COMP5111 – Fundamentals of Software Testing and Analysis

Experiments – Traps and Tricks

Shing-Chi Cheung

Computer Science & Engineering

HKUST

Slides adapted from Steve Easterbrook's FSE-DS 2006 keynote speech http://www.cs.toronto.edu/~sme
Reference: Claes Wohlin et al., Experimentation in Software Engineering, Kluwer Academic Publishers, 2000.

Experiment results should be measurable

"We can't improve what we can't measure"

… John Penix, Google
@ICSE 2013 Panel Highlights

Where’s our contribution?

- Process
- Properties
- Problem solving
- Performance comparison

How should we validate our claims?

Where's our contribution?

A better understanding of how software engineers work? **Process**

- How do software engineers test deep learning systems?
- What is the practice of code mentoring in large scale projects?
- How are code reviews performed? Does the process differ between open-source and commercial projects?
- What are the major factors for successful open source projects?

How should we validate our claims?

Where's our contribution?

A characterization of the properties of new tools/techniques/data sources?

Properties

- Are mutation scores correlated with fault detection?
- Are code clones harmful?
- What are the effects of code obfuscations on the test generation of Android apps?
- Are code examples on online Q&A forum safe?

How should we validate our claims?

Where's your contribution?

Identification of problems with the current state-of-the-art?

- How to improve the coverage of Randoop? **Problem solving**
- How to locate missing faults using coverage based fault localization techniques?
- How to strengthen test suites using chosen patches?
- Precise concolic testing using symbolic calling contexts
- Automated testing of DNN-driven autonomous cars

How will you validate your claims?

Where's our contribution?

Performance comparison

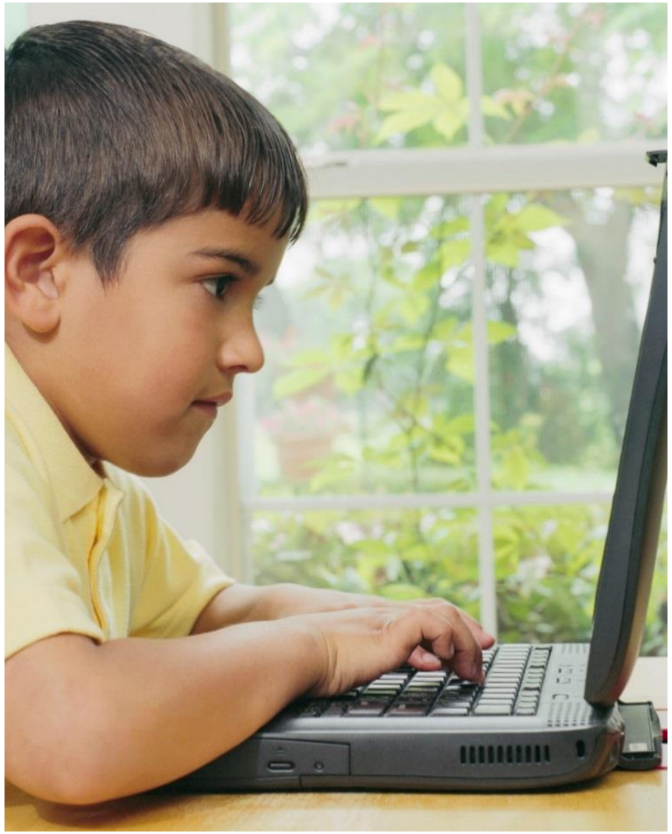
Evidence that approach A is better than approach B?

- Is symbolic analysis more effective than random testing?
- Is "Better Data" better than "Better Data Miners"?
- Testing versus code reviews
- Preserve or not to preserve invalid solutions in search-based program repair

How should we validate our claims?

Meet Donald Bean

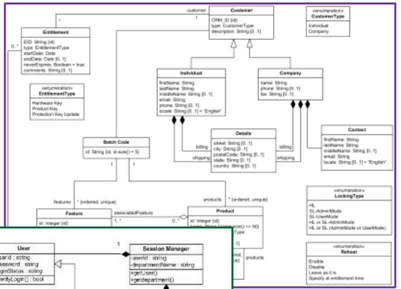
- Name:
  - Donald Bean (aka ‘Don’)
- Topic:
  - Merging Stakeholder views in Model Driven Development
- Status:
  - 2 years into PG study → PQE
  - Has built a tool
  - Needs an evaluation plan



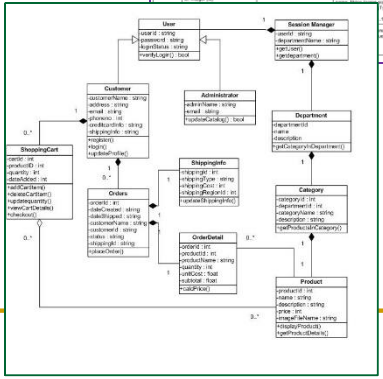
Don’s Evaluation Plan

- **Formal Experiment**
  - Independent Variable: Don-Merge vs. Rational Architect (RA)
  - Dependent Variables: Correctness, Speed, Assessment by human subjects
  - Task: Merging Class Diagrams from two different stakeholders’ models
  - Subjects: Grad Students in SE
  - H_1: “Don-Merge produces correct merges more often than RA”
  - H_2: “Subjects produce merges faster with Don-Merge than with RA”
  - H_3: “Subjects prefer using Don-Merge to RA”
- **Results**
  - H_1 accepted (strong evidence)
  - H_2 & H_3 rejected
  - Subjects found the tool unintuitive

M2:



M1:



What is your advice for Don?

- The topic is hopeless … Find another topic
- Forget the experiment … run it again … until …
- Replace RA with another tool, tell advisor that RA is not working
- Ask the advisor not to be so pushy …
- Tell the advisor that there is some bad news … postpone PQE, pleeease …
- Tell the advisor that there are promising results, but …
- Keep silent … Nothing has happened …

Threats to Validity

- **Construct Validity**
  - What do we mean by a merge? What is correctness?
  - 5-point scale for subjective assessment - insufficient discriminatory power
    - (both tools scored very low)
- **Conclusion Validity**
  - Inappropriate use of statistical techniques
- **Internal Validity**
  - Confounding variables: Time taken to learn the tool; familiarity
  - Subjects were all familiar with RA, not with Don-merge
  - Researcher bias: subjects knew Don-merge was Don's own tool
- **External Validity**
  - Task representativeness: class models were of a toy problem
  - Subject representativeness: Grad students as sample of what population?

Threats to Validity

- Construct Validity
  - What do we mean by a merge? What is correctness?
  - 5-point scale too subjective assessment - insufficient discriminatory power (both tools scored very low)

Clarity of problem formulation; Selection of variables and measurements

- Conclusion Validity
  - Inappropriate use of statistical techniques

Statistical and conclusion methodologies

- Internal Validity
  - Confounding variables: Time taken to learn the tool; familiarity
  - Subjects were all familiar with RA, not with Don-merge
  - Researcher bias: subjects knew Don-merge was Don's own tool

Existence of factors affecting variable dependency and measurement reliability

- External Validity
  - Task representativeness: class models were of a toy problem
  - Subject representativeness: Grad students as sample of what population?

Are subjects and tasks biased/representative

What went wrong?

- What was the research question?
  - "Is tool A better than tool B?"
- What would count as an answer?
  - "better" in terms of completion time, learning curve, accuracy, scalability, interoperability, portability, robustness or ... ?
- Why is the answer interesting?
  - How does it "contribute to knowledge"?
- How is this evaluation related to the existing literature?

