

Lecture 14: Controlled Experiments

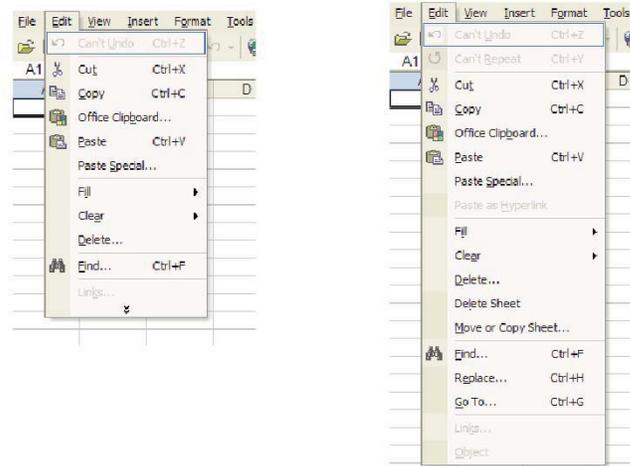
Content in this lecture indicated as "All Rights Reserved" is excluded from our Creative Commons license. For more information, see <http://ocw.mit.edu/fairuse>.

Spring 2011

6.813/6.831 User Interface Design and Implementation

1

UI Hall of Fame or Shame?



© Microsoft. All rights reserved.

Spring 2011

6.813/6.831 User Interface Design and Implementation

2

Today's candidate for the Hall of Fame or Shame is **adaptive menus**, a feature of Microsoft Office 2003. Initially, a menu shows only the most commonly used commands. Clicking on the arrow at the bottom of the menu expands it to show all available commands. Adaptive menus track how often a user invokes each command, in order to display frequently-used commands and recently-used commands.

Let's discuss the usability of this idea.

This interface is striving for a compromise between **simplicity** (i.e., providing as few features as possible) and **task analysis** (supporting the tasks required by many users, and trying to adapt to the common tasks of each individual user). Both properties are important for usability. Unfortunately they also compete with each other. Adaptive menus are an interesting hybrid technique that's trying to satisfy both.

The downside is lack of predictability. Because the menu may change in complex and unpredictable ways – with new items appearing and underused items disappearing for no visible reason – the user can no longer rely on a lot of useful cues to find menu items. One of these cues that's lost is spatial memory – Paste may be found at different distances down the menu each time the menu appears. Another missing cue is context: Paste's neighbors may change as well.

Another downside is lack of user control. The adaptation happens automatically; the user can't manually fixate or remove items from a menu.

This particular implementation of adaptive menus has some specific usability problems. When the full menu appears, the new items are inserted into place, and there's very little **contrast** in the graphic design to distinguish between the old items and the new items. So the user has to search through the entire menu again.

But this particular implementation addresses other usability problems very well. When the user is hunting through all the menus, looking for a command, the full menu only has to be requested once; then all subsequent menus are fully displayed.

UI Hall of Fame or Shame?



© Microsoft. All rights reserved.

Spring 2011

6.813/6.831 User Interface Design and Implementation

3

Here's an alternative approach to providing easy access to high-frequency menu choices: a **split menu**. You can see it in Microsoft Office's font drop-down menu. A small number of fonts that you've used recently appear in the upper part of the menu, while the entire list of fonts is available in the lower part of the menu. The upper list is sorted by recency, while the lower part is sorted alphabetically.

Let's discuss the split menu approach too.

These menu approaches are particularly relevant to today's lecture because they've been tested by a couple of excellent controlled experiments. The split menu idea was evaluated by Sears & Shneiderman, "Split menus: effectively using selection frequency to organize menus", *ACM TOCHI*, March 1994. And the adaptive menu was tested by Findlater & McGrenere, "A comparison of static, adaptive, and adaptable menus", *CHI 2004*.

Sears & Shneiderman: compared alphabetic, frequency-ordered, and split menus (with up to 4 high-frequency items at top of menu, **ordered in same way** as rest of menu, **removed** from the rest of menu – so not the same as Office's font split menu). 15-item menus (randomly selected from a dictionary of 1000 common words). Three different frequency distributions across the alphabetic menu (end of menu, middle of menu, top of menu). 36 subjects, within-subjects, each saw 3x3 menus (random unique items) counterbalanced. Pulldown menus, timed from mouse press on menu bar until selection of item. 100 trials per menu, chosen from frequency distribution of menu. Measured time and ranked preference of menu type (1-3). Subjective ranking had split (1.4) > alphabetic (2.0) > frequency (2.6). Selection time for frequent items at end of menu had split (1.4s) > freq (1.5s) > alphabetic (1.7s); for frequent items at start of menu had split, alphabetic (1.4s) > freq (1.5s). Also proposed and fitted a cognitive model that high-frequency menu items take time logarithmic in item position (a la Fitts's Law) while low-frequency items take linear time in position (visual scan).

Findlater & McGrenere: compared split menus that were static (unchanging), adaptive (changed by system), and adaptable (changed by user). Adaptive menu has two most recent and two most frequent items in the split part. Adaptable had arrow buttons next to each item to promote or demote it. Used frequency distributions from three most frequently-used Word menus (File, Format, Insert), collected from one user over 20 weeks, but changed all item names to mask them. Measured time and ranked preference (on several dimensions: overall, efficiency, ease, error, frustration, etc.). 27 subjects, within-subjects 3x3. Selection time had static (1.5), adaptable (1.6) >> adaptive (1.65). Overall preference had adaptable >> adaptive >> static.

