Object-Oriented Measurement

Khaled El Emam

National Research Council of Canada

Agenda

- Definition of Object-Oriented Measures
- Theory Behind Object-Oriented Measures
- Object-Oriented Measurement in Practice
- Utility of Object-Oriented Measures
  - Preventative Action
  - Quality Prediction
  - Design Guidelines
- Cautions and Myths
  - Confounding Effects
  - Using Prediction Models Across Projects
  - Optimal Class Size & Faults
- Conclusions

Definition

Contents

- Object-oriented measures quantitatively characterize the structure of a software system
- We are only concerned with static measures that can be collected from design documents or source code
- We are only concerned with class-level measures (as opposed to, for example, method-level measures)

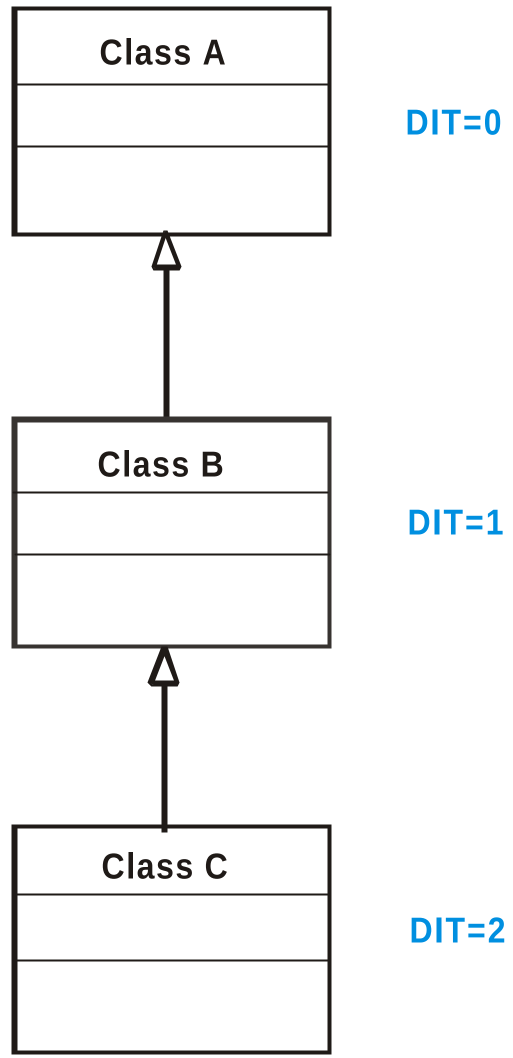
Types of Measures

Contents

- Many OO measures have been developed (at least 100)
- Types of measures developed:
  - size (measure how big a class is)
  - inheritance (characterize inheritance hierarchy)
  - coupling (characterize relations amongst classes)
  - cohesion (characterize relations within classes)
  - complexity (aggregates of method measures)

