

Understanding and serving users

Usability evaluation methods

Heuristics

Cognitive walkthroughs

Why definition matters?

- What you define usability to be, determines what you design into being
- Understanding how usability is determined improves the quality of its evaluation and the interpretation of test results
 - Definition drives evaluation

The need for multiple measures

<i>Usability Measure</i>	User 1	User 2
Effective	100 %	100 %
Time	45	45
Steps	7	3

Framework for Usability Evaluation

- Approach and Type
 - Approach refers to source of data
 - User, Expert, or Model
 - Type refers to purpose of evaluation
 - Diagnostic (Formative) or Metrication (Summative)
- Any evaluation method is a combination of *approach* and *type*

Evaluation Approach

The approach defines the *source* of the data i.e., where does the evaluator gain the data about usability?

- from real users? (**User-based**)
- from usability experts or self evaluation? (**Expert-based**)
- from the application of a formal theory or model? (**Model-based**)

Type of Usability Evaluation

- Diagnostic/Formative
 - Evaluation is aimed at uncovering shortcomings in design
 - May use all approaches described already
 - This is most common
 - Are all usability evaluations diagnostic (formative)?

Type of Usability Evaluation (2)

- Metric/certification/summative
 - Seeks to certify or measure product against pre-defined criteria(standards, certification)
 - Usually relies on user-based tests
 - More common for non-usability testing
 - Even this type of evaluation can be used for diagnostic purposes in on-going design iterations

Expert-based techniques

- We will study two classic inspection methods:
 - Heuristic evaluation
 - Cognitive Walkthrough

The Heuristic Evaluation Method

- A cheap form of expert usability test
- Based on examining interface compliance with design guidelines ('heuristics')
- According to Nielsen (1994) "it is easy, it is fast and it is as cheap as you want it" (p.25)
 - In Nielsen and Mack (eds) *Usability Inspection Methods* (Wiley)

Procedure

http://www.useit.com/papers/heuristic/heuristic_evaluation.html

- Usually requires evaluator to work alone
- Duration = 1-2 hours
- 2 passes through the interface recommended
 - 1st pass: get a sense of the flow through
 - 2nd pass: identify specific interface problems
- Evaluators note problems
- All problems are pooled for analysis

Revised (1994) heuristics

- Visible system status
- Match between system and real-world
- User control and freedom
- Consistency and standards
- Error prevention
- Recognition over recall
- Flexibility and efficiency
- Aesthetic and minimalist design
- Help users recognize and recover from error
- Help and documentation

Top designer maxims (Lund 2000)

- Know thy user, and you are not thy user
- Things that look the same should act the same
- The information for the decision needs to be there when the decision is needed
- Error msgs should actually mean something to the user and tell user how to fix a problem
- Every action should have a reaction
- Everyone makes mistakes so every mistake must be fixable
- Don't overload the user's buffers
- Consistency, consistency, consistency
- Minimize the need for mighty memory

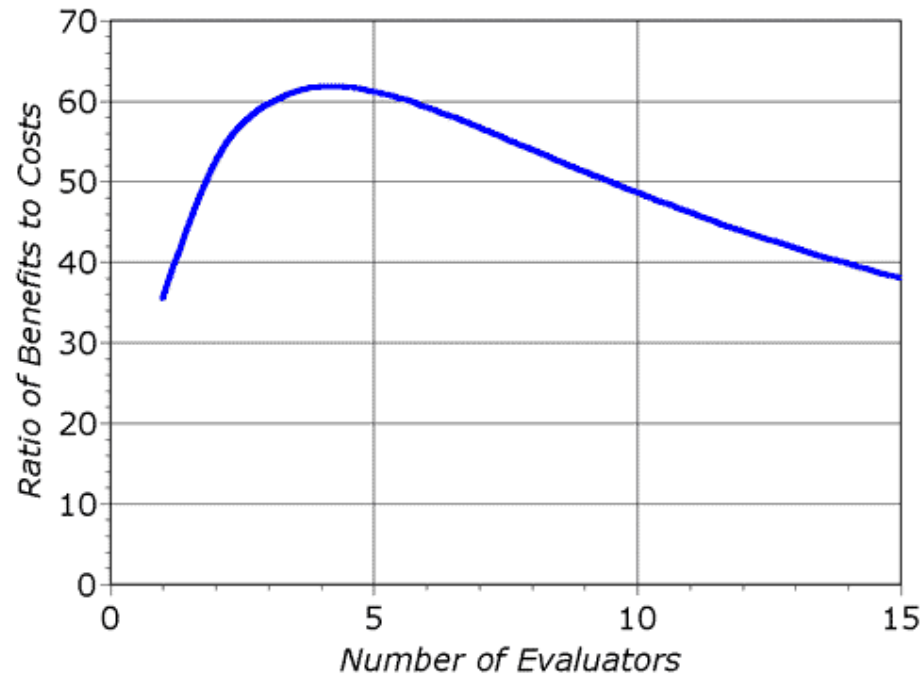
Output

- List of usability problems
- Each problem should be related to heuristics
- Each problem listed separately
- Output does not include re-design advice
- Does not guarantee finding all problems

How many evaluators are used?

