UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

A Hybrid Approach to Change Impact Analysis in Object-oriented Systems

Mívian Marques Ferreira

Advisor Mariza Bigonha
Co-advisor Kecia Ferreira

Agenda

- Introduction
- State of the Practice
- Systematic Mapping Review
- Commits Characterization
- A Heuristic for Co-change
- The Proposed Change Impact Model
- Conclusion

Introduction

Context

- To “survive”, software MUST evolve
- Evolving implies
  - Insert new functionalities
  - Correct bugs
  - Improve usability and performance

Context

- To “survive”, a software MUST evolve
- Evolving implies
  - Insert new functionalities
  - Correct bugs
  - Improve usability and performance

Modifying the software is crucial for it to continue to serve the purpose for which it was developed.

Context

- Source code is the most common software artifact modified

- Ripple Effect - a punctual modification in a part of the system’s source code can cause other parts to be modified

Context

Identifying which different parts of the system were impacted by a modification is a process called **Change Impact Analysis (CIA)**

Problem

- To perform CIA, developers need to
  - ✓ Understand the modification scope
  - ✓ A lot of knowledge about the system’s structure

Problem

- When developers don’t
  - × Understand the modification scope
  - × Have a lot of knowledge about the system’s structure

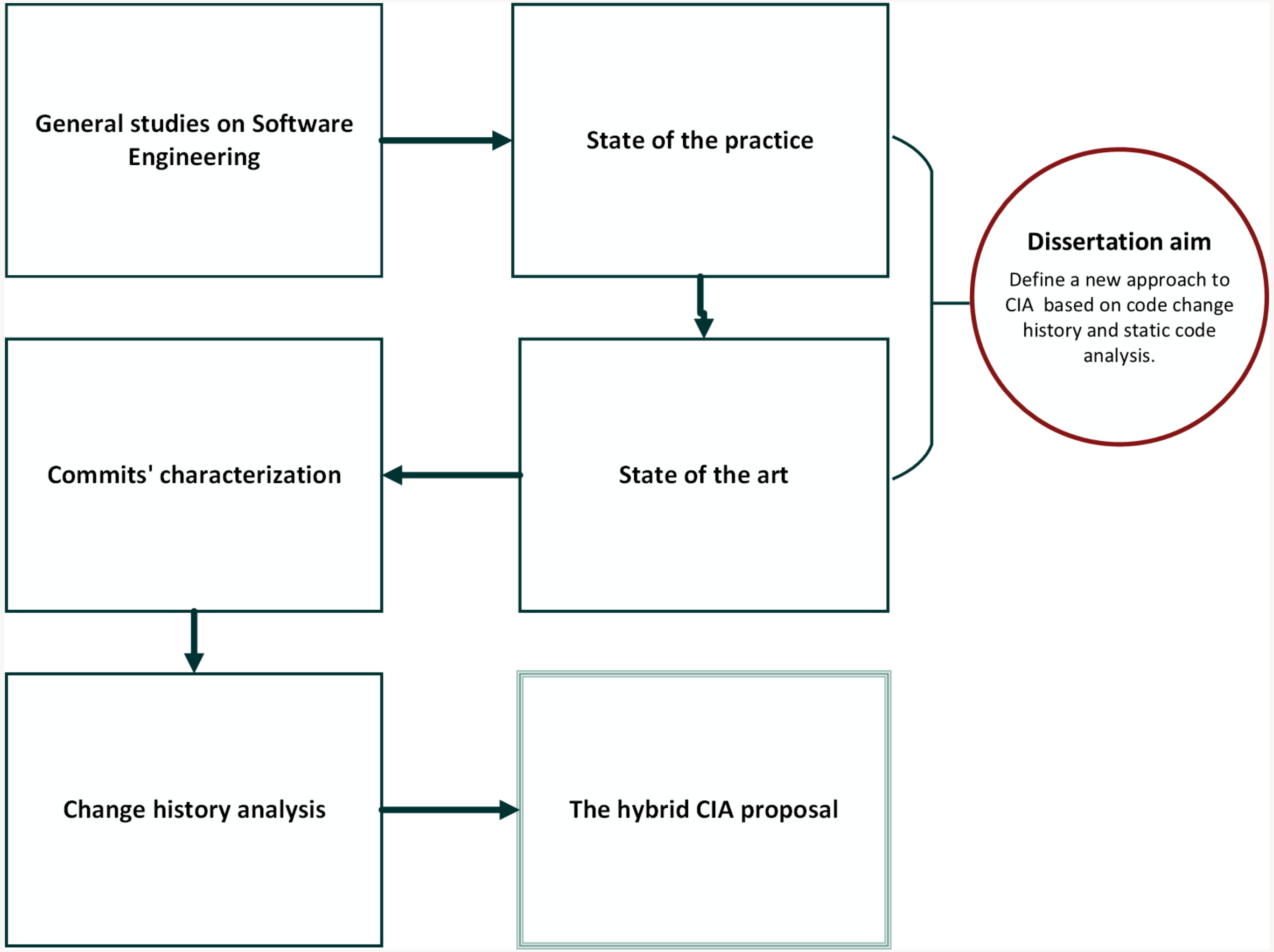
Problem

Need for Methods and Tools for CIA

Aim

This Ph.D. dissertation aims to
define, **implement** and **evaluate**
a new method for change impact analysis
of **classes** in **OOS**

Dissertation Method



General studies on Software Engineering
State of the practice
State of the art
Commits' characterization
Change history analysis
The hybrid CIA proposal
Dissertation aim
Define a new approach to CIA based on code change history and static code analysis.

State of the Practice

Aim

This study aimed to understand the gap between the research and the practice of software maintenance.

Software Maintenance Topics

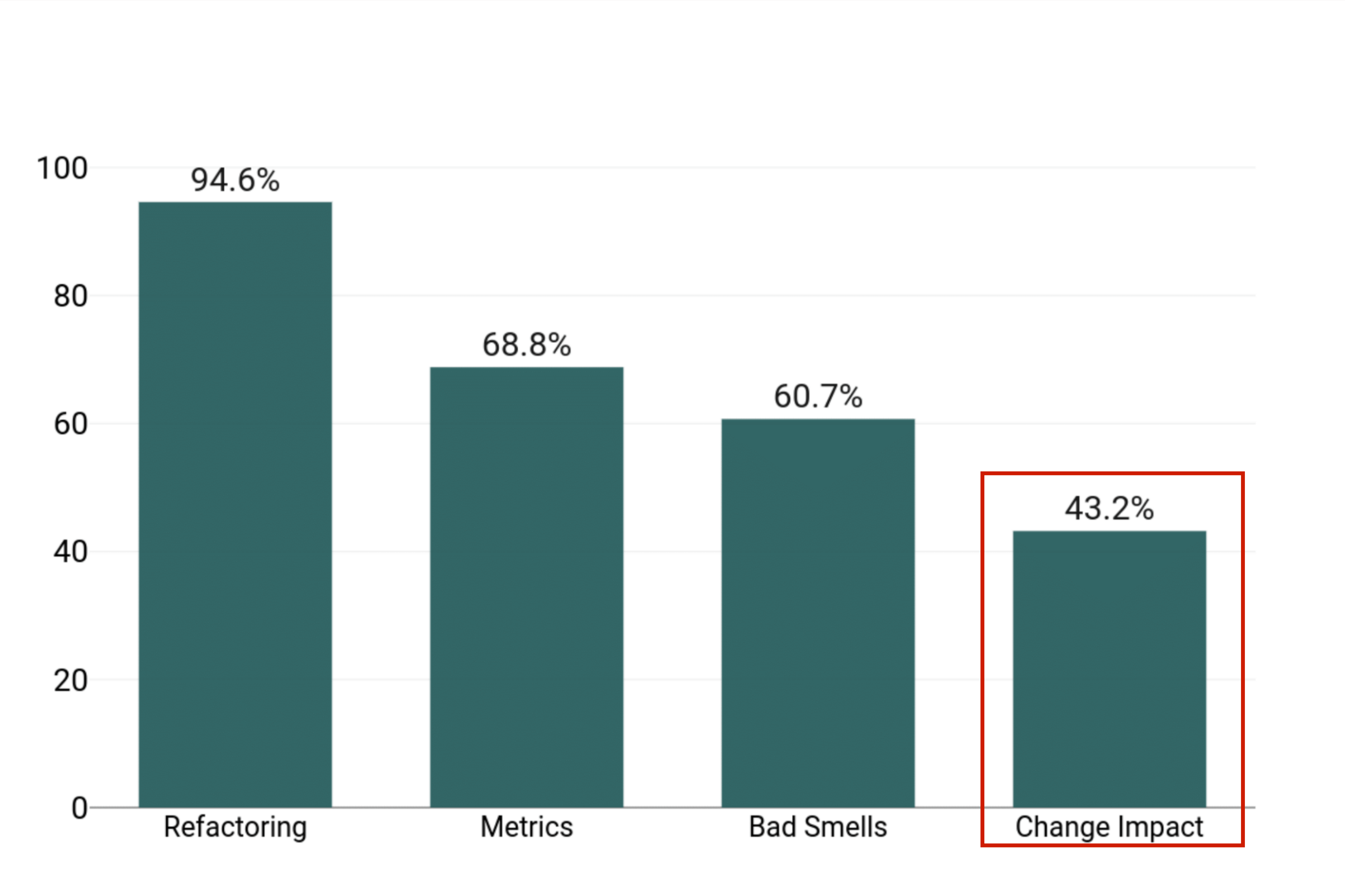
- Refactoring
- Software Metrics
- Bad Smell
- **Change Impact Analysis**