Today's Topics

- HCI research methods
- Experiment design
- Threats to validity
- Randomization, blocking, counterbalancing

This is the first lecture in the thread about research methods in human-computer interaction. We'll talk at a high level about the three major kinds of empirical research in HCI, but our primary interest will be in testing the usability of a user interface by rigorous, controlled, laboratory experiment. This lecture covers some of the issues involved in designing such a controlled experiment. The issues are general to all scientific experiments, but we'll look specifically at how they apply to user interface testing.

In our next lecture, we'll talk about how to analyze the results of an experiment.

Much of the material for this lecture comes from Joseph McGrath, "Methodology matters: Doing research in the behavioural and social sciences", published in R. Baecker, J. Grudin, W. Buxton and S. Greenberg (eds), *Readings in Human Computer Interaction: Towards the Year 2000* (Morgan-Kaufmann, 1996)

Research Methods in HCI

- Lab experiment
- Field study
- Survey

Spring 2011

6.813/6.831 User Interface Design and Implementation

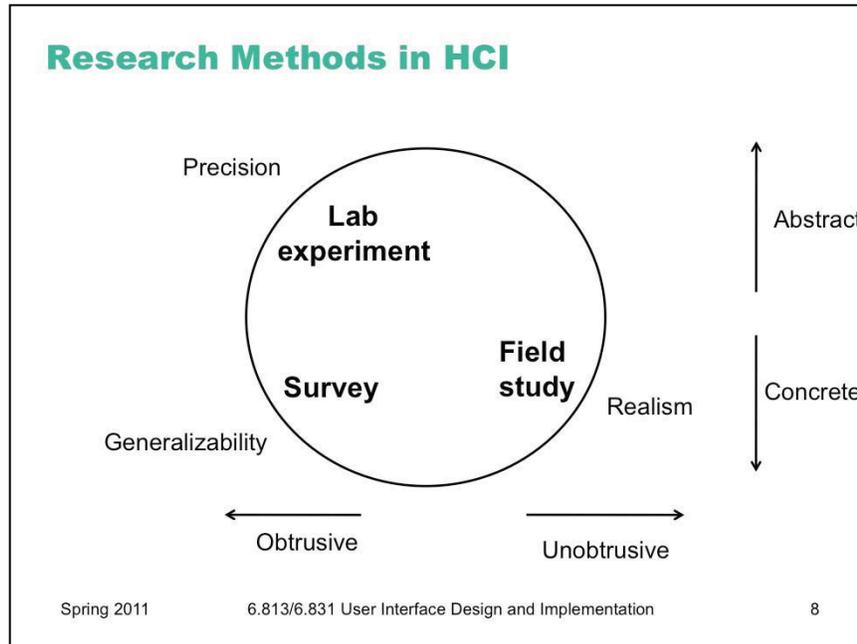
7

We'll start by talking about the main kinds of research methods in human-computer interaction. Here we mean **empirical methods**, techniques for investigating the world and collecting evidence to prove or disprove our hypotheses about how people interact with computers, and about the usability of interfaces. These methods are widely used across other kinds of research involving people, including psychology, cognitive science, sociology, and economics. When we talk about an HCI research paper and look for the **evaluation** of the paper's claim, we will often find one of these methods. Note that these are not the *only* acceptable kinds of evaluation in human-computer interaction. In particular, this picture (and this lecture) largely ignore evaluation methods that are relevant to the **computer** side of the human-computer interface: issues like performance, reliability, security; proof that a system or toolkit is broadly applicable; etc.

A **lab experiment** is an artificial situation, created by and highly controlled by the experimenter, that typically compares alternative user interfaces or measures how usability varies with some design parameter. One example might be a test of font readability, done by bringing subjects into the experimenter's lab, asking them to read texts displayed with different fonts, and timing their reading speed.

A **field study** is a real situation, happening in the actual environment where people use the interface under consideration, and using real tasks (rather than tasks concocted by the experimenter). Social scientists make distinctions between pure field studies (where the researcher acts like an anthropologist or ethnographer, intervenes as little as possible, hiding in the bushes, so to speak) and field *experiments* (where the researcher introduces something new, like a new system, new process, or new UI). In HCI, which is still a young field and still striving to make something new, these distinctions end up being a matter of *when* the study happens to be done in the evolution of your project. Initial field studies just observe without intervening (contextual inquiry, which we discussed in the task analysis lecture, is a technique like this). Final field studies deliver the new UI and see how it's used.

A **survey** is a questionnaire, conducted by paper, phone, web, or in person.



The picture shows how these methods compare on several interesting dimensions. In the vertical dimension, lab experiments often use highly **abstract** tasks, which are simplified and highly controlled in order to make strong statistical claims, but also farther removed from the real-world. An example is the classic Fitts's Law experiment, which measures the time it takes to hit targets on the screen by using plain rectangles rather than realistic pointing targets like buttons, menus, and hyperlinks. Field studies and surveys, by contrast, can use more concrete, real-world examples.

In the horizontal direction, lab experiments and surveys are **obtrusive**: people need to be interrupted and forced to give their attention to the study. Because they're aware they're being studied, their behavior can change; we'll mention some of the ways that can happen later in this lecture. Field studies, on the other hand, can be far less obtrusive, since the subjects are doing their own tasks in their own environment, and so the likelihood of obtaining natural behavior is much higher.

