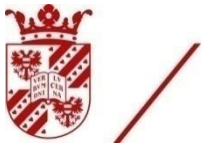


university of groningen

Apostolos Ampatzoglou
a.ampatzoglou@rug.nl
University of Groningen
The Netherlands

Assessing Code Smell Interest Probability

A Case Study

Sofia Charalampidou, Apostolos Ampatzoglou, Alexander Chatzigeorgiou, Paris Avgeriou

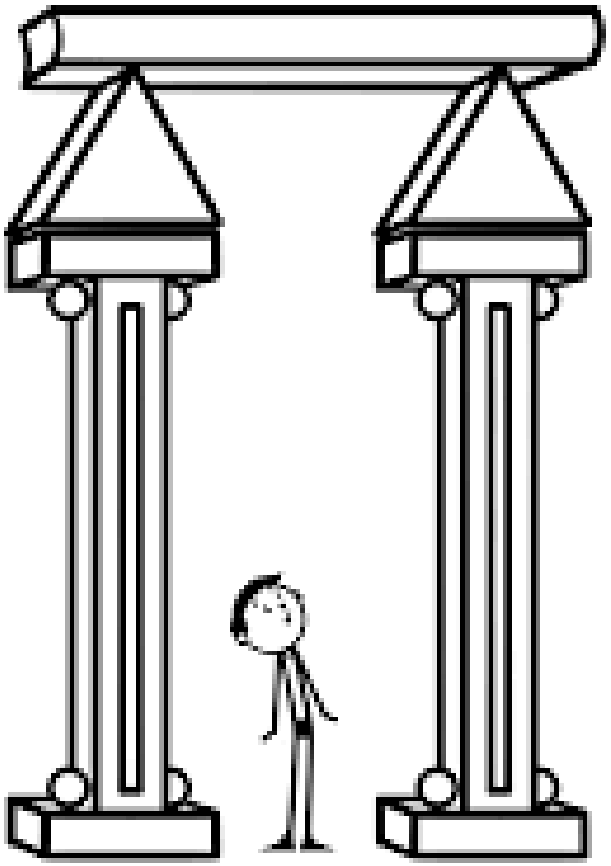
university of groningen

HELLENIC REPUBLIC