Survey

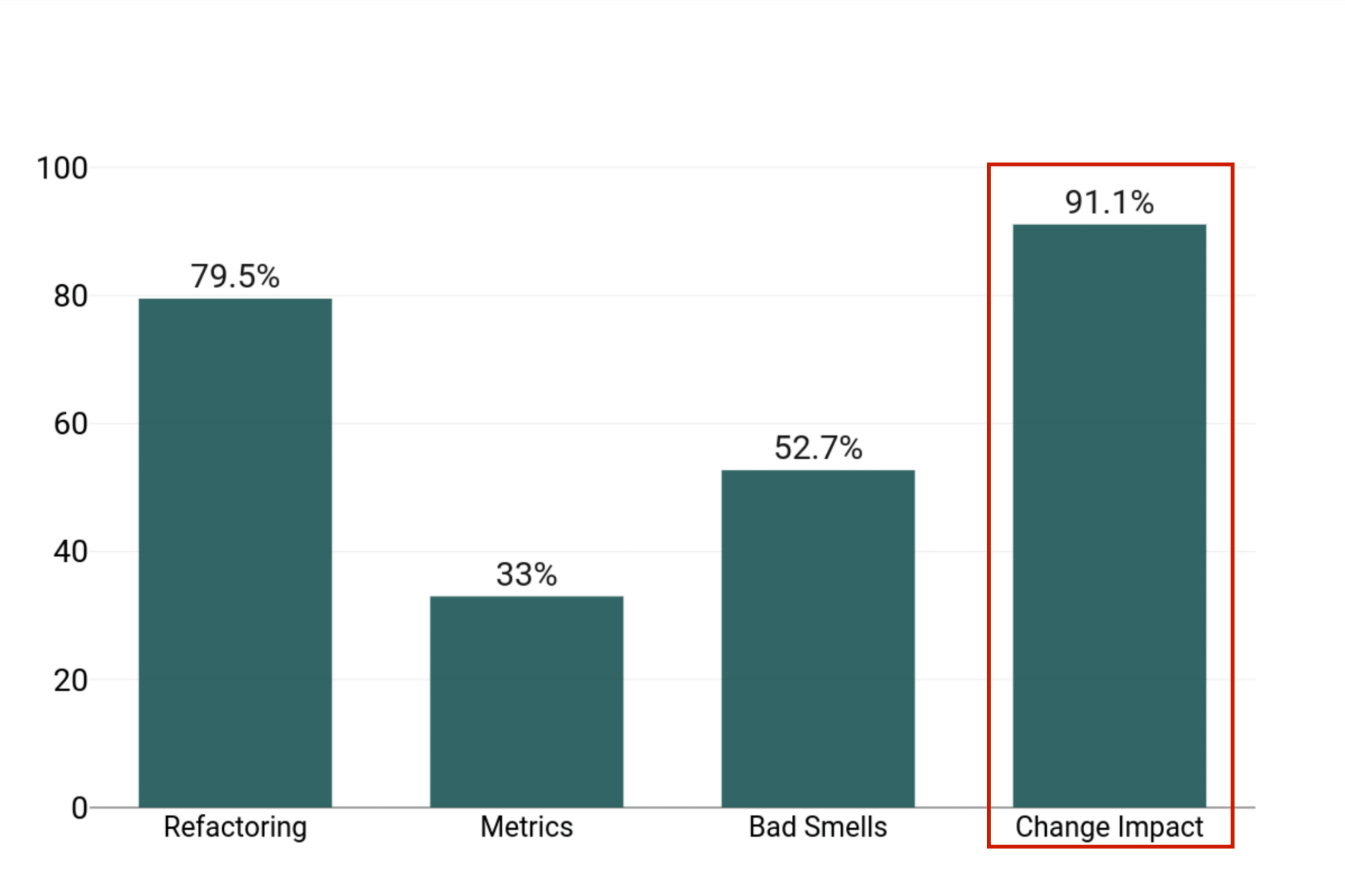
- 112 software development professionals
- 92 companies
- 12 countries
- Participants Characterization
  - Undergraduate degrees, certificate programs, or a master's degrees correspond to 95.5% of the sample
  - 89.3% have more than 5 years of professional experience

Research Questions & Results

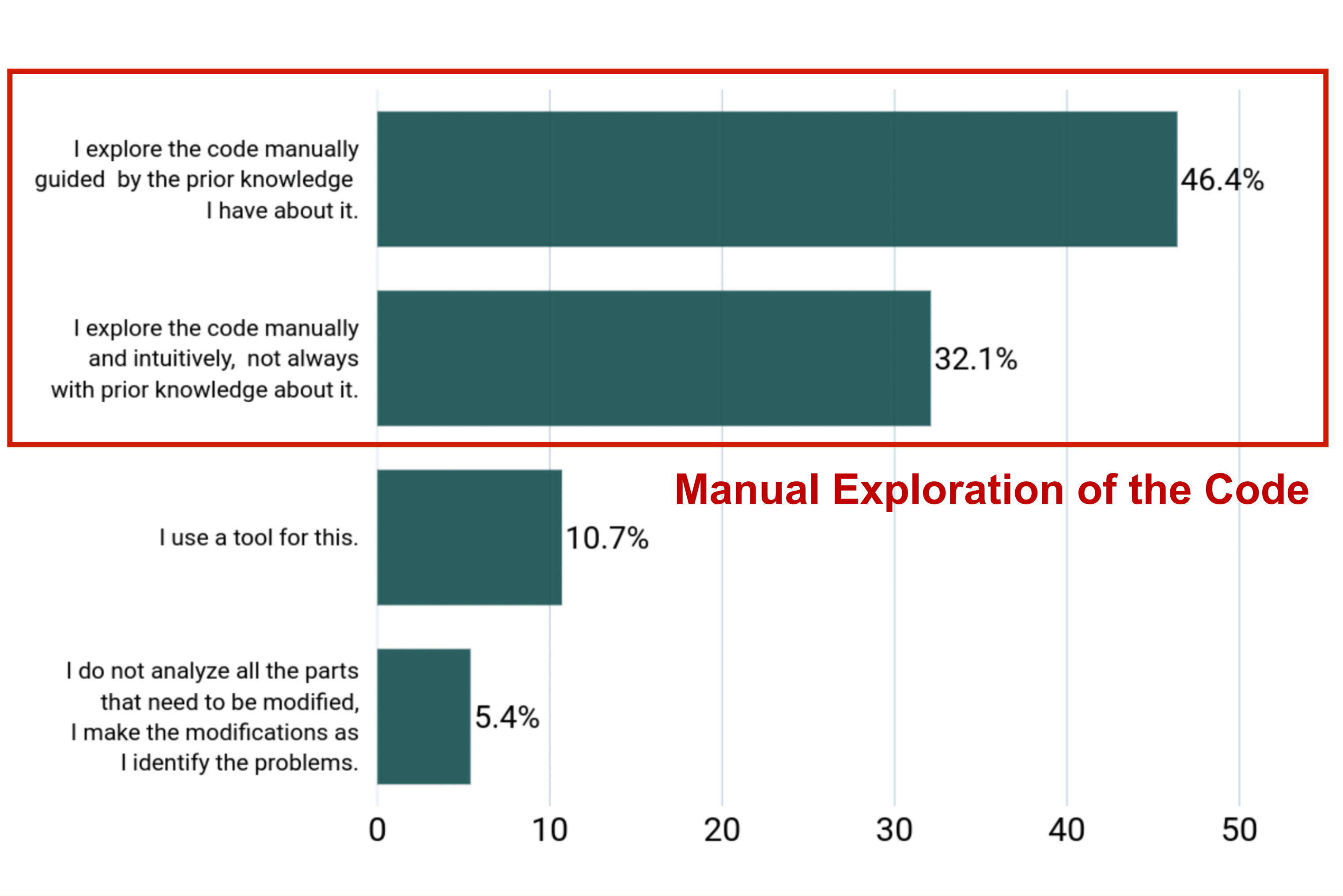
RQ1. Are developers familiar with the concepts of software metrics, bad smells, refactoring, and change impact analysis?



RQ2. Do practitioners apply software metrics, refactoring, bad smells, and change impact analysis in practice?



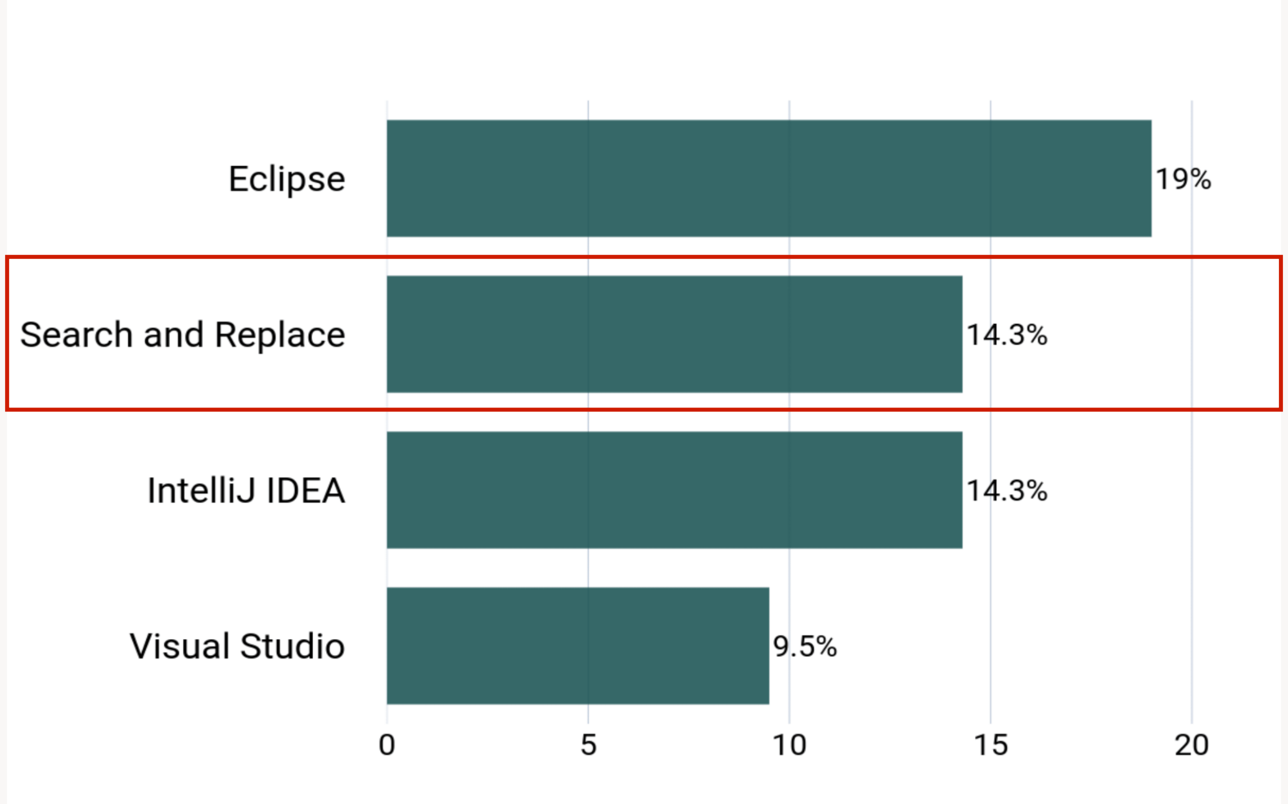
RQ3. How do practitioners perform change impact analysis?



Manual Exploration of the Code

RQ4.How practitioners perform change impact analysis?

Change Impact Analysis Tools



Final Remarks

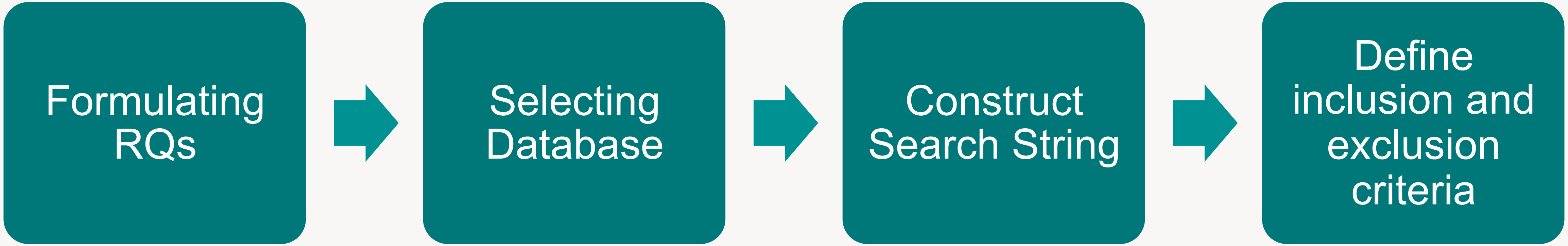
- ✓ Software metrics are not fully applied in practice
- ✓ Refactoring is a popular concept, but only simple refactoring techniques are used
- ✓ **Change impact analysis is not adequately performed in practice**
- ✓ Practitioners still face difficulties in performing source code maintenance
- ✓ We still have many challenges to bringing theoretical knowledge into practice

Systematic Mapping Review

Aim

The mapping aims to carry out a broad characterization of the methods and tools proposed for CIA.