- Single evaluators tend to find only 35% of known problems
 - range: 19-51%
- Different evaluators find different problems!
 - cannot predict this in advance
 - suggests very subjective criteria for evaluation
 - weakness of method?

Nielsen's cost-benefit estimate



Nielsen (1992) Reading

- Banking system example
- Evaluators given sample dialog
- Context assumes user knows how to input data, commands and corrections
- Limitations of underlying touch-tone technology ignored by evaluators

- (1) S: Enter 1 for account information, 3 for transfers between your accounts,....
- (2) U: 3# (user interrupts system)
- (3) S: Enter your account to transfer from:
- (4) U: 1234567890# (savings acc. number)
- (5) S: Enter account to transfer to
- (6) U: # (an abbreviation for checking)
- (7) S: Enter amount in cents
- (8) U: 100000#
- (9) S: From account number twelve, thirty-four, fifty-six, seventy-eight, ninety, to account number primary account, a transfer of \$1000 is made.
Press one to confirm, zero to cancel
- (10) U: 1#
- (11) S: You do not have access to this function

Study method

- 3 groups of evaluators
 - 31 students,
 - no interface evaluation exp.
 - first programming course completed
 - 19 'regular' evaluators
 - with several years experience of evaluation.
 - no expertise on voice response systems
 - 14 'double specialists'
 - with evaluation experience AND technology expertise