UNIVERSITY OF MACEDONIA

university of groningen

Context

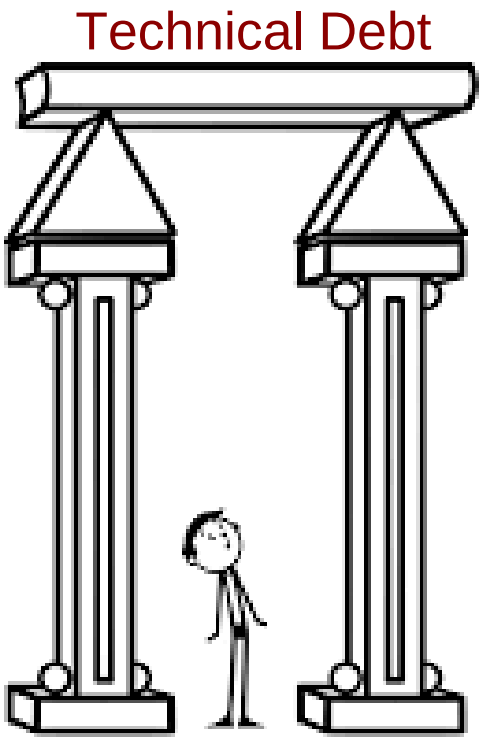
Technical Debt



Principal Interest

university of groningen

Context

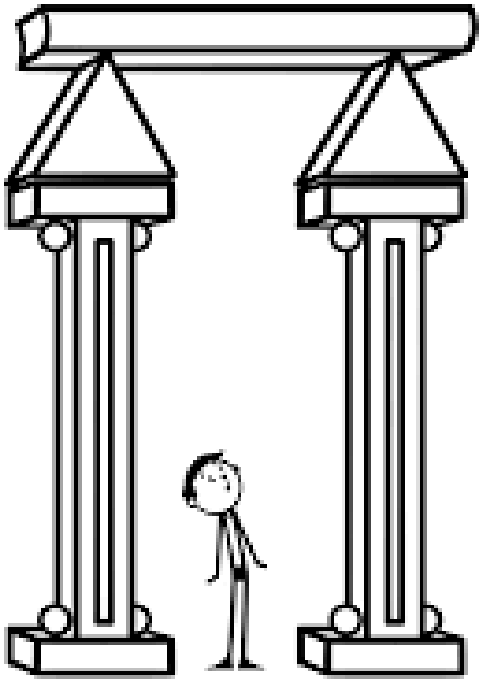


- Interest Amount
- Interest Probability

university of groningen

Context

Technical Debt



Principal

Interest

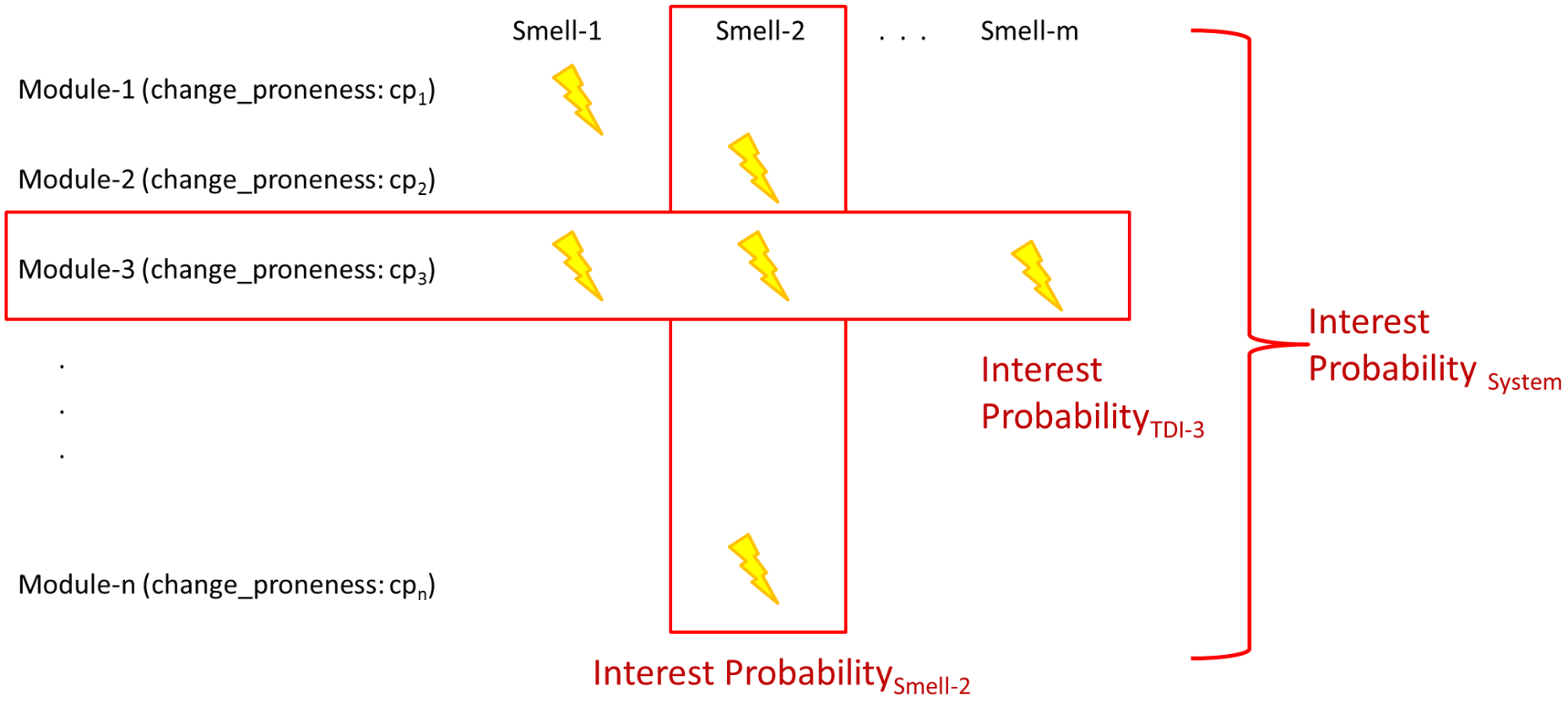
Interest Amount