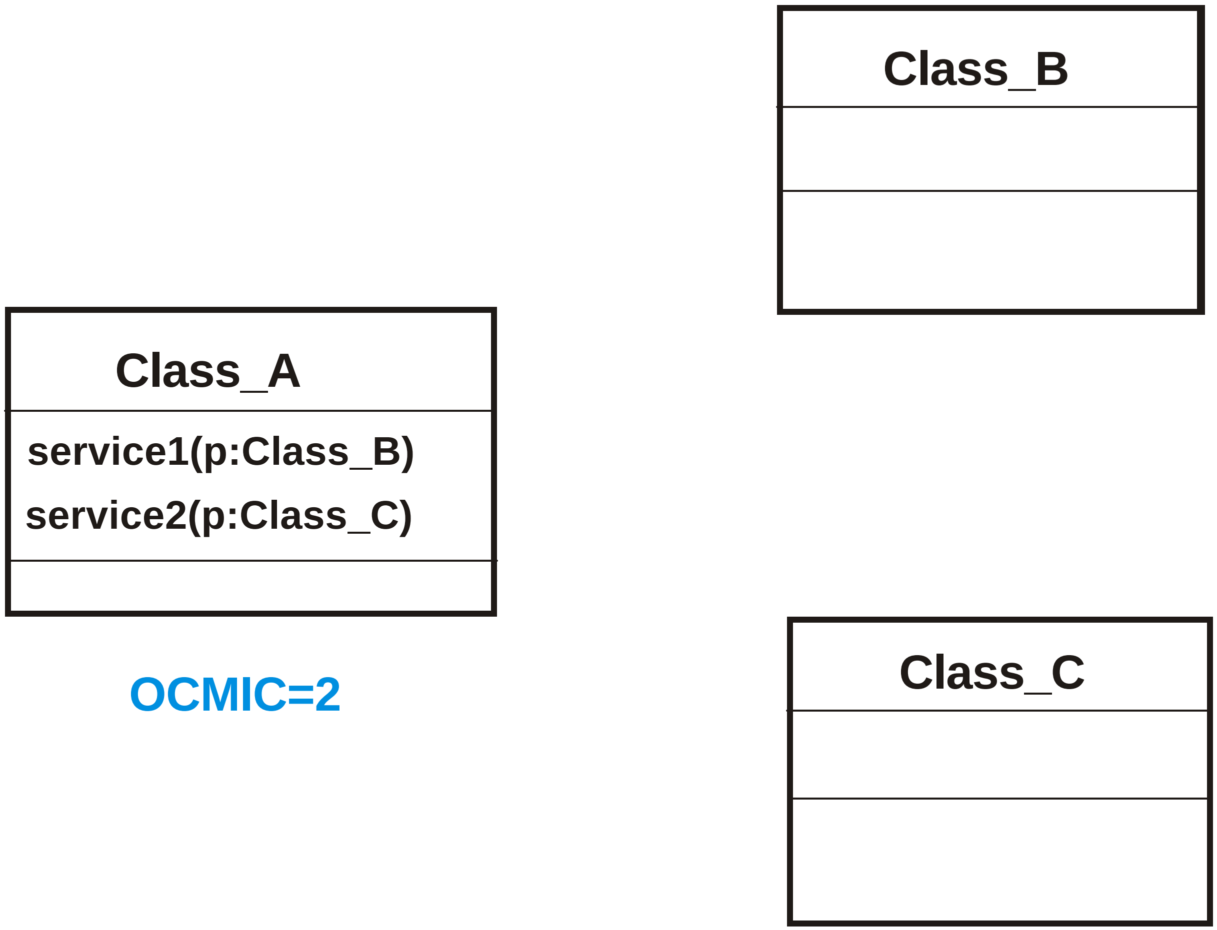
Example of Inheritance Measure: DIT

Contents



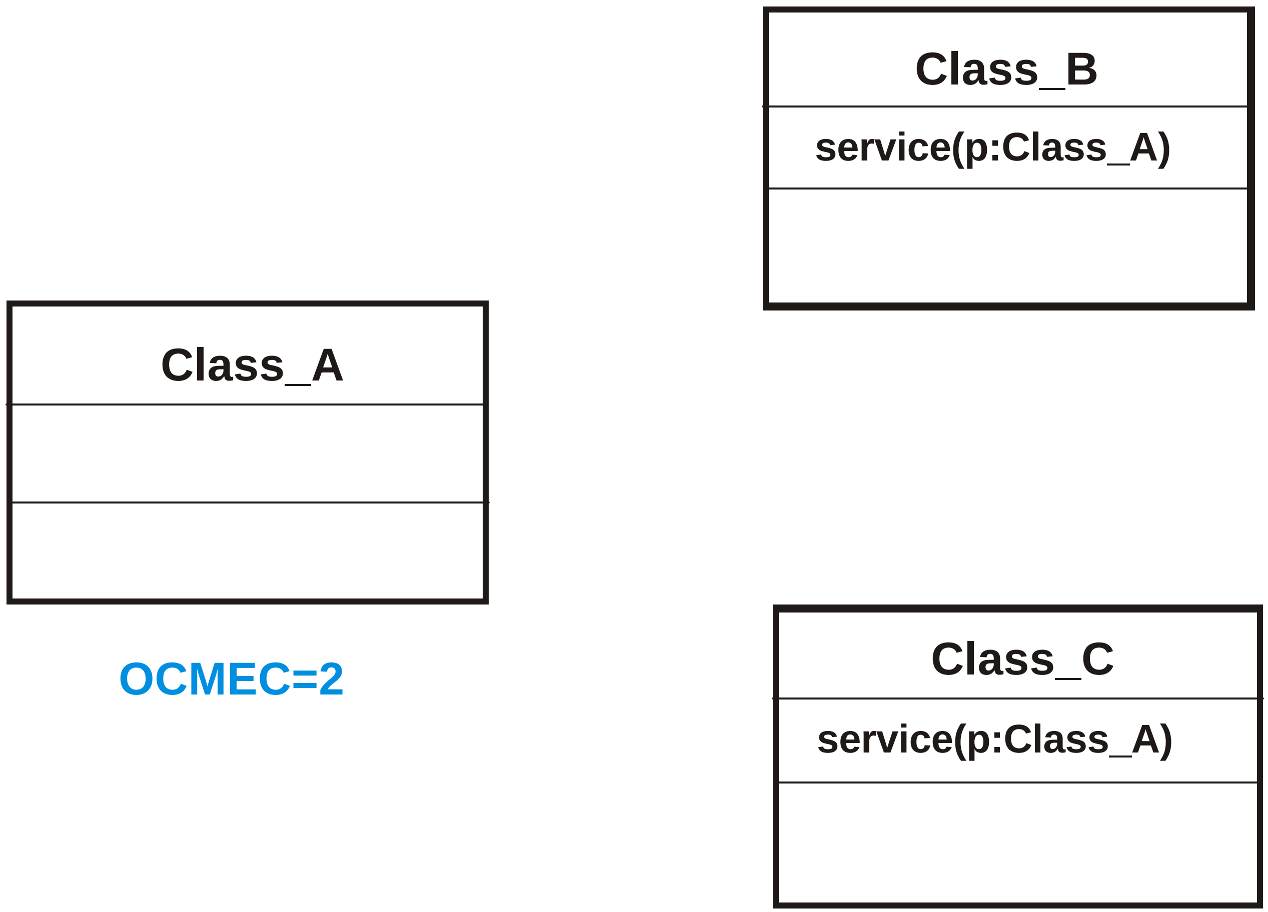
Example of Coupling Measure: Class-Method Import Coupling

Contents



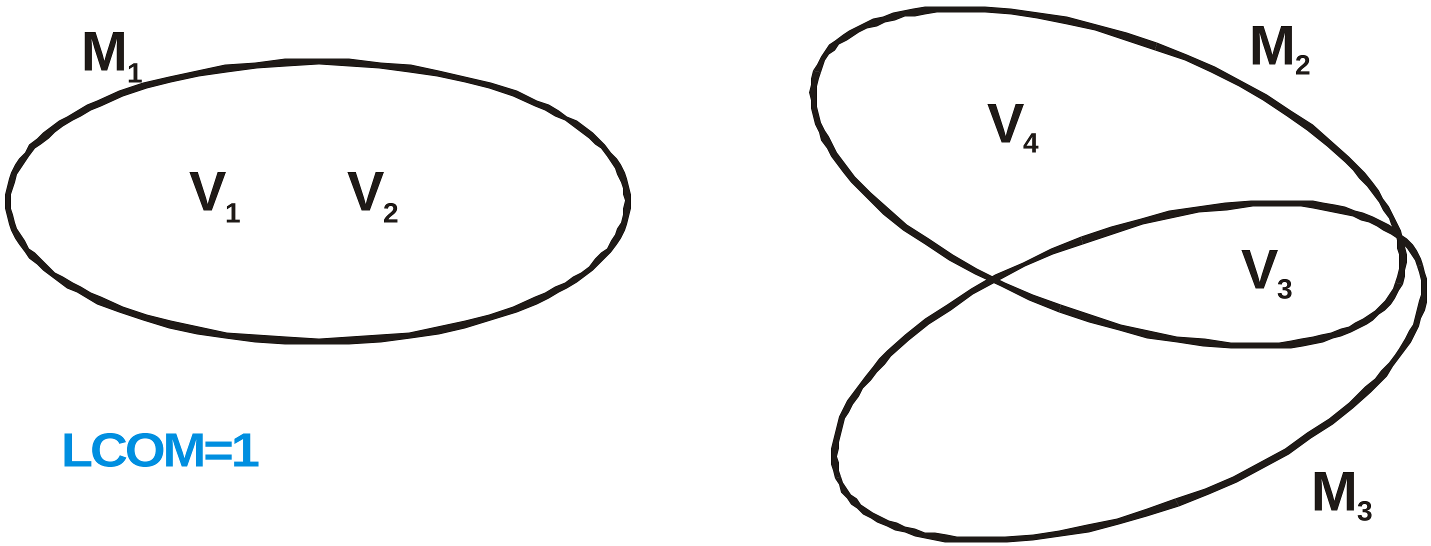
Example of Coupling Measure: Class-Method Export Coupling

