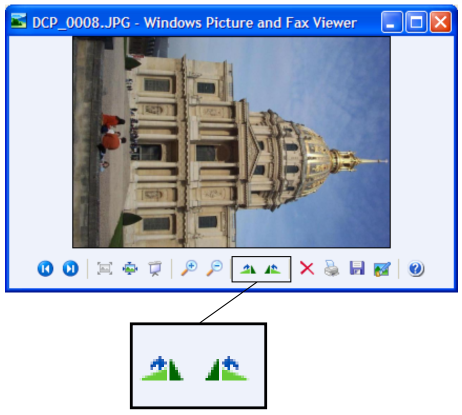
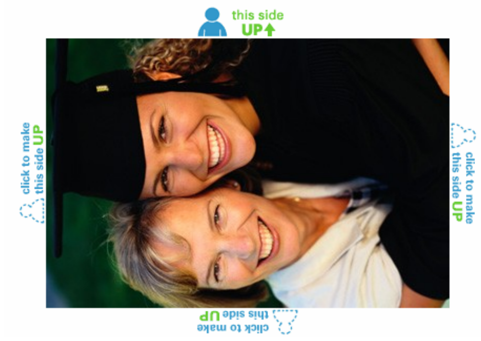
Lecture 13: Controlled Experiments

UI Hall of Fame or Shame?




this side UP
click to make this side UP
click to make this side UP
this side UP click to make

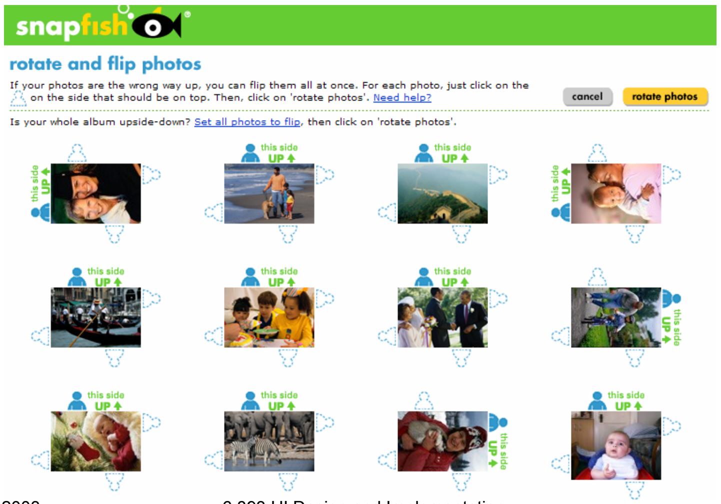
Source: Madleina Scheidegger

For today's UI Hall of Fame and Shame, we'll focus on the Rotate commands in photo browsers and drawing editors. These commands rotate an image by 90 degree increments, either clockwise or counterclockwise.

In the Windows XP Image Viewer, the rotation commands are represented by toolbar buttons. Unfortunately, the icons on these buttons don't work well. They're very similar to each other, and the arrow doesn't stand out (poor **contrast**). The icon tells a little story, showing before and after representations of a simplified abstract object. That's not such a bad thing in general, but it obscures the important differences between the two icons and forces you to study them carefully to figure out what they mean. Worse, the **mapping** is backwards: the Rotate Right button (with the right-pointing arrow) actually appears on the left.

The Snapfish web site (for storing and printing digital photo albums) has a neat solution to this problem. It does away with the notion of rotating entirely; instead, you just click on the side of the photo that you want to be on top. A little head-and-shoulders icon provides an **affordance** for clicking, while reminding about the heads-up orientation. This interface is neat because the controls are **directly mapped** to their effect (the side of the image that becomes the top). There's no need to mention right or left, clockwise or counterclockwise, or 90 or 180 degrees. The rotation is done by direct manipulation of the image itself. The labels are unfortunate – particularly the unreadable upside-down label! -- but new idioms often need extra help at first.