Interest Probability

Key-Indicator for
TD Prioritization

university of groningen

Interest Probability



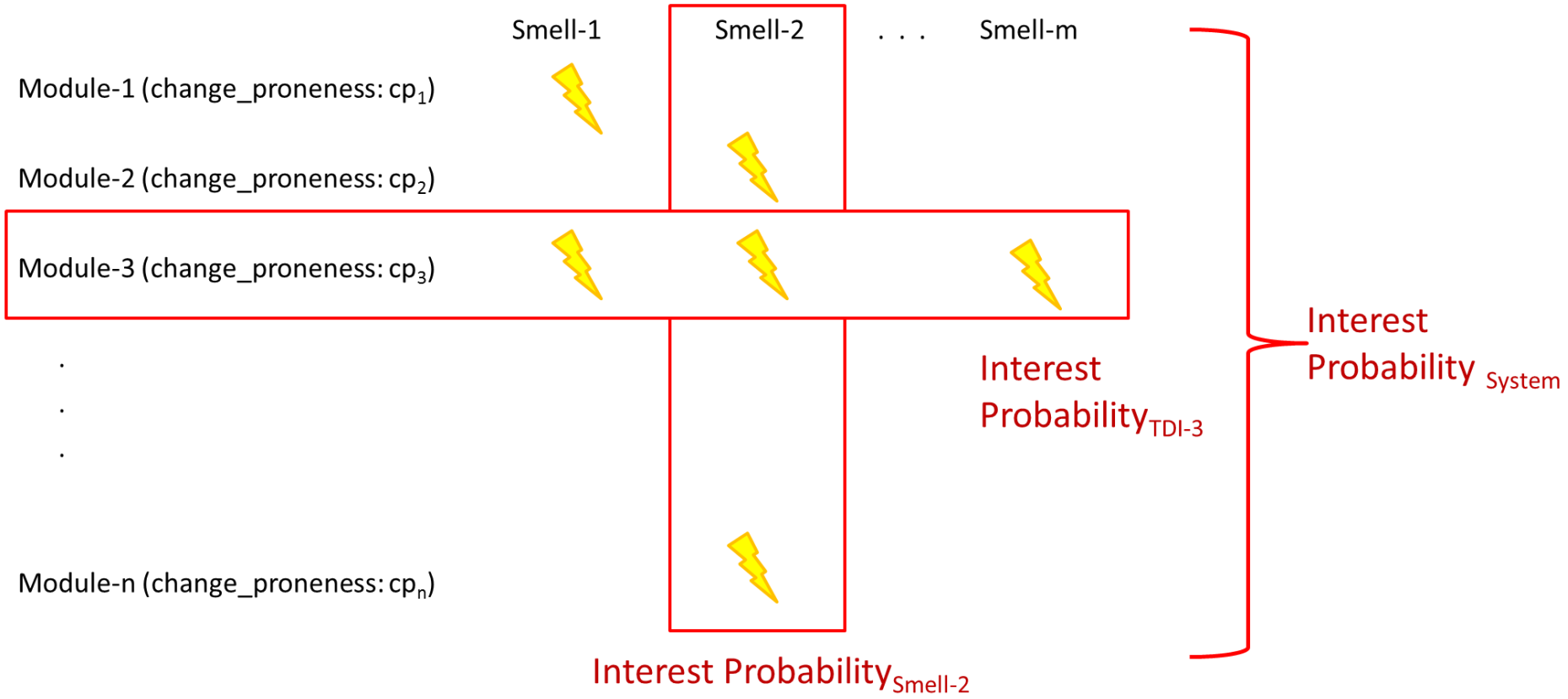
university of groningen

Smell Interest Probability

Interest probability $_{\text{smell X}}$ = 0.5 ➔

How to read it?

There is a 50% chance that at least one module suffering from smell X will change in the next version of the system



