When and Why Your Code Starts to Smell Bad

Michele Tufano, Fabio Palomba, Gabriele Bavotaz,
Rocco Olivetox, Massimiliano Di Penta, Andrea De Luciay,
Denys Poshyvanyk

Johnatan Oliveira

2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE)

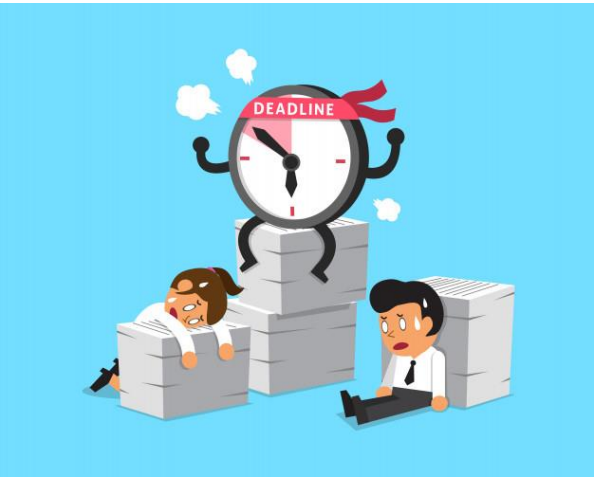
Agenda

- Introduction and Motivation
- Goal
- Related Work
- Study Design
- Results
- Threats to Validity
- Conclusion and Future Work

Introduction

Introduction

Technical debt is a metaphor introduced by Cunningham to indicate “not quite right code which we postpone making it right”



Common wisdom suggests that urgent maintenance activities and pressure to deliver features

Goal

Analyze change history of projects with the purpose to find **when** and **why** smells are introduced

Related Work

Related Work

1. Khomh et al. studied the relationship between the presence of code smells and software change

2. Abbes et al. conducted three controlled experiments with the aim of investigating the impact of Blob and Spaghetti Code smells on program comprehension

3. Arcoverde et al. investigated how developers react to the presence of bad smells in their code.

Research Questions

Research Questions

RQ1 – When are code smells introduced?

RQ2 – Why are code smells introduced?

Study Design

Data set

3

The Apache Software Foundation

different ecosystems analyzed

Data set

The context of the study consists of the change history of 200 projects belonging to three software ecosystems

CHARACTERISTICS OF ECOSYSTEMS UNDER ANALYSIS.

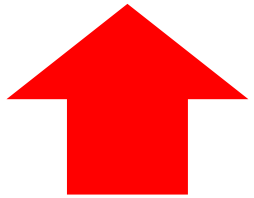
Ecosystem	#Proj.	#Classes	KLOC	#Commits	#Issues	Mean Story Length	Min-Max Story Length
Apache	100	4-5,052	1-1,031	207,997	3,486	6	1-15
Android	70	5-4,980	3-1,140	107,555	1,193	3	1-6
Eclipse	30	142-16,700	26-2,610	264,119	124	10	1-13
Overall	**200**	-	-	**579,671**	**4,803**	**6**	**1-15**

Data set

The context of the study consists of the change history of 200 projects belonging to three software ecosystems

CHARACTERISTICS OF ECOSYSTEMS UNDER ANALYSIS.

Ecosystem	#Proj.	#Classes	KLOC	#Commits	#Issues	Mean Story Length	Min-Max Story Length
Apache	100	4-5,052	1-1,031	207,997	3,486	6	1-15
Android	70	5-4,980	3-1,140	107,555	1,193	3	1-6
Eclipse	30	142-16,700	26-2,610	264,119	124	10	1-13
Overall	**200**	-	-	**579,671**	**4,803**	**6**	**1-15**

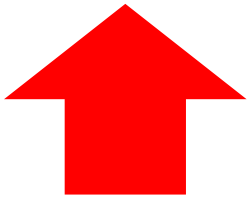


Data set

The context of the study consists of the change history of 200 projects belonging to three software ecosystems

CHARACTERISTICS OF ECOSYSTEMS UNDER ANALYSIS.

Ecosystem	#Proj.	#Classes	KLOC	#Commits	#Issues	Mean Story Length	Min-Max Story Length
Apache	100	4-5,052	1-1,031	207,997	3,486	6	1-15
Android	70	5-4,980	3-1,140	107,555	1,193	3	1-6
Eclipse	30	142-16,700	26-2,610	264,119	124	10	1-13
Overall	**200**	-	-	**579,671**	**4,803**	**6**	**1-15**



Data set

The context of the study consists of the change history of 200 projects belonging to three software ecosystems

CHARACTERISTICS OF ECOSYSTEMS UNDER ANALYSIS.

Ecosystem	#Proj.	#Classes	KLOC	#Commits	#Issues	Mean Story Length	Min-Max Story Length
Apache	100	4-5,052	1-1,031	207,997	3,486	6	1-15
Android	70	5-4,980	3-1,140	107,555	1,193	3	1-6
Eclipse	30	142-16,700	26-2,610	264,119	124	10	1-13
Overall	**200**	-	-	**579,671**	**4,803**	**6**	**1-15**

Code Smells (1/2)

Blob

Class Data Should Be Private

Complex Class

Functional Decomposition

Spaghetti Code

smells considered from the catalogues by Fowler and Brown

Code Smells (2/2)

1. **Blob Class:** a large class with different responsibilities that monopolizes most of the system's processing

2. **Class Data Should be Private:** a class exposing its attributes, violating the information hiding principle