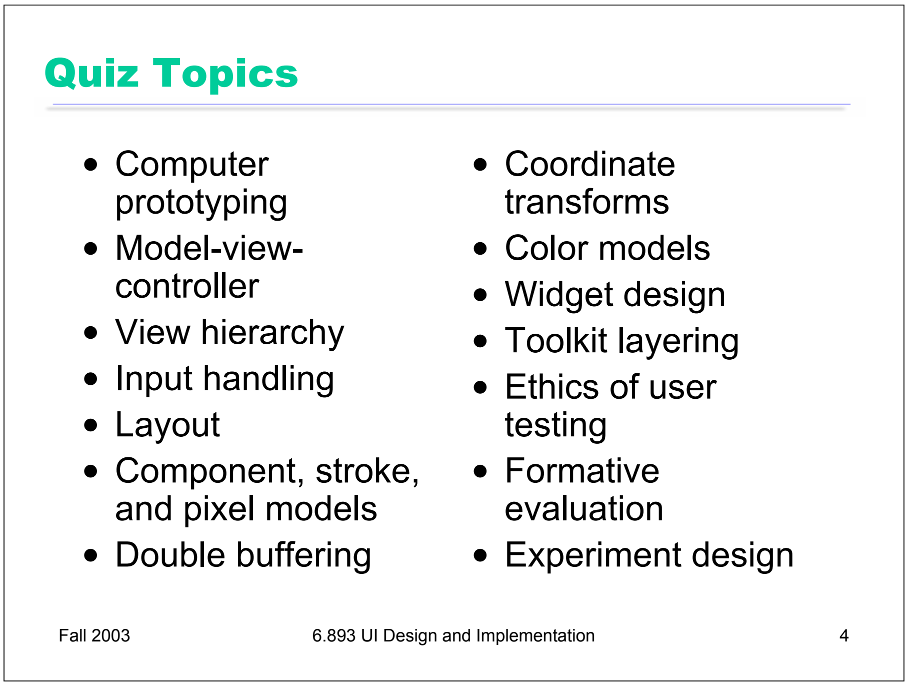
UI Hall of Fame



If your photos are the wrong way up, you can flip them all at once. For each photo, just click on the on the side that should be on top. Then, click on 'rotate photos'. Need help?

Is your whole album upside-down? Set all photos to flip, then click on 'rotate photos'.

Snapfish's technique is particularly efficient for rotating many images individually. That's particularly important for digital photos, because it's common to turn the camera to take portrait-aspect picture, but all camera photos use landscape-aspect by default. (Why? A much better default would be to always put the force of gravity downward. How much would it cost to put a little accelerometer in the camera, so that it can detect which way is down and rotate the image automatically?)

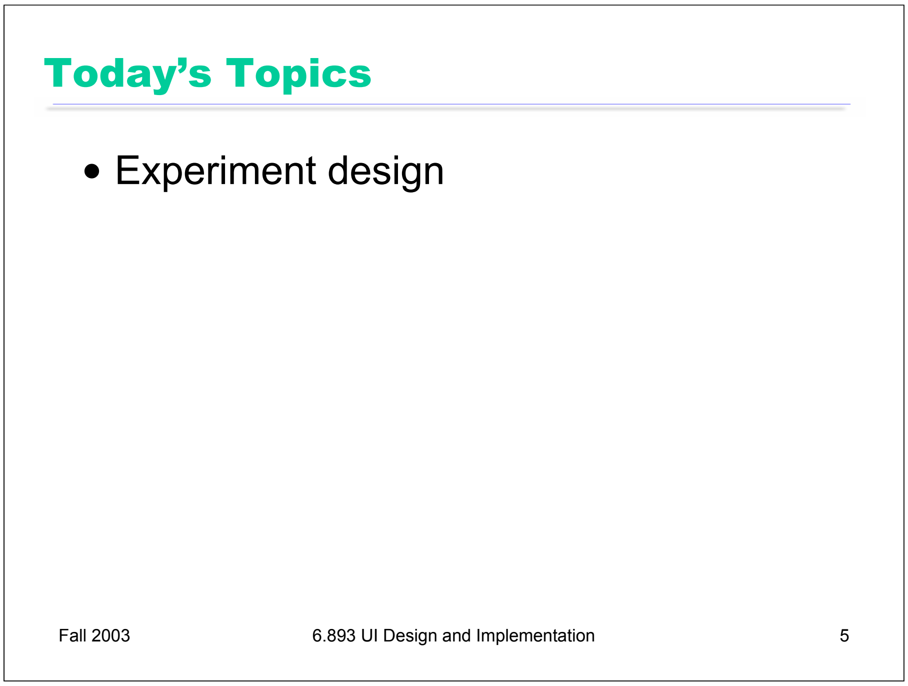


Quiz Topics

- Computer prototyping
- Model-view-controller
- View hierarchy
- Input handling
- Layout
- Component, stroke, and pixel models
- Double buffering
- Coordinate transforms
- Color models
- Widget design
- Toolkit layering
- Ethics of user testing
- Formative evaluation
- Experiment design