Experiments as Clinical Trials

Why would we expect it to be better?

Why do we need to know?

What will we do with the answer?

Is drug A better than drug B?

Better at doing what?

Better in what way?

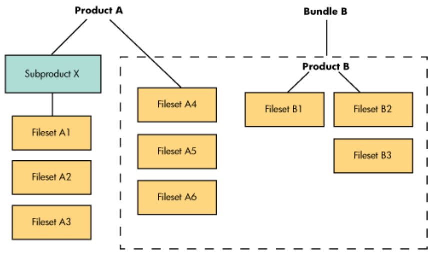
Better in what situations?

Why would we expect it to be better?

We gotta have a theory!

Science and Theory

- A (scientific) theory is:
  - More than just a description - it explains and predicts
  - Logically complete, internally consistent, experimentally refutable
  - Simple and elegant.
- Components of a theory:
  - Concepts & laws which state the relation of concepts
    - E.g. Conway's Law - structure of software reflects the structure of the team that builds it. A theory should explain why.
- Theories lie at the heart of what it means to do science.
  - Production of generalizable knowledge
- Theories provide orientation for data collection
  - Cannot observe the world without a theoretical perspective



Theories are good for generalization…

Statistical Generalization

- First level generalization:
  - From samples to population
- Well understood methodology and widely used in empirical studies
- Can only be used for quantifiable variables
- Based on random sampling:
  - Standard statistical tests check if results on samples apply to the whole population
- Not useful when:
  - We can't characterize the population
  - We can't sample randomly or by objective criteria
  - We can't get enough data points

Analytical Generalization

- Second level generalization:
  - From findings to theory (e.g., fault localization)
- Applicable to quantitative and qualitative studies
- Compares findings with theory
  - Do the data support or refute the theory?
  - Or: do they support this theory better than rival theories? (e.g., spectrum FL vs statistical FL)
- Supports empirical induction:
  - Evidence builds if subsequent studies also support the theory (& fail to support rival theories)
- More powerful than stats
  - Doesn't rely on correlations
  - Examines underlying mechanisms

Examples of Uninteresting Theory

Software defect prediction

- Program code written on Friday tends to be buggy.
- Modules with more code written on Friday more likely contain defects.
- Program code written during World Cup Finals tends to be buggy.
- Program code written in a rainy day tends to be buggy.
- Programs written by developers in a crowded office are more buggy.

Don’s Theory

- **Background & Assumptions** (introduction & background sections)
  - Large team projects, models contributed by many actors
  - Models are fragmented, each captures a partial view
  - Partial views are inconsistent and incomplete most of the time

Either well known or requires validation using real project data

- **Basic Theory** (methodology section)
  - (Brief summary:)
  - **Model merging** is an exploratory process that aims to **discover** intended **relationships** between **views**. ‘**Goodness**’ of a merge is a **subjective judgment**.
  - **Why interesting:** If an attempted merge doesn’t seem ‘good’, many need to change either the models, or the way in which they were mapped together.
- **Supplementary Theories**
  - Useful merge tools need to represent relationships explicitly
  - Useful merge tools need to be complete (work for any models, even if inconsistent)

Don’s Theory

- **Basic Theory**
  - (Brief summary:)
  - **Model merging** is an exploratory process that aims to **discover** intended **relationships** between **views**. ‘**Goodness**’ of a merge is a **subjective judgment**.
  - **Why interesting**: If an attempted merge doesn’t seem ‘good’, many need to change either the models, or the way in which they were mapped together.
- **Anatomy**
  - What do we mean by a merge?
  - What do we mean by correctness? // quantifiable?
  - Are we missing the target? // in line with background and assumption?
  - Is “goodness” measurable? How should we measure ‘goodness’?
  - What data can we collect?
  - Can the ground truth of intended relationships be collected? // collection process

Important questions in building scientific knowledge

What type of questions are we asking?

- Existence:
  - Does X exist?
  - How often does X exist?
- Description & Classification
  - What is X like?
  - What are its properties?
  - How can it be categorized?
  - How can we measure it?
  - What are its components?
- Descriptive-Process
  - How does X work?
  - What is the process by which X h
  - In what are the steps as X evolve
  - How does X achieve its purpose?
- Descriptive-Comparative
  - How does X differ from Y?

Do code clones exist?

What are performance bugs like?

How does code review work?

- Relationship
  - Are X and Y related?
  - Do occurrences of X correlated wi
- Causality
  - Does X cause Y?
  - Does X prevent Y?
  - What causes X?
  - What effect does X have on Y?
- Causality-Comparative
  - Does X cause more Y than does Z?
  - Is X better at preventing Y than is Z?
  - Does X cause more Y than does Z under one condition but not others?

Design
  - What is an effective way to achieve X?
  - How can we improve X?

Are real faults coupled with mutation faults?

Causes for Android bugs induced by WebView?

How do flaky tests differ from failing tests?

Don's Research Question(s)

Model merging is an exploratory process that aims to discover intended relationships between views. 'Goodness' of a merge is a subjective judgment.

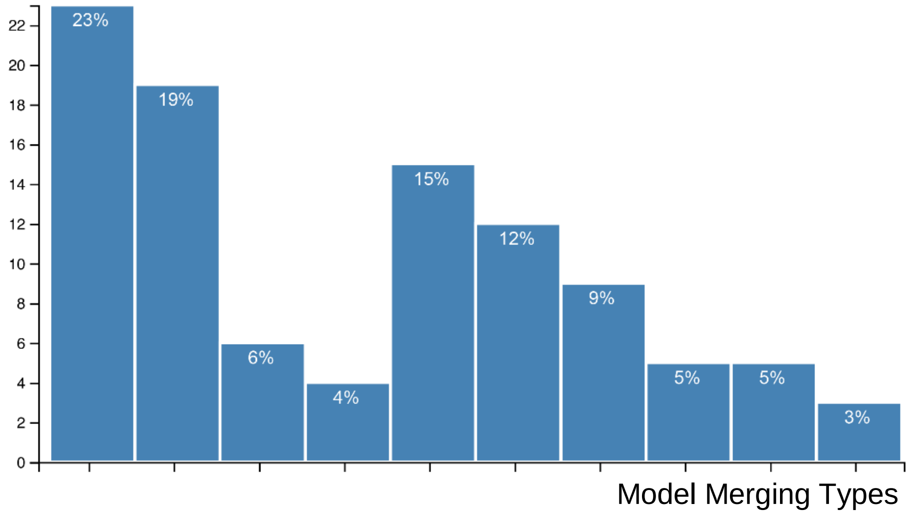
Existence

- **Does model merging ever happen in practice?**
- We normally show the demographics based on historical data.
- E.g., To show the existence of null-pointer problem in practice:
  - *"1,340,561 (82.6%) out of the 1,622,375 code revisions of IF-clauses filed at GitHub as at Sept 2015 involve null-pointer checks."*
- E.g., To show the existence of incomplete code fixing:
  - *"Over 50% of closed bug reports are re-opened in SourceForge projects."*
- If historical data are unavailable, we may need to conduct a user survey.

Don's Research Question(s)

Model merging is an exploratory process that aims to **discover** intended **relationships** between **views**. 'Goodness' of a merge is a subjective judgment.

