Empirical Investigation

Empirical SE

- Fill the gap between research and practice by:
  - Developing methods for studying SE practice
  - Building a body of knowledge of SE practice
  - Validating research before deployment in industrial settings

SE Investigation

- What is software engineering investigation?
- Applying “scientific” principles and techniques to investigate properties of software and software related tools and techniques.

- Why talking about software engineering investigation?
- Because the standard of empirical software engineering research is quite poor.

SE Investigation: Examples

- Experiment to confirm rules-of-thumb
  - Should the LOC in a module be less than 200?
  - Should the number of branches in any functional decomposition be less than 7?
- Experiment to explore relationships
  - How does the project team experience with the application affect the quality of the code?
  - How does the requirements quality affect the productivity of the designer?
  - How does the design structure affect maintainability of the code?
- Experiment to initiate novel practices
  - Would it be better to start OO design with UML?
  - Would the use of SRE improve software quality?

SE Investigation: Why?

- To improve (process and/or product)
- To evaluate (process and/or product)
- To prove a theory or hypothesis
- To disprove a theory or hypothesis
- To understand (a scenario, a situation)
- To compare (entities, properties, etc.)

SE Investigation: What?

- Person's performance
- Tool's performance
- Person's perceptions
- Tool's usability
- Document's understandability
- Program's complexity

etc.

SE Investigation: Where & When?

- In the field
- In the lab
- In the classroom

- Anytime depending on what questions you are asking

SE Investigation: How?

- Hypothesis/question generation
- Data collection
- Data evaluation
- Data interpretation

- Feed back into iterative process

SE Investigation: Characteristics

- Data sources come from industrial settings
  - This may include people, program code, etc.
- Usually
  - Surveys
  - Case-studies (→ hypothesis generation)
  - Experiments (→ hypothesis testing)

Where Data Come From?

- First Degree Contact
  - Direct access to participants
- **Example:**
  - Brainstorming
  - Interviews
  - Questionnaires
  - System illustration
  - Work diagrams
  - Think-aloud protocols
  - Participant observation

Where Data Come From?

- Second Degree Contact
  - Access to work environment during work time, but not necessarily participants
- **Example:**
  - Instrumenting systems
  - Real time monitoring

Where Data Come From?

- Third Degree Contact
  - Access to work artifacts, such as source code, documentation
- **Example:**
  - Problem report analysis
  - Documentation analysis
  - Analysis of tool logs
  - Off-line monitoring

Investigation Principles

There are 4 main principles of investigation:

1. **Stating the hypothesis:** What should be investigated?
2. **Selecting investigation technique:** conducting surveys, case studies, formal experiments
3. **Maintaining control over variables:** dependent and independent variables
4. **Making meaningful investigation:** verification of theories, evaluating accuracy of models, validating measurement results.

SE Investigation Techniques

- Three ways to investigate:
  - **Formal experiment:** A controlled investigation of an activity, by identifying, manipulating and documenting key factors of that activity.
  - **Case study:** Document an activity by identifying key factors (inputs, constraints and resources) that may affect the outcomes of that activity.
  - **Survey:** A retrospective study of a situation to try to document relationships and outcomes.

Case-study or Experiment?

- How to decide whether conduct an experiment or perform a case-study?

Factor	Experiment	Case-study
Retrospective	Yes (usually)	No (usually)
Level of control	High	Low
Difficulty of control	Low	High
Level of replication	High	Low
Cost of replication	Low	High
Can Generalize?	Yes (may be)	No

Control is the key factor

Hypothesis

- The first step is deciding what to investigate.
- The goal for the research can be expressed as a hypothesis in quantifiable terms that is to be tested.
- The test result (the collected data) will confirm or refute the hypothesis.
- **Example:**

  Can Software Reliability Engineering (SRE) help us to achieve an overall improvements in software development practice in our company?