Contents



Example of Cohesion Measure: Lack of Cohesion

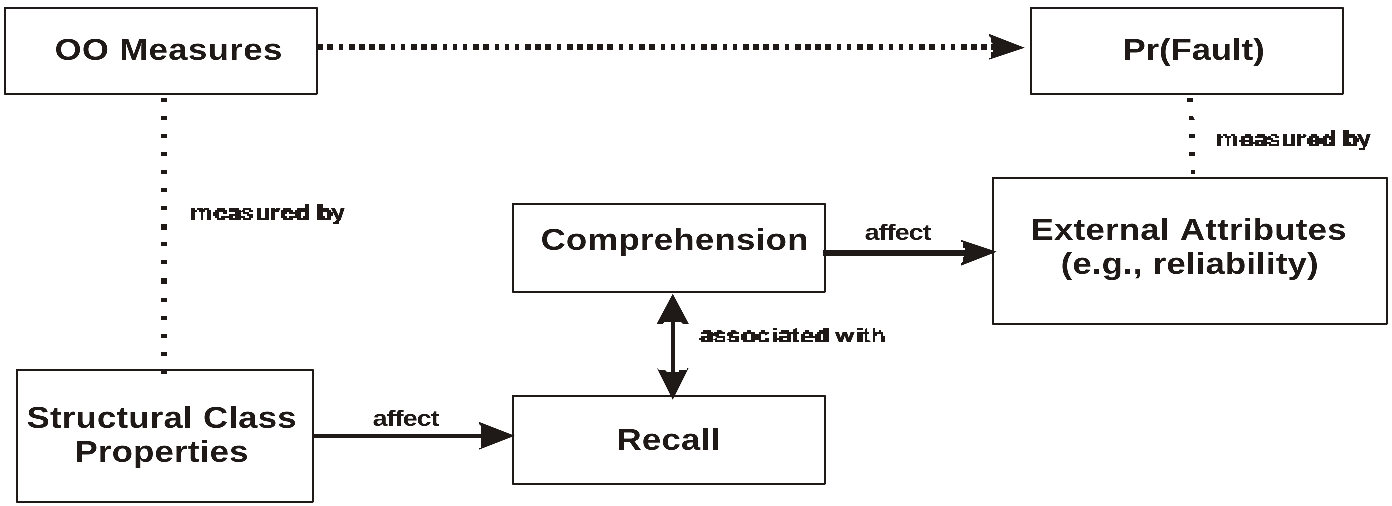
Contents



LCOM=1

A Theory of Object-Oriented Measures - I

Contents



A Theory of Object-Oriented Measures - II

Contents

Cognitive Complexity Theory

- The structural properties of a class have an impact on recall
  - interference effects
  - fan effects
  - familiarity
- Recall is associated with comprehension
- Comprehension is associated with reliability

The Most Useful Measures

Contents

- It is necessary to have a good measure of class size
- Import and export coupling measures, separately, have been found to work well across studies
- Depth of Inheritance Tree has equivocal evidence
- Number of Children is not a good measure
- Cohesion measures are not related to much

Putting Object-Oriented Measurement Into Practice

Contents

- Tools:
  - Tools to collect the measures
  - Tools to perform data analysis
- Expertise:
  - Object-oriented measurement
  - Statistics
  - The systems being analyzed
- Process:
  - ISO/IEC 15939

Utility of Object-Oriented Measures

Contents

- Object-oriented measures can potentially be used for:
  - Taking preventative action
  - Quality (and cost) estimation
  - Developing design guidelines
- You can get different results depending on which criterion you are talking about:
  - pre-release faults
  - post-release faults
  - development & maintenance effort

Preventative Action - I

Contents

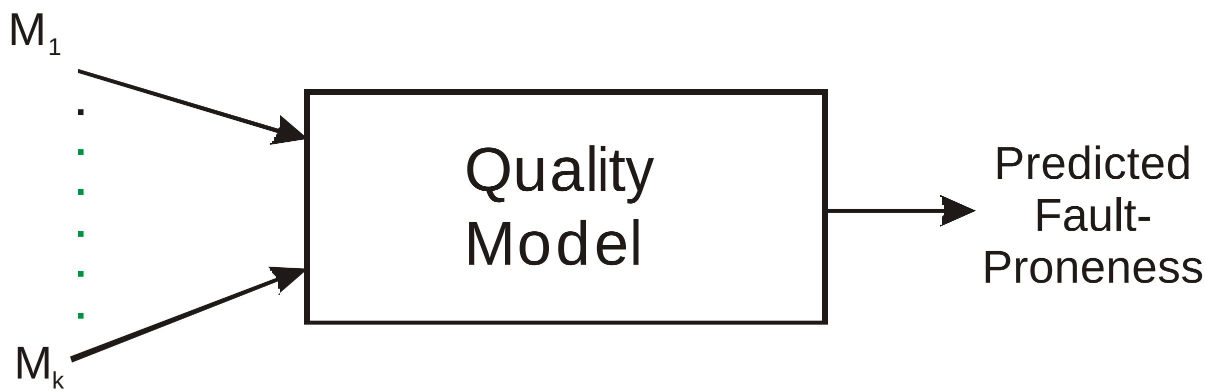
- There exists evidence showing that most faults in a software system are detected in a small percentage of its components
- To improve defect detection effectiveness and efficiency it is desirable to identify these components early on
- Then one can take mitigating actions, such as more inspections, targeted testing, or even a redesign

Preventative Action - II

Contents

Quality Models

- Quantitative models are constructed to predict which components are likely to have a fault



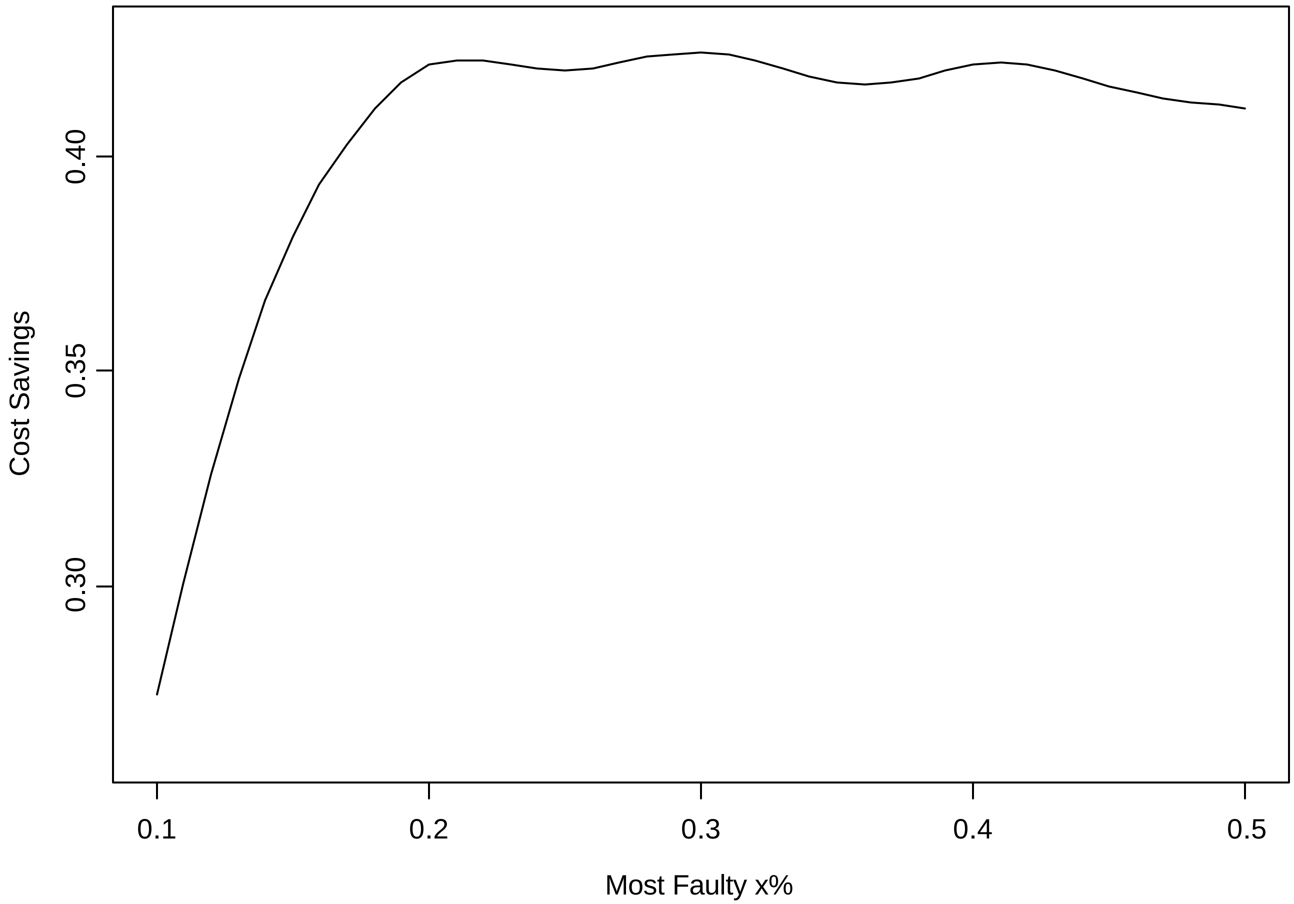
Preventative Action - Example (1)