Finally, the picture also shows how the three methods compare on three criteria which are all desirable but virtually impossible to obtain at once from a single method. **Realism** means whether the phenomena captured are actually what happens in the real world (in real contexts on real tasks); field studies are strongest on that. **Generalizability** concerns whether the results apply to the whole population of *people* relevant to the study; a survey is strongest on that, because it's far cheaper to survey a large number of people, and good statistical sampling techniques exist to make the results generalizable. **Precision** means control over measurement error and extraneous factors that might introduce noise into the results; lab experiments are strongest on that, because they can tightly control the tasks and environment to eliminate as much error as possible.

(This picture actually omits one important class of methods, which is occasionally but not widely used in HCI research: a theoretical model or computer simulation of human behavior. We talked about some theoretical models for HCI, like KLM and GOMS, in the efficiency lecture.)

Quantifying Usability

- Usability: how well users can use the system's functionality
- Dimensions of usability
 - Learnability: is it easy to learn?
 - Efficiency: once learned, is it fast to use?
 - Errors: are errors few and recoverable?
 - Satisfaction: is it enjoyable to use?

In this lecture, we'll be focusing on controlled experiments – particularly those that try to measure the **usability** of alternative user interface designs. Recall our definition of usability, and how we broke it down into several dimensions. We can **quantify** all these measures of usability, which will be essential to doing an experiment. Just as we can say algorithm X is faster than algorithm Y on some workload, we can say that interface X is more learnable, or more efficient, or less error-prone than interface Y for some set of tasks and some class of users, by designing an experiment that measures the two interfaces.

Controlled Experiment

- Start with a testable **hypothesis**
 - e.g. Mac menu bar is faster than Windows menu bar
- Manipulate **independent variables**
 - different interfaces, user classes, tasks
 - in this case, y-position of menubar
- Measure **dependent variables**
 - times, errors, # tasks done, satisfaction
- Use statistical tests to accept or reject the hypothesis

Spring 2011

6.813/6.831 User Interface Design and Implementation

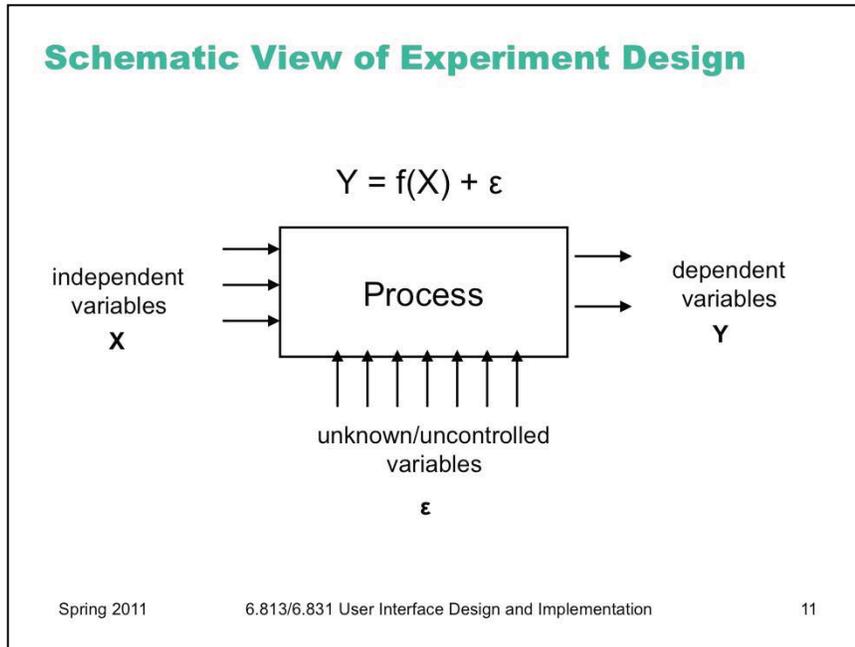
10

Here's a high-level overview of a controlled experiment. You start by stating a clear, **testable** hypothesis. By testable, we mean that the hypothesis must be quantifiable and measurable. Here's an example of a hypothesis that we might want to test: that the Macintosh menu bar, which is anchored to the top of the screen, is faster to access than the Windows menu bar, which is separated from the top of the screen by a window title bar.

You then choose your **independent variables** – the variables you're going to manipulate in order to test the hypothesis. In our example, the independent variable is the kind of interface: Mac menubar or Windows menubar. In fact, we can make it very specific: the independent variable is the y-position of the menubar (either $y = 0$ or $y = 16$, or whatever the height of the title bar is). Other independent variables may also be useful. For example, you may want to test your hypothesis on different user classes (novices and experts, or Windows users and Mac users). You may also want to test it on certain kinds of tasks. For example, in one kind of task, the menus might have an alphabetized list of names; in another, they might have functionally-grouped commands.

You also have to choose the **dependent variables**, the variables you'll actually measure in the experiment to test the hypothesis. Typical dependent variables in user testing are time, error rate, event count (for events other than errors – e.g., how many times the user used a particular command), and subjective satisfaction (usually measured by numerical ratings on a questionnaire).

Finally, you use statistical techniques to analyze how your changes in the independent variables affected the dependent variables, and whether those effects are **significant** (indicating a genuine cause-and-effect) or not (merely the result of chance or noise). We'll say more about statistical tests in the next lecture.



Here's a block diagram to help you visualize what we're trying to do with experiment design. Think of the process you're trying to understand (e.g., menu selection) as a black box, with lots of inputs and a few outputs. A controlled experiment twiddles some of the input knobs on this box (the independent variables) and observes some of the outputs (the dependent variables) to see how they are affected.