Planning



Formulating RQs

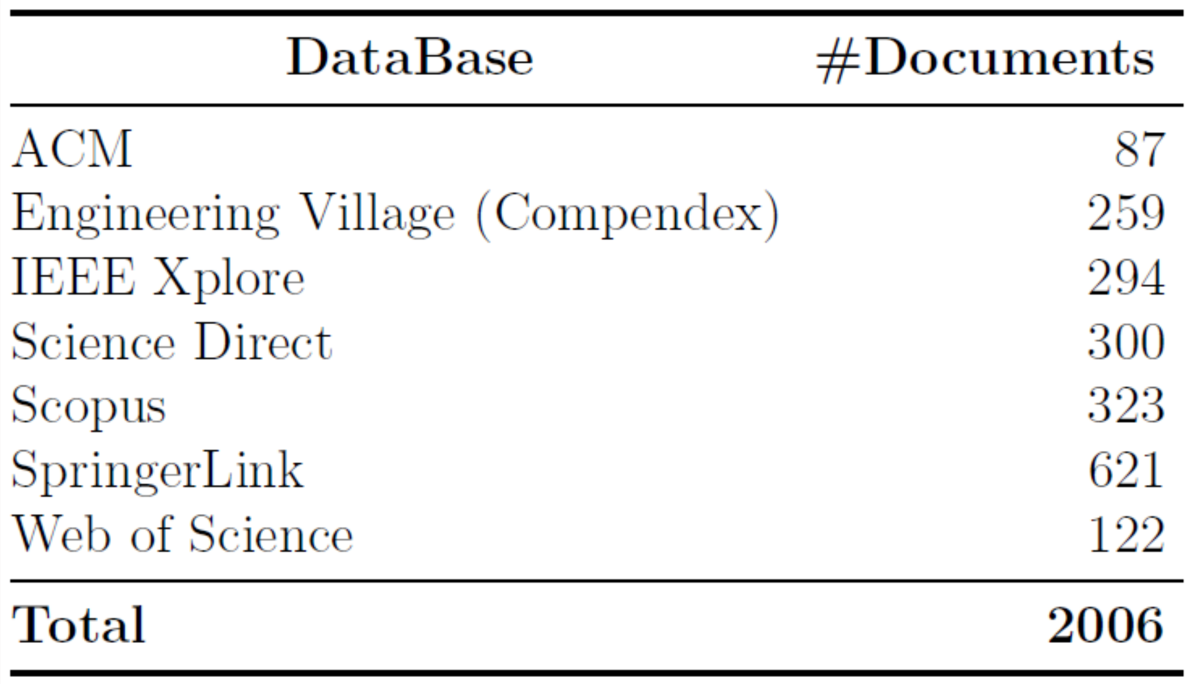
Selecting Database

Construct Search String

Define inclusion and exclusion criteria

Eletronic Database

- Main digital libraries and eletronic database of software engineering publications.

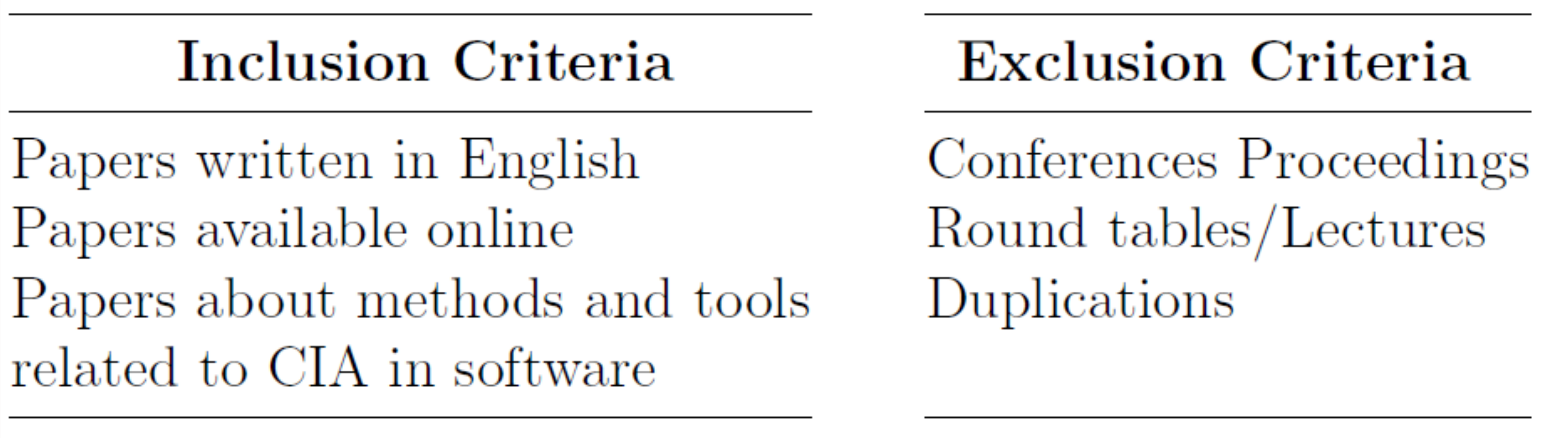


DataBase	#Documents
ACM	87
Engineering Village (Compendex)	259
IEEE Xplore	294
Science Direct	300
Scopus	323
SpringerLink	621
Web of Science	122
Total	**2006**

Search String

(“change impact” OR “change propagation” OR “modification impact” OR “modification propagation” OR “ripple effect” OR “co-change” OR “software modification”) AND “software maintenance”

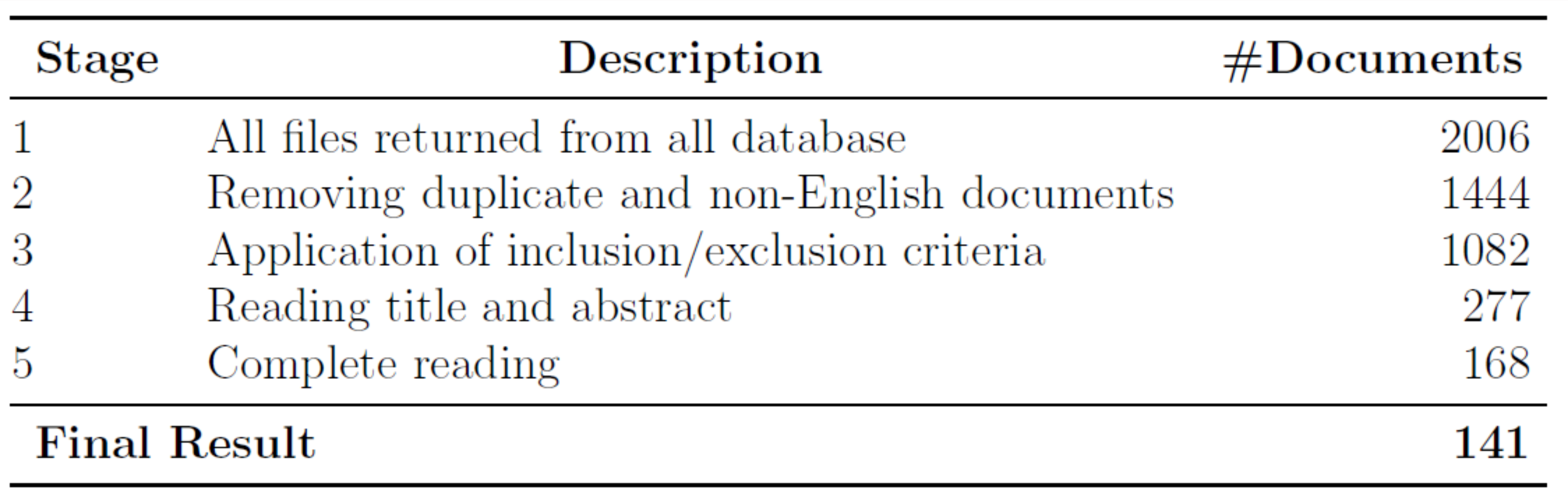
Inclusion & Exclusion Criteria



Inclusion Criteria
Papers written in English
Papers available online
Papers about methods and tools related to CIA in software

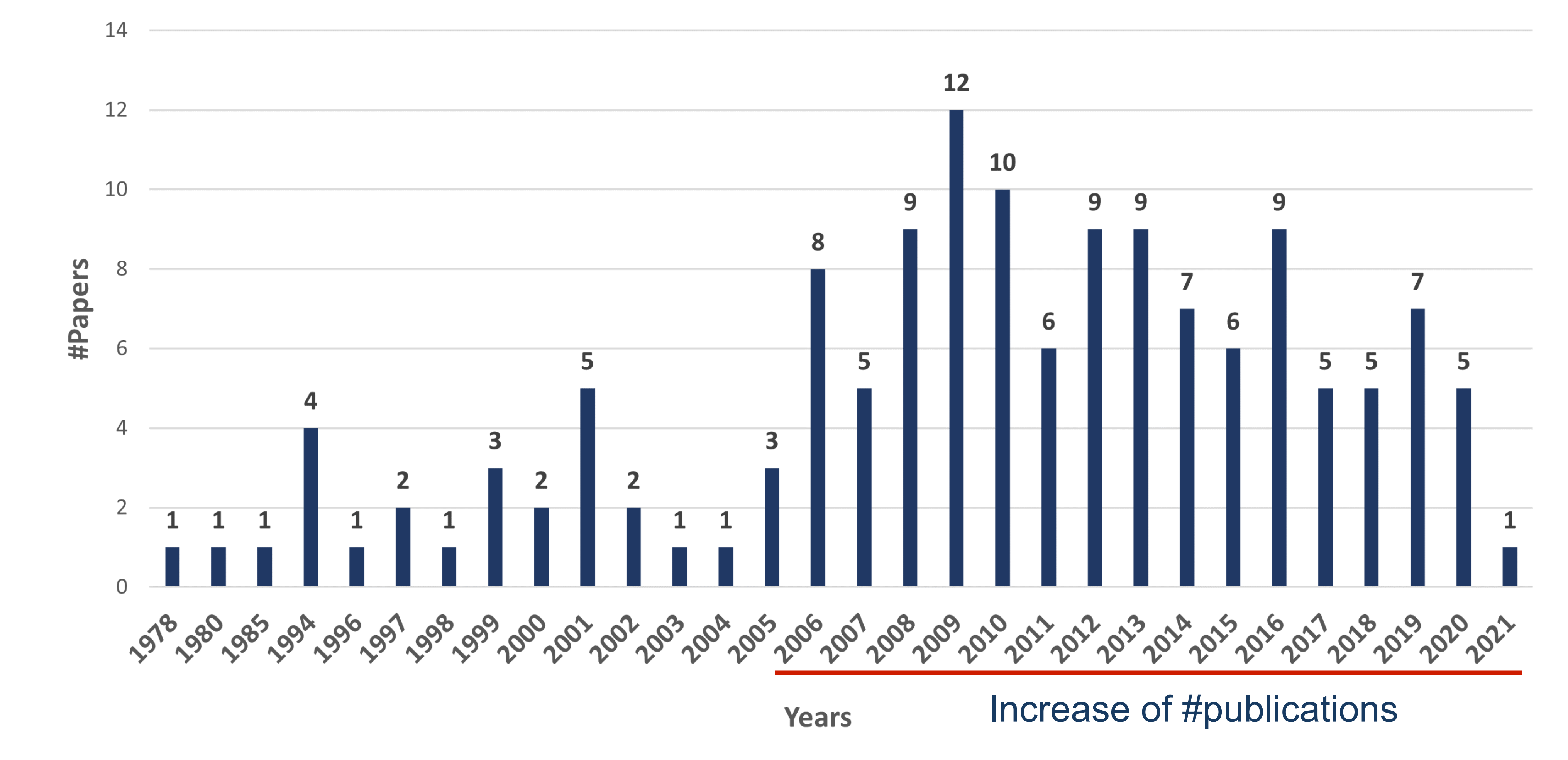
Exclusion Criteria
Conferences Proceedings
Round tables/Lectures
Duplications