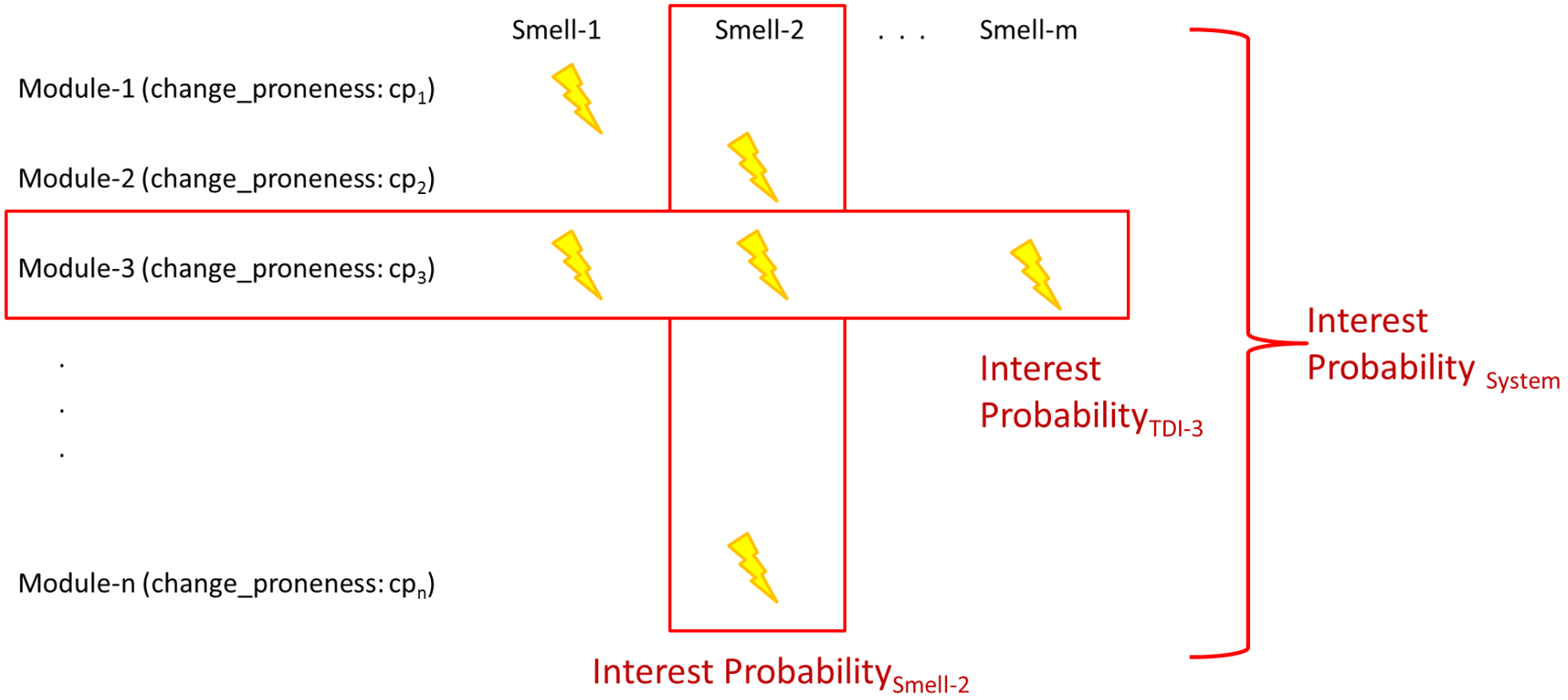
university of groningen

Smell Interest Probability

Why to use it?