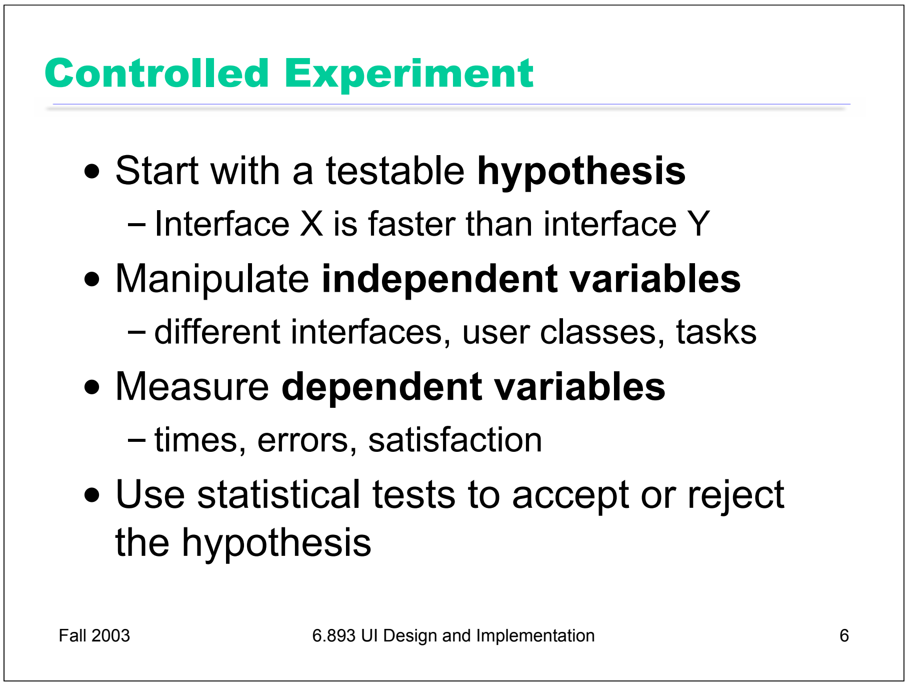
Quiz 2 is coming up in 2 weeks. It will cover Lectures 8 through 13 (today's lecture), plus any of the readings for those lectures.

Here are some of the topics you can expect to see on the quiz. Since most of the relevant lectures were about UI implementation, you should also expect to read and write some Java code on the quiz.


Today's Topics

- Experiment design

Today's lecture covers some of the issues involved in designing a controlled experiment. The issues are general to all scientific experiments, but we'll look specifically at how they apply to user interface testing.



Controlled Experiment

- Start with a testable **hypothesis**
  - Interface X is faster than interface Y
- Manipulate **independent variables**
  - different interfaces, user classes, tasks
- Measure **dependent variables**
  - times, errors, satisfaction
- Use statistical tests to accept or reject the hypothesis

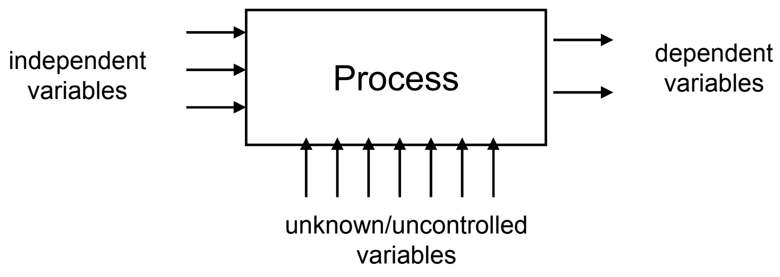
Here's a high-level overview of a controlled experiment. You start by stating a clear, **testable** hypothesis. By testable, we mean that the hypothesis must be quantifiable and measurable. For example, your hypothesis might be, "menu bars are faster than a Gimp-style right-click menu with hierarchical submenus". Here's another example that we'll use throughout this lecture: suppose you've developed two materials for the soles of children's shoes. Then your hypothesis might be, "material A wears slower than material B."