- Description/Classification
  - **What are the different types of model merging that occur in practice on large scale systems?**
  - We can use bar charts to show the distribution based on historical data.
  - Threats: Model types may not be all disjoint.



Don's Research Question(s)

Model merging is an exploratory process that aims to **discover** intended **relationships** between **views**. 'Goodness' of a merge is a subjective judgment.

- Descriptive-Comparative
  - **How does model merging with explicit representation of relationships differ from model merging without such representation?**
  - Can be conducted in the form of case studies under industrial settings
  - Useful to discover or motivate other research questions

Don's Research Question(s)

Model merging is an exploratory process that aims to **discover** intended **relationships** between **views**. 'Goodness' of a merge is a subjective judgment.

Causality

- Does an explicit representation of the relationship between models cause developers to explore different ways of merging models?

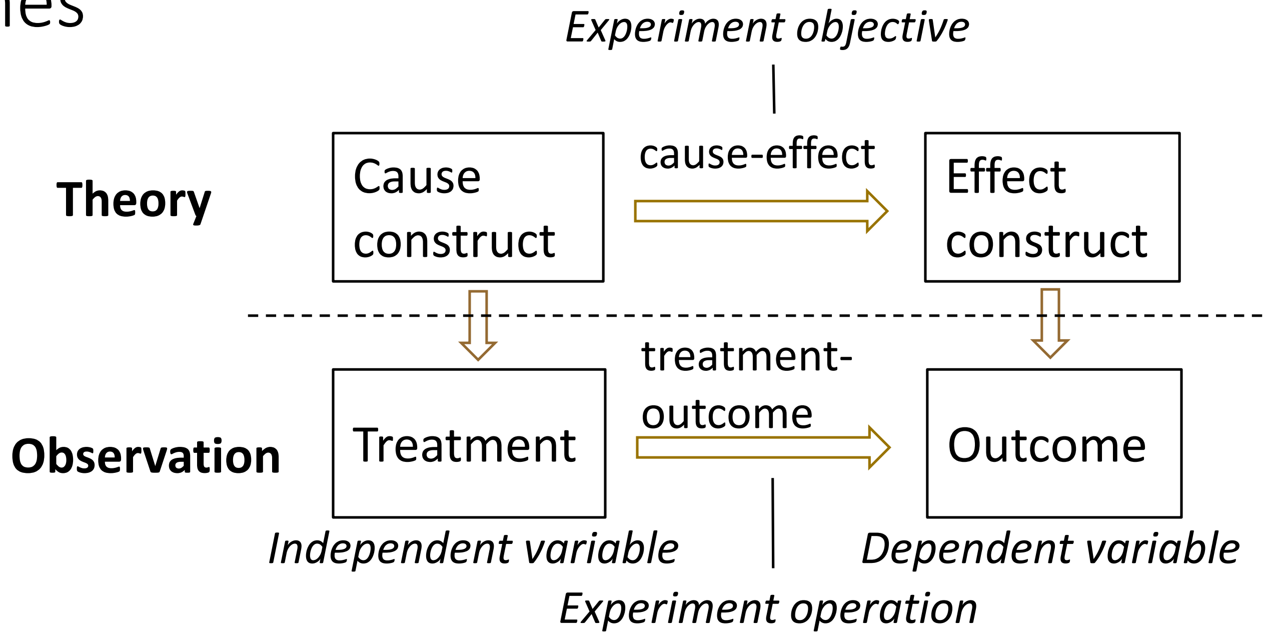
- Causality-Comparative
  - Does the algebraic representation of relationships in Don's tool lead developers to explore more than do relationships represented by Object Constraint Language (OCL) in RA?

May draw conclusion using hypotheses.

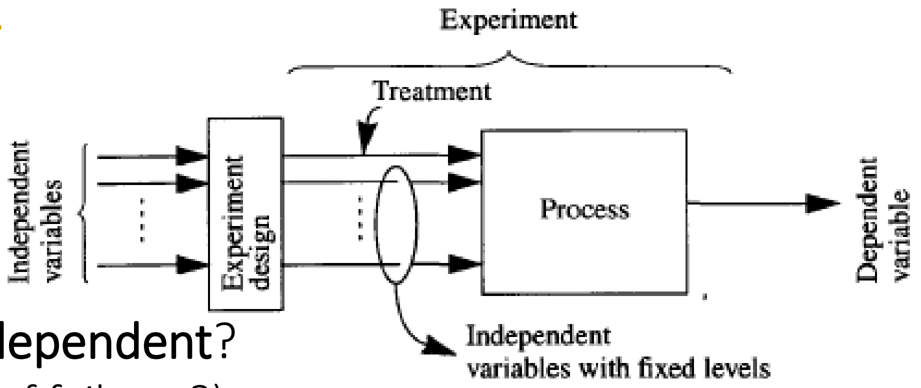
- Better coupled with explanation and discussion of corner cases.

Experiments

- We confirm a relation between treatments and outcomes



Variables



- What are the **variables**?
- Which of them are **independent** and **dependent**?
  - Number of faults (how about number of failures?)
  - Method, experience of subjects, familiarity with the software, environment, tool support, learning curve, team size, applications, …
- Focus on **one dependent variable** in each hypothesis
- All **independent variables** are **manipulated** and **controlled**
  - Variables that are manipulated to assume different values are **factors**
  - A **treatment** is a value that can be assumed by a factor
  - Variables that are controlled to remain unchanged are **control variables**
- State a theory, e.g., a cause effect relation between independent variables and a dependent variable, formally in a hypothesis.

Treatment

- A treatment can be applied to a combination of subjects and objects
  - Subjects apply a treatment upon objects
    - E.g., Experienced developers apply method X to developing a bank account system. // dep var: correctness
  - We apply a treatment upon subjects
    - E.g., We apply Randoop to generating tests for Hadoop project. // dep var: #faults detected
- An experiment consists of a set of tests (or trials), each of which is a combination of treatment, subjects and objects.

Don's Research Question(s)

Theory

Model merging is an exploratory process that aims to **discover** intended **relationships** between **views**.

- Existence
  - Does model merging ever happen at large systems?
- Description/Classification
  - What are the different types of model merging that occur in practice on large scale systems?
- Descriptive-Comparative
  - How does model merging with explicit representation of relationships differ from model merging without such representation? // subjects not explicitly specified
- Causality
  - Does an explicit representation of the relationship between models cause developers to explore different ways of merging models?
- Causality-Comparative
  - Does the algebraic representation of relationships in Don's tool lead developers to explore more than do relationships represented by Object Constraint Language (OCL)?

Controlled Experiments

Experimental investigation of a testable hypothesis, in which conditions are set up to isolate the variables of interest ("independent variables") and test how they affect certain measurable outcomes (the "dependent variables")

- good for
  - quantitative analysis of benefits of a particular tool/technique
  - establishing cause-and-effect in a controlled setting
  - (demonstrating how scientific we are!)

Is this research question subject to controlled experiments?

- Project efforts can be reduced by 30% by deploying fuzz tests using Randoop.

Controlled Experiments

Experimental investigation of a testable hypothesis, in which conditions are set up to isolate the variables of interest ("independent variables") and test how they affect certain measurable outcomes (the "dependent variables")

- limitations
  - hard to apply if you cannot simulate the right conditions in the lab
  - uncertain if the laboratory setup reflects the real situation
  - ignores contextual factors (e.g. social/organizational/political factors)
  - extremely time-consuming!