Selection of the Studies



Stage	Description	#Documents
1	All files returned from all database	2006
2	Removing duplicate and non-English documents	1444
3	Application of inclusion/exclusion criteria	1082
4	Reading title and abstract	277
5	Complete reading	168
Final Result		**141**

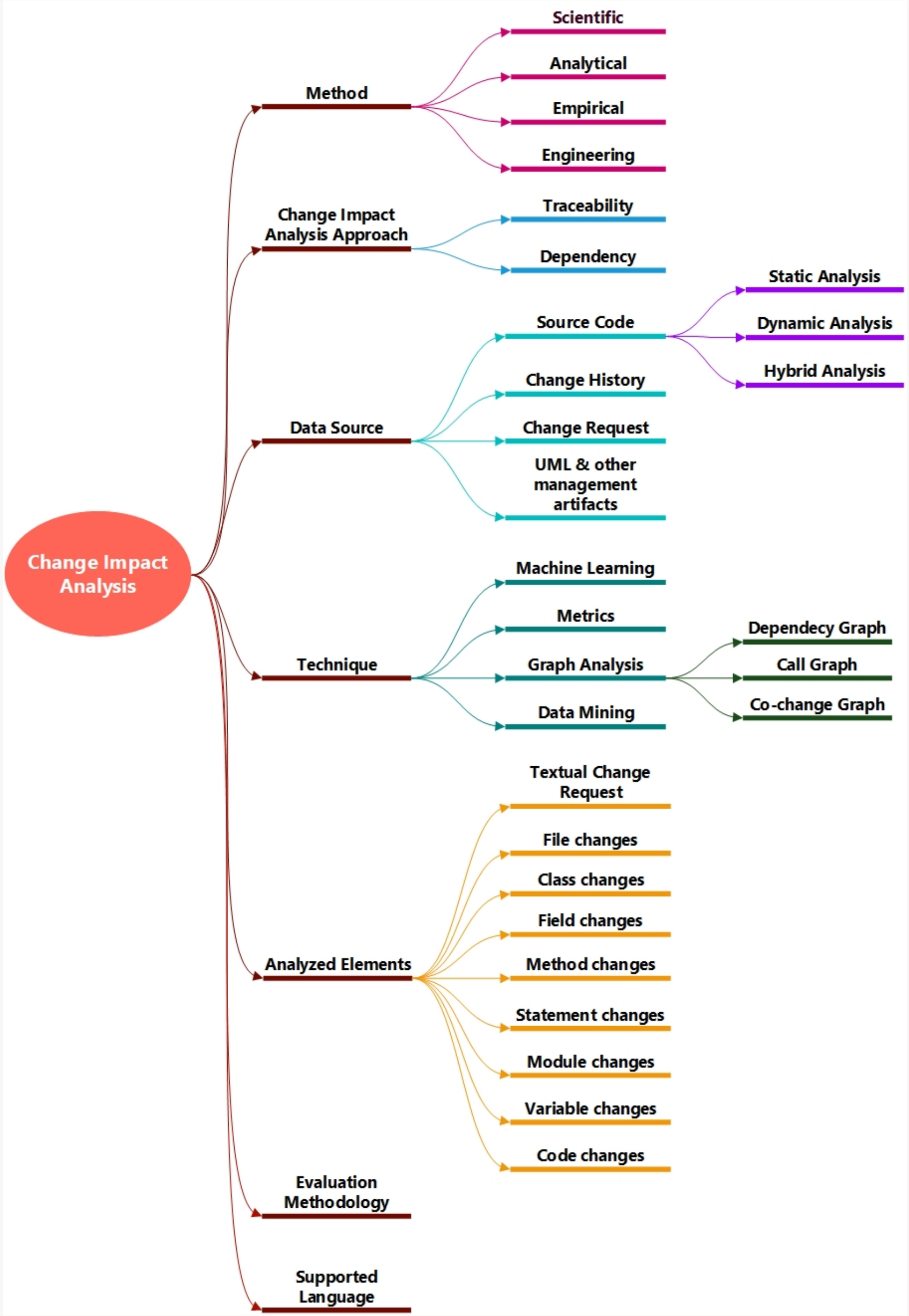
Change Impact Analysis thru the Time



Increase of #publications

Framework for CIA Studies' Characteristics

- Extension of a framework proposed by Li et al. (2013)



Research Questions & Results

RQ1. Which approaches and tools are proposed for CIA?

- Classification of the work into
  - Method (76%)
  - Method and Tool (17.7%)
  - Tool (4.3%)

Many methods, fewer tools.

RQ2. Which are the characteristics of these approaches and tools?

- **Change Impact Analysis Approach**
  - Dependency (69.7%)
  - Traceability (30.3%)

The academic community understands that analyzing dependencies between software artifacts helps in CIA

RQ2. Which are the characteristics of these approaches and tools?

- **Data Source**
  - Souce code (56.3%)

    ↳ Static analysis is the most used by researchers
  - Change History (30.3%)

✓ Methods and Tools use more than one data source for CIA

RQ2. Which are the characteristics of these approaches and tools?

- **Technique**
  - Graph Analysis (42.6%)

    ↳ The dependency graph is the most used by researchers
  - Data Mining (12.7%)
  - Metrics (7.9%)

RQ2. Which are the characteristics of these approaches and tools?

- **Analyzed Elements**
  - Source code change (43.7%)
    - ↳ Classes, method, module, variable...
      - ↳ Most common (33.9%)
  - File (10.6%)
  - Requirements (8.5%)

RQ2. Which are the characteristics of these approaches and tools?

- **Supported Language**
  - Java (17.7%)
  - Language Independent (16%)

The object-oriented paradigm is the most used by researchers when proposing methods and tools for CIA

RQ3. Which methods and metrics did the studies use in evaluating these approaches and tools?

- **Evaluation Methods**
  - Empirical Studies (31%)
  - Case Study (29%)
  - Comparative Analysis and Usage Examples (9,9%)

RQ3. Which methods and metrics did the studies use in evaluating these approaches and tools?

- **Evaluation Metrics**
  - Used by 68.3% of the analyzed papers
    - Recall and Precision (16.2%)
    - Recall, Precision and F-measure (8.5%)

Final Remarks

- We analyzed 141 studies published between 1978 and 2021
- We extended the framework proposed by Li et al. and extracted data from the publications
    - ✓ The studies proposed more methods than tools for change impact analysis, it is necessary to develop tools to support them.
    - ✓ The most applied technique for CIA is graph analysis.
    - ✓ Source code is the most commonly artifact used in methods and tools for CIA
    - ✓ Most methods and tools for CIA support object-oriented software systems.

Final Remarks

- State of the Practice + State of the Art
  - There is a relevant demand for more practical and effective approaches for CIA.

- We considered that change history analysis based on commits' data and static analysis are promissing approaches

Commits Characterization

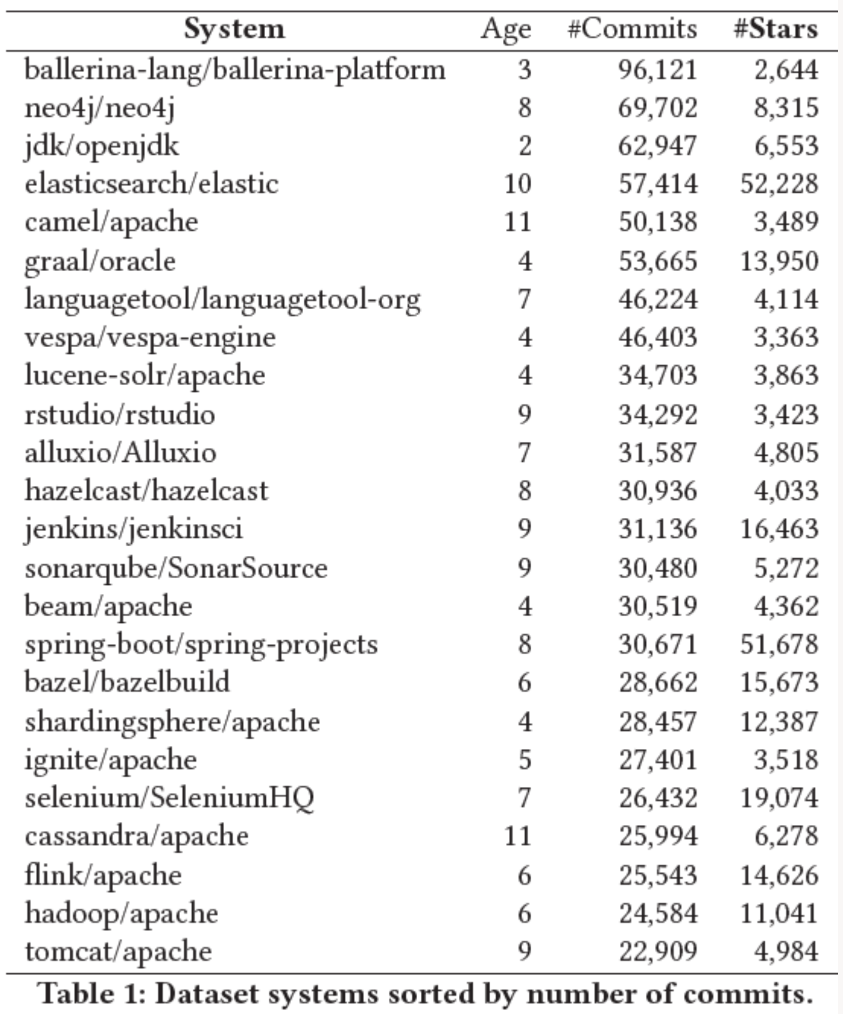
Aim

- Characterize commits regarding
  - Number of modified files
  - Number of modified source-code files
  - Category of activities
  - Number of modified files by category
  - Co-occurrences of activities
  - Time interval in which developers perform commits