Control /1

- What variables may affect truth of a hypothesis? How do they affect it?
- **Variable:**
  - ***Independent*** (values are set by the experiment or initial conditions)
  - ***Dependent*** (values are affected by change of other variables)
- **Example:** Effect of "programming language" on the "quality" of resulting code.
  - Programming language is an independent and quality is a dependent variable.

Control /2

- A common mistake: ignoring other variables that may affect the values of a dependent variable.
- **Example:**
  - Suppose you want to determine whether a change in ***programming language*** (independent variable) can affect the ***productivity*** (dependent variable) of your project. For instance, you currently use FORTRAN and you want to investigate the effects of changing to Ada. The values of all other variables should stay the same (e.g., application experience, programming environment, type of problem, etc.)
  - Without this you cannot be sure that the difference in productivity is attributable to the change in language.
- But list of other variables may grow beyond control!

Formal Experiments: Planning

1. **Conception**
   - Defining the goal of investigation
2. **Design**
   - Generating *quantifiable* (and *manageable*) hypotheses to be tested
   - Defining experimental objects or units
   - Identifying experimental subject
   - Identifying the response variable(s)

Formal Experiments: Planning

3. **Preparation**
   - Getting ready to start, e.g., purchasing tools, hardware, training personnel, etc.
4. **Execution**
5. **Review and analysis**
   - Review the results for soundness and validity
6. **Dissemination & decision making**
   - Documenting conclusions

Formal Experiments: Principles

- An experimental unit is the experimental object to which a single treatment is applied.
- Experimental error describes the failure of two identically treated experimental units to yield identical results.
- To address the problem of variability by giving guidance on forming experimental units so as to minimize experimental error, there are three major principles.

Formal Experiments: Principles

1. Replication

- Experiment under identical conditions should be repeatable.
- Confounded results (unable to separate the results of two or more variables) should be avoided.

2. Randomization

- The experimental trials must be organized in a way that the effects of uncontrolled variables are minimized

Formal Experiments: Principles

3. Local control

- **Blocking:** allocating experimental units to blocks or groups so the units within a block are relatively homogeneous. The blocks are designed so that the experimental design captures the anticipated variation in the blocks by grouping like varieties, so that the variation does not contribute to the experimental error.
- **Balancing:** is the blocking and assigning of treatments so that an equal number of subjects is assigned to each treatment. Balancing is desirable because it simplifies the statistical analysis.

Example: Blocking & Balancing

- You are investigating the comparative effects of three design techniques on the quality of the resulting code.
- The experiment involves teaching the techniques to 12 developers and measuring the number of defects found per 1000 LOC to assess the code quality.
- It may be the case that the twelve developers graduated from three universities. It is possible that the universities trained the developers in very different ways, so that being from a particular university can affect the way in which the design technique is understood or used.
- To eliminate this possibility, three blocks can be defined so that the first block contains all developers from university X, the second block from university Y, and the third block from university Z. Then, the treatments are assigned at random to the developers from each block. If the first block has six developers, two are assigned to design method A, two to B, and two to C.

Types of experimental designs

- An independent variable is called a factor in the experimental design.
- Various values or classifications for each factor are called the levels of the factor.
- Levels can be continuous or discrete, quantitative or qualitative.

Factorial design:

Types of experimental designs

Factorial design:

- **Crossing** (each level of each factor appears with each level of the other factor
- **Nesting** (each level of one occurs entirely in conjunction with one level of another)
- Proper nested or crossed design may reduce the number of cases to be tested.

		Factor *B*		
		Level 1	Level 2	Level 3
Factor *A*	Level 1	$a_1 b_1$	$a_1 b_2$	$a_1 b_3$
	Level 2	$a_2 b_1$	$a_2 b_2$	$a_2 b_3$

Factor *A*					
Level 1			Level 2		
Factor *B*			Factor *B*		
Level 1	Level 2	Level 3	Level 1	Level 2	Level 3
$a_1 b_1$	$a_1 b_2$	$a_1 b_3$	$a_2 b_1$	$a_2 b_2$	$a_2 b_3$