The problem is that there are many *other* input knobs as well (unknown/uncontrolled variables), that may affect the process we're observing in unpredictable ways. The purpose of experiment design is to eliminate the effect of these unknown variables, or at least render harmless, so that we can draw conclusions about how the independent variables affect the dependent variables.

What are some of these unknown variables? Consider the menubar experiment. Many factors might affect how fast the user can reach the menubar: the pointing device they're using (mouse, trackball, isometric joystick, touchpad); where the mouse pointer started; the surface they're moving the mouse on; the user's level of fatigue; their past experience with one kind of menubar or the other. All of these are unknown variables that might affect the dependent variable (speed of access).

Design of the Menubar Experiment

- Users
 - Windows users or Mac users?
 - Age, handedness?
 - How to sample them?
- Implementation
 - Real Windows vs. real Mac
 - Artificial window manager that lets us control menu bar position
- Tasks
 - Realistic: word processing, email, web browsing
 - Artificial: repeatedly pointing at fake menu bar
- Measurement
 - When does movement start and end?
- Ordering
 - of tasks and interface conditions
- Hardware
 - mouse, trackball, touchpad, joystick?
 - PC or Mac? which particular machine?

Spring 2011

6.813/6.831 User Interface Design and Implementation

12

Here are some of the issues we'd have to consider in designing the menubar experiment.

First, what user population do we want to sample? Does experience matter? Windows users will be more experienced with one kind of menu bar, and Mac users with the other. On the other hand, the model underlying our hypothesis (Fitts's Law for pointing tasks) is largely independent of long-term memory or practice, so we might expect that experience doesn't matter. But that's another hypothesis we should test, so maybe past experience should be an independent variable that we select when sampling.

How do we sample the user population we want? The most common technique (in academia) is advertising around a college campus, but this biases against older users and less-educated users. *Any* sampling method has biases; you have to collect demographic information, report it, and consider whether the bias influences the generalizability of your results. What implementation should we test? One possibility is to test the menu bars in their real context: inside the Mac interface, and inside the Windows interface, using real applications and real tasks. This is more realistic, but the problem is now the presence of confounding variables -- the *size* of the menu bars might be different, the reading speed of the font, the mouse acceleration parameters, etc. We need to control for as many of these variables as we can. Another possibility is implementing our own window manager that holds these variables constant and merely changes the position of the menu bar.

What tasks should we give the user? Again, having the user use the menu bar in the context of realistic tasks might provide more generalizable results; but it would also be noisier. An artificial experiment that simply displays a menu bar and cues the user to hit various targets on it would provide more reliable results. But then the user's mouse would always be in the menu bar, which isn't at all realistic. We'd need to force the user to move the mouse out of the menu bar between trials, perhaps to some home location in the middle of the screen.

How do we **measure** the dependent variable, time? Maybe from the time the user is given the cue ("click Edit") to the time the user successfully clicks on Edit.

What **order** should we present the tasks and the interface conditions? Using the same order all the time can cause both learning effects (e.g., you do better on the interface conditions because you got practice with the tasks and the study protocol during the first interface condition) and fatigue effects (where you do worse on later interface conditions because you're getting tired).

Finally, the hardware we use for the study can introduce lots of confounding variables. We should use the same computer for the entire experiment -- across both users and interface conditions.

Concerns Driving Experiment Design

- Internal validity
 - Are observed results actually **caused** by the independent variables?
- External validity
 - Can observed results be **generalized** to the world outside the lab?
- Reliability
 - Will consistent results be obtained by **repeating** the experiment?

Spring 2011

6.813/6.831 User Interface Design and Implementation

13

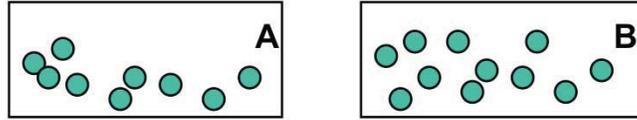
Unknown variation is the bugaboo in experiment design, and here are the three main problems it can cause.

Internal validity refers to whether the effect we see on the experiment outputs was actually caused by the changes we made to the inputs, or caused by some unknown variable that we didn't control or measure. For example, suppose we designed the menubar experiment so that the Mac menubar position was tested on an actual Mac, and the Windows menubar position was tested on a Windows box. This experiment wouldn't be internally valid, because we can't be sure whether differences in performance were due to the difference in the position of the menubar, or to some other (unknown) difference in two machines -- like font size, mouse acceleration, mouse feel, even the system timer used to measure the performance! Statisticians call this effect **confounding**; when a variable that we didn't control has a systematic effect on the dependent variables, it's a **confounding variable**.

One way to address internal validity is to hold variables constant, as much as we can: for example, conducting all user tests in the same room, with the same lighting, the same computer, the same mouse and keyboard, the same tasks, the same training. The cost of this approach is **external validity**, which refers to whether the effect we see can be generalized to the world outside the lab, i.e. when those variables are not controlled. But when we try to control *all* the factors that might affect menu access speed -- a fixed starting mouse position, a fixed menubar with fixed choices, fixed hardware, and a single user -- then it would be hard to argue that our lab experiment generalizes to how menus are used in the varying conditions encountered in the real world.

Finally, **reliability** refers to whether the effect we see (the relationship between independent and dependent variables) is repeatable. If we ran the experiment again, would we see the same effect? If our experiment involved only one trial -- clicking the menubar just once -- then even if we held constant every variable we could think of, unknown variations will still cause error in the experiment. A single data sample is rarely a reliable experiment.

How Many Marbles in Each Box?