Dataset

24 Java projects from GitHub

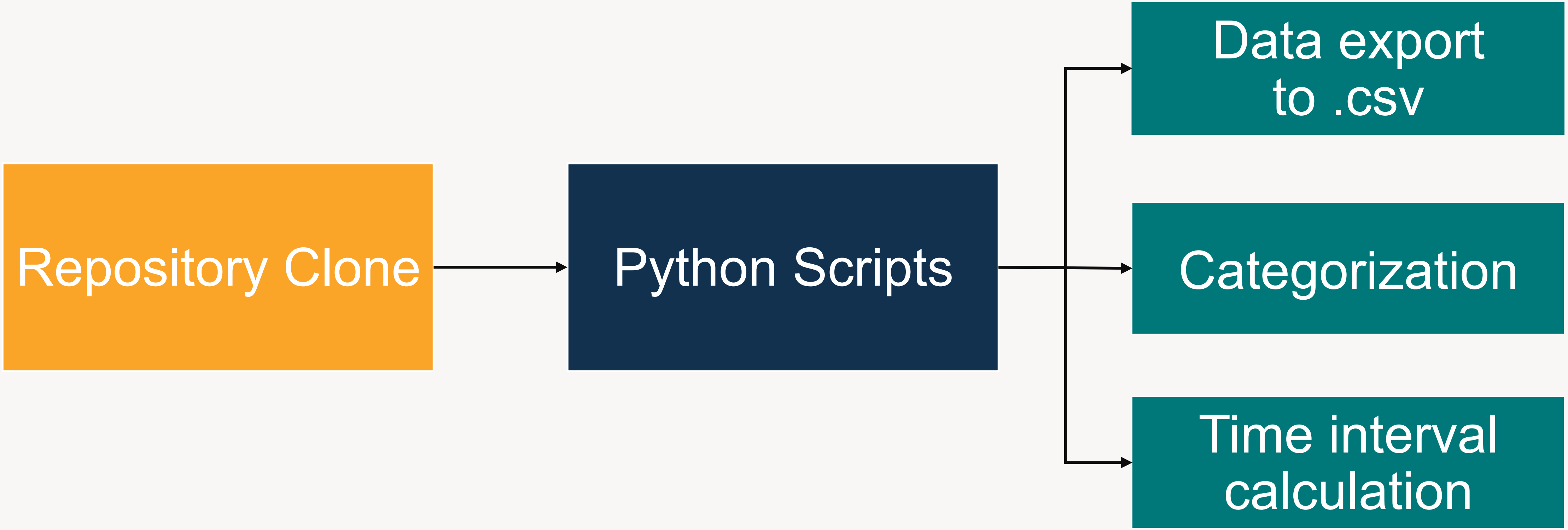
- ✓ Most popular projects - #stars
- ✓ Mature systems – 3 to 11 Years
- ✓ ≈ 1 million commits analyzed



System	Age	#Commits	**#Stars**
ballerina-lang/ballerina-platform	3	96,121	2,644
neo4j/neo4j	8	69,702	8,315
jdk/openjdk	2	62,947	6,553
elasticsearch/elastic	10	57,414	52,228
camel/apache	11	50,138	3,489
graal/oracle	4	53,665	13,950
languagetool/languagetool-org	7	46,224	4,114
vespa/vespa-engine	4	46,403	3,363
lucene-solr/apache	4	34,703	3,863
rstudio/rstudio	9	34,292	3,423
alluxio/Alluxio	7	31,587	4,805
hazelcast/hazelcast	8	30,936	4,033
jenkins/jenkinsci	9	31,136	16,463
sonarqube/SonarSource	9	30,480	5,272
beam/apache	4	30,519	4,362
spring-boot/spring-projects	8	30,671	51,678
bazel/bazelbuild	6	28,662	15,673
shardingsphere/apache	4	28,457	12,387
ignite/apache	5	27,401	3,518
selenium/SeleniumHQ	7	26,432	19,074
cassandra/apache	11	25,994	6,278
flink/apache	6	25,543	14,626
hadoop/apache	6	24,584	11,041
tomcat/apache	9	22,909	4,984

Table 1: Dataset systems sorted by number of commits.

Data Extraction



Repository Clone

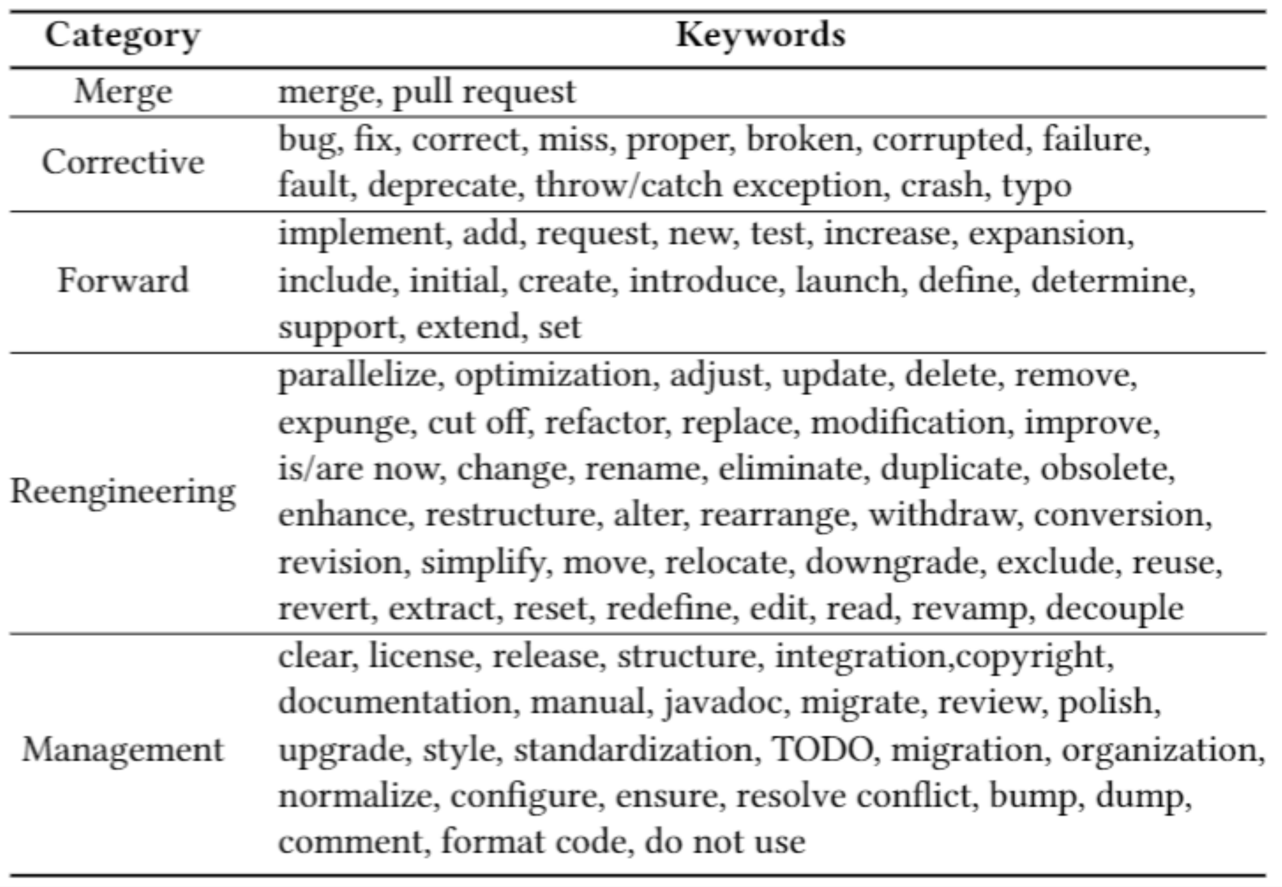
Python Scripts

Data export to .csv

Categorization

Time interval calculation

Categorization



Category	Keywords
Merge	merge, pull request
Corrective	bug, fix, correct, miss, proper, broken, corrupted, failure, fault, deprecate, throw/catch exception, crash, typo
Forward	implement, add, request, new, test, increase, expansion, include, initial, create, introduce, launch, define, determine, support, extend, set
Reengineering	parallelize, optimization, adjust, update, delete, remove, expunge, cut off, refactor, replace, modification, improve, is/are now, change, rename, eliminate, duplicate, obsolete, enhance, restructure, alter, rearrange, withdraw, conversion, revision, simplify, move, relocate, downgrade, exclude, reuse, revert, extract, reset, redefine, edit, read, revamp, decouple
Management	clear, license, release, structure, integration,copyright, documentation, manual, javadoc, migrate, review, polish, upgrade, style, standardization, TODO, migration, organization, normalize, configure, ensure, resolve conflict, bump, dump, comment, format code, do not use

Basead on Hattori e Lanza (2008)