(a) Prioritize refactoring of most risky smells

(b) Training



university of groningen

Smell Interest Probability

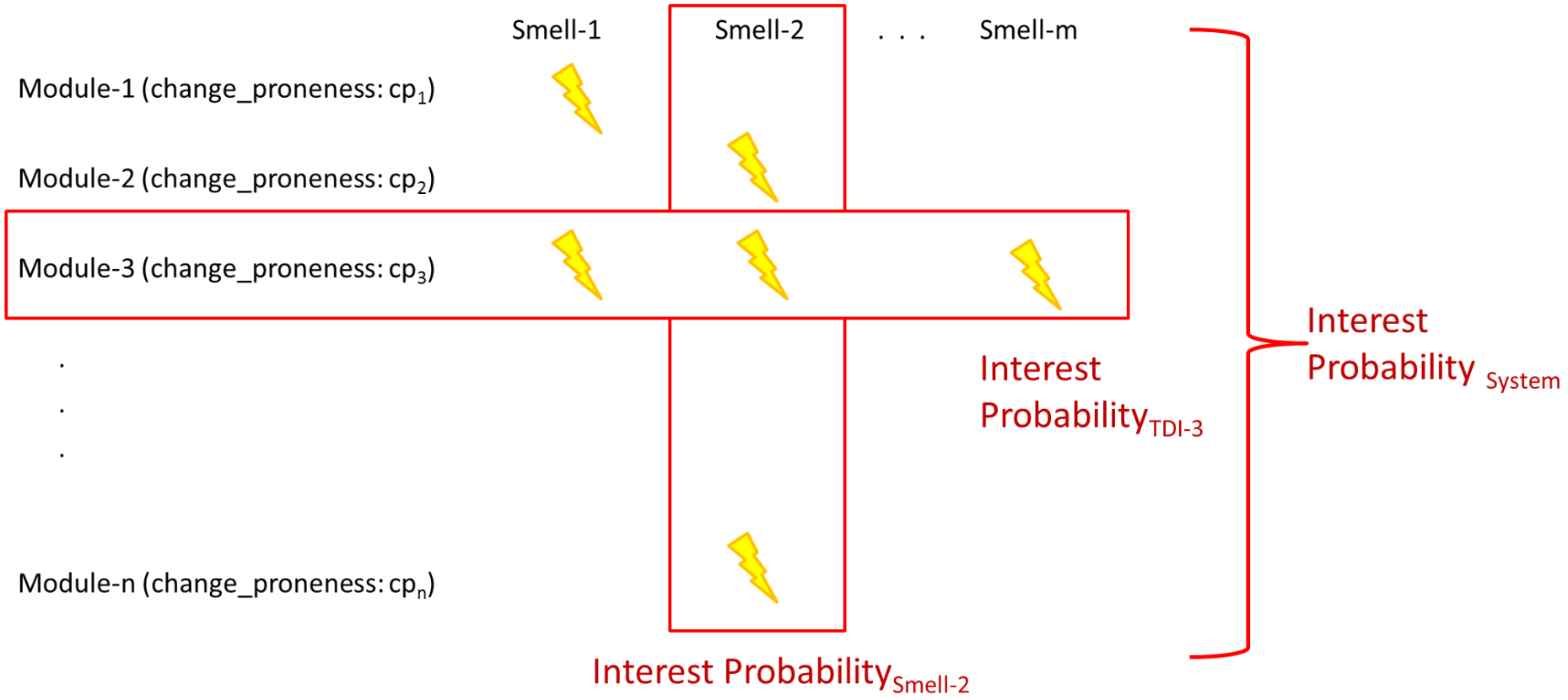
How to calculate?