Contents

- A commercial Java application
- A quality model was constructed to predict the probability of a post-release fault
- Quality model was used to identify the classes that should be inspected
- Evaluated the cost savings from using the quality model (we had data on the cost of inspections and dealing with post-release faults)

Preventative Action - Example (2)

Contents



Quality Prediction

Contents

- Developed a quality model on one release of a commercial Java application
- Used that model to predict the proportion of classes in the next release that will have a fault in them
- The prediction has approx. 9% error (i.e., underestimated the proportion by 9%)
- This can be considered a good prediction accuracy

Design Guidelines - I

Contents

- It is also possible to empirically identify which structural properties of the software cause problems (e.g., which types of coupling)
- This information can then be used to construct proscriptive design guidelines to minimize problems in future development efforts
- Guidelines can be enforced automatically or through inspections

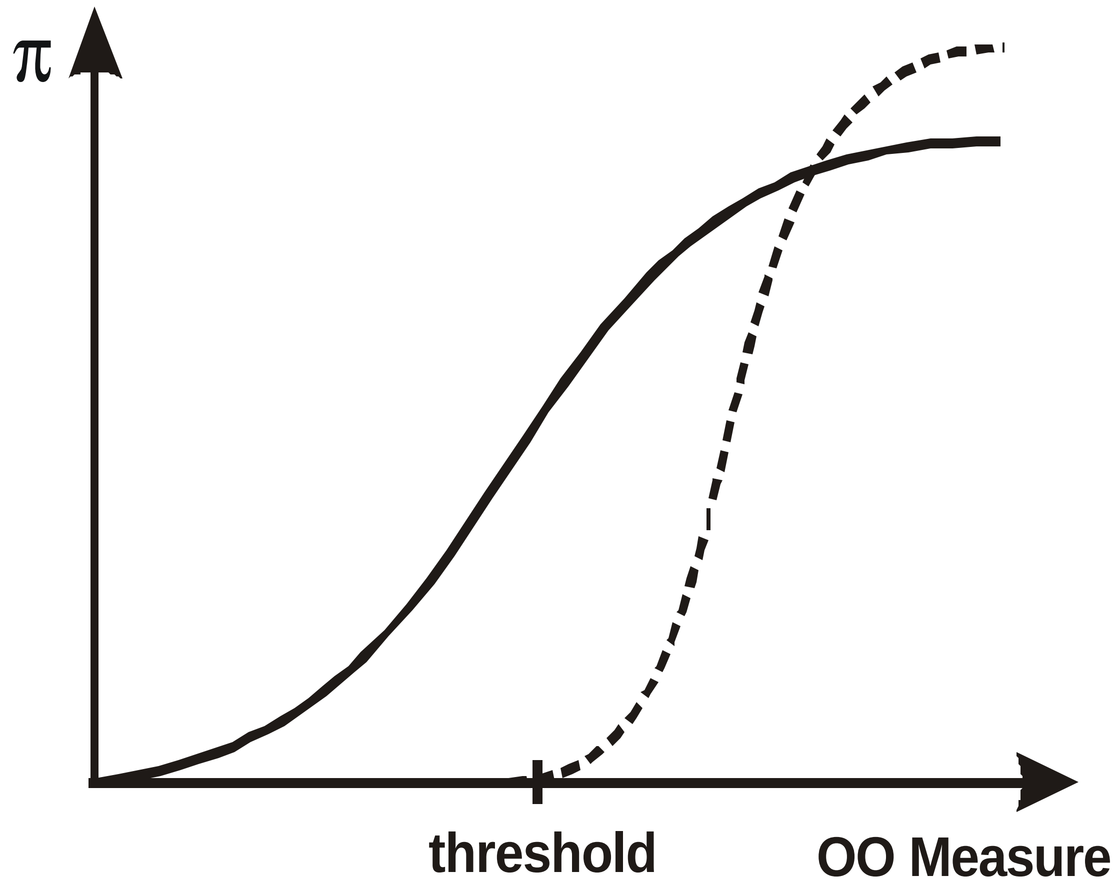
Design Guidelines - II

Contents

- Design guidelines expressed in the form of thresholds on the measures
- Types of design guidelines (hypothetical):
  - coupling should be less than 7
  - cohesion should be greater than 20
- If rule is not satisfied then there is a greater risk of a fault

Design Guidelines - III

Contents



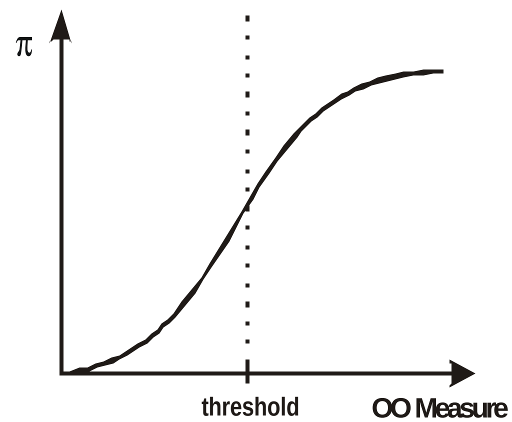
Design Guidelines - IV

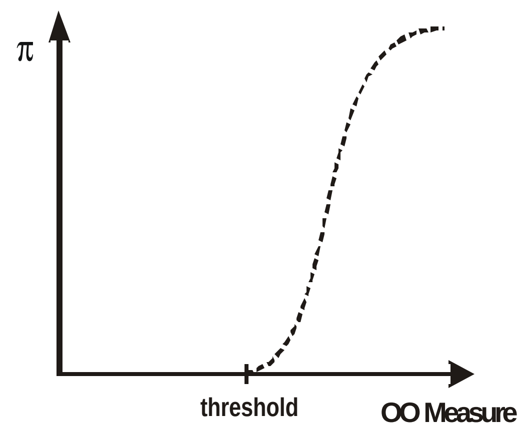
Contents

- Thus far, no evidence that thresholds exist
- We could not identify a specific value on any of the object-oriented measures where the probability of a fault suddenly increases
- Does this mean that thresholds are useless ?