Time Interval Calculation

1. Group commits by author
2. Calculate the time interval

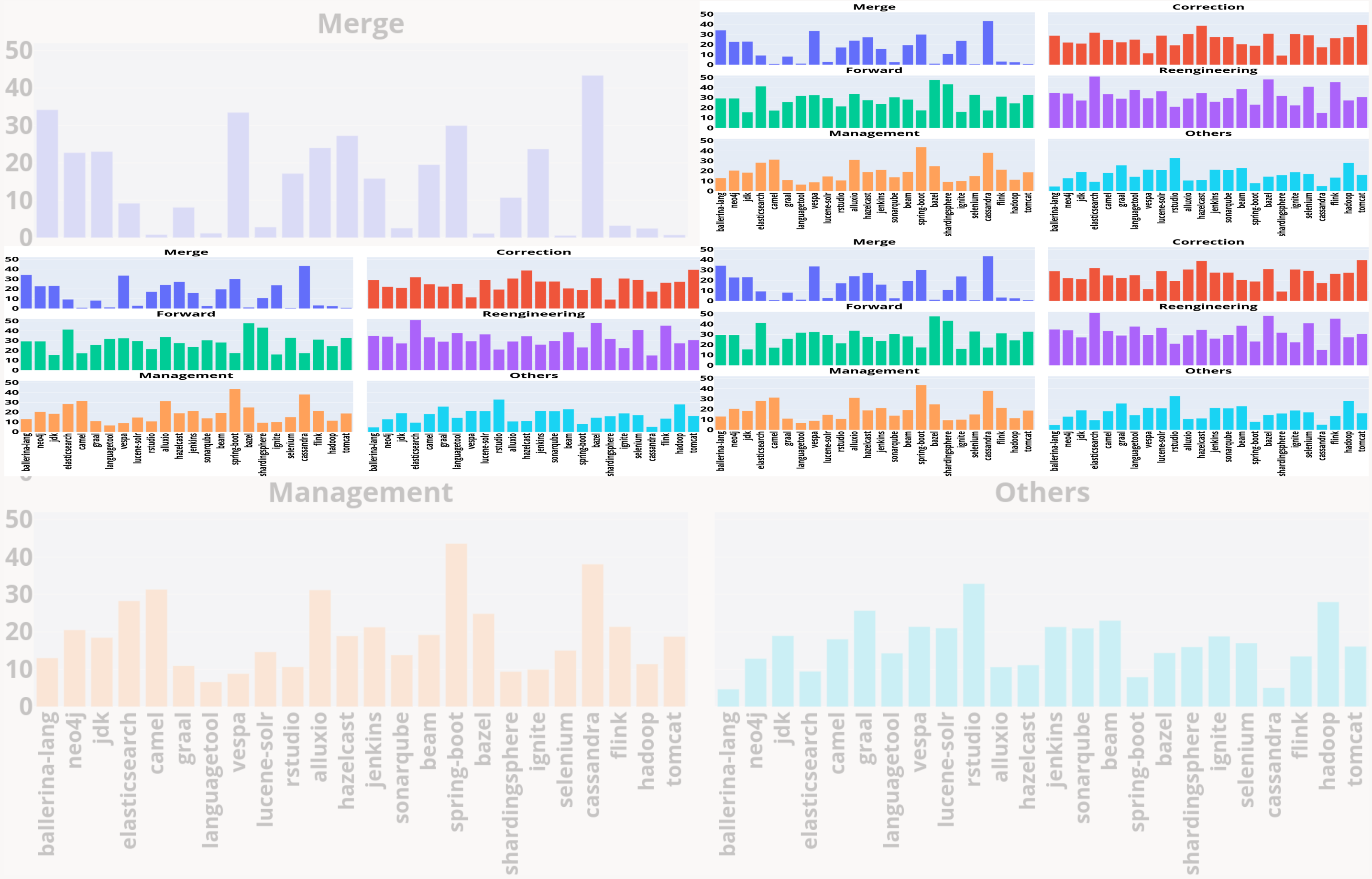
> Author A push a commit at 12h, 13h, and 13h30min
> Two time intervals: 60 and 30 minutes

3. Calculate the time average for each author

Research Questions & Results

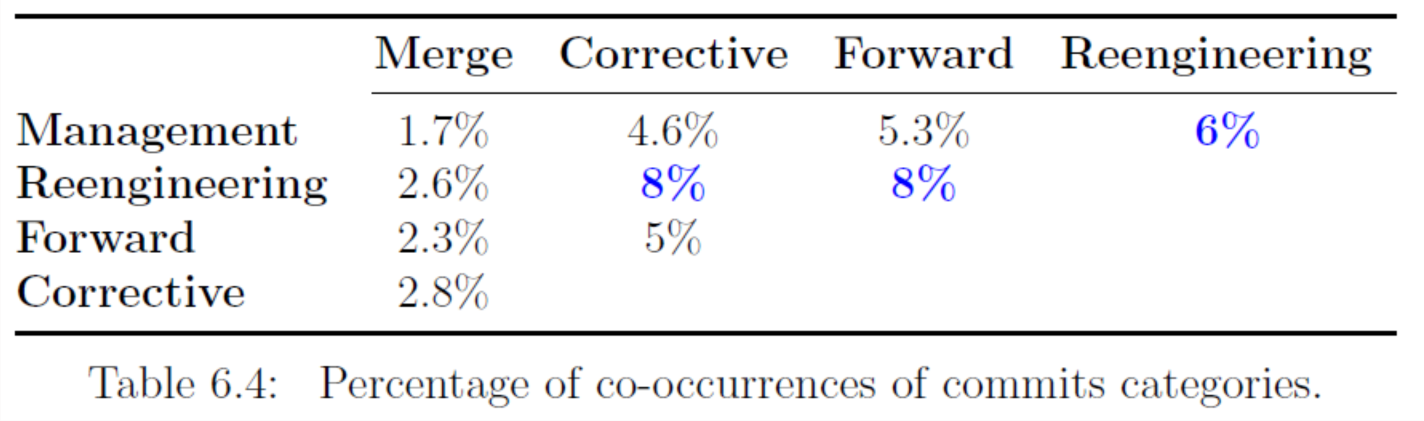
RQ1. How often are the activity types performed in commits?

1. Reengineering - 32.97%
2. Forward Engineering - 28.2%
3. Corrective Engineering - 25%



RQ2. How often do co-occurrences between the activity types appear in commits?

30% of commits analyzed involve more than one activity type



	Merge	Corrective	Forward	Reengineering
Management	1.7%	4.6%	5.3%	6%
Reengineering	2.6%	8%	8%	
Forward	2.3%	5%		
Corrective	2.8%			

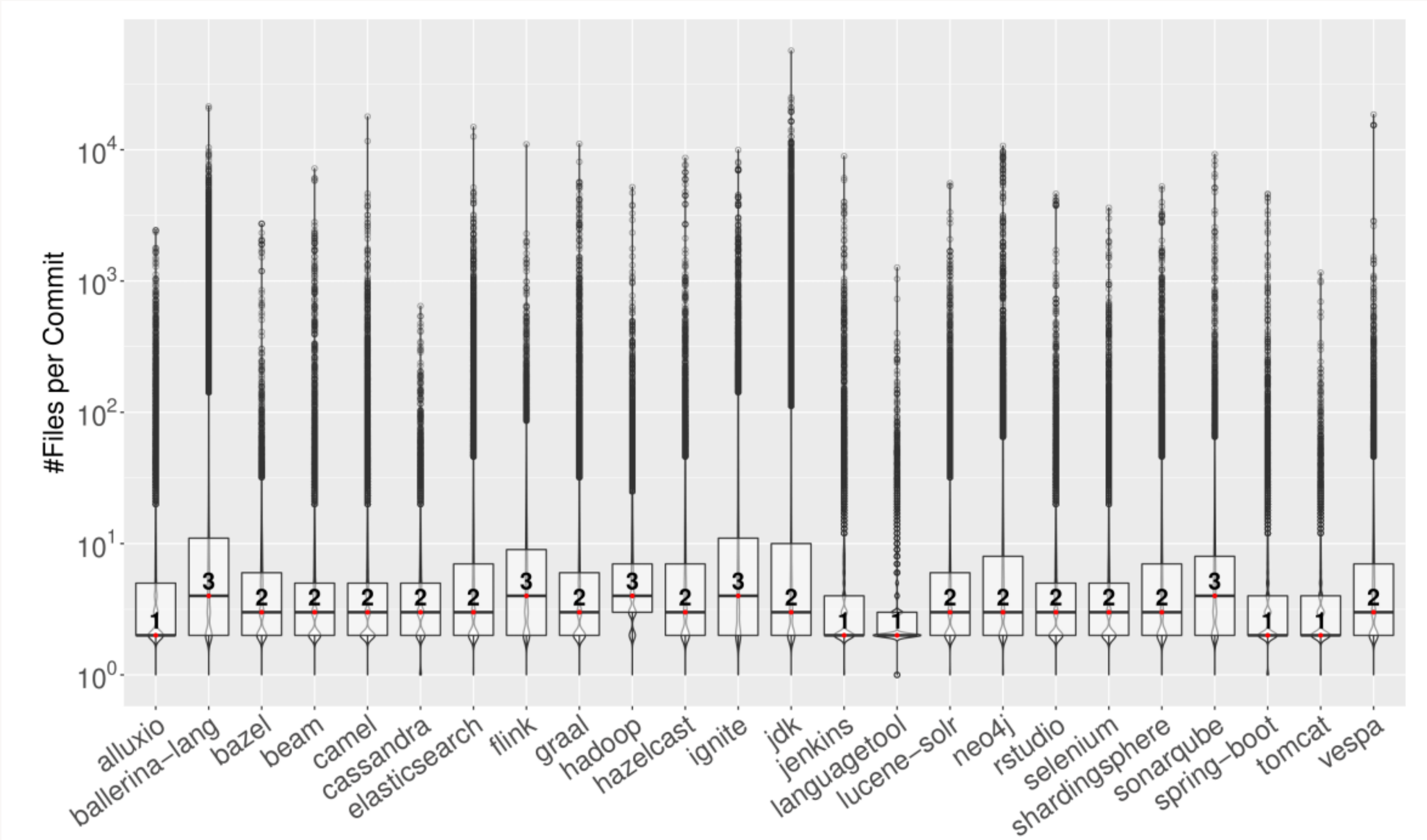
Table 6.4: Percentage of co-occurrences of commits categories.

RQ3. What is the size of commits in software system repositories?