3. **Complex Class:** a class having a high cyclomatic complexity

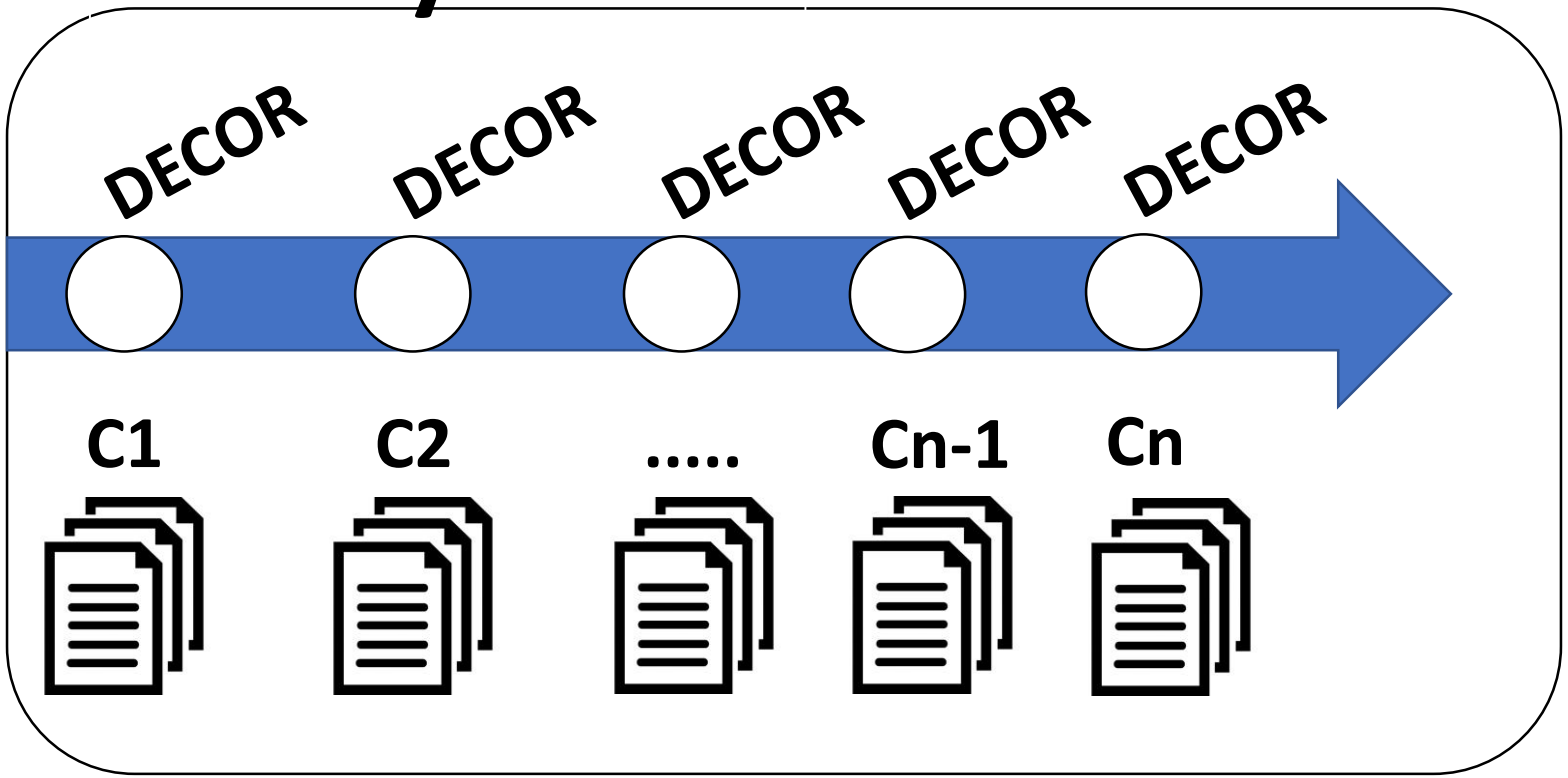
Code Smells (1/2)

4. Functional Decomposition: a class where inheritance and polymorphism are poorly used, declaring many private fields and implementing few methods