Design Guidelines - V

Contents





Cautions and Myths

Contents

- Confounding effects
- The same models can be used across projects
- There is an optimal class size

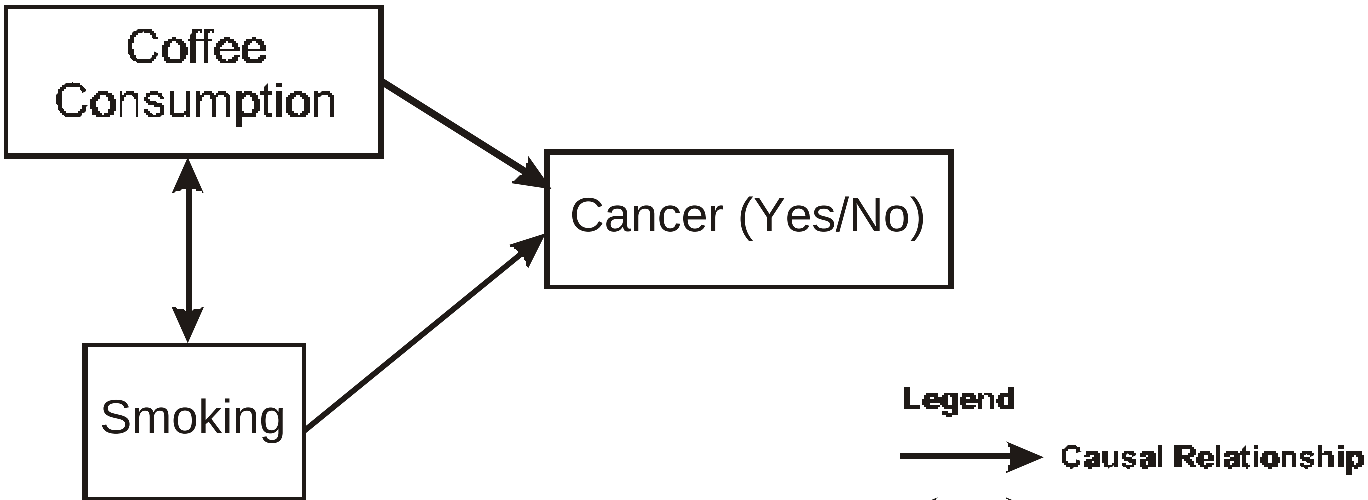
Common Validation Method

Contents

- You collect data on the object-oriented measure and on whether a fault occurs in a class
- If there is a positive relationship between the measure and fault-proneness, then the measure is said to be validated
- At least 12 “successful” validation studies

An Epidemiological Example

Contents



- If you look only at the coffee consumption ® cancer relationship, you can get very misleading results
- Smoking is a confounder

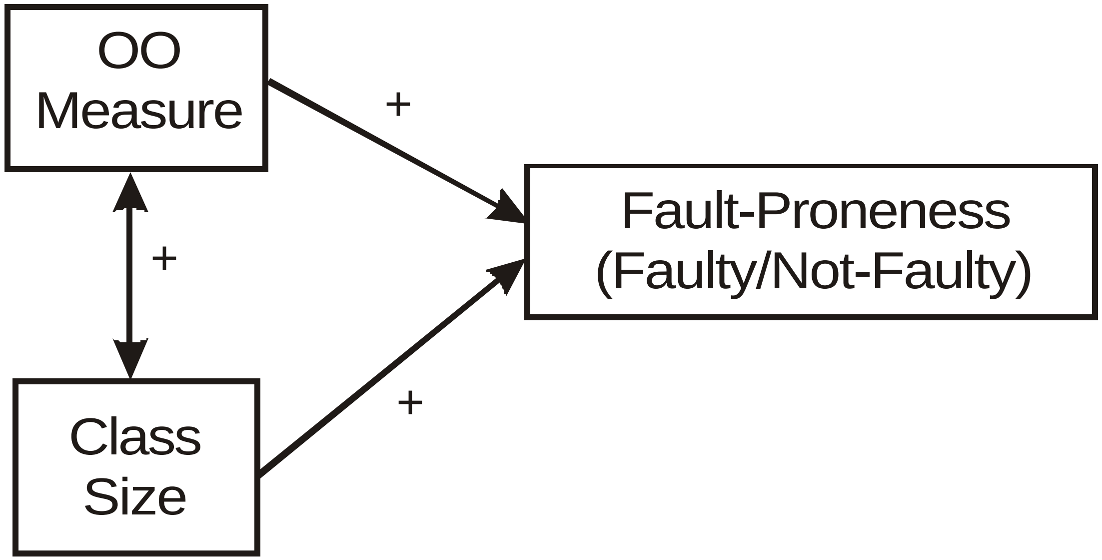
Confounding Effects

Contents

- It is known that smoking and coffee consumption are associated
- It is known that smoking causes cancer
- If smoking is not controlled during the study, then it is likely that an inflated coffee effect will be found
- This is a classic confounding effect
- Without proper controls, the study results can show a relationship where none really exists

Confounding with Object-Oriented Measures

Contents



- In this case, size is the confounder
- There is substantial evidence that most object-oriented measures are associated with class size

Example Relationship with Size

Contents

OO Metric	LOC	
	Rho	p-value
WMC	0.88	<0.0001
DIT	0.098	0.19
CBO	0.46	<0.0001
RFC	0.88	<0.0001
LCOM	0.24	0.0011
NMA	0.86	<0.0001
NPAVG	0.27	0.000256

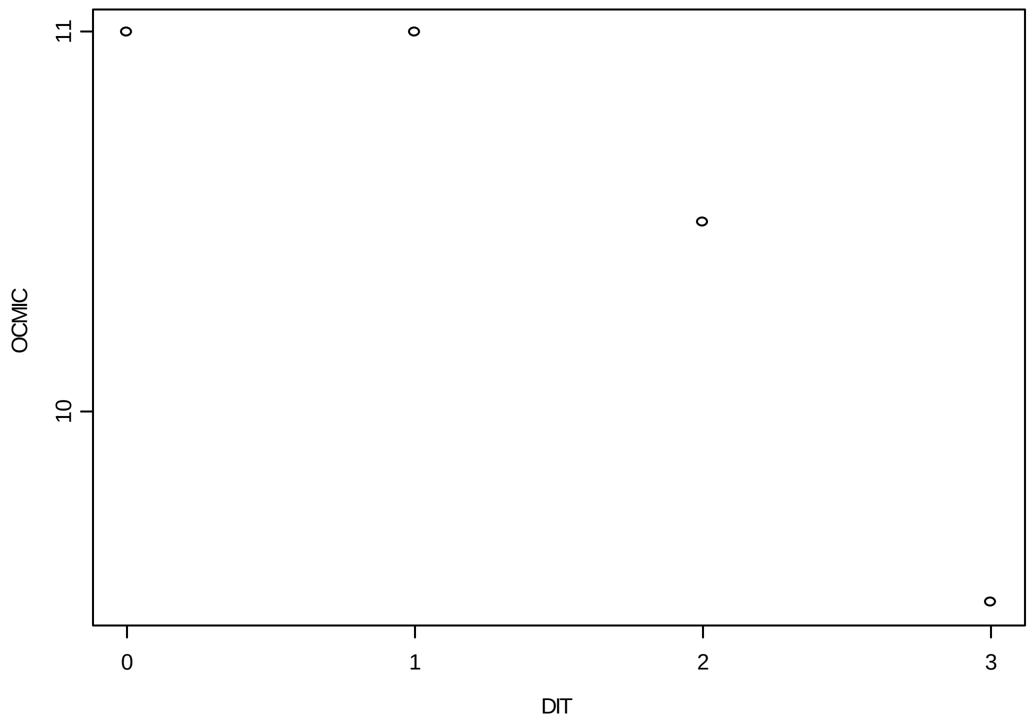
The Confounding Effect of Size

Contents

- Just looking at the relationship between an object-oriented measure and faults does not make sense
- It is necessary to at least control for size