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, menubar or right-click menu. 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. In the shoe sole example, the independent variable would be the type of material used to make the shoe sole.

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 a questionnaire). In the shoe example, the dependent variable might be the thickness of the sole after a subject has worn it for a while.

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 won't say much in this class about the statistical tests, but it's essential to know and understand them if you want to analyze the results of an experiment. You can learn these techniques in other MIT courses (e.g. 9.07, 14.30). You'll also find some reference books in the General Information section of the course web site.

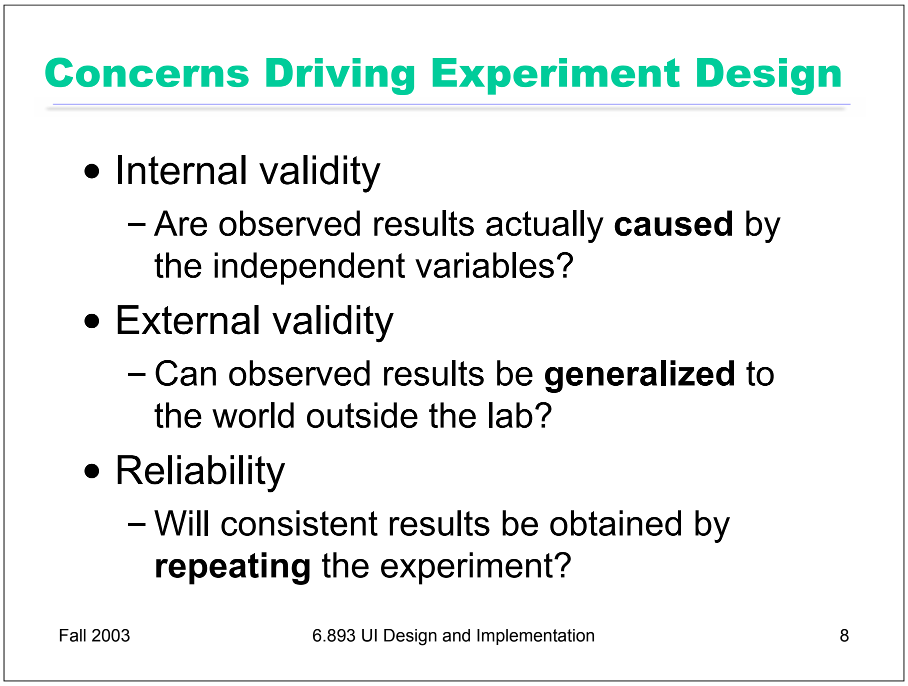
Schematic View of Experiment Design



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? Let's consider the shoe example. Many factors might affect the rate of wear of a shoe sole: the kind of surface walked on; the weight of the child; the way they walk (e.g., dragging their feet); their overall level of activity (sedentary or athletic); the kinds of activities they do (dancing vs. bicycling); maybe even the ambient temperature (which might soften the sole). All of these are unknown variables that might affect the dependent variable (amount of wear).