- Hypothesis: box A has a different number of balls than box B
- Reliability
 - Counting the balls manually is reliable only if there are few balls
 - Repeated counting improves reliability
- Internal validity
 - Suppose we weigh the boxes instead of counting balls
 - What if an A ball has different weight than a B ball?
 - What if the boxes themselves have different weights?
- External validity
 - Does this result apply to all boxes in the world labeled A and B?

Spring 2011

6.813/6.831 User Interface Design and Implementation

14

Here's a simple example to illustrate internal validity, external validity, and reliability.

Suppose we have two boxes labeled A and B, and each box has some marbles inside. We want to test the **hypothesis** that these two boxes have different numbers of marbles. (For example, maybe each box is from a different manufacturer, but both boxes are the same price, and we want to claim that one box is a better deal than the other.) In this case, the **independent variable** is the identity of the box (the brand of the manufacturer), and the **dependent variable** we're trying to measure is the number of marbles inside the box.

This hypothesis is **testable** because we can do measurements on the boxes to test it. One measurement we might do is open the boxes and count the marbles inside each one. If we counted 3 marbles in one box, and 5 marbles in the other box, then we'd have strong evidence that A has fewer marbles than B. But if we counted 99 marbles in A and 101 marbles in B, would we be just as confident? No, because we know that our counting may have errors -- we may skip marbles, or we may double-count marbles. If we counted A again, we might come up with 102 marbles. The **reliability** of a single measurement isn't good enough. We can fix this reliability problem by better experiment design: counting each box several times, and computing a **statistic** (a summary of the measurements) instead of using a single measurement. The statistic is often the mean of the measurements, but others are possible.

Counting is pretty slow, though, so to make our experiment cheaper to run, suppose we weigh the boxes instead. If box A is lighter than box B, then we'll say that it has fewer marbles in it. It's easy to see the possible flaws in this experiment design: what if each marble in box A is lighter than each marble in box B? Then the whole box A might be lighter than B even if it has more marbles. If this is true, then it threatens the experiment's **internal validity**: the dependent variable we've chosen to measure (total weight) is a function not only of the number of marbles in the box, but also of the weight of each marble, and the weight of the marble also varies by the independent variable (the identity of the box).

Suppose the weight of marbles is regulated by an international standard, so we can argue that A marbles should weigh the same as B marbles. But what if box A is made out of metal, and box B is made out of cardboard? This would also threaten internal validity -- the weight of the box itself is a confounding variable. But now we can imagine a way to **control** for this variable, by pouring the marbles out of their original boxes into box C, a box that we've chosen, and weighing both sets of marbles separately in box C. We've changed our experiment design so that the weight of the box itself is held constant, so that it won't contribute to the measured difference between box A and box B.

But we've only done this experiment with the box A and box B we have in front of us. What confidence do we have in concluding that *all* boxes from manufacturer A have a different number of marbles than those from manufacturer B -- that if we went to the store and picked up new boxes of marbles, we'd see the same difference? That's the question of **external validity**.

Threats to Internal Validity

- Ordering effects
 - People learn, and people get tired
 - Don't present tasks or interfaces in same order for all users
 - Randomize or counterbalance the ordering
- Selection effects
 - Don't use pre-existing groups (unless group is an independent variable)
 - Randomly assign users to independent variables
- Experimenter bias
 - Experimenter may be enthusiastic about interface X but not Y
 - Give training and briefings on paper, not in person
 - Provide equivalent training for every interface
 - Double-blind experiments prevent both subject and experimenter from knowing if it's condition X or Y
 - Essential if measurement of dependent variables requires judgement

Spring 2011

6.813/6.831 User Interface Design and Implementation

15

Let's look closer at typical dangers to internal validity in user interface experiments, and some solutions to them. You'll notice that the solutions tend to come in two flavors: randomization (which prevents unknown variables from having systematic effects on the dependent variables) and control (which tries to hold unknown variables constant).

Ordering effects refer to the order in which different levels of the independent variables are applied. For example, does the user work with interface X first, and then interface Y, or vice versa? There are two effects from ordering: first, people learn. They may learn something from using interface X that helps them do better (or worse) with interface Y. Second, people get tired or bored. After doing many tasks with interface X, they may not perform as well on interface Y. Clearly, holding the order constant threatens internal validity, because the ordering may be responsible for the differences in performance, rather than inherent qualities of the interfaces. One good solution to this threat is **randomization**: present the interfaces, or tasks, or other independent variables in a random order to each user. Another good solution is counterbalancing (see a later slide).

Selection effects arise when you use pre-existing groups as a basis for assigning different levels of an independent variable. Giving the Mac menubar to artists and the Windows menubar to engineers would be an obvious selection effect. More subtle selection effects can arise, however. Suppose you let the users line up, and then assigned the Mac menubar to the first half of the line, and Windows menubar to the second half. This procedure seems "random", but it isn't – the users may line up with their friends, and groups of friends tend to have similar activities and interests. The same thing can happen even if people are responding "randomly" to your study advertisement – you don't know how "random" that really is! The only safe way to eliminate selection effects is honest randomization.

A third important threat to internal validity is **experimenter bias**. After all, you have a hypothesis, and you're hoping it works out – you're rooting for interface X. This bias is an unknown variable that may affect the outcome, since you're personally interacting with the user whose performance you're measuring. One way to address experimenter bias is **controlling** the protocol of the experiment, so that it doesn't vary between the interface conditions. Provide equivalent training for both interfaces, and give it on paper, not live.