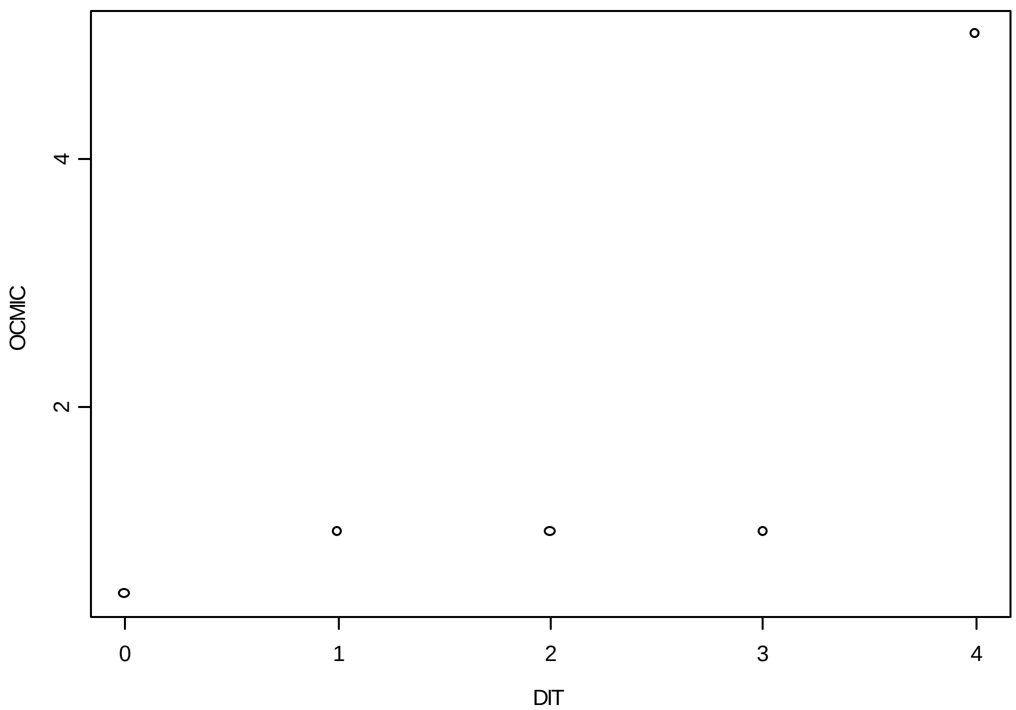
The Confounding Effect of Inheritance Depth - I

Contents



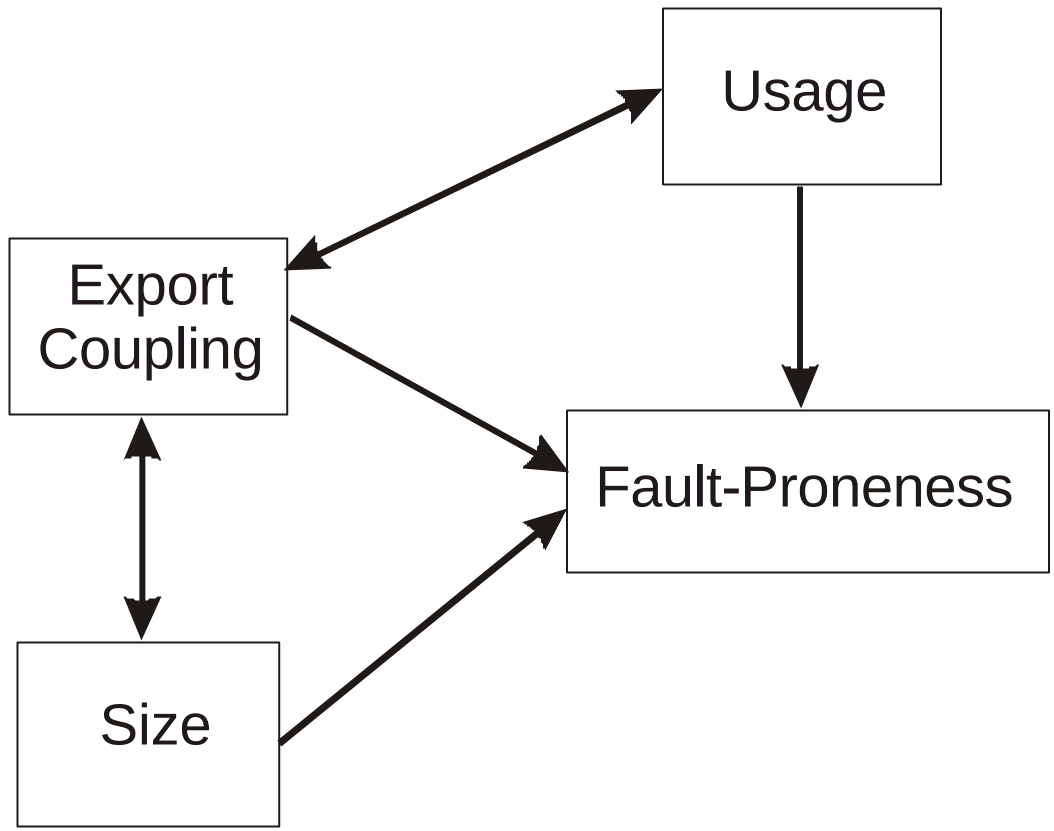
The Confounding Effect of Inheritance Depth - II

Contents



Usage Effects

Contents



Logistic Regression

Contents

Analysis Technique

- The form of a logistic regression model is:

$$p = \frac{1}{1 + e^{-\left(b_0 + \sum_{i=1}^{k} b_i x_i\right)}}$$

- Parameter estimates through the maximization of a log-likelihood, including estimates of the threshold
- Standard technique for validation of object-oriented measures