Concerns Driving Experiment Design

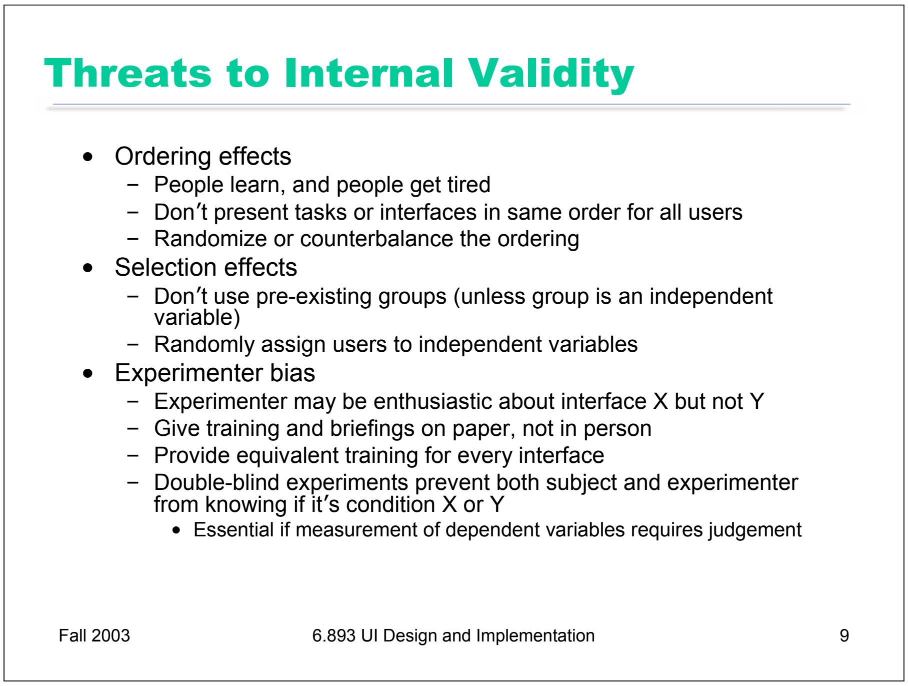
- Internal validity
  - Are observed results actually **caused** by the independent variables?
- External validity
  - Can observed results be **generalized** to the world outside the lab?
- Reliability
  - Will consistent results be obtained by **repeating** the experiment?

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 shoe experiment so that sneakers made with material A were given to boys, and sneakers made with material B were given to girls. This experiment wouldn't be internally valid, because we can't be sure whether different amounts of wear are due to the difference in materials, or to some (unknown) difference in the behavior of boys and girls. (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. If we tried to control all the factors that might affect shoe sole wear – choosing a single surface, with one designated activity, by a single person – then it would be hard to argue that our lab experiment generalizes to how soles might wear 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 tested only one pair of shoes, 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.



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

Let's look closer at typical dangers to internal validity, and some solutions to them. You'll notice that the solutions 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).

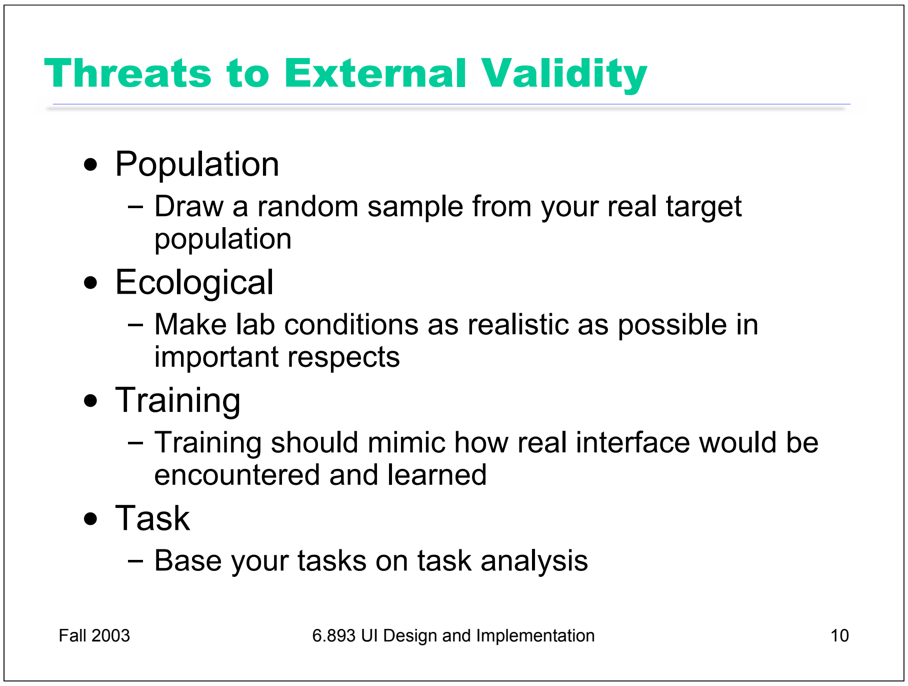
Ordering effects refer to the order in which different levels of the independent variables are applied. For example, does the user work with interface X first, and then interface Y, or vice versa? There are two effects from ordering: first, people learn. They may learn something from using interface 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. The solution to this threat is **randomization**: present the interfaces, or tasks, or other independent variables in a random order to each user.