An even better technique for eliminating experimenter bias is the **double-blind experiment**, in which neither the subject nor the experimenter knows which condition is currently being tested. Double-blind experiments are the standard for clinical drug trials, for example; neither the patient nor the doctor knows whether the pill contains the actual experimental drug, or just a placebo. With user interfaces, double-blind tests may be impossible, since the interface condition is often obvious on its face. (Not always, though! The behavior of cascading submenus isn't obviously visible, for example.)

Experimenter-blind tests are essential if measurement of the dependent variables requires some subjective judgement. Suppose you're developing an interface that's supposed to help people compose good memos. To measure the quality of the resulting memos, you might ask some people to evaluate the memos created with the interface, along with memos created with a regular word processor. But the memos should be presented in random order, and you should hide the interface that created each memo from the judge, to avoid bias.

Threats to External Validity

- Population
 - Draw a random sample from your real target population
- Ecological
 - Make lab conditions as realistic as possible in important respects
- Training
 - Training should mimic how real interface would be encountered and learned
- Task
 - Base your tasks on task analysis

Spring 2011

6.813/6.831 User Interface Design and Implementation

16

Here are some threats to external validity that often come up in user studies. If you've done a thorough user analysis and task analysis, in which you actually went out and observed the real world, then it's easier to judge whether your experiment is externally valid.

Population asks whether the users you sampled are representative of the target user population. Do your results apply to the entire user population, or only to the subgroup you sampled? The best way to ensure population validity is to draw a random sample from your real target user population. But you can't really, because users must choose, of their own free will, whether or not to participate in a study. So there's a **self-selection** effect already in action. The kinds of people who participate in user studies may have special properties (extroversion? curiosity? sense of adventure? poverty?) that threaten external validity. But that's an inevitable effect of the ethics of user testing. The best you can do is argue that on important, measurable variables – demographics, skill level, experience – your sample resembles the overall target user population.

Ecological validity asks whether conditions in the lab are like the real world. An office environment would not be an ecologically valid environment for studying an interface designed for the dashboard of a car, for example.

Training validity asks whether the interfaces tested are presented to users in a way that's realistic to how they would encounter them in the real world. A test of an ATM machine that briefed each user with a 5-minute tutorial video wouldn't be externally valid, because no ATM user in the real world would watch such a video. For a test of an avionics system in an airplane cockpit, on the other hand, even hours of tutorial may be externally valid, since pilots are highly trained.

Similarly, **task validity** refers to whether the tasks you chose are realistic and representative of the tasks that users actually face in the real world. If you did a good task analysis, it's not hard to argue for task validity.

Threats to Reliability

- Uncontrolled variation
 - Previous experience
 - Novices and experts: separate into different classes, or use only one class
 - User differences
 - Fastest users are **10 times** faster than slowest users
 - Task design
 - Do tasks measure what you're trying to measure?
 - Measurement error
 - Time on task may include coughing, scratching, distractions
- Solutions
 - Eliminate uncontrolled variation
 - Select users for certain experience (or lack thereof)
 - Give all users the same training
 - Measure dependent variables precisely
 - Repetition
 - Many users, many trials
 - Standard deviation of the mean shrinks like the square root of N (i.e., quadrupling users makes the mean twice as accurate)

Spring 2011

6.813/6.831 User Interface Design and Implementation

17

Once we've addressed internal validity problems by either controlling or randomizing the unknowns, and external validity by sampling and experiment protocol design, reliability is what's left.

Here's a good way to visualize reliability: imagine a bullseye target. The center of the bullseye is the true effect that the independent variables have on the dependent variables. Using the independent variables, you try to aim at the center of the target, but the uncontrolled variables are spoiling your aim, creating a spread pattern. If you can reduce the amount of uncontrolled variation, you'll get a tighter shot group, and more reliable results.

One kind of uncontrolled variation is a user's previous experience. Users enter your lab with a whole lifetime of history behind them, not just interacting with computers but interacting with the real world. You can limit this variation somewhat by screening users for certain kinds of experience, but take care not to threaten external validity when you artificially restrict your user sample.

Even more variation comes from differences in ability – intelligence, visual acuity, memory, motor skills. The best users may be **10 times better** than the worst, an enormous variation that may swamp a tiny effect you're trying to detect.

Other kinds of uncontrolled variation arise when you can't directly measure the dependent variables. For example, the tasks you chose to measure may have their own variation, such as the time to understand the task itself, and errors due to misunderstanding the task, which aren't related to the difficulty of the interface and act only to reduce the reliability of the test. Time is itself a gross measurement which may include lots of activities unrelated to the user interface: coughing, sneezing, asking questions, responding to distractions.

One way to improve reliability eliminates uncontrolled variation by holding variables constant: e.g., selecting users for certain experience, giving them all identical training, and carefully controlling how they interact with the interface so that you can measure the dependent variables precisely. If you control too many unknowns, however, you have to think about whether you've made your experiment externally invalid, creating an artificial situation that no longer reflects the real world.

The main way to make an experiment reliable is **repetition**. We run many users, and have each user do many trials, so that the mean of the samples will approach the bullseye we want to hit. As you may know from statistics, the more trials you do, the closer the sample mean is likely to be to the true value. (Assuming the experiment is internally valid of course! Otherwise, the mean will just get closer and closer to the *wrong* value.) Unfortunately, the standard deviation of the sample mean goes down slowly, proportionally to the square root of the number of samples N. So you have to *quadruple* the number of users, or trials, in order to double reliability.