5. Spaghetti Code: a class without structure that declares long methods without parameters

Data Extraction

History Miner



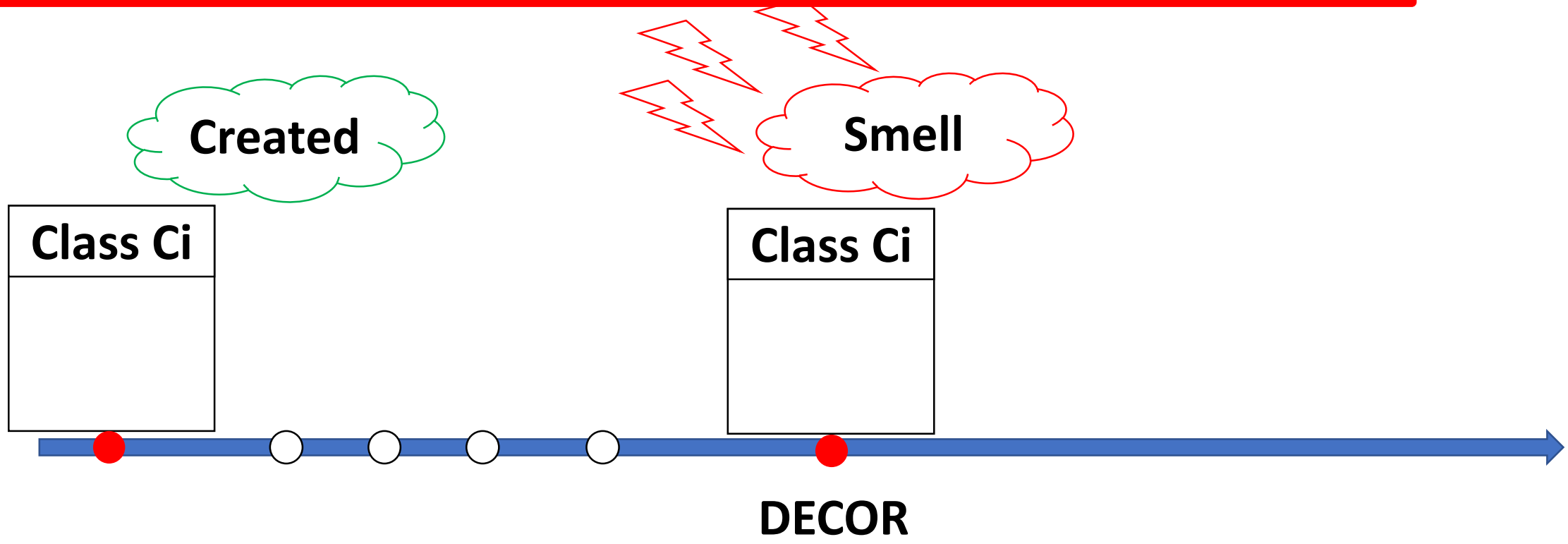
579,671

commits analyzed

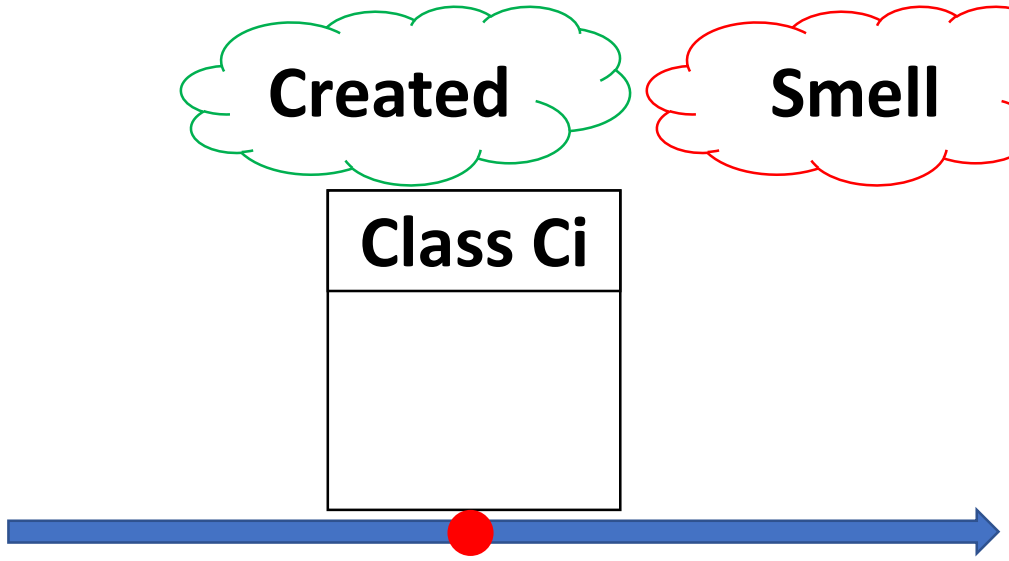
8

weeks of computation

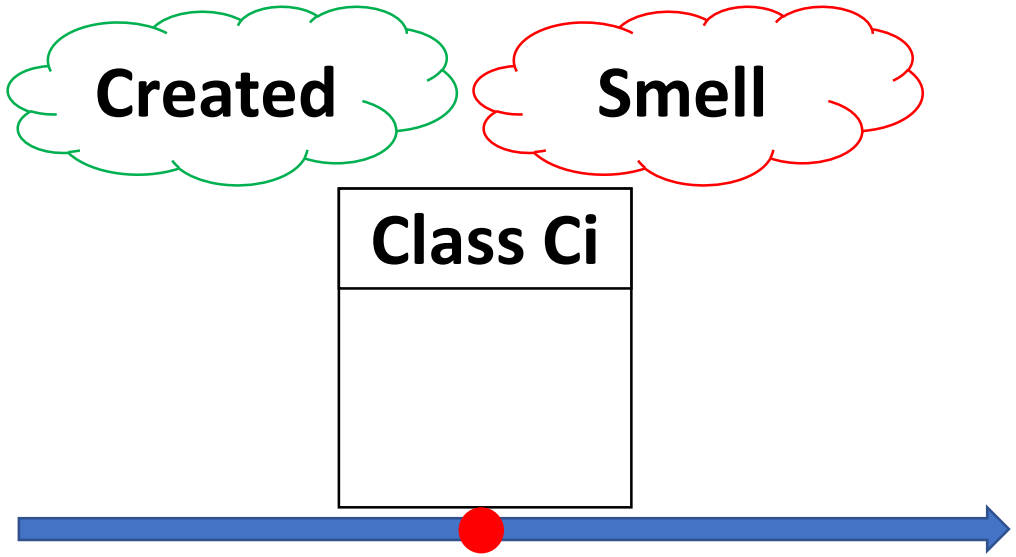
Data Extraction



Two possible scenarios



Smell introduced during the creation of the artifact



Smell introduced after several maintenance activities

Metrics

LOC

WMC

RFC

CBO

LCOM

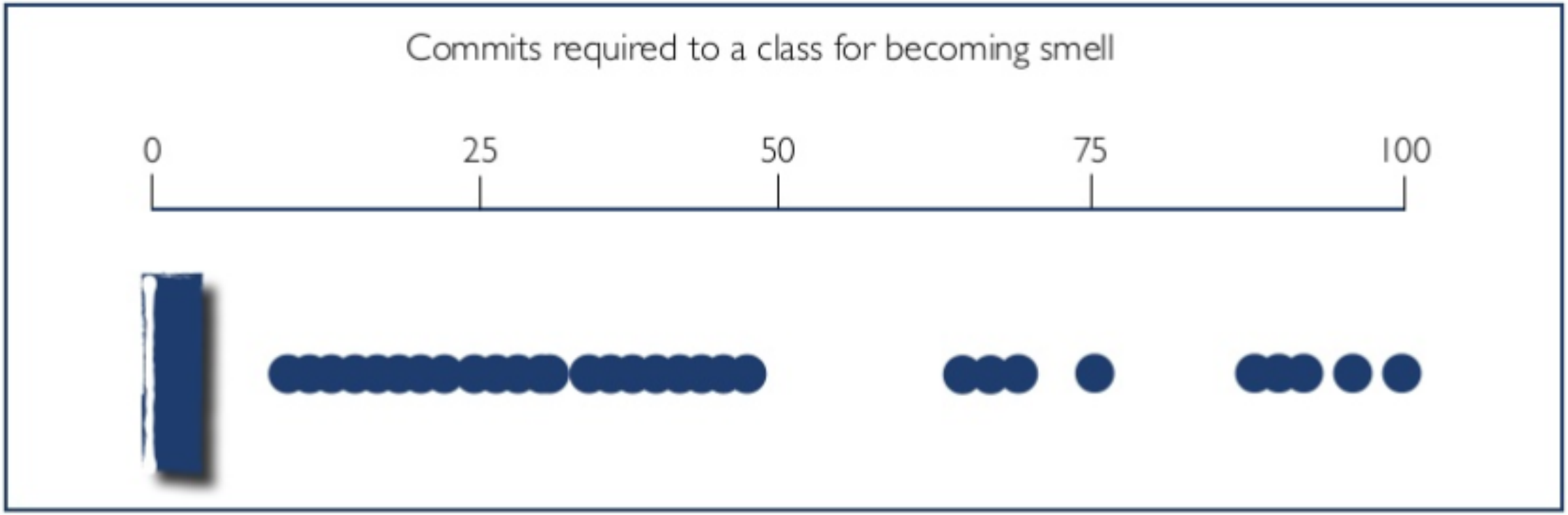
NOA

NOM

Results

RQ1

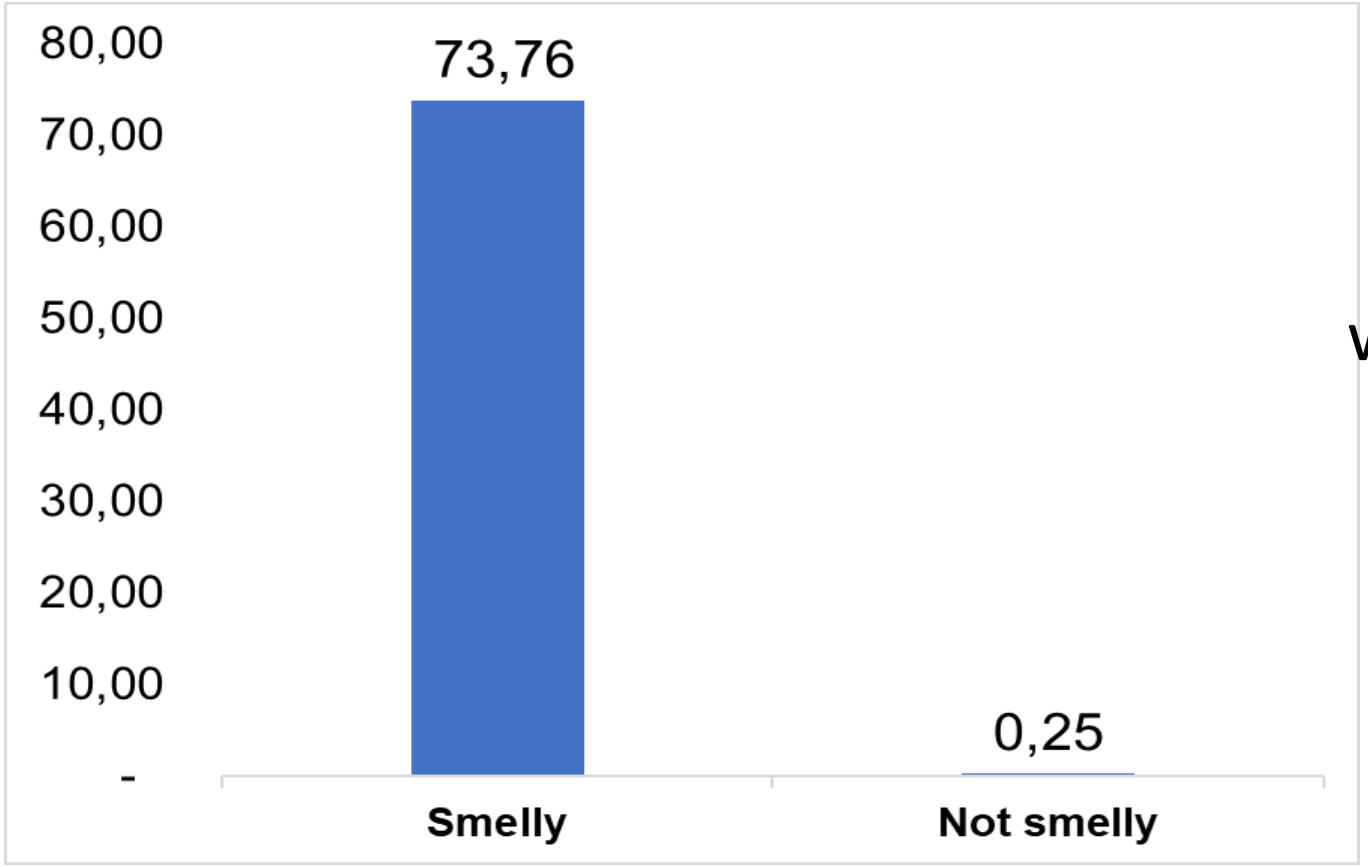
RQ1 – When are code smells introduced?