E.g., Project cost estimation, cloud computing, crowd debugging, …

E.g., The effect of method X to the number of faults found in testing

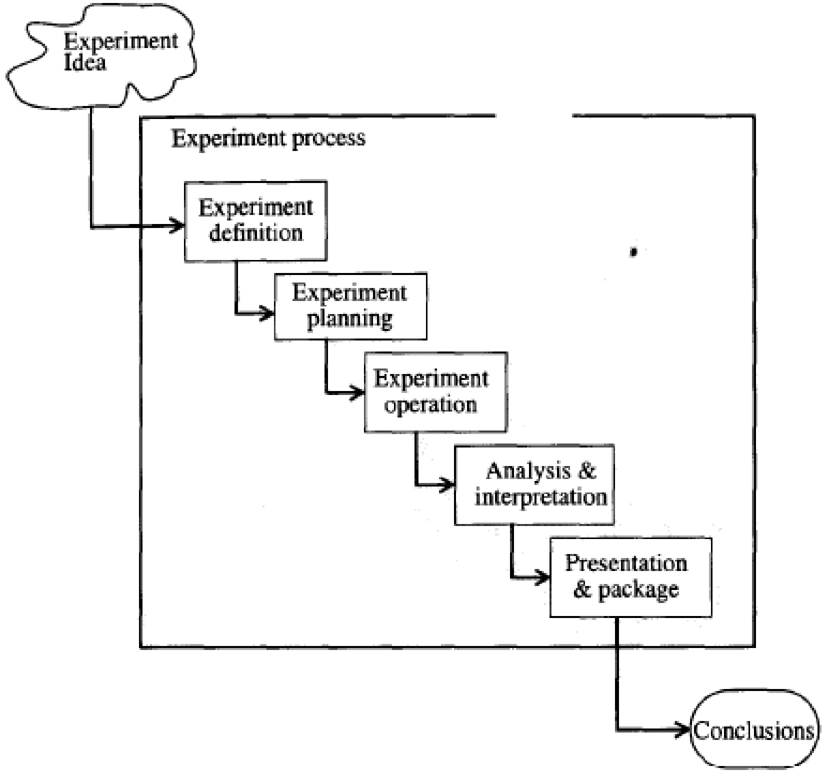
Pfleeger, S.L.; Experimental design and analysis in software engineering. Annals of Software Engineering 1, 219-253. 1995

Controlled Experiments

- Suitable when there are only a few factors (independent variables), which can be fully controlled.
- Effects can be observed relatively shortly.
- Normally used to confirm a theory or to explore a relation
- Methodology: definition -> planning -> operation -> analysis and interpretation -> presentation and package.

E.g., The effect of method X to the number of faults found in testing

Overview of a Controlled Experiment



Definitions

- Object of study (what is studied?)
  - Method / process / performance/ property / metrics / theory ?
- Purpose (What is the intention?)
  - Evaluate the performance of two different techniques
  - Characterize the effects of design patterns in the software maintenance process
- Quality focus
  - Effectiveness / cost / reliability / fault rates
- Perspective (optional)
  - Viewpoint from which the results are interpreted

Definitions

- Context (where is the study conducted?)
  - Environment in which the experiment is run
  - Defines subjects who carry out the experiment
    - Characterized by experience, team size, knowledge
  - Defines subjects/objects (i.e., software artifacts) that are used in the experiment
    - Characterized by size, complexity, priority, application domain, …

Definitions

Object of study	Purpose	Quality focus	Perspective	Context
Product Process Property Performance Model Metric Theory	Characterize Monitor Evaluate Predict Control Change	Effectiveness Cost Reliability Maintainabilit y Portability	Developer Modifier Maintainer Project manager Corporate manager Customer User Researcher	Subjects Objects

- *Definition template*:

  This experiment is to analyze **<object of study>** for the purpose of **<purpose>** with respect to **<quality focus>** from the point of view of **<perspective>** in the context of **<context>**.

Planning

- State hypotheses formally
  - Null hypothesis
  - Alternative/Rival hypotheses (rival hypothesis is common in case studies)
- Determine independent and dependent variables
- Identify possible treatments and outcomes
  - Determine measurement scale (should not be too coarse, e.g., complex vs simple programs)
  - Identify subjects and objects
    - Selection criteria
- Design experiments
- Evaluate threats to validity

Planning – Hypothesis Formulation

- Two hypotheses have to be formulated:
- A null hypothesis, H0:
  - There are no real underlying trends or patterns in the experiment setting; the only reasons for differences in our observations are coincidental.
  - A hypothesis that we want to reject with high significance (p-value < 0.05).
  - *Example: There is no significant difference between the number of faults detected by method X as compared with that by the random approach.*
- An alternative hypothesis, H1:
  - This is the hypothesis in favor when H0 is rejected.
  - *Example: There is significant difference between the number of fault detected by method X as compared with that by the random approach.*

Planning – Risk in hypothesis testing

- **Type-I-error**: It occurs when a statistical test has indicated a pattern or relationship even if there actually is no real pattern.
  - P(type-I-error) = P(reject H0 | H0 is true) ← p-value
  - The significant level α is the highest p-value we accept for rejecting H0.
  - The normal value of α is 0.05. What if we obtain a p-value of 0.054?
- **Type-II-error**: It occurs when a statistical test has not indicated a pattern or relationship even if there actually is a real pattern.
  - P(type-II-error) = P(not reject H0 | H0 is false)

Cohen, J.: Statistical Power Analysis for the Behavioral Sciences. 1988.

Risk in Hypothesis Testing

- **Power**: The power of a statistical test is the probability that the test will reveal a true pattern if H0 is false. Power should be as high as possible.
  - Power = P(reject H0 | H0 is false) = 1 – P(type-II-error)
  - Greater than 0.80 as a standard for adequacy.
  - Its value depends on which statistical tests, number of subjects and significance level

Cohen's d-index (Effect Size)

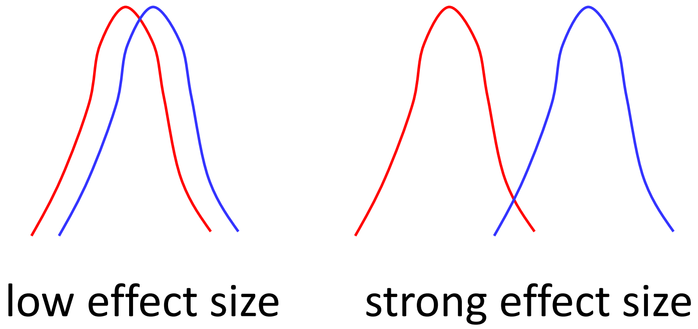
- Weakness of p-value
  - It measures how unlikely that the null hypothesis is true (< 0.5%)
  - But it does not directly measure the effects made by two treatments
- Effect size d