Blocking

- Divide samples into subsets which are more homogeneous than the whole set
 - Example: testing wear rate of different shoe sole material
 - Lots of variation between feet of different kids
 - But the feet on the same kid are far more homogeneous
 - Each child is a block
- Apply all conditions within each block
 - Put material A on one foot, material B on the other
- Measure difference within block
 - $\text{Wear(A)} - \text{Wear(B)}$
- Randomize within the block to eliminate internal validity threats
 - Randomly put A on left foot or right foot

Spring 2011

6.813/6.831 User Interface Design and Implementation

18

Blocking is another good way to eliminate uncontrolled variation, and therefore increase reliability. The basic idea is to divide up your samples up into blocks that are more homogeneous than the whole set. In other words, even if there is lots of uncontrolled variation between blocks, the blocks should be chosen so that there is little variation within a block. Once you've blocked your data, you apply **all** the independent variable conditions **within** each block, and compare the dependent variables only within the block.

Here's a simple example of blocking. Suppose you're studying materials for the soles of childrens' shoes, and you want to see if material A wears faster or slower than material B. There's much uncontrolled variation between feet of different children – how they behave, where they live and walk and play – but the two feet of the same child both see very similar conditions by comparison. So we treat each child as a block, and apply one sole material to one foot, and the other sole material to the other foot. Then we measure the difference between the sole wear as our dependent variable. This technique prevents inter-child variation from swamping the effect we're trying to see.

In agriculture, blocking is done in space. A field is divided up into small plots, and all the treatments (pesticides, for example) are applied to plants in each plot. Even though different parts of the field may differ widely in soil quality, light, water, or air quality, plants in the same plot are likely to be living in very similar conditions.

Blocking helps solve reliability problems, but it doesn't address internal validity. What if we always assigned material A to the left foot, and material B to the right foot? Since most people are right-handed and left-footed, some of the difference in sole wear may be caused by footedness, and not by the sole material. So you should still randomize assignments within the block.

Between Subjects vs. Within Subjects

- “Between subjects” design
 - Users are divided into two groups:
 - One group sees only interface X
 - Other group sees only interface Y
 - Results are compared **between** different groups
 - Is $\text{mean}(x_i) > \text{mean}(y_j)$?
 - Eliminates variation due to ordering effects
 - User can’t learn from one interface to do better on the other
- “Within subjects” design
 - Each user sees both interface X and Y (in random order)
 - Results are compared **within** each user
 - For user i , compute the difference $x_i - y_i$
 - Is $\text{mean}(x_i - y_i) > 0$?
 - Eliminates variation due to user differences
 - User only compared with self

Spring 2011

6.813/6.831 User Interface Design and Implementation

19

The idea of blocking is what separates two commonly-used designs in user studies that compare two interfaces. Looking at these designs also gives us an opportunity to review some of the concepts we’ve discussed in this lecture.

A **between-subjects** design is unblocked. Users are randomly divided into two groups. These groups aren’t blocks! Why? First, because they aren’t more homogeneous than the whole set – they were chosen randomly, after all. And second, because we’re going to apply only one independent variable condition within each group. One group uses only interface X, and the other group uses only interface Y. The performance of the X group is then compared with the performance of the Y group. This design eliminates variation due to interface ordering effects. Since users only see one interface, they can’t transfer what they learned from one interface to the other, and they won’t be more tired on one interface than the other. But it suffers from reliability problems, because the differences between the interfaces may be swamped by the innate differences between users. As a result, you need more repetition – more users – to get reliable and significant results from a between subjects design.

A **within-subjects** design is blocked by user. Each user sees both interfaces, and the differential performance (performance on X – performance on Y) of all users is averaged and compared with 0. This design eliminates variation due to user differences, but may have reliability problems due to ordering effects.

Which design is better? User differences cause much more variation than ordering effects, so the between-subjects design typically needs more users than the within-subjects design. But the between-subjects design may be more externally valid, because users in the real world will probably end up using only one interface (X or Y), not both.

Counterbalancing

- Defeats ordering effects by varying order of conditions systematically (not randomly)
- Latin Square designs
 - randomly assign subjects to equal-size groups
 - A,B,C,... are the experimental conditions
 - Latin Square ensures that each condition occurs in every position in the ordering for an equal number of users

G1	G2
A	B
B	A

2x2

G1	G2	G3
A	C	B
B	A	C
C	B	A

3x3

G1	G2	G3	G4
A	D	C	B
B	A	D	C
C	B	A	D
D	C	B	A

4x4

... etc.

Spring 2011

6.813/6.831 User Interface Design and Implementation

20

Within-subjects designs suffer from ordering effects (particularly learning, which makes people get better, and fatigue, which makes them do worse). Randomizing the order of tasks and experimental conditions is one way to deal with these effects. Another way, particularly when the number of users is small, is **counterbalancing**, which ensures that every experimental condition occurs the same number of times at each position in the order. You counterbalance your experiment with the help of a **Latin square**, which is an $N \times N$ matrix with the property that a symbol occurs exactly once in each row and each column. A few Latin squares are shown here; it's not hard to recognize the pattern and reproduce it for higher N .

To use counterbalancing, you first determine N , the number of experimental conditions you have, which is done by taking the product of the different values of each independent variable you are using. For example, the Windows vs. Mac menubar experiment has one independent variable with two values, hence two experimental conditions. If we also decided to test how the speed of access varied with starting distance from the menu (say, 50 pixels, 500 pixels, and 1000 pixels), then we'd have 6 experimental conditions (2 menubars x 3 distances). Given the number of conditions N , we divide the users randomly into N equal groups ($G1 \dots GN$), and present each group with the conditions in the order of a different column from the Latin square.