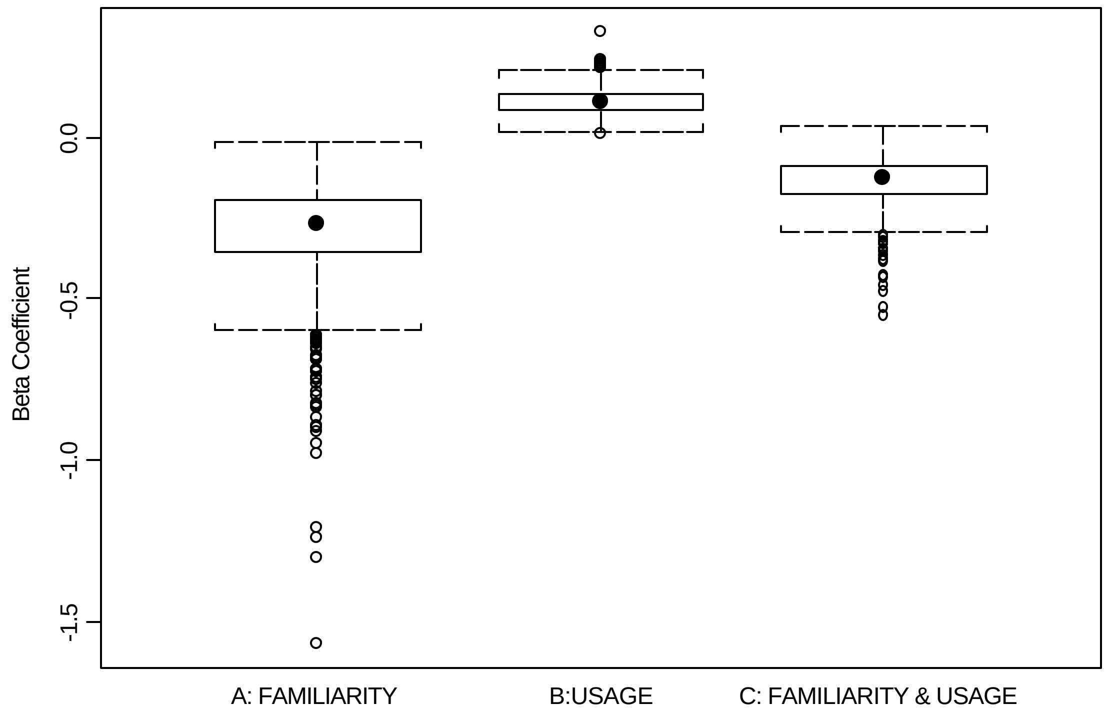
Monte Carlo Simulation

Contents

- We performed a Monte Carlo simulation to illustrate how usage can give very different results
- The scenario was that of looking at the relationship between export coupling and fault-proneness assuming:
  - Usage has a bigger effect on fault-proneness
  - Export-coupling has a bigger effect on fault-proneness

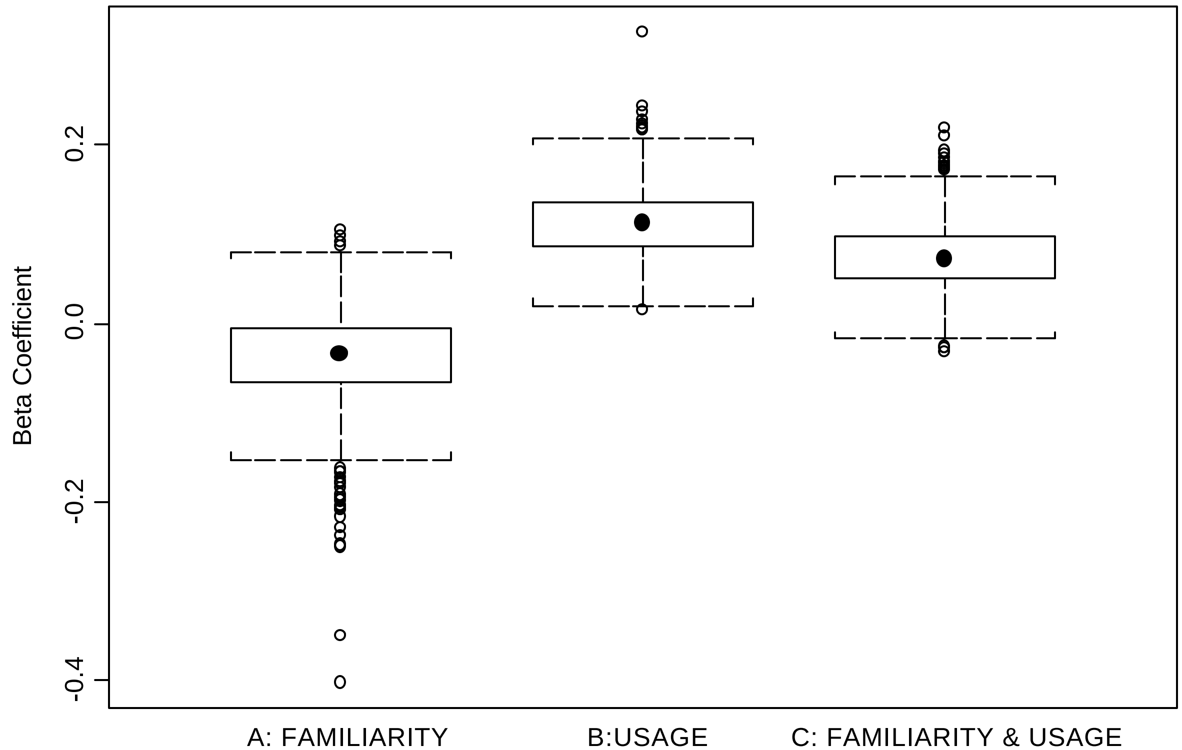
Export Coupling Has a Bigger Effect

Contents



Usage Has a Bigger Effect

Contents



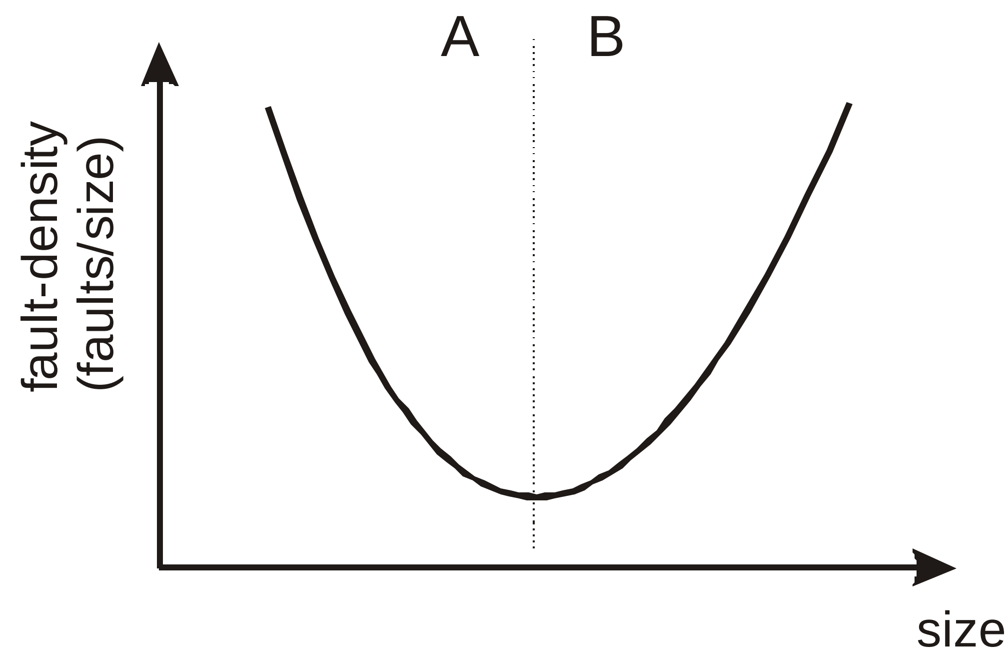
Using Models Across Projects

Contents

- Unless you have a measure for usage or unless the usage patterns are the same across projects, then it is dangerous to use models across projects
- Using models across multiple releases is probably a safer bet

Optimal Class Size & Faults - I

Contents



- Smaller classes (or modules) are more faulty than larger ones (program decomposition is bad)
- There is an optimal class size