$$d = M_1 - M_2 / SD$$
$$SD^2 = (SD_1^2 + SD_2^2)/2$$

Assuming two groups have the SD_1 and SD_2 are comparable

Calculator: https://www.socscistatistics.com/effectsize/default3.aspx

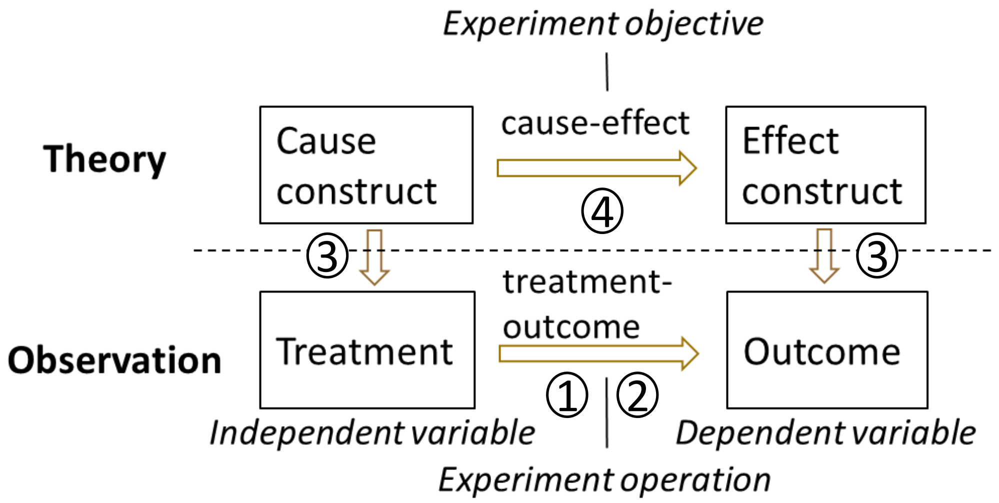
https://www.simplypsychology.org/effect-size.html



Relative size	Effect size (d)
Very small	0.01
Small	0.2
Medium	0.5
Large	0.8
Very large	1.2

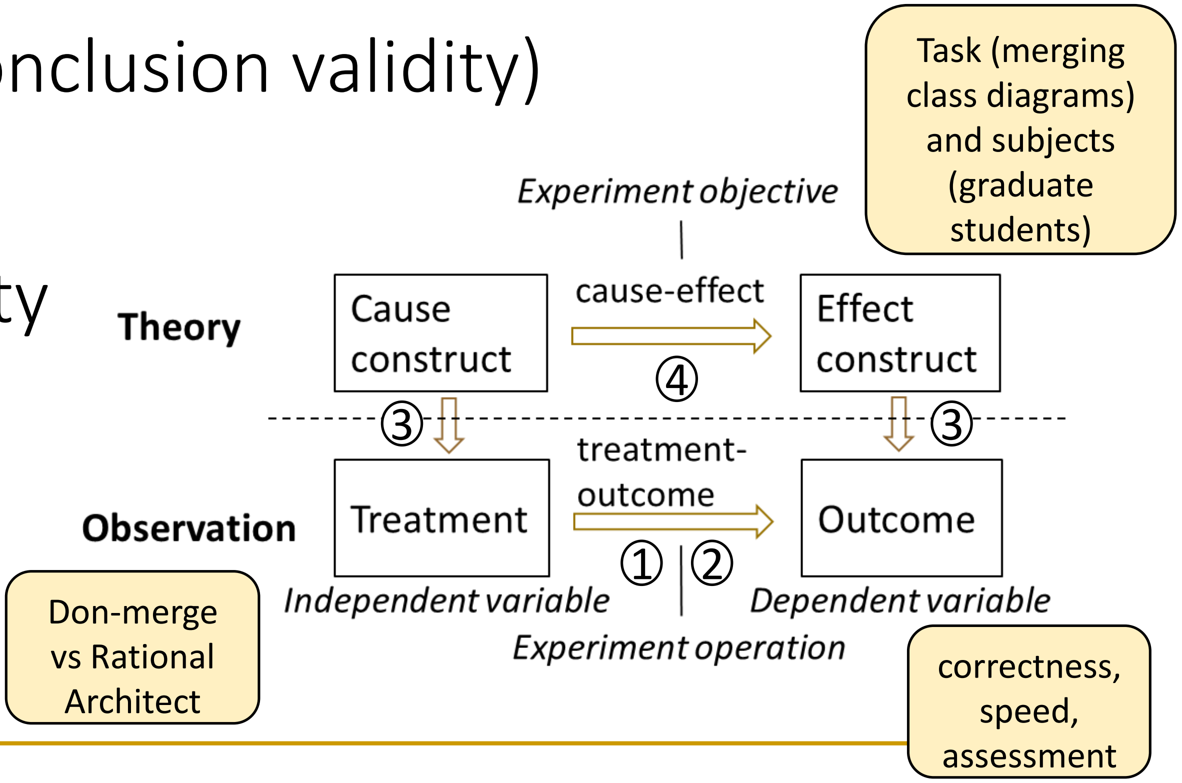
Planning – Threat to Validity

1. Reliability (or conclusion validity)
2. Internal validity
3. Construct validity
4. External validity

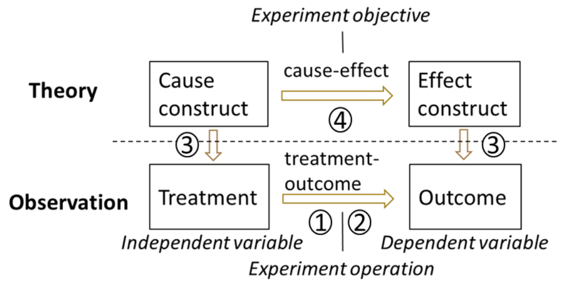


Planning – Threat to Validity

1. Reliability (or conclusion validity)
2. Internal validity
3. Construct validity
4. External validity

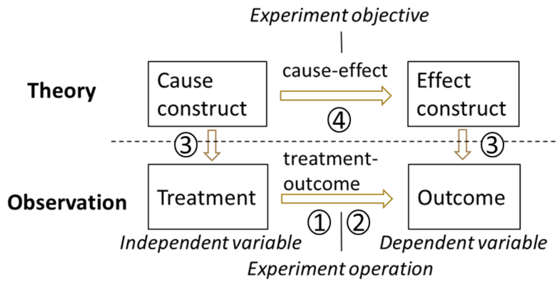


Planning – Threat to Validity



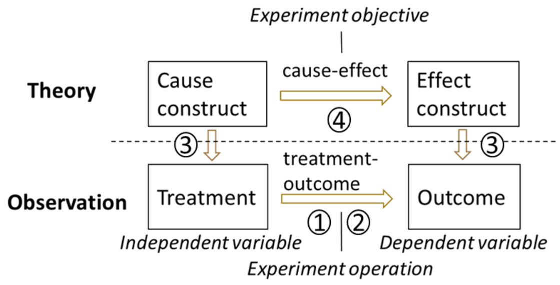
- Construct Validity ③
  - Do the treatments and outcomes faithfully measures the constructs
    - Theoretical concepts are operationalized and measured correctly.
    - Are we measuring the construct we intended to measure?
    - Have we translated these constructs correctly into observable measures?
    - Do the metrics we use have suitable discriminatory power?
  - Examples
    - What do we mean by a merge? What are the criteria for correctness?
    - Would 0-5 point scale for subjective assessment provide sufficient discriminatory power
      - The scores for both tools are low

Planning – Threat to Validity



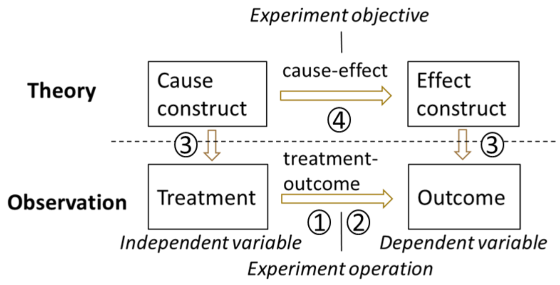
- **Internal Validity ②**
  - Do the outcomes really follow from the treatments (concerns var dependency)
    - Do the results really follow from the data? Existence of confounding variables.
    - Have we properly eliminated confounding variables?
  - Example of confounding variables
    - Time taken to learn the tool (subjects were all familiar with RA, not with Don-merge)
    - Don was the TA of the course taken by the subjects