Generally, blobs affect a class since its creation

RQ1

RQ1 – When are code smells introduced?



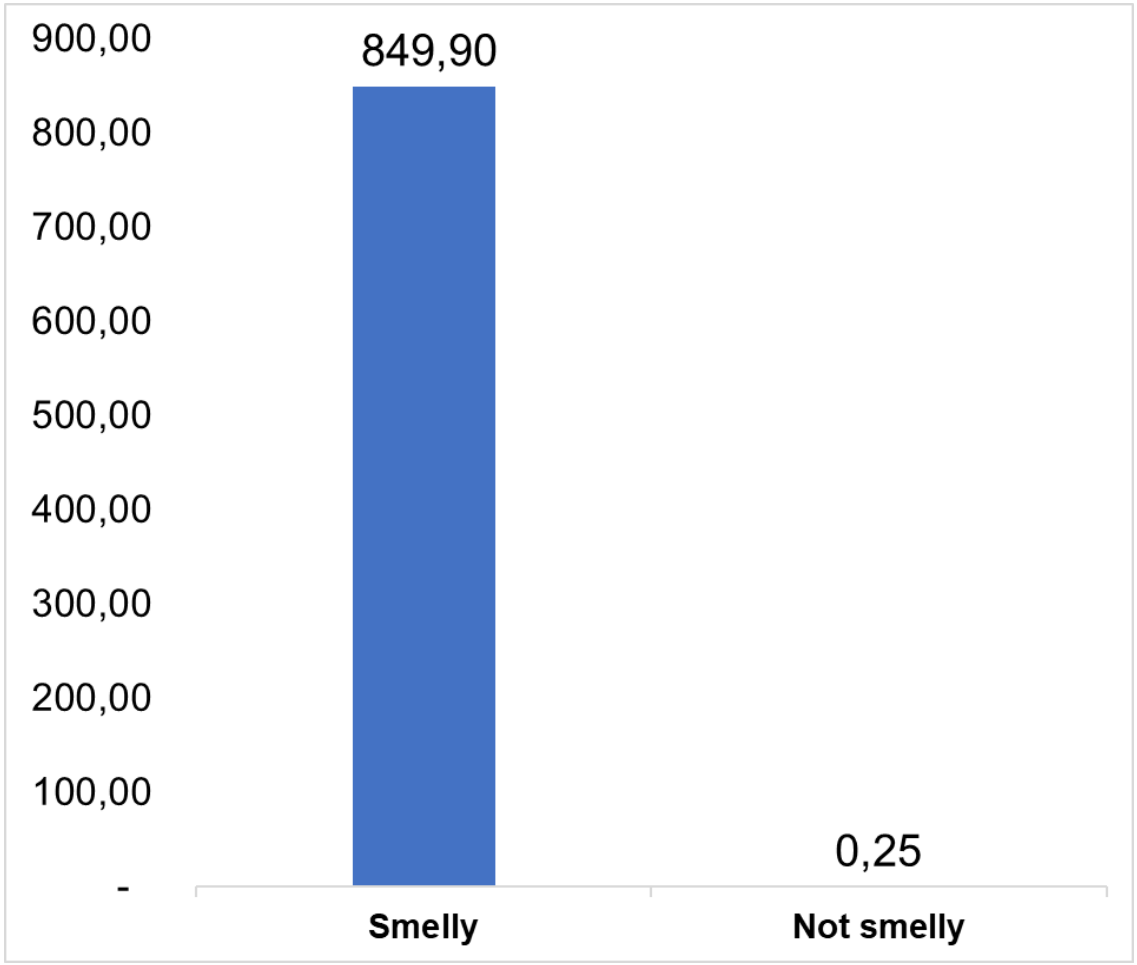
LOC slope of smelly classes is

295 times higher

with respect to non-smelly classes

RQ1

RQ1 – When are code smells introduced?

