [hypothesis testing]
- How large is the difference? [point estimation, averages]
- How accurate are the results? [standard deviation, confidence intervals]
- Sample Size
  - Rule of thumb: 16 – 20 test users.

# Summary

- Formal studies can reveal detailed information but take extensive time/effort
- Human participants entail special requirements
- Experiment design involves
  - Factors, levels, participants, tasks, hypotheses
  - Important to consider which factors are likely to have real effects on the results, and isolate these
- Analysis
  - Often need to involve a statistician to do it right
  - Need to determine statistical significance
  - Important to make plots and explore the data

# Comparing Evaluation Methods

Jeffries et al., CHI '91

**Table 6:** Summary of the study's findings.

	Advantages	Disadvantages
Heuristic evaluation	<ul style="list-style-type: none"> <li>Identifies many more problems</li> <li>Identifies more serious problems</li> <li>Low cost</li> </ul>	<ul style="list-style-type: none"> <li>Requires UI expertise</li> <li>Requires several evaluators</li> </ul>
Usability testing	<ul style="list-style-type: none"> <li>Identifies serious and recurring problems</li> <li>Avoids low-priority problems</li> </ul>	<ul style="list-style-type: none"> <li>Requires UI expertise</li> <li>High cost</li> <li>Misses consistency problems</li> </ul>
Guidelines	<ul style="list-style-type: none"> <li>Identifies recurring and general problems</li> <li>Can be used by software developers</li> </ul>	<ul style="list-style-type: none"> <li>Misses some severe problems</li> </ul>
Cognitive Walk-through	<ul style="list-style-type: none"> <li>Helps define users' goals and assumptions</li> <li>Can be used by software developers</li> </ul>	<ul style="list-style-type: none"> <li>Needs task definition methodology</li> <li>Tedious</li> <li>Misses general and recurring problems</li> </ul>

# Comparing Evaluation Methods

Jeffries et al., CHI '91

- Heuristic Evaluation is best from a cost/benefit analysis, but requires access to several experienced designers
- Usability testing second best – found recurring, general, and critical errors but is expensive to conduct
- Guideline-based evaluators missed a lot but did not realize this
  - They were software engineers, not usability specialists
- Cognitive walkthrough process was tedious