Joint probability of events

(a) number of events

(b) probability of each maintenance event to occur

(c) $P(A|B) = P(A) + P(B) - P(A)*P(B)$

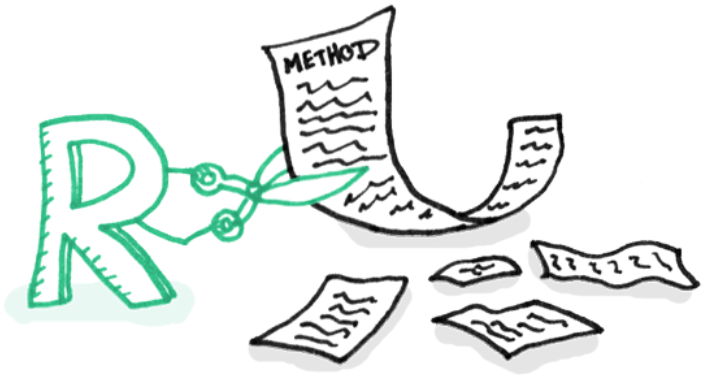


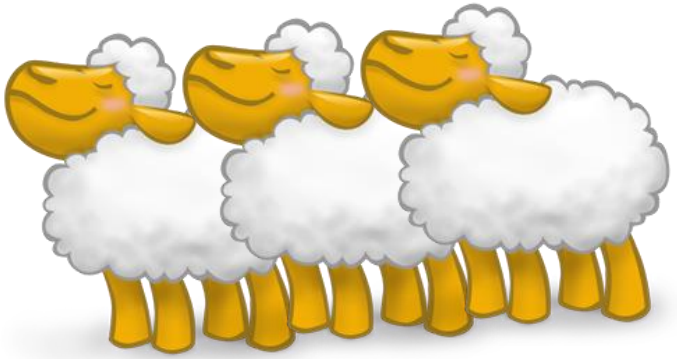
university of groningen

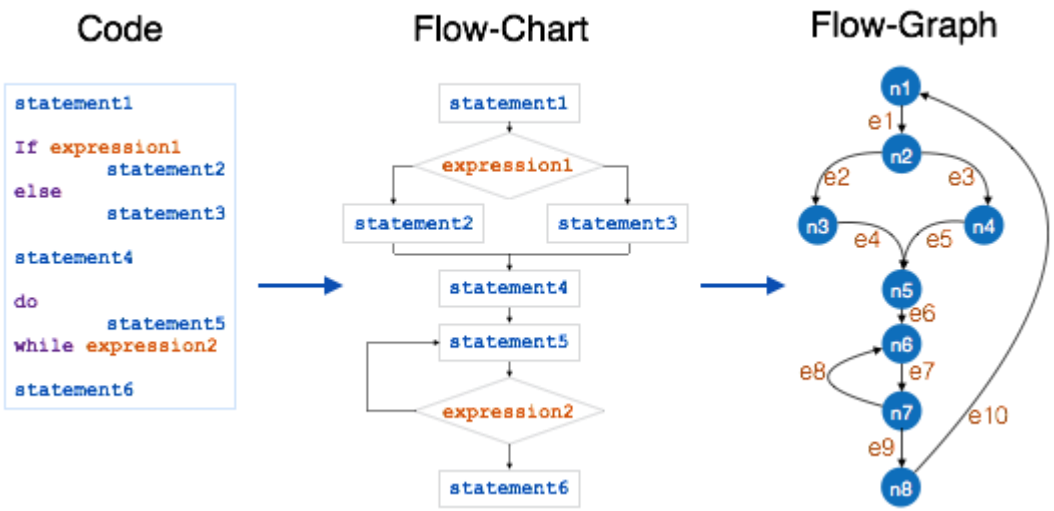
Case Study Design

university of groningen

What smells are we interested in?







university of groningen

Case Study Design

Goal of this study:

What is the interest probability incurred by code smells?

- What is the occurrence frequency for each code smell?
- What is the mean change proneness of the modules in which each type of code smell is identified?

university of groningen

Case & Data Collection

5,5K classes
48K methods ~ Units of analysis
16K commits

[V1] ***Method name***: The name of the considered method

[V2] ***Class name***: The class in which the method belongs to

[V3] ***Long Method:*** Is the method classified as long by the SEMI tool (yes / no)?

[V4] ***Code Clone:*** Number of clones identified in the method's body by NiCad

[V5] ***Conditional Complexity:*** Number of conditional statements in the method that have been flagged as unnecessary (i.e., they can be replaced with polymorphism) by Deodorant

[V6] ***PCMC***: Percentage of commits in which the method has changed. The complete history of the method is considered.

university of groningen

Data Analysis

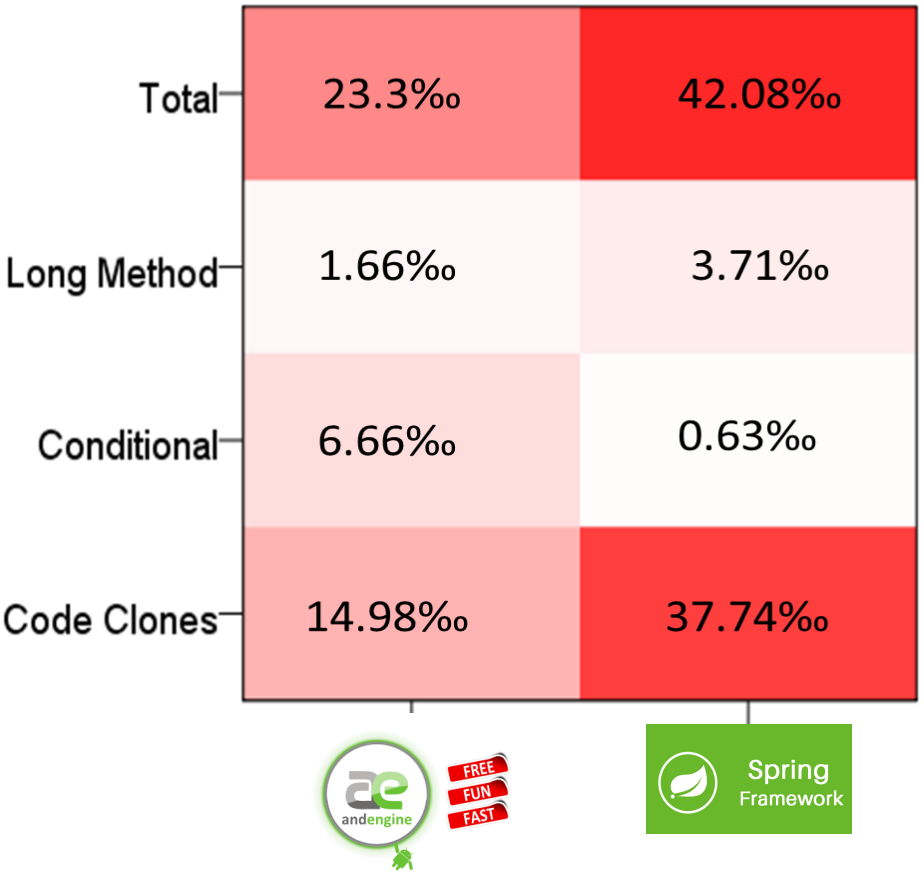
Question	Variables	Statistical Analysis
RQ_1	[V3] [V4] [V5]	Frequency Table (actual value) Heatmap (per mille)
RQ_2	[V3] [V4] [V5] [V6]	One-sample Hypothesis Testing of [V6] against the mean [V6] of all project's methods Select cases based on [V3], [V4], or [V5]