- Some commits change hundreds of files at once

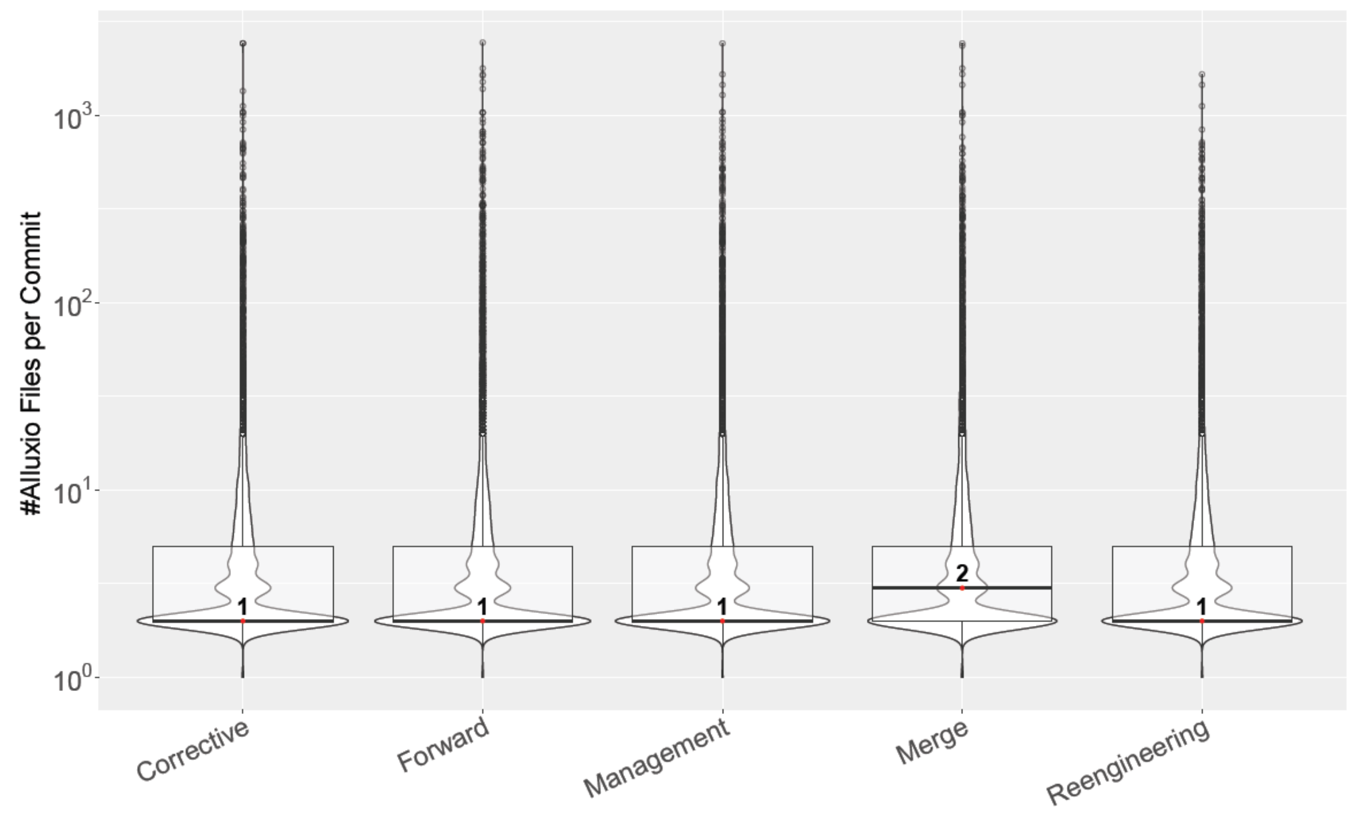
- #files range from 1 to 10

- #java files range from 1 to 4



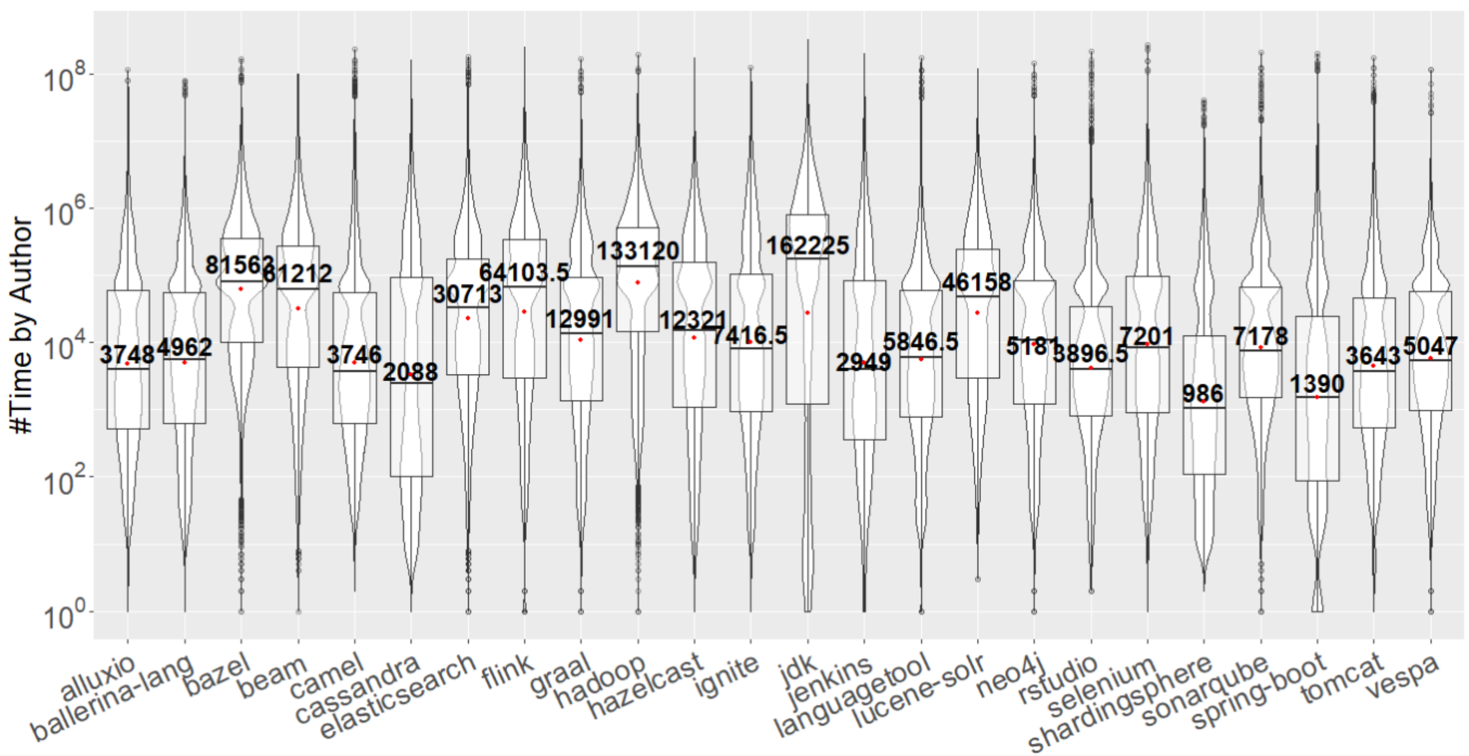
RQ4. What is the size of commits according to their aims?

- There is no significant difference between the number of files per category



RQ4. What is the time interval a developer registers a commit in a repository?

Developers perform commits on average every 8 hours

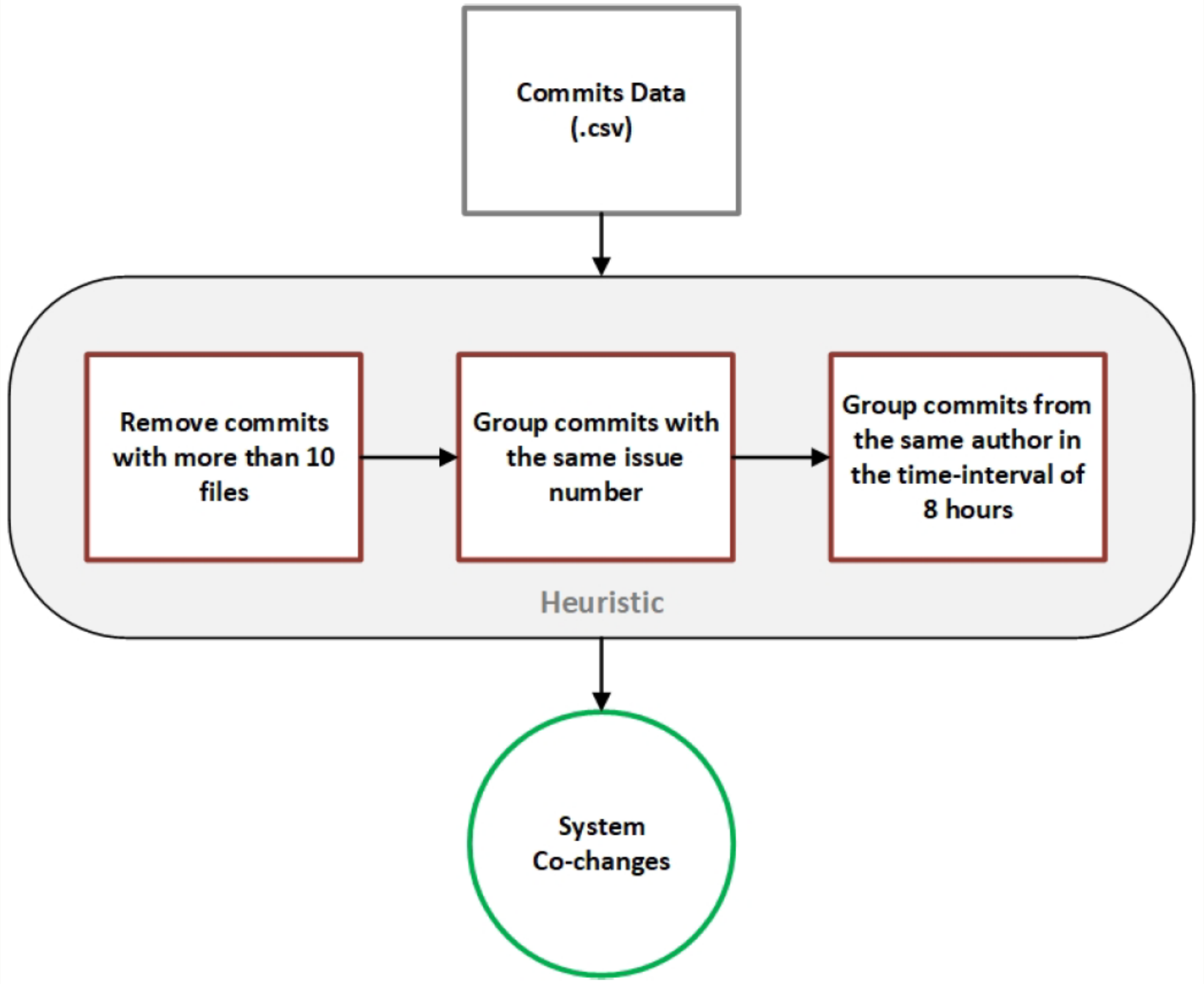


Final Remarks

- ✓ Reengineering is the most frequent activity, followed by Forward Engineering and Corrective Engineering.
- ✓ 30% of commits involve more than one type of activity.
- ✓ Most commits involve 1 to 10 files and 1to 4 source-code files.
- ✓ Many commits involve hundreds of files and those commits not only refer to Merge or Management.
- ✓ On average, a developer proceeds a commit every eight hours.

A Heuristic for Co-change

The Heuristic



Commits Data (.csv)

Heuristic

Remove commits with more than 10 files

Group commits with the same issue number

Group commits from the same author in the time-interval of 8 hours

System Co-changes

Evaluation Approach

- Comparison of two heuristics

  Proposed Heuristic vs. Commit Heuristic

- **Proposed Heuristic**
  - Applies the three steps to filter the commits to detect co-change

- **Commit Heuristic**
  - A co-change occurs if at least one commit involves the set of classes