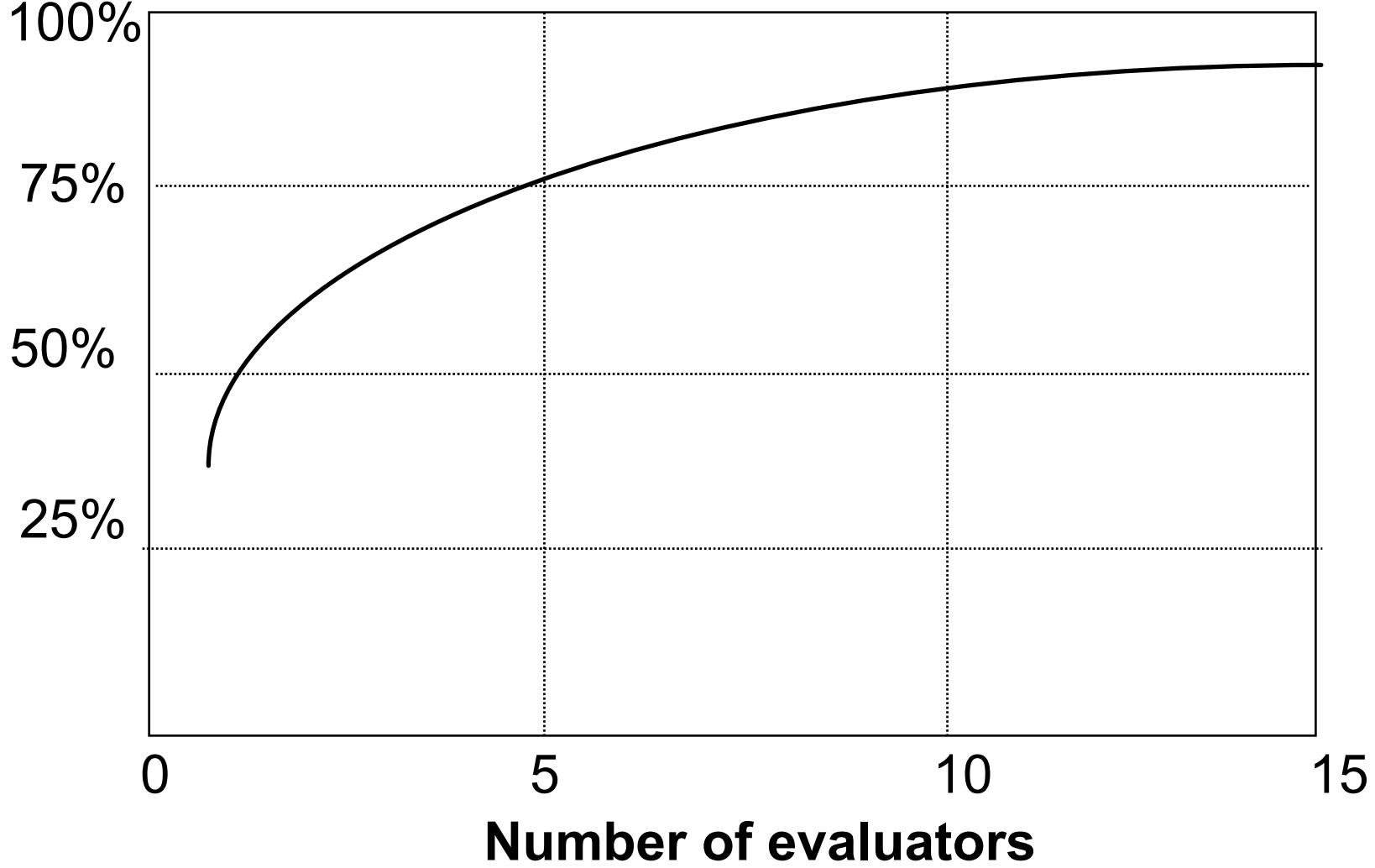
Best to Worst Evaluator Ratio

Total / Major/ Minor

- Novice evaluators 22% / 29% / 15%
- Usability experts 41% / 46% / 36%
- Double experts 60% / 61% / 59%