university of groningen

Results

university of groningen

Smell Frequency



The most frequent type of code TD is code clones. However, their frequency-level is project-related. Concerning long methods, approximately 2-4 can be identified in a thousand methods. The frequency of Conditional Complexity is also project related since it varies between less than one to 6 per mille in the two projects.

university of groningen

Change Proneness

Smell	Mean	Std. Dev.	Sig.	95% conf. interval	
				Low	Up
Long Method	2.00	4.549	*.000*	0.91	2.31
Conditional Complexity	2.36	3.793	*.011*	0.50	3.44
Code Clones	0.45	1.434	.106	-0.01	0.12

Smell	Mean	Std. Dev.	sig.	95% conf. interval	
				Low	Up
Long Method	3.60	4.615	.235	-2.85	8.61
Conditional Complexity	3.21	3.735	*.009*	0.69	4.29
Code Clones	1.86	3.921	.060	-0.05	2.34

Methods that suffer from code smells are more change prone than TD-free methods. Among specific types of code smells, long methods and the use of conditionals instead of polymorphism are usually encountered in change prone methods. On the other hand code clones are usually positioned in system parts that do not change frequently.

university of groningen

Interest Probability

	Long Method	Conditional Complexity	Code Clones
#TDIs (#events)	5	20	45
Mean Change Probability (mean probability of event to occur)	$2.00e^{-3}$	$1.78e^{-3}$	$1.03e^{-3}$
Interest Probability	0.99%	3.50%	4.53%

	Long Method	Conditional Complexity	Code Clones
#TDIs (#events)	166	28	1689
Mean Change Probability (mean probability of event to occur)	$0.14e^{-3}$	$0.16e^{-3}$	$0.03e^{-3}$
Interest Probability	2.07%	0.44%	14.34%

Code clones is the smell that has the higher probability to produce interest in future maintenance activities in the two examined projects. This characteristic is mostly attributed to the smell occurrence frequency rather than its identification in change prone methods. The long method smell is the code TD type that presents the most similar smell interest probability in the examined projects.

university of groningen

Implications

Practitioners

Existence of smells and method change proneness → Extra care in change prone methods

High levels of interest probability → Training in TD prevention and repayment

The modification of a clone can cause interest in multiple modules → Alert on types of code TD

Researchers

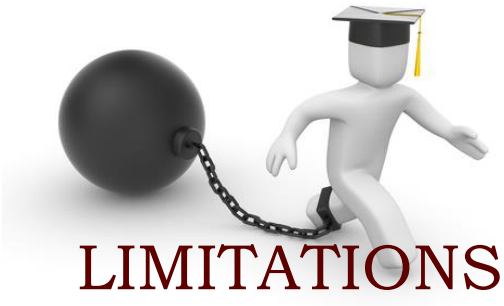
More Smells

Different Levels of Granularity

More projects

university of groningen

Threats to Validity



Construct Validity:

- Tool Accuracy
- Existence of Smells other than the three examined

Lack of Generalization to:

- Programming Language / Paradigm
- Other smells

Reliability:

- No research bias
- Public repositories

university of groningen

Thank you for your attention!

Questions???