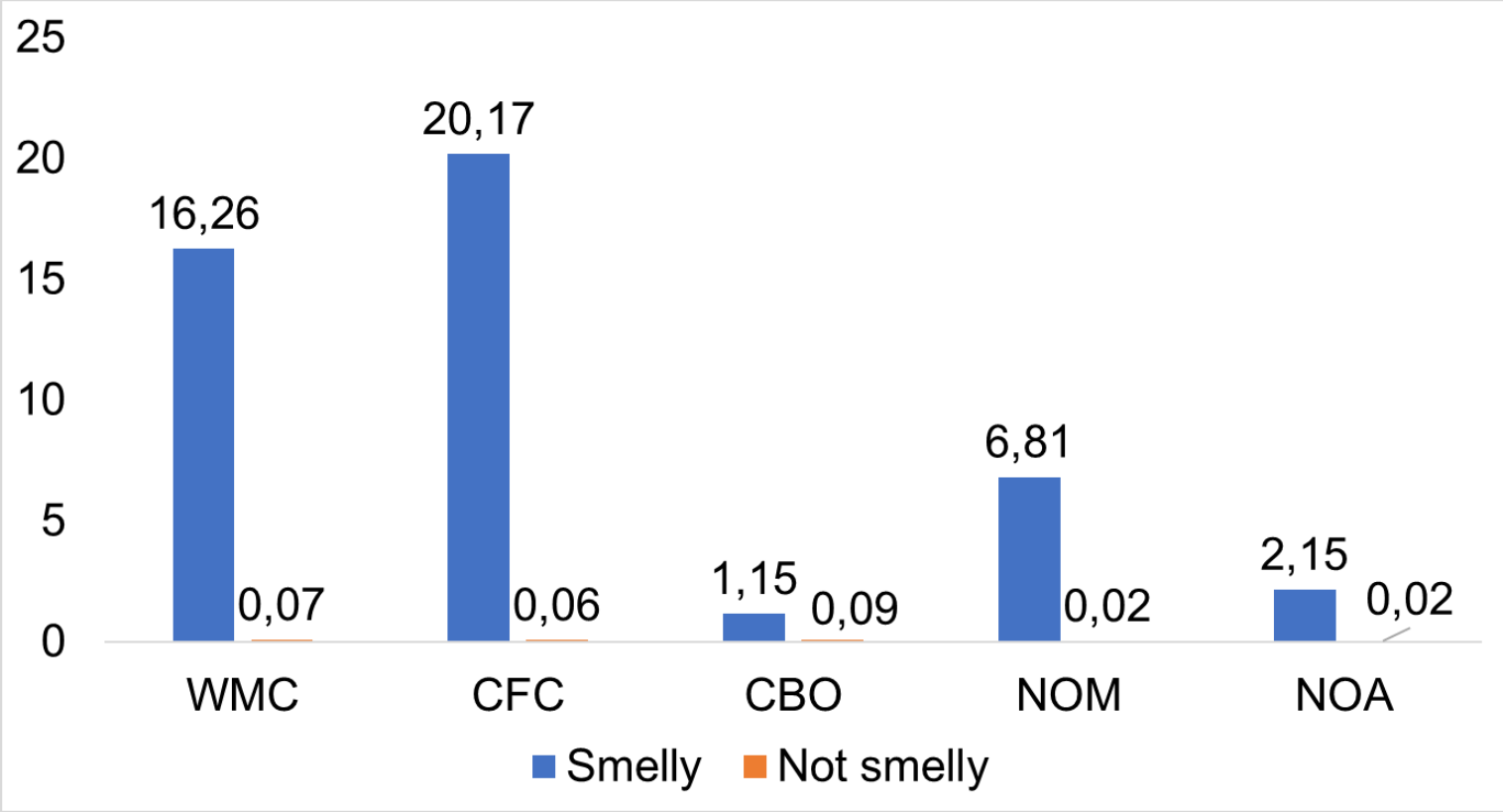


LCOM slope of smelly classes is

3399.6 times higher with respect to non-smelly classes

RQ1

RQ1 – When are code smells introduced?



Results are confirmed for other metrics

RQ1

RQ1 – When are code smells introduced?

- Most of the smell instances are **introduced** when files are **created**.
- **Blob** and **Complex Class**, where the smells manifest themselves after several changes performed on the file.
- Files that will become smelly exhibit specific trends for some quality **metric** values

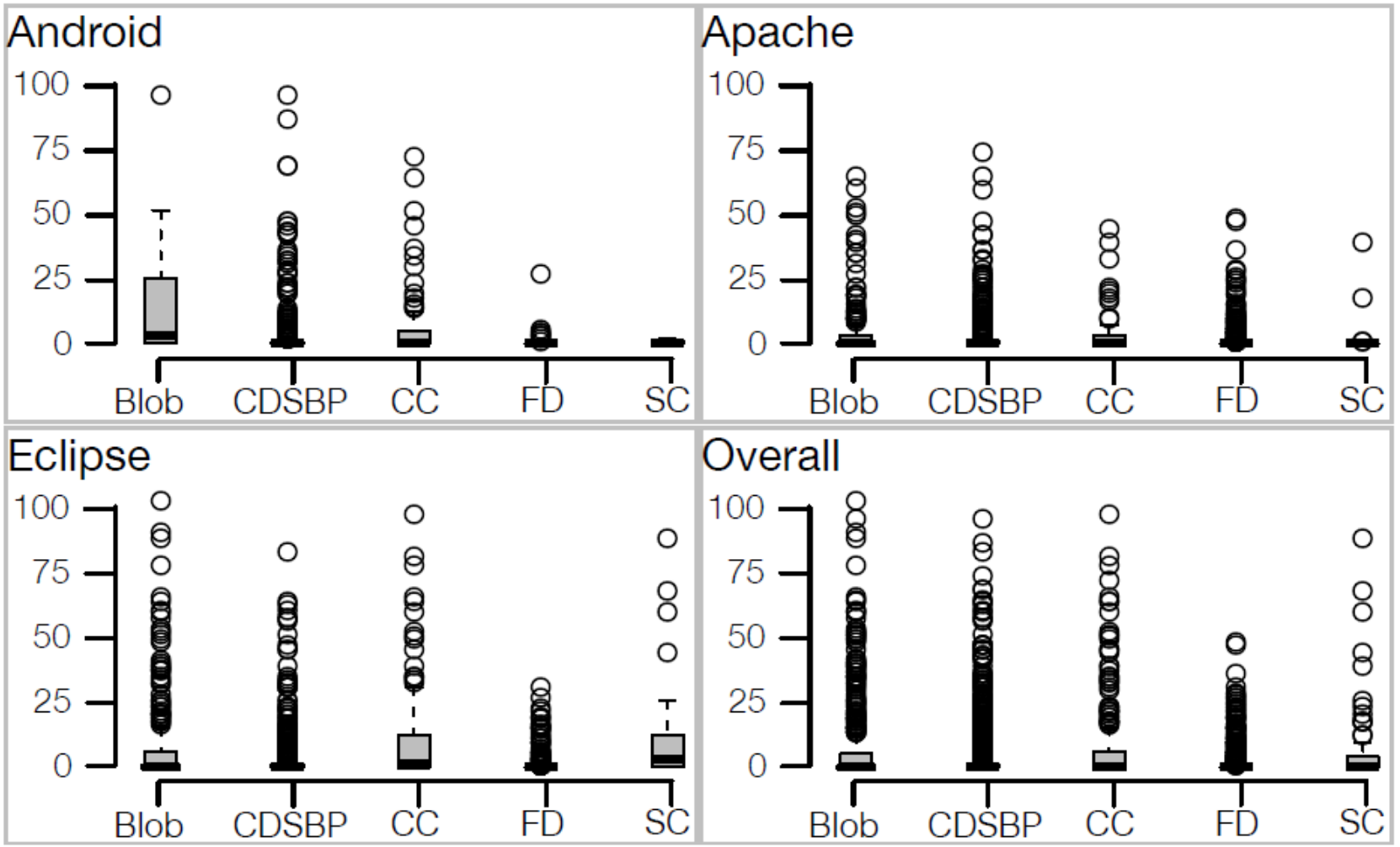
RQ2

RQ2 – Why are code smells introduced?