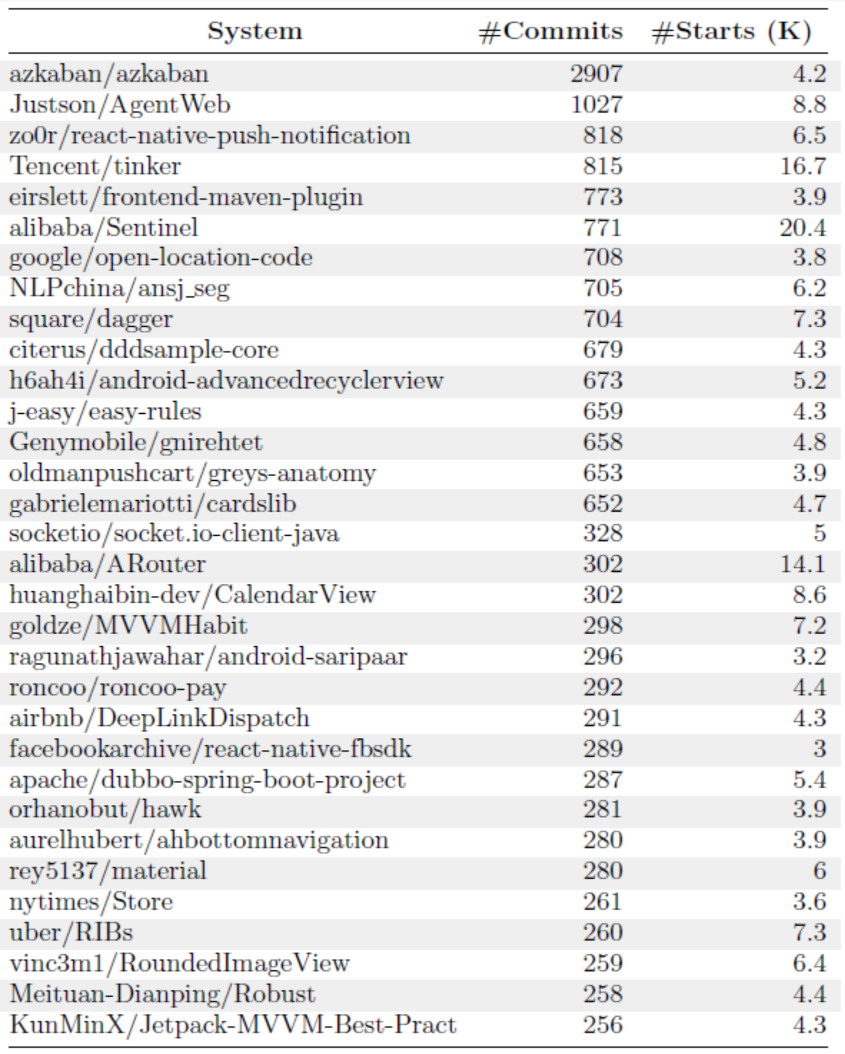
Evaluation Approach

- **Dependency Graph - Oracle**
  - Coupling among modules is a cause of change propagation
  - Link between co-change and static dependencies

- **Co-change Graph**
  - The system classes are the vertices
  - Edges represent a co-change between the classes

Dataset

- 32 Java systems
- ≈ 18K commits



System	#Commits	#Starts (K)
azkaban/azkaban	2907	4.2
Justson/AgentWeb	1027	8.8
zo0r/react-native-push-notification	818	6.5
Tencent/tinker	815	16.7
eirslett/frontend-maven-plugin	773	3.9
alibaba/Sentinel	771	20.4
google/open-location-code	708	3.8
NLPchina/ansj_seg	705	6.2
square/dagger	704	7.3
citerus/dddsample-core	679	4.3
h6ah4i/android-advancedrecyclerview	673	5.2
j-easy/easy-rules	659	4.3
Genymobile/gnirehtet	658	4.8
oldmanpushcart/greys-anatomy	653	3.9
gabrielemariotti/cardslib	652	4.7
socketio/socket.io-client-java	328	5
alibaba/ARouter	302	14.1
huanghaibin-dev/CalendarView	302	8.6
goldze/MVVMHabit	298	7.2
ragunathjawahar/android-saripaar	296	3.2
roncoo/roncoo-pay	292	4.4
airbnb/DeepLinkDispatch	291	4.3
facebookarchive/react-native-fbsdk	289	3
apache/dubbo-spring-boot-project	287	5.4
orhanobut/hawk	281	3.9
aurelhubert/ahbottomnavigation	280	3.9
rey5137/material	280	6
nytimes/Store	261	3.6
uber/RIBs	260	7.3
vinc3m1/RoundedImageView	259	6.4
Meituan-Dianping/Robust	258	4.4
KunMinX/Jetpack-MVVM-Best-Pract	256	4.3

Research Questions & Results

RQ1. How precise is the proposed heuristic?

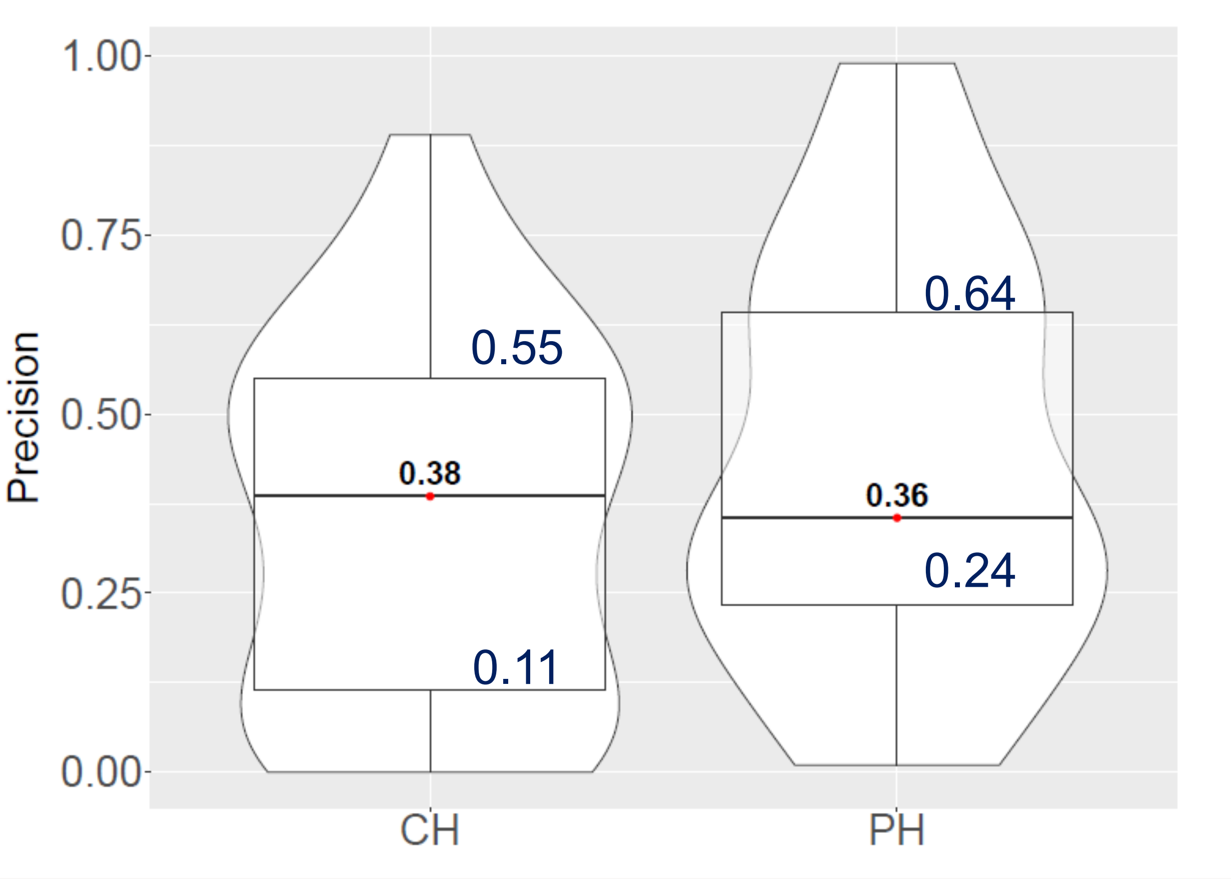
- **Precision**

$True\ Positive / True\ Positive \cup False\ Positive$

- **True Positive =** heuristic identifies the co-change between A and B, and there is path from A to B in the dependency graph
- **False Positive** = heuristic identifies the co-change between A and B, and there is no path from A to B in the dependency graph

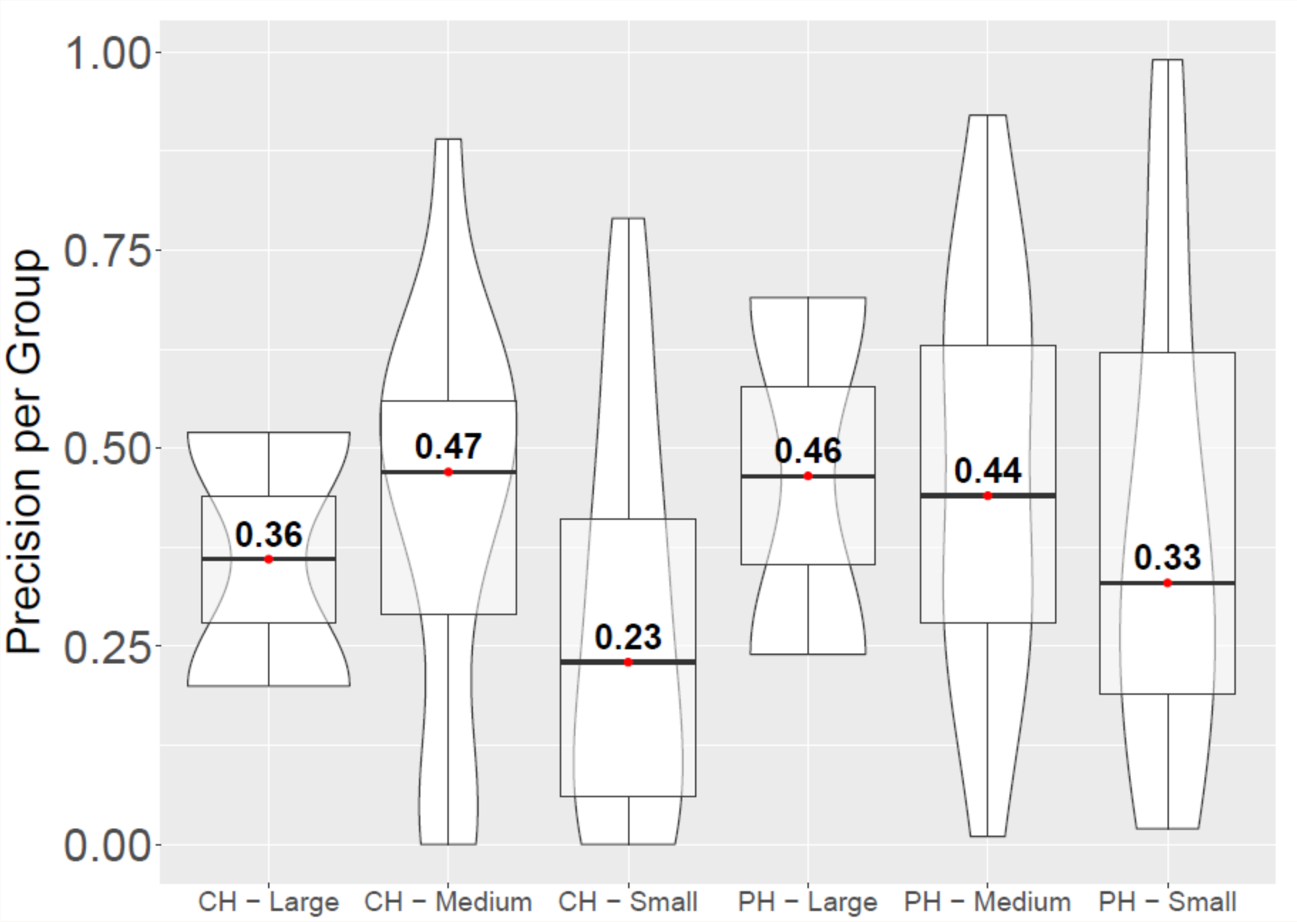
RQ1. How precise is the proposed heuristic?

The Proposed Heuristic (PH) has higher precision than Commit Heuristic (CH)