Note that it's important for the number of users to be a multiple of N , so that the groups are equal in size. Otherwise the conditions won't occur the same number of times at each position in the order.

The simple Latin squares shown here have a flaw – *pairwise* learning effects are not controlled. In the 3x3 matrix, for example, B appears after A two-thirds of the time. In the 4x4 matrix, B follows A three-quarters of the time. So high performance on B may be due to practice on A, rather than inherent to B. This problem can be fixed by a **balanced Latin square**, left as an exercise for the reader. (Or see I. Scott MacKenzie, "Research Note: Within-subjects vs. Between-subjects Designs: Which to Use?", 2008, <http://www.yorku.ca/mack/RN-Counterbalancing.html>).

Kinds of Measures

- Self-report
- Observation
 - Visible observer
 - Hidden observer
- Archival records
 - Public records
 - Private records
- Trace

Spring 2011

6.813/6.831 User Interface Design and Implementation

21

One thing we've glossed over a bit is how to **measure** the dependent variables we're interested in. Different measurement approaches have different kinds of noise, and some have biases induced by the user's awareness that their behavior is being studied. These biases are generally called **reactivity effects** by the social science community. Here's a simple taxonomy from the McGrath paper ("Methodology Matters", cited earlier), ranked by the obtrusiveness (and hence susceptibility to reactivity) of the method.

Self-report means asking the user to provide the data. Examples might be "How many times a day do you check email?" or "How much did you like this interface?" Self-reports are applicable to many different questions, and very cheap to collect. They are also noisy (particularly where quantitative data is requested), and may be biased by a variety of reactive effects, like politeness (trying to please the experimenter) and social desirability (saying what they think they *should* say, rather than the truth). Surveys generally exclusively use self-reported measures, though carefully-designed surveys can eliminate some of the biases. Lab experiments must resort to self-reports for some variables, particularly satisfaction, but better alternatives can often be found.

Observation means either the experimenter or an instrument (stopwatch, logging, screen capture, camera) is capturing the data. Observation is more expensive, but also more objective and controlled. A key distinction is whether the observation is visible (known to the subject) or hidden. Visible observation can produce distorting effects, like the classic "Hawthorne effect", in which people perform better simply because they know they're being studied. Hidden observation raises ethical questions of informed consent. Usually in lab experiments, the dilemma is resolved by using visible observation (or at least informed consent about observation), but using it on *all* conditions, and making it as unobtrusive as possible so that users stop thinking about it.

The last two measures are not used in lab experiments, because they use data that was not created solely for the purpose of the experiment, so they cannot be controlled. **Archival records** are records of past behavior, not made expressly for the purpose of the current research. Like observation, archival records can be distinguished by whether the user was aware that the records might be studied or read by another person (public) or not (private). Comments on a blog are public archival records; files on your hard drive are private. Finally, **traces** are laid down by behavior without the people involved even being *aware* that they are leaving something behind. The classic example of a trace is "read wear", the dogears and damage to books that indicate where and how they've been used. Web server access logs probably also fall into this category, since the vast majority of users are probably unaware that their page visits are being recorded.

Triangulation

- Any given research method has advantages and limitations
 - lab experiment not externally valid
 - field study not controlled
 - survey biased by self-report
- For higher confidence, researchers **triangulate**
 - try several different methods to attack the same question; see if they concur or conflict
- Triangulation is rarely seen in a single HCI paper
 - More important that triangulation be happening **across the field**
 - encourage a diversity of research methods from different researchers aimed at the same question

Spring 2011

6.813/6.831 User Interface Design and Implementation

22

Let's conclude the lecture by returning to the main kinds of research methods in HCI (lab experiments, field studies, and surveys). We're focusing on lab experiments in this course, because there's much to say about them that isn't covered by any other course in the EECS curriculum. But frankly, each of these methods has advantages and disadvantages. Lab experiments are abstract and obtrusive, and may not be representative of the real world. Field studies can't be controlled, so it's hard to make strong, precise claims about comparative usability. Surveys are biased by reactivity.

So a research program that depends entirely on only one of these methods as a source of data and validation is likely to be biased and incomplete. One way to deal with this problem is by **triangulation** – using multiple methods (usually at least three, so that they can vote) to tackle the same research question. If they all support your claim, then you have much stronger evidence, without as many biases.

Triangulation is rarely seen in a single paper, but is not uncommon in a single researcher's whole research program (the series of papers generated by their work), and is frankly essential for the HCI field as a whole. A diversity of research methods is necessary for producing reliable knowledge. There's an active debate right now in the HCI community about this question, because of a sense that the flagship conference (CHI) has become dominated by papers using controlled lab experiments.

Summary

- Research methods in HCI include lab experiments, field studies, and surveys
- Controlled experiments manipulate independent variables and measure dependent variables
- Must consider and defend against threats to internal validity, external validity, and reliability
- Blocking, randomization, and counterbalancing techniques help

UI Hall of Fame or Shame?



© Bryan MacKinnon. All rights reserved.

Spring 2011

6.813/6.831 User Interface Design and Implementation

24

Here's next lecture's candidate for the UI Hall of Fame or Shame: Password Engine, an iPhone app (<http://appshopper.com/utilities/password-engine>). Its purpose is to generate passwords.

What do you think about its simplicity?

What about the Save and Revert buttons at the bottom? What in general are good usability approaches for saving changed data to a backend?

MIT OpenCourseWare
<http://ocw.mit.edu>

6.831 / 6.813 User Interface Design and Implementation
Spring 2011

For information about citing these materials or our Terms of Use, visit: <http://ocw.mit.edu/terms>.