Tag	Description
	COMMIT GOAL TAGS
Bug fixing	The commit aimed at fixing a bug
Enhancement	The commit aimed at implementing an enhancement in the system
New feature	The commit aimed at implementing a new feature in the system
Refactoring	The commit aimed at performing refactoring operations
	PROJECT STATUS TAGS
Working on release	The commit was performed [value] before the issuing of a major release
Project startup	The commit was performed [value] after the starting of the project
	DEVELOPER STATUS TAGS
Workload	The developer had a [value] workload when the commit has been performed
Ownership	The developer was the owner of the file in which the commit introduced the smell
Newcomer	The developer was a newcomer when the commit was performed

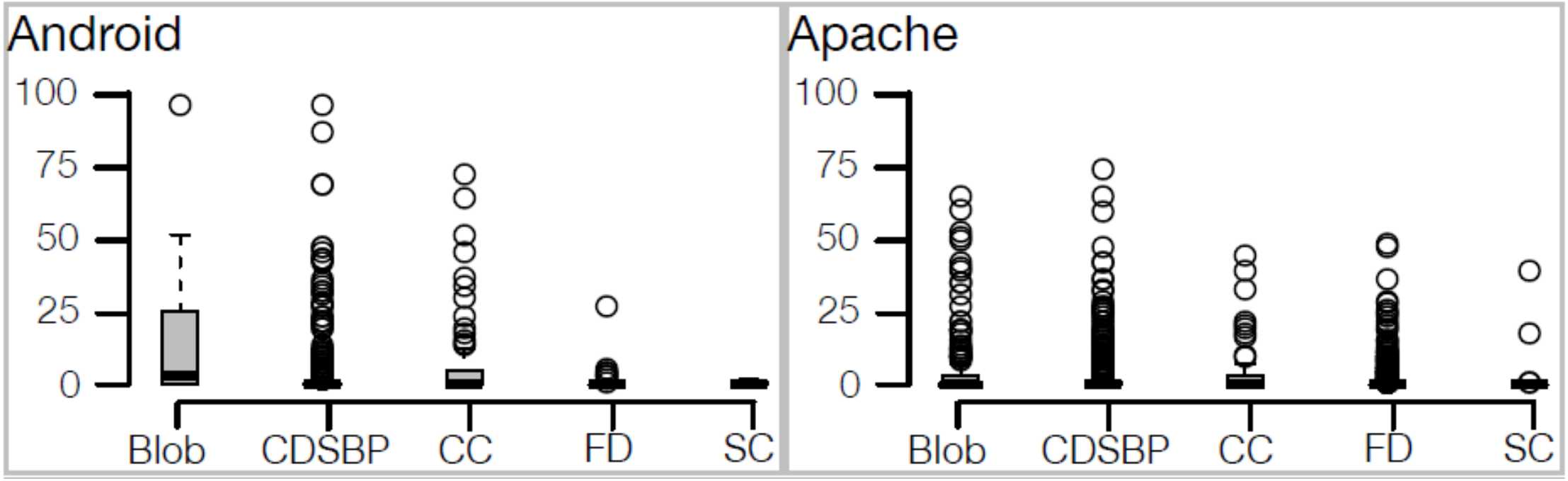
RQ2

RQ2 – Why are code smells introduced?



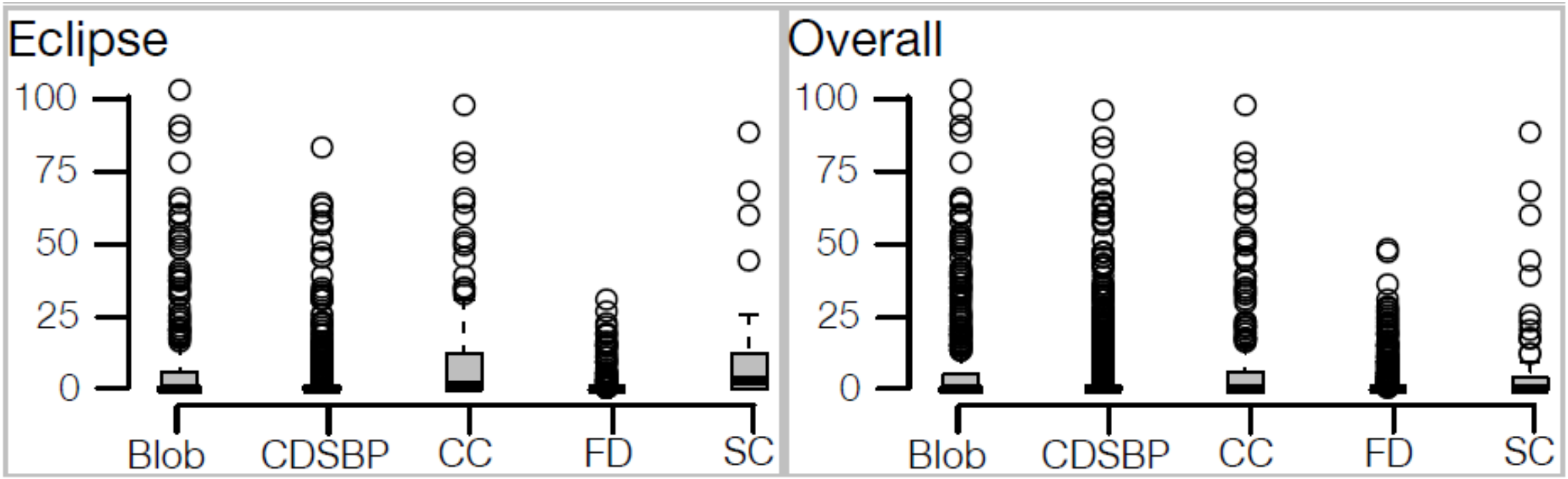
RQ2

RQ2 – Why are code smells introduced?



RQ2

RQ2 – Why are code smells introduced?



RQ2

RQ2 – Why are code smells introduced?

Smell	Android				Apache				Eclipse				Overall			
	BF	E	NF	R	BF	E	NF	R	BF	E	NF	R	BF	E	NF	R
Blob	15	59	23	3	5	83	10	2	19	55	19	7	14	65	17	4
CDSP	11	52	30	7	6	63	30	1	14	64	18	4	10	60	26	4
CC	0	44	56	0	3	89	8	0	17	52	24	7	13	66	16	5
FD	8	48	39	5	16	67	14	3	18	52	24	6	16	60	20	4
SC	0	0	100	0	0	81	4	15	8	61	22	9	6	66	17	11

BF: BUG FIXING; **E**: ENHANCEMENT;
NF: NEW FEATURE; **R**: REFACTORING.

RQ2

RQ2 – Why are code smells introduced?

Smell	Android				Apache				Eclipse				Overall			
	BF	E	NF	R	BF	E	NF	R	BF	E	NF	R	BF	E	NF	R
Blob	15	59	23	3	5	83	10	2	19	55	19	7	14	65	17	4
CDSP	11	52	30	7	6	63	30	1	14	64	18	4	10	60	26	4
CC	0	44	56	0	3	89	8	0	17	52	24	7	13	66	16	5
FD	8	48	39	5	16	67	14	3	18	52	24	6	16	60	20	4
SC	0	0	100	0	0	81	4	15	8	61	22	9	6	66	17	11

BF: BUG FIXING; **E**: ENHANCEMENT;
NF: NEW FEATURE; **R**: REFACTORING.