Types of experimental designs

- COBOL programmers with less than two years of experience with COBOL (a_1b_1)
- COBOL programmers with less than two years of experience with Pascal (a_1b_3)
- COBOL programmers with at least two years of experience with COBOL (a_1b_2)
- COBOL programmers with at least two years of experience with Pascal (a_1b_4)
- Pascal programmers with less than two years of experience with COBOL (a_2b_1)
- Pascal programmers with less than two years of experience with Pascal (a_2b_3)
- Pascal programmers with at least two years of experience with COBOL (a_2b_2)
- Pascal programmers with at least two years of experience with Pascal (a_2b_4)

Factor A: Language			
Level 1: COBOL		Level 2: Pascal	
<2 years experience with COBOL	≥2 years experience with COBOL	<2 years experience with Pascal	≥2 years experience with Pascal
a_1b_1	a_1b_2	a_2b_3	a_2b_4

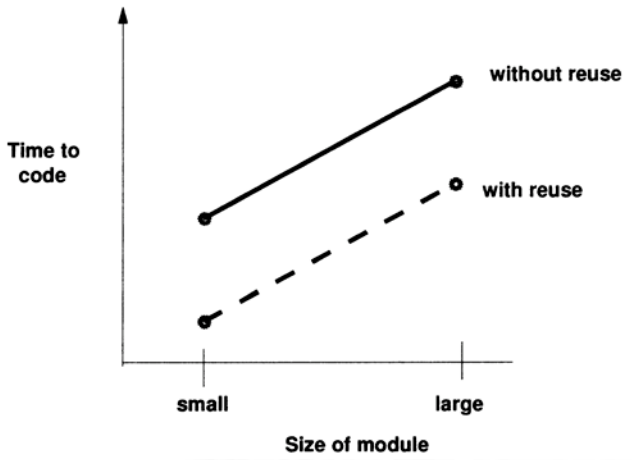
Figure 4.6: Nested design for language and

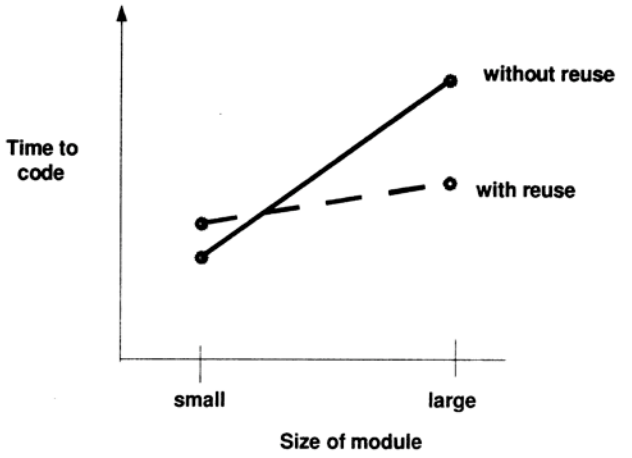
The eight categories are generated because we are examining experience with language type, rather than experience in general. However, with a nested design we can take advantage of the fact that the two factors are related. As depicted in Figure 4.6, we need only consider four treatments with a nested design, rather than eight treatments a_1b_3, a_1b_4, a_2b_1 and a_2b_2 are not used. Thus, nested designs take advantage of factor dependencies, and they reduce the number of cases to be considered.

Nesting can involve more than two factors.

Selecting an experimental design

- Selecting the number of variables/factors:
  - Single variable
  - Multiple variables
- **Example:** Measuring time to code a program module with or without using a reusable repository
  - Without considering the effects of experience of programmers
  - With considering the effects of experience of programmers