Selection effects arise when you use pre-existing groups as a basis for assigning different levels of an independent variable. Our earlier example in which A-shoes were given to boys and B-shoes to girls was an obvious selection effect. More subtle selection effects can arise, however. Suppose you let the kids line up, and then assigned A-shoes to the first half of the line, and B-shoes to the second half. This procedure seems "random", but it isn't – the kids may line up with their friends, and groups of friends tend to have similar activities. The only safe way to eliminate selection effects is honest randomization.

A third important threat to internal 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 the face. (Not always, though! The behavior of hierarchical submenus isn't obviously visible.)

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

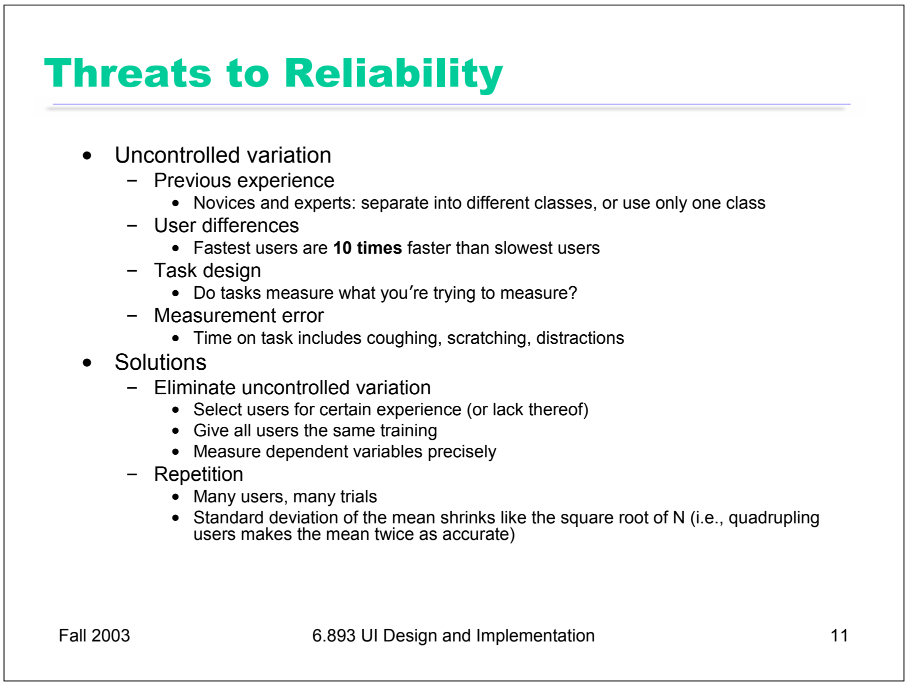
Here are some threats to external validity that often come up in user studies. If you've done a thorough user analysis and task analysis, in which you actually went out and observed the real world, then it's easier to judge whether your experiment is externally valid.

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 unknown variables (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 includes 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)

Fall 2003 6.893 UI Design and Implementation 11

Once we've addressed internal validity problems by either controlling or randomizing the unknowns, 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.