RQ2

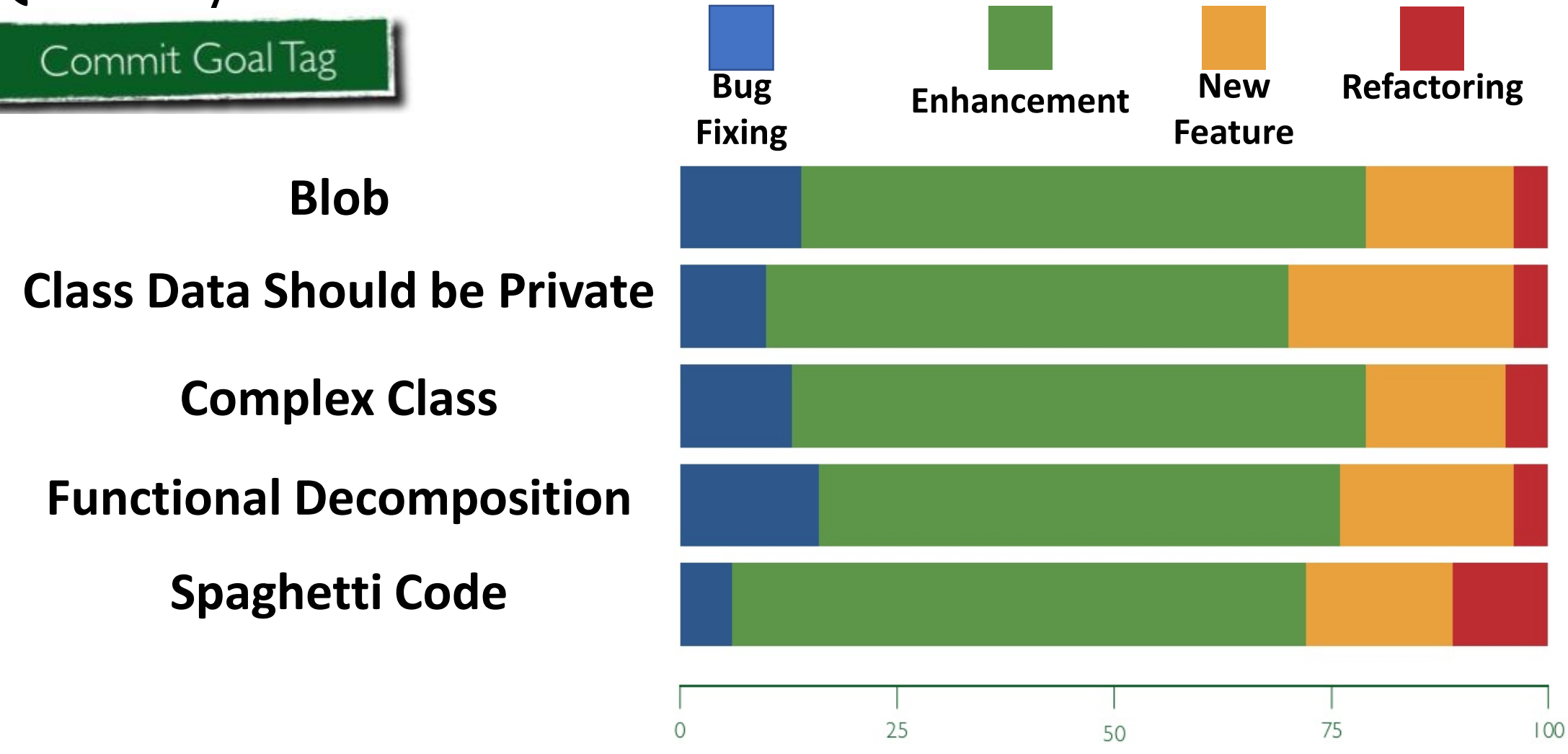
RQ2 – Why are code smells introduced?

Tag	Description
	COMMIT GOAL TAGS
Bug fixing	The commit aimed at fixing a bug
Enhancement	The commit aimed at implementing an enhancement in the system
New feature	The commit aimed at implementing a new feature in the system
Refactoring	The commit aimed at performing refactoring operations
	PROJECT STATUS TAGS
Working on release	The commit was performed [value] before the issuing of a major release
Project startup	The commit was performed [value] after the starting of the project
	DEVELOPER STATUS TAGS
Workload	The developer had a [value] workload when the commit has been performed
Ownership	The developer was the owner of the file in which the commit introduced the smell
Newcomer	The developer was a newcomer when the commit was performed

RQ2

RQ2 – Why are code smells introduced?

Commit Goal Tag



RQ2

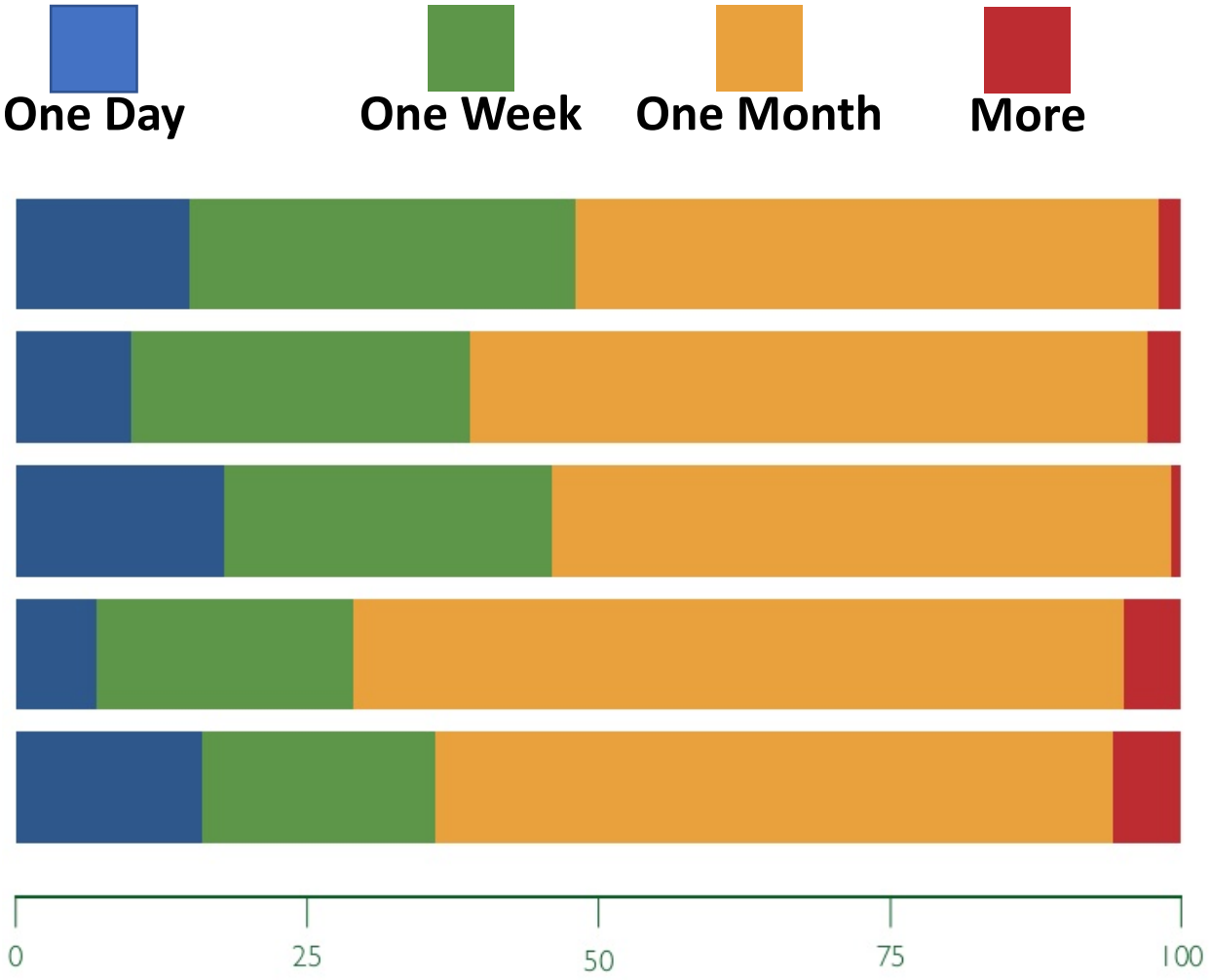
RQ2 – Why are code smells introduced?