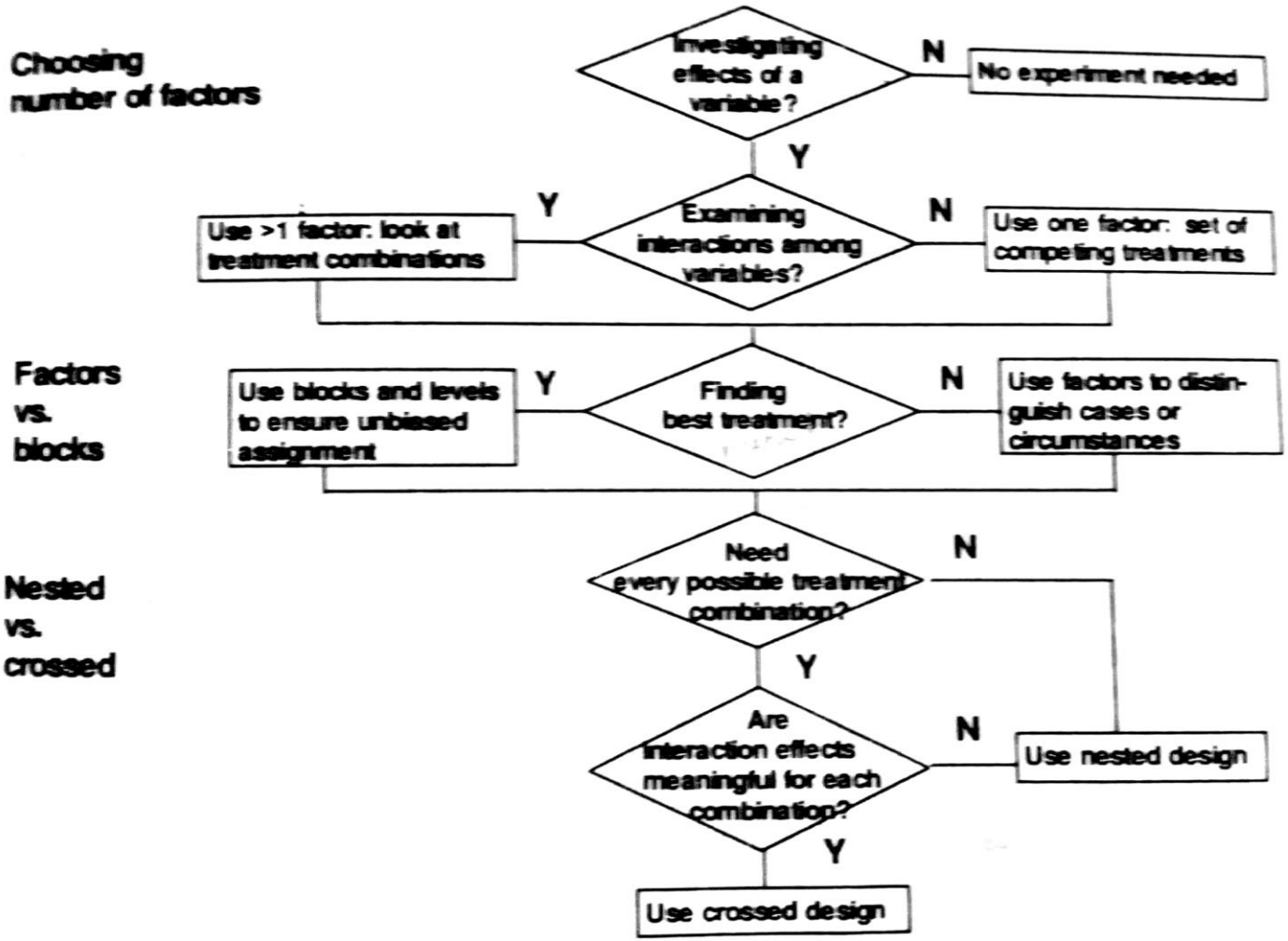




Selecting an experimental design

- Factors Vs Blocks
- Choosing between nested and crossed designs

Selecting an experimental design



Planning case studies

- Sister Projects
- Baselines
- Random Selection

References

TEXT BOOK-Software Metrics- Norman E Fenton

PPT-B.H. Far (far@ucalgary.ca)
http://www.enel.ucalgary.ca/~far/Lectures/SENG421/04/

Acknowledgment

Prof.B.H. Far

Department of Electrical & Computer Engineering, University of Calgary