Software Engineering Experimentation

Software Engineering Specific Issues
(Mostly CS as well)

Jeff Offutt

http://www.ise.gmu.edu/~offutt/

Software Engineering

1. The biggest obstacle to software engineering experimentation is that our <u>populations are unknown</u>
   - What is a representative collection of programs?
   - Faults?
   - Developers?
2. Second : Industry won't cooperate
   - In other engineering fields, companies provide access to data, resources, processes, and people
3. Third : "Knowledge inversion" – senior scientists often do not know as much about experimentation as younger scientists

1. Unknown Populations

- How many programs are enough for external validity?
- Are seeded faults as good as natural faults?
- Does using students bias the results?
- How do we analyze our results?

1. Statistical Tests and Software

- Experimental data based on programs cannot, with validity, be subjected to inferential statistical tests since the population is unknown
- An unknown population nullifies any statistical result that would be obtained, regardless of the number of programs
- Only descriptive statistics can be used
  - For example, log linear analysis
- That is, statistical hypothesis testing, at least in the statistical sense, is not accurate

2. Industry Cooperation

- Researchers need access to data from industry to know how techniques work in practice
- Two years ago, my student applied a high-end testing technique to real-time, safety-critical software, finding several bugs
  - She was refused permission to publish, because "customers might think our software is not perfect"
- Seven years ago, a former student applied mutation testing to Cisco's routing software, finding several bugs, one very severe, saving millions of dollars
  - $750,000 bonus!
  - Almost fired for telling me
  - Her boss asked me to sign a non-disclosure agreement, afterwards
- Very difficult to get research funding from industry

3. Knowledge Inversion

- Every "generation" of computer scientists has taken a step forward
  - '70s – '80s: No validation at all
  - '80s : We built systems
  - '80s – '90s : Results on small sets of data
  - '90s : Careful experimental design, larger data sets
  - 2000s : Sophisticated statistical analysis of results
- Many journal and conference reviewers do not have the knowledge to evaluate experiments

Confounding Variables in Software

A partial list of confounding variables, from previous courses, started Fall 1994

1. The Human Element
2. Scalability and Generality
3. Conduct of Experiment
4. Other

1. The Human Element

- Grant money
  - source
  - amount
- Motivation
  - Sponsor
  - Subject
- Capabilities of programmer
  - knowledge
  - experience
  - skills
  - birth order
  - IQ
  - status
  - handedness
  - personality traits
  - attention to detail
- Physical environment
- Feedback to subjects
- Subjects expectations
- Learning curve
- Amount and type of training
- Natural language
- Time of day experimentation conducted
- Group organization
- Communication
  - skills
  - type allowed
  - Structure
- Researchers
  - knowledge
  - experience
  - skills
- Culture of subjects

2. Scalability and Generality

- Size of project
- Application domain
- Programming language
- Number of artifacts/subjects
- Sampling of artifacts
- Source of artifacts
  - real or custom built
  - how created

3. Conduct of Experiment

- Duration of experiment
- Measure of artifact
- Support tools
- Specifications
- Hardware
- Support software
- Method of data collection
- Order of experimental process

4. Other

- Method
- Oracle quality
- Complexity of change
- Experimental design
- Direction of hypothesis

Principles to Follow

1. Improvement is through continuous, sustained change, not technological breakthrough
   - Scientists take baby steps
   - The "big step" is the last of many
   - OO and the Web were last of thousands of baby steps
2. Take great care in your data collection
   - Identify and control variables carefully
   - Document all decisions
   - Save all data – you may have to repeat the experiment years later

Principles to Follow (2)

3. Data collection is not the goal, analysis and application are the goals
   - Don't lose the forest in the trees
   - Conclusions matter, measurement does not
4. Data are uncertain and fallible – design experiments to be fault tolerant
   - Too many variables
5. Non-developers need to collect and analyze data
   - Developers' goal is the current product, not next
   - Research lab or university who can cooperate with company

Principles to Follow (3)

6. The goal of an experiment is to help companies develop better software, cheaper
   - The goal is NOT to publish a paper