Planning – Threat to Validity



- Reliability (or conclusion validity) ①
  - Are the conclusions made on treatment-outcome reliable?
    - Is the study repeatable with the consistent results?
    - Have we eliminated all researcher biases?
    - Have we used the right test? Parametric vs Non-parametric
  - Example
    - Don selected subjects that he is familiar with
    - Don did not select certain kind of subjects without justification

Planning – Threat to Validity

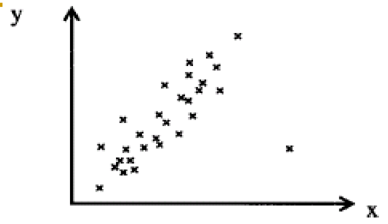


- External Validity ④
  - **Is the observation (treatment-outcome) generalizable to theory (cause-effect)?**
    - Are the findings generalizable beyond the immediate study?
    - Do the results support the claims of generalizability?
  - **Task is not representative:** class diagram models used in the experiment were toy problems
  - **Subjects are not representative:** unclear which population samples were represented by the graduate students

Operation

- Preparation
  - Prepare subjects
    - Training? Consent from subjects to conduct the experiment? Do subjects need to know the intention? …
  - Prepare objects
    - Data collection form? Collection method, e.g., time? Permission from the company/organizations?
  - Data reliability
    - How to ensure the data collected is reliable and valid?
    - Are the data collected relevant to the experiment?

Analysis and Interpretation



- Analysis
  - Use descriptive statistics to get informal understanding of the data (bar chart, variance, mean, median, …)
  - Data set reduction
    - Remove invalid data points
    - Reduce the number of factors that provide similar information
  - Perform hypothesis test on the reduced data set
    - Could conduct multiple tests to increase reliability of the analysis
- Interpretation
  - Whether the analysis lead to acceptance or rejection of the hypothesis?
  - Can it be related to previous published results?
    - Generalization of previous concrete results
    - Refinement of previous inconclusive results

More treatments or factors
More independent variables

Presentation and Package

- Documentation of results
  - Research paper
  - Lab package to replicate the experiment later
  - Archiving of data for future use
  - Documentation of tools built in the experiment
- Lessons learnt
- Make the experimental results available online
- Release the subjects/tools to the research community

Why Build a Tool?

- Build a Tool to Test a Theory
  - Tool is part of the experimental materials needed to conduct your study
- Build a Tool to Develop a Theory
  - Theory emerges as you explore the tool
- Build a Tool to Explain your Theory
  - Theory as a concrete instantiation of (some aspect of) the theory

Take home messages

Articulate the theory(s) underlying your work

Be precise about your research question

Be explicit about your philosophical stance

Use the theory to guide the study design

Validate the Theory not the Tool

Illustration: Evaluate the effectiveness of search-based software testing

Gordon Fraser and Andrea Arcuri, Sound Empirical Evidence in Software Testing, ICSE 2012.

Definition

- This experiment is to analyze search-based software testing for the purpose of its effectiveness with respect to branch coverage from the point of view of researchers in the context of open source software.

Planning

- RQ1: What is the probability distribution of achievable branch coverage on open source software? *Description/Classification*
- RQ2: How often can classes execute unsafe operations? *Existence*
- RQ3: What are the consequences of choosing a small case study in a biased way? *Causality*

Preparation

- Select EvoSuite as the evaluation platform to represent search-based software testing tool.
- Prepare a data set based on an unbiased sample of 100 SourceForge open source projects.

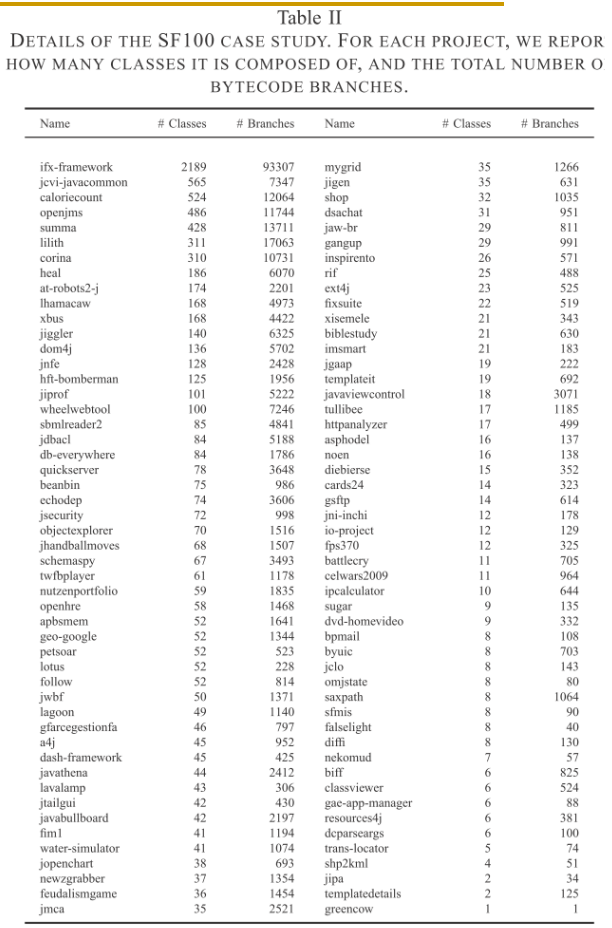


Table II
DETAILS OF THE SF100 CASE STUDY. FOR EACH PROJECT, WE REPOR
HOW MANY CLASSES IT IS COMPOSED OF, AND THE TOTAL NUMBER O
BYTECODE BRANCHES.

Name	# Classes	# Branches	Name	# Classes	# Branches
ifx-framework	2189	93307	mygrid	35	1266
jcvi-javacommon	565	7347	jigen	35	631
caloriecount	524	12064	shop	32	1035
openjms	486	11744	dsachat	31	951
summa	428	13711	jaw-br	29	811
lilith	311	17063	gangup	29	991
corina	310	10731	inspirento	26	571
heal	186	6070	rif	25	488
at-robots2-j	174	2201	ext4j	23	525
lhamacaw	168	4973	fixsuite	22	519
xbus	168	4422	xisemele	21	343
jiggler	140	6325	biblestudy	21	630
dom4j	136	5702	imsmart	21	183
jnfe	128	2428	jgaap	19	222
hft-bomberman	125	1956	templateit	19	692
jiprof	101	5222	javaviewcontrol	18	3071
wheelwebtool	100	7246	tullibee	17	1185
sbmlreader2	85	4841	httpanalyzer	17	499
jdbacl	84	5188	asphodel	16	137
db-everywhere	84	1786	noen	16	138
quickserver	78	3648	diebierse	15	352
beanbin	75	986	cards24	14	323
echodep	74	3606	gsftp	14	614
jsecurity	72	998	jni-inchi	12	178
objectexplorer	70	1516	io-project	12	129
jhandballmoves	68	1507	fps370	12	325
schemaspy	67	3493	battlecry	11	705
twfbplayer	61	1178	celwars2009	11	964
nutzenportfolio	59	1835	ipcalculator	10	644
openhre	58	1468	sugar	9	135
apbsmem	52	1641	dvd-homevideo	9	332
geo-google	52	1344	bpmail	8	108
petsoar	52	523	byuic	8	703
lotus	52	228	jclo	8	143
follow	52	814	omjstate	8	80
jwbf	50	1371	saxpath	8	1064
lagoon	49	1140	sfmis	8	90
gfarcegestionfa	46	797	falselight	8	40
a4j	45	952	diffi	8	130
dash-framework	45	425	nekomud	7	57
javathena	44	2412	biff	6	825
lavalamp	43	306	classviewer	6	524
jtailgui	42	430	gae-app-manager	6	88
javabullboard	42	2197	resources4j	6	381
fim1	41	1194	dcparseargs	6	100
water-simulator	41	1074	trans-locator	5	74
jopenchart	38	693	shp2kml	4	51
newzgrabber	37	1354	jipa	2	34
feudalismgame	36	1454	templatedetails	2	125
jmca	35	2521	greencow	1	1

RQ1 - Results

- There are 8,784 classes and 291,639 branches.
- 122 days to run experiments.

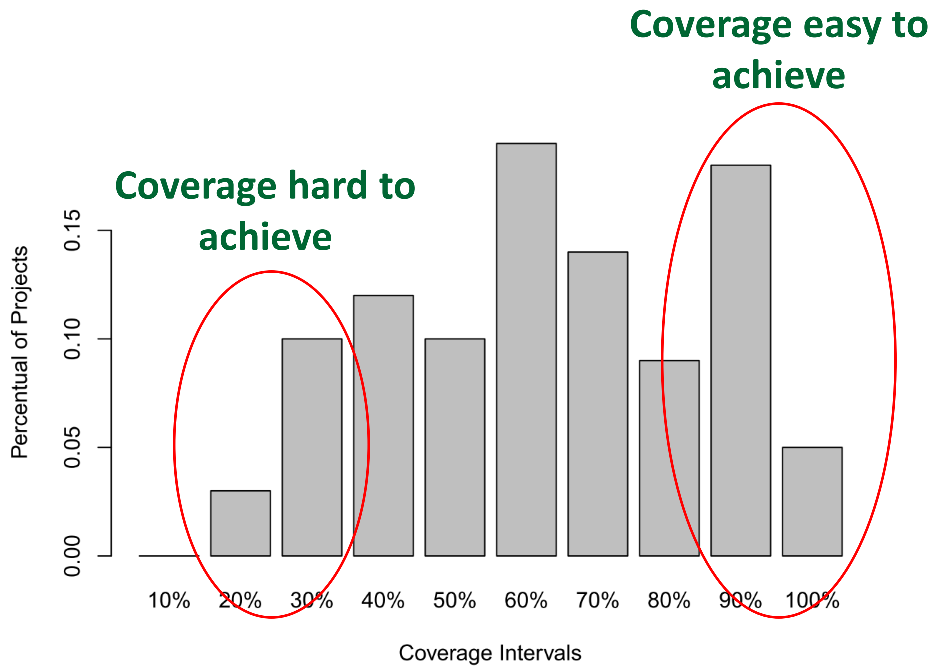


Figure 1. For each 10% code coverage interval, we report the proportion of projects that have an average coverage (averaged out of 10 runs on all their classes) within that interval. Labels show the upper limit (inclusive). For example, the group 40% represent all the projects with average coverage greater than 30% and lower or equal to 40%.

RQ1 - Results

- There are about one third of classes whose coverage are easy to achieve.
- There are about one third of classes whose coverage are hard to achieve.

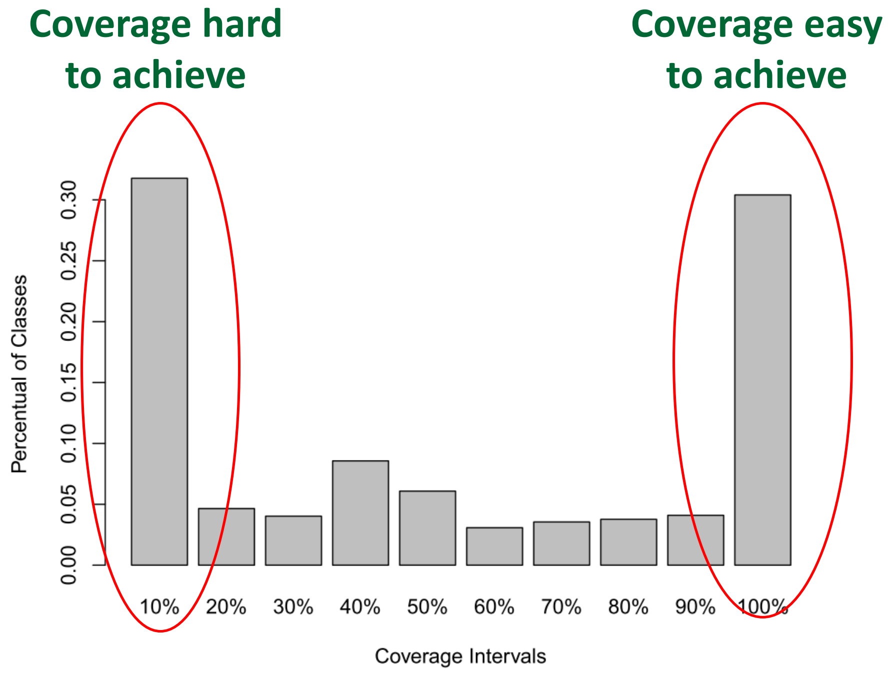


Figure 2. For each 10% code coverage interval, we report the proportion of classes that have an average coverage (averaged out of 10 runs) within that interval. Labels show the upper limit (inclusive). For example, the group 40% represent all the classes with average coverage greater than 30% and lower or equal to 40%.

RQ1 - Results

- The branches of classes involving multi-threading are not necessarily more difficult to cover.
- *Comment: May be accompanied by hypothesis testing*

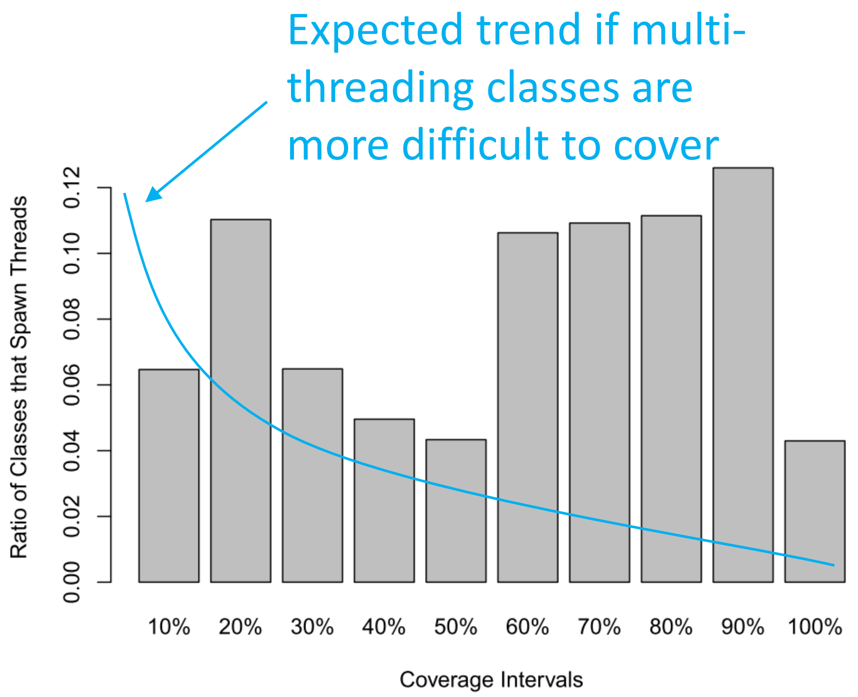


Figure 4. Average number of threads for classes within each 10% code coverage interval: Multi-threaded code does not per se inhibit coverage.

Analysis

- RQ1: What is the probability distribution of achievable branch coverage on open source software?
- Observation: There is a large number of classes that can easily be fully covered by EvoSuite (coverage 90%~100%), and also a large number of classes with problems (coverage 0%~10%), while the rest is evenly distributed across the 10%~90% range.

Distribution is somewhat bipolar

Analysis

- RQ2: How often can classes execute unsafe operations?
- Observation: Interactions with the environment are a prime source of problems in achieving high coverage. It is striking that the execution of 71% of all classes led to some kind of FilePermission.

Analysis

- RQ3: What are the consequences of choosing a small case study in a biased way?
- Observation: Experiments show that 90.7% of classes may lead to interactions with their environment. When there are no interactions with the environment EvoSuite can achieve an average coverage as high as 90%.

Follow-up Analysis

- Manual inspection of 40 classes:
  - 10 classes that had low coverage but no permission problems
  - 10 classes that had file permission problems
  - 10 classes that had network problems
  - 10 classes with runtime permission problems

Classes are randomly sampled in each category

Main reasons for low coverage

- 10 classes that had low coverage but no permission problems
  - Complex string handling
  - Java generics and dynamic type handling
  - Branches in exception handling code
  - Other problems were linking up with the environment for testing classes as the major technical obstacle

Main reasons for low coverage

- 10 classes that had file permission problems
  - Branches do not necessarily depend on file contents, but sometimes on file existence or file names
  - Branches are usually followed by code that manipulates files
  - When code tries to read property files, such files need to exist and contain appropriate content for testing
  - Setting up a suitable file environment for test classes is a major obstacle

Main reasons for low coverage

- 10 classes that had network problems
  - NetPermissions were more frequent than SocketPermissions
  - NetPermissions were mainly induced by the generation of invalid URLs that the program does not have access to
    - E.g. file:/foo/fum/
    - Likely propagate to URL from the random strings generated by Evosuite
  - NetPermission checks are very similar to FilePermission checks concerning I/O

Main reasons for low coverage

- 10 classes with runtime permission problems
  - Most due to code trying to set up a graphical user interface
  - Most GUI applications try to access files (e.g., .accessibility.properties)
  - Coverage does not increase even granting permissions to load libraries
  - Most prominent permission problem is exitVM.0, which is required to shut down the running virtual machine
  - Remaining problems include actions on running threads, loading libraries, queueing printer jobs, changing security manager.

Threats to validity

- Statistics could be affected by extreme outliers
  - Distribution may not be normal
  - There are some extreme outliers in terms of #classes per project or #branches per class in 100 selected projects
  - But the extreme outliers do not particular affect median values
  - Randomness of Evosuite → run each experiment 10 times
- Might not generalize to all open source projects
  - Software in Google code and other repositories may follow different distribution properties (e.g., number of classes per project)

Conclusion

- 23 projects for which EvoSuite achieves on average a coverage higher than 80%.
  - Small sampling based on these 23 projects instead of using 100 projects can lead to biased experiment results
- EvoSuite on average achieves 48% branch coverage
- Invite others to challenge EvoSuite with its released dataset SF100 corpus

Further References

- Victor R. Basili, Richard Selby and David Hutchens, Experimentation in Software Engineering, IEEE TSE, July 1986 (invited paper). [a key paper introducing experimentation to software engineering]
- Barbara A. Kitchenham et al., Preliminary Guidelines for Empirical Research in Software Engineering, IEEE TSE 28:8, Aug 2002, pp. 721-734.
- Dag I.K. Sjoberg et al., Conducting Realistic Experiments in Software Engineering, Proc. of 2002 International Symposium on Empirical Software Engineering.
- Tore Dyba, Evidence-Based Software Engineering for Practitioners, IEEE Software, Jan/Feb 2005, pp. 58-65.
- Amela Karahasanovic et al., Collecting Feedback During Software Engineering Experiments, Empirical Software Engineering 10, 2005, pp. 113-147.
- Martin Host et al., Experimental Context Classification: Incentives and Experience of Subjects, ICSE 2005, pp. 470-478.
- Jo E. Hannay et al., A Systematic Review of Theory Use in Software Engineering Experiments, TSE 33:2, Feb 2007, pp. 87-107.
- Barbara A. Kitchenham et al., Case Studies for Method and Tool Evaluation, IEEE Software, July 1995, pp. 52-62. [A nice template for simple case study]
- Andreas Zendler, A Preliminary Software Engineering Theory as Investigated by Published Experiments, Empirical Software Engineering 6, 2001, pp. 161-180. [Good review of software experimentation history]
- Natalia Juristo et al., Reviewing 25 Years of Testing Technique Experiments, Empirical Software Engineering 9, 2004: pp. 7-44. [archive analysis example]
- M. Aggrawal and K. Chari, Software Effort, Quality, and Cycle Time: A Study of CMM Level 5 Projects, TSE 23(3), March 2007, pp. 145-156. [archive analysis example]
- Erik Arisholm et al., Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise, IEEE TSE 33:2, Feb 2007, pp. 65-86. [an experiment example]
- Norman E. Fenton and Niclas Ohlsson, Quantitative Analysis of Faults and Failures in a Complex Software System, TSE 26:8, Aug 2002, pp. 797-814. [case study example]
- Claes Wohlin and Anneliese Andrews, Prioritizing and Assessing Software Project Success Factors and Project Characteristics using Subjective Data, Empirical Software Engineering 8, 2003, pp. 285-308. [case study eg]
- Gregg Rothermel et al., On Test Suite Composition and Cost-Effective Regression Testing, TOSEM 13 (3), July 2004, pp. 277-331. [an experiment example based on multiple factors and treatments – Randomized Factorial Design]
- Ian Sommerville and Jane Ransom, An Empirical Study of Industrial Requirements Engineering Process Assessment and Improvement, TOSEM14 (1), Jan 2005, pp. 85-117. [case study example]
- Z. Li, M. Harman and R.M. Hierons, Search Algorithms for Regression Test Case Prioritization, TSE 33 (4), Apr 2007, pp. 225-237. [an experiment example based on multiple factors – use ANOVA]
- Cemal Yilmaz et al., Main Effects Screening: A Distributed Continuous Quality Assurance Process for Monitoring Performance Degradation in Evolving Software Systems, ICSE’05, May 15-21, St Louis, Missouri, USA, pp. 294-302. [fractional factorial design]

Motivation
Software fault and RIP model
Structural coverage
Code coverage
Code instrumentation
Unit testing, parameterized unit tests
Mutation testing
Regression testing

Pattern-based analysis
Random test generation
Symbolic execution/concolic testing
Search-based test generation
Automated fault localization
Automated program repair
Pointer analysis
Slicing and tainting
Vulnerability and fuzz testing
Mining public software repository
Experimentation

END OF CLASS LECTURES!