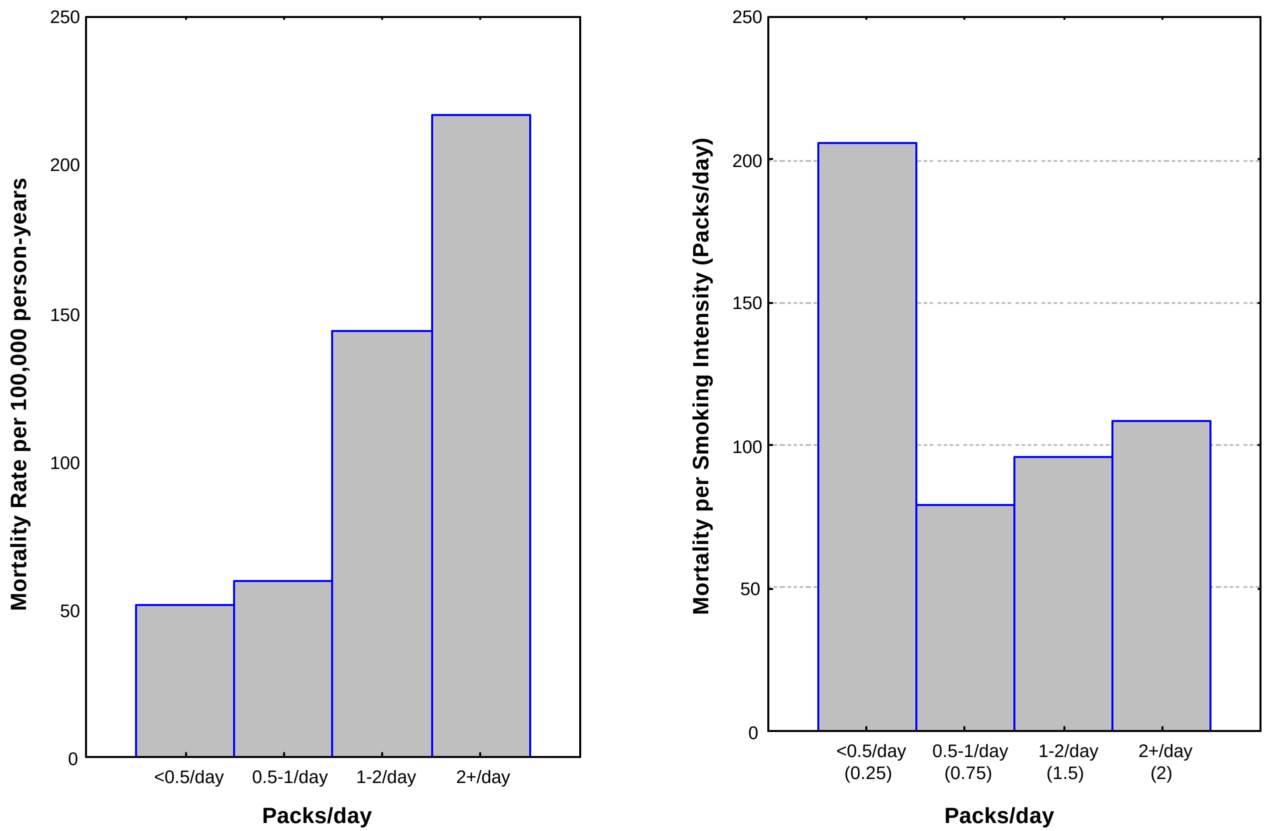
Optimal Class Size & Faults - II

Contents

- The first conclusion is due to a mathematical artifact of plotting Y/X vs. X
- You will always get a negative association, even if there is no association between the raw variables
- It is easy to obtain absurd conclusions (turn a positive association to a negative one) this way
- Some examples ...

Smoking and Mortality Rate

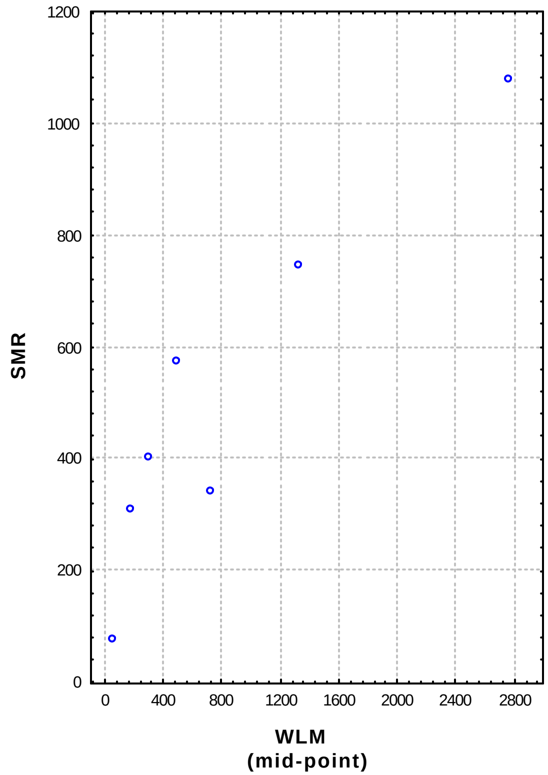
Contents

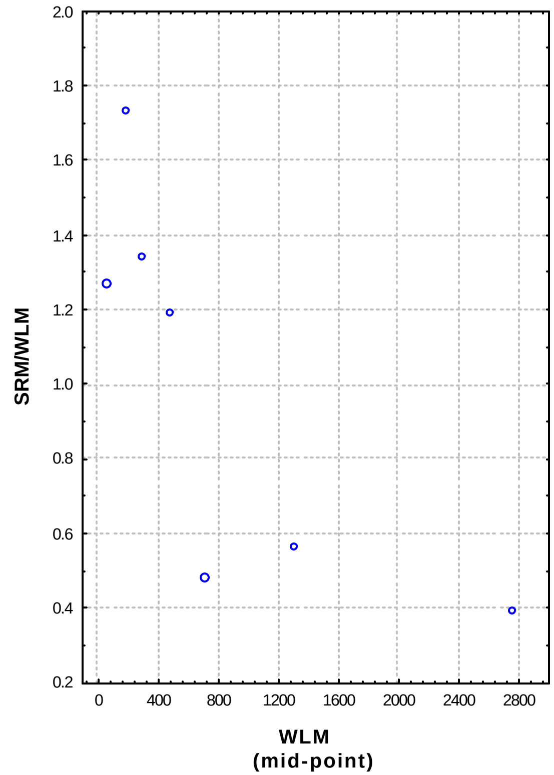


- Optimal smoking: 0.75 packs/day

Uranium Exposure & Mortality

Contents





- Quickly, move closer to a uranium mine

Conclusions

Contents

- Object-oriented measures can be very useful for identifying fault-prone classes early on, but:
  - be cautious about thresholds
  - it is safer to use the prediction models across releases of the same product
  - watch out for statistical artifacts

More Information

Contents

- The research work supporting this presentation can be found at:

http://www.object-oriented.org

National Research Council Canada
Conseil national de recherches Canada
Institute for Information Technology

Acknowledgements

Contents

- Saida Benlarbi, Alcatel
- Nishith Goel, Cistel Technology
- Shesh Rai, Statistics Canada & St. Jude Children’s Research Hospital
- Walcelio Melo, Oracle Brazil
- Hakim Lounis, CRIM
- Daniela Glasberg, McGill University