Tag	Description
	COMMIT GOAL TAGS
Bug fixing	The commit aimed at fixing a bug
Enhancement	The commit aimed at implementing an enhancement in the system
New feature	The commit aimed at implementing a new feature in the system
Refactoring	The commit aimed at performing refactoring operations
	PROJECT STATUS TAGS
Working on release	The commit was performed [value] before the issuing of a major release
Project startup	The commit was performed [value] after the starting of the project
	DEVELOPER STATUS TAGS
Workload	The developer had a [value] workload when the commit has been performed
Ownership	The developer was the owner of the file in which the commit introduced the smell
Newcomer	The developer was a newcomer when the commit was performed

RQ2

RQ2 – Why are code smells introduced?

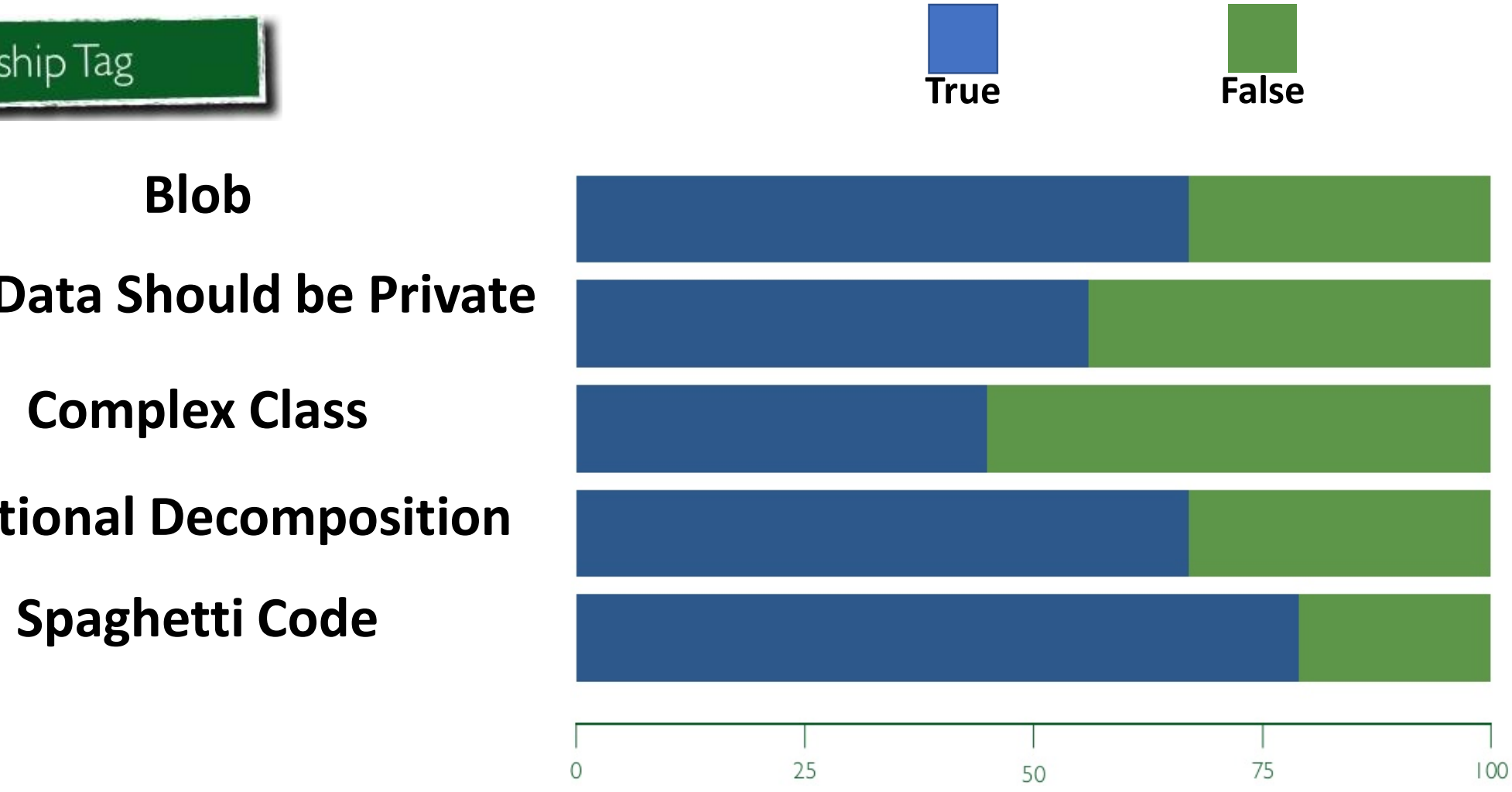
Working on ReleaseTag



RQ2

RQ2 – Why are code smells introduced?

Ownership Tag



RQ2

RQ2 – Why are code smells introduced?

- Smells are generally introduced by developers when enhancing existing features or implementing new ones.
- Last month before issuing a deadline
- Owners of the file are more prone to introduce smells

Threats to Validity

Threats to Validity

- Construct Validity. Relationship between theory and observation. **DECOR (precision above 60% and a recall of 100%)**

- Internal Validity. As for the threats that could have influenced the results. They performed the study by comparing classes affected (and not) by a specific type of smell

- External Validity. This validity is related to the possibility to generalize the results.

Conclusion and Future Work

Conclusion and Future Work

This paper presented a large-scale empirical study conducted over the commit history of 200 open source projects and aimed at understanding when and why bad code smells are introduced.

Future work. Focus on designing and developing a new generation of code quality-checkers.

Pros and Cons

Pros and Cons

+ The paper evaluate 5 types of code smells

- Only tool detection.

+ The paper evaluate many Java Projects

- Only Java Projects

Questions ?

Contact us

lab-soft

Software Engineering Lab

labsoft.dcc.ufmg.br

labsoft@dcc.ufmg.br

johnatan.si@dcc.ufmg.br