Evidence-Based Software Engineering: Controlled (Laboratory) Experiments

Why and When?

The role for an experiment is to provide a means of investigating a *cause & effect* relationship (a *causal* link) between some factor and the observed outcomes. Researchers are likely to begin with a testable hypothesis of some form (e.g. smoking cigarettes causes lung cancer; code inspections find more bugs than black box testing; …) and seek to perform suitably rigorous studies to 'prove' or 'disprove' it[1]. The design needs to isolate the key factor and its effect as far as possible, so that the effects of changes to the factor can be identified and measured. (This isolation from context is why we refer to 'laboratory experiments'.)

"A controlled experiment in software engineering is a randomised or quasi-experiment, in which individuals or teams (the study units) conduct one or more software engineering tasks for the sake of comparing different populations, processes, methods, techniques, languages or tools (the treatments)." (Sjøberg et al, 2005)

Hence conducting an experiment is only sensible where there is some model and an associated hypothesis that can be tested. In a software engineering context we may have many possible sources of bias because of the human element that is usually involved, such as: prior experience; 'learning' effects; training bias (eg the order in which things are taught); selection of participants… Conducting an experiment can be challenging!

Planning & Designing an Experiment

Do not under-rate this one… As for any empirical study, should begin by writing a *protocol* for the study, which addresses the following key issues:

- the *research question(s)* which should lead to a testable hypothesis
- the *dependent and independent variables* arising from the research question
- determine how the variables will be *controlled* (the experimental environment)
- select the *context* – how the study will be structured
- develop a *measurement plan* for collecting the data
- determine how the *participants* will be recruited and selected
- assess the internal and external threats to validity
- plan the *analysis* that we will undertake [covered in later lectures]

Again, it is essential to conduct a 'dry run' beforehand, using carefully chosen participants to get effective feedback.

The hypothesis

Usually written as a prediction, e.g. "a falling body will accelerate at a uniform and standard rate" and should be *testable* so that it can be 'proved' or 'disproved', which means that we also need a *null hypothesis* that states that there are no real underlying trends (causality) and that any differences observed are coincidental and can be ascribed to statistical fluctuations

Independent & dependent variables

The *independent* variable(s) will be associated with *cause* and should change as a result of the activities of the investigator (e.g. number of errors 'seeded', length of an item of software, time allocated to a task, the training provided, …). (Sometimes the only independent variable will be the experimental treatment.) Using more than one makes analysis (and the experiment) more difficult. The *dependent* variable is associated with *effect* and its value is expected to change as a result of changes made to the independent variable. Measuring this is necessary in order to assess the outcomes of the experiment. Examples might be time taken, quality of solution, order in which tasks are completed, number completed,…

The controls

To provide a convincing demonstration of cause and effect, we need to minimise the effects of any other factors that might influence the dependent variable. Some ways of doing this include:

[1] Strictly, *prove* is a mathematical concept and an empirical study can only *demonstrate*. However, as 'prove' is used widely it is used here—but recognising the different meaning that it has in this context.

- eliminate the factor
- hold the factor constant
- use random selection of subjects
- use control groups (divide into two groups, and for the control group have no manipulation of the independent variable so that any difference between the groups can be attributed to the variable)

Also need to establish what the ‘baseline’ is (what are the control group doing) since it forms the basis for comparison with the intervention. This may well be ill-defined (eg ‘existing practices’) and it may also be difficult to control. In contrast, where participants are *recipients* of a treatment (as in clinical medicine), then it is possible to perform *randomised controlled trials* (RCTs) that use **double blinding,** by which neither the researcher nor the participant knows who is in which group, but this is rarely practical in computing.

Experimental forms
An experiment uses a *between subject* design that involves randomly allocating each participant to one of two groups, the ‘experimental group’ uses the intervention, while the ‘control group’ performs the task in the way they would have done before. A problem for SE is that it may be difficult to prescribe how a control group should work. For example, in a study of pair programming the intervention is well-defined, but the ‘control’ (solo programming) is not well-defined.

SE mainly uses *quasi-experiments*. These are used where it is not possible to randomise allocation of participants to groups, perhaps because they need to possess particular skills, and take many forms. One form commonly used in SE is a *within subject* design, which involves the participants in performing sequential tasks whereby:

- each participant is involved in more than one treatment (task)
- order of treatment is randomised
- can be used to detect small effects **but** need to beware of ‘learning effects’ that might interact with the experimental conditions

A within subject study may employ *crossover* forms to try to isolate learning effects, by which:

- group A gets intervention 1 followed by intervention 2
- group B gets intervention 2 followed by intervention 1

Observation/Measurement
Experiments often require both *pre-test* and *post-test* measurements of the dependent variable to help determine cause and effect (‘before and after’). Examples of such measurements include:

- project and task data such as *time to completion*, *cost* (in whatever form)
- self-reported responses by which the participants fill in a questionnaire
- behaviour counts, such as how often a help system is invoked
- number of bugs found in a block of code

Experiments usually involve participants performing *tasks* that will generate the necessary measures, such as debugging code, analysing a design structure, developing code via pair programming,… So the experimental design also involves designing these tasks, together with any necessary training material and the necessary data collection mechanisms. All of which should really be addressed in the protocol.

Threats to validity
We need to have confidence in the findings from an experiment and hence need to assess its validity. The design needs to consider many possible sources of *internal* threats such as: imbalanced groups; history (events occurring between measures that affect the participants); reactivity (participants may try to ‘help’ the experimenter); learning (such as when participants in cross-over studies use experience gained in earlier treatments). For *external* validity (how well do the findings apply outside the experiment itself), the ideal might be ‘replication studies’ taking place in different places and (obviously) with different participants. Two common forms used are *close* replication of a study, to give confidence in the results, and *differentiated* replication, to help identify how widely the results apply.