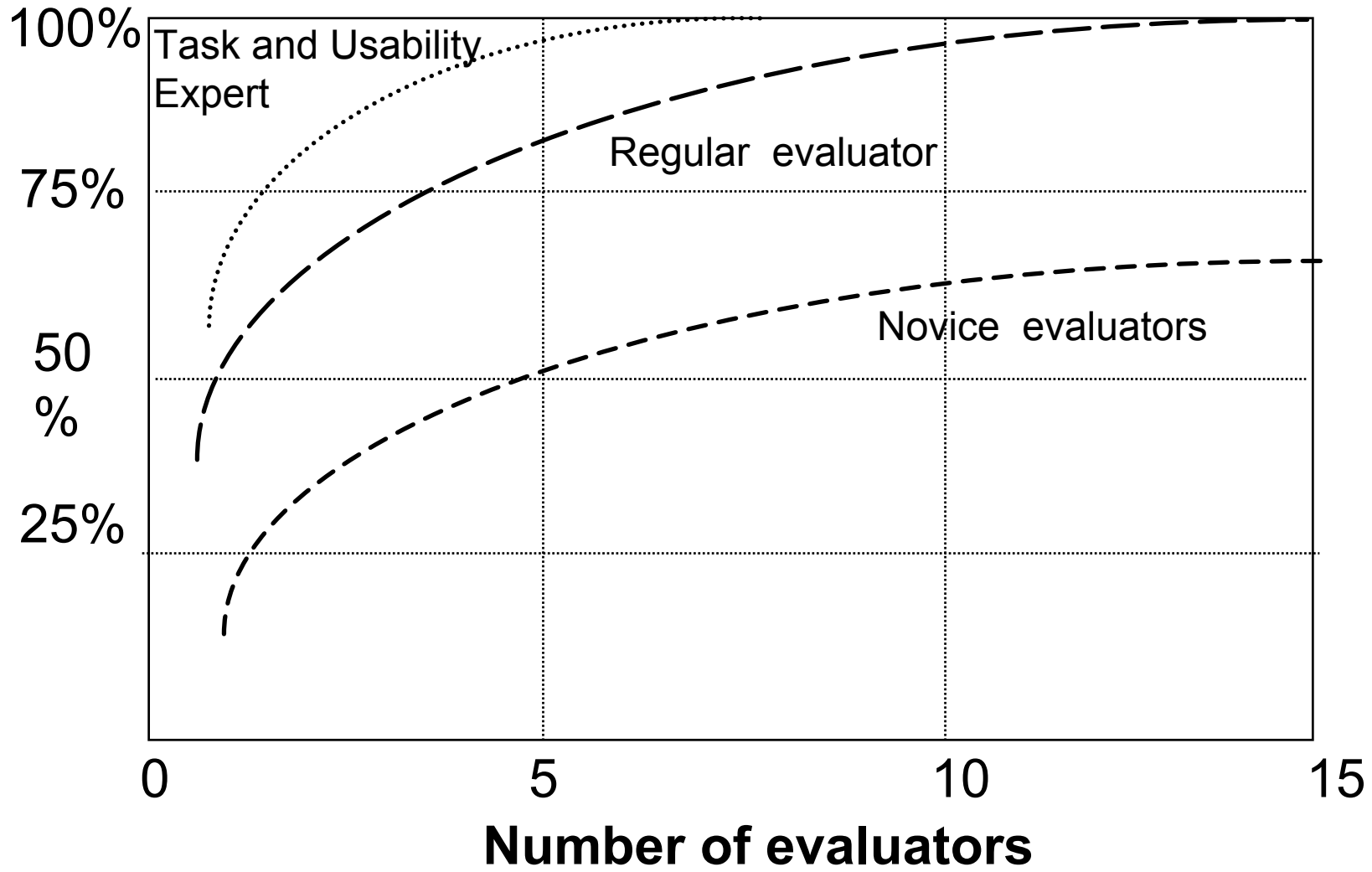
Problems Found

Nielsen (1993) Usability Engineering



Problems Found

Nielsen (1993)



In general -

- Most evaluators miss some problems
- Most evaluators 'overhit' problems
- Some 'heuristics' are favored
- Heuristics are not equally understood

- You probably have your own stories about this....

Nielsen and Landauer (1993)

- Prediction formula for problems found:

$$\text{Different Problems found}(i) = N(1-(1-x)^i)$$

i = number of evaluators

N = total number of problems

x = proportion of problems found by a single evaluator

2 evaluators, 50 problems exist =

Problems found by 2 evaluators =

$$50(1-(1-.35)^2)=28.88$$

(assuming 35% rate for each evaluator)

Extending this equation:

- From this formula we can now estimate how many problems MAY exist, given our observations

- If we observe 10 different problems with 5 evaluators, then the total number of likely problems (N) can be estimated from:

$$\begin{aligned}10 &= N(1-(1-0.35)^5) \\ &= N(1-0.116) = N(0.884) \\ N &= 10/0.884 = 11.31\end{aligned}$$

Using this to determine # problems

- If you assume a typical hit rate of 31%, then solving for N gives estimate of total problems that exist:
 - $(i) = N(1-(1-x)^i)$ thus
 - $N = (i)/(1-(1-x)^i)$
- So if 3 evaluators find a total of 12 problems, we can estimate that there are:
 $12/(1-(1-.31)^3) = 12/0.672 = 17.86...$ or roughly 18 problems existing in our resource or application being evaluated

Or very roughly -- just divide unique problems found by $1-(0.65)^i$

Where i = # of evaluators, and 0.65 reflects estimate of 35% hit rate i.e., $(1-0.35)$

Implications:

- If evaluator hit rate is low, we need more of them -
 - IF we observe 10 different problems with evaluators whose hit rate is only 10% then 5 evaluators will produce an estimate of 24 total problems, and 6 evaluators will produce an estimate of 21.
- Corollary: If evaluator hit rate is high, we need very few of them to estimate N.

Nielsen and Phillips (1993)

(see readings)

- Compared two dbase query interfaces
 - dialog box and pop-up menu
- Heuristic evaluations were performed under various conditions
 - Cold: written description of interfaces
 - Warm: prototype of dialog box, description of pop-up, and had performed cold evaluation
 - Hot: Running versions of both interfaces

Comparison of evaluations

Type	N	Time spent
Cold*	12	9.9 mins
Warm*	10	60 mins
Hot*	15	Demo then 14.2 mins
GOMS	19	108 mins
Users	20	

*Typical evaluator: 9 yrs “usability experience”

Results: estimates of time for user

	Box 1 query	Box 2 queries	Menu 1 query	Menu 2 queries	SD % of mean
Cold	20.3 (26.1)	29.4 (30,4)	8.0 (7.4)	14.0 (15.1)	108%
Warm	12.9 (10.7)	21.4 (21.6)	4.7 (2.7)	9.1 (5.3)	75%
Hot	13.8 (6.1)	20.0 (9.5)	6.1 (3.5)	9.4 (5.5)	52%
GOMS	16.6 (3.7)	22.6 (5.4)	5.8 (0.8)	11.2 (1.9)	19%
Users	15.4 (3.0)	25.5 (4.7)	4.3 (0.6)	6.5 (0.9)	17%

Other points

- Estimation ability not related to usability experience!
- Times for 2 query menu tasks are overestimated since evaluators assumed no carryover in cognition
- User testing provides best estimates
- High variability suggests need for more than 1 evaluator.

OUTPUT

- List assumptions you make about users
- Identify every problem and state why it is a usability issue
- Identify every violation of a heuristic
- Relate violations to specific problems
- Rank the problems in importance

Checklist:

- How many problems were found?
 - Does each one have at least one associated heuristic violation?
- How many violations of each heuristic were found?
 - There will probably be more violations observed than problems found
- What problems did you find independent of the heuristics?