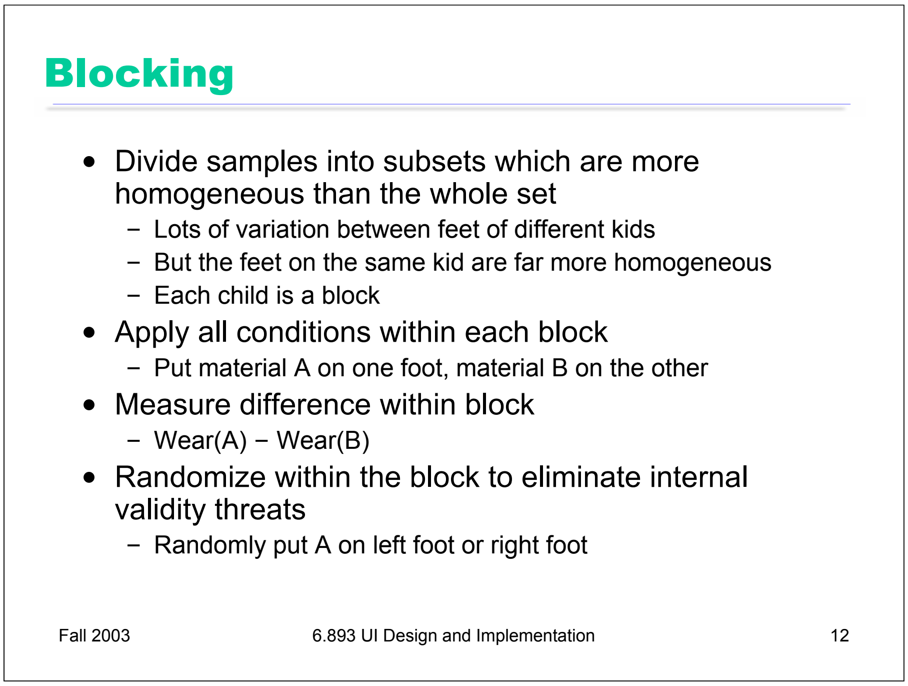
One kind of uncontrolled variation is a user's previous experience. Users enter your lab with a whole lifetime of history behind them, not just interacting with computers but interacting with the real world. You can limit this variation somewhat by screening users for certain kinds of experience, but 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 are **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

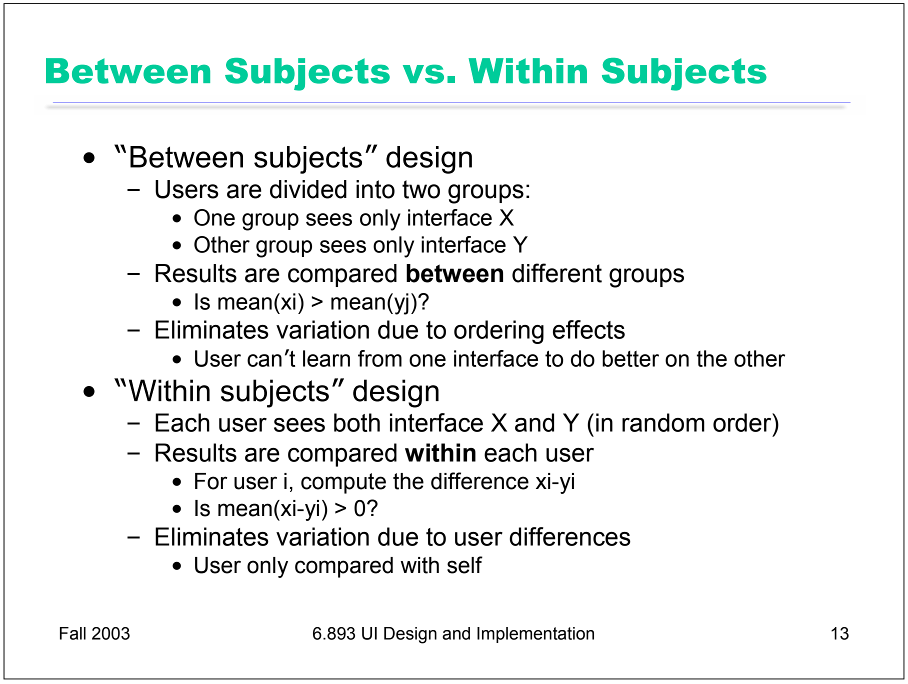
- Divide samples into subsets which are more homogeneous than the whole set
  - Lots of variation between feet of different kids
  - But the feet on the same kid are far more homogeneous
  - Each child is a block
- Apply all conditions within each block
  - Put material A on one foot, material B on the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

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.

Blocking is a natural technique for the shoe sole material example. 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 mean(xi) > mean(yj)?
  - 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 xi-yi
    - Is mean(xi-yi) > 0?
  - Eliminates variation due to user differences
    - User only compared with self

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

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 mean performance of the X group is then compared with the mean 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 needs more users than the within-subjects design. But the between-subjects design may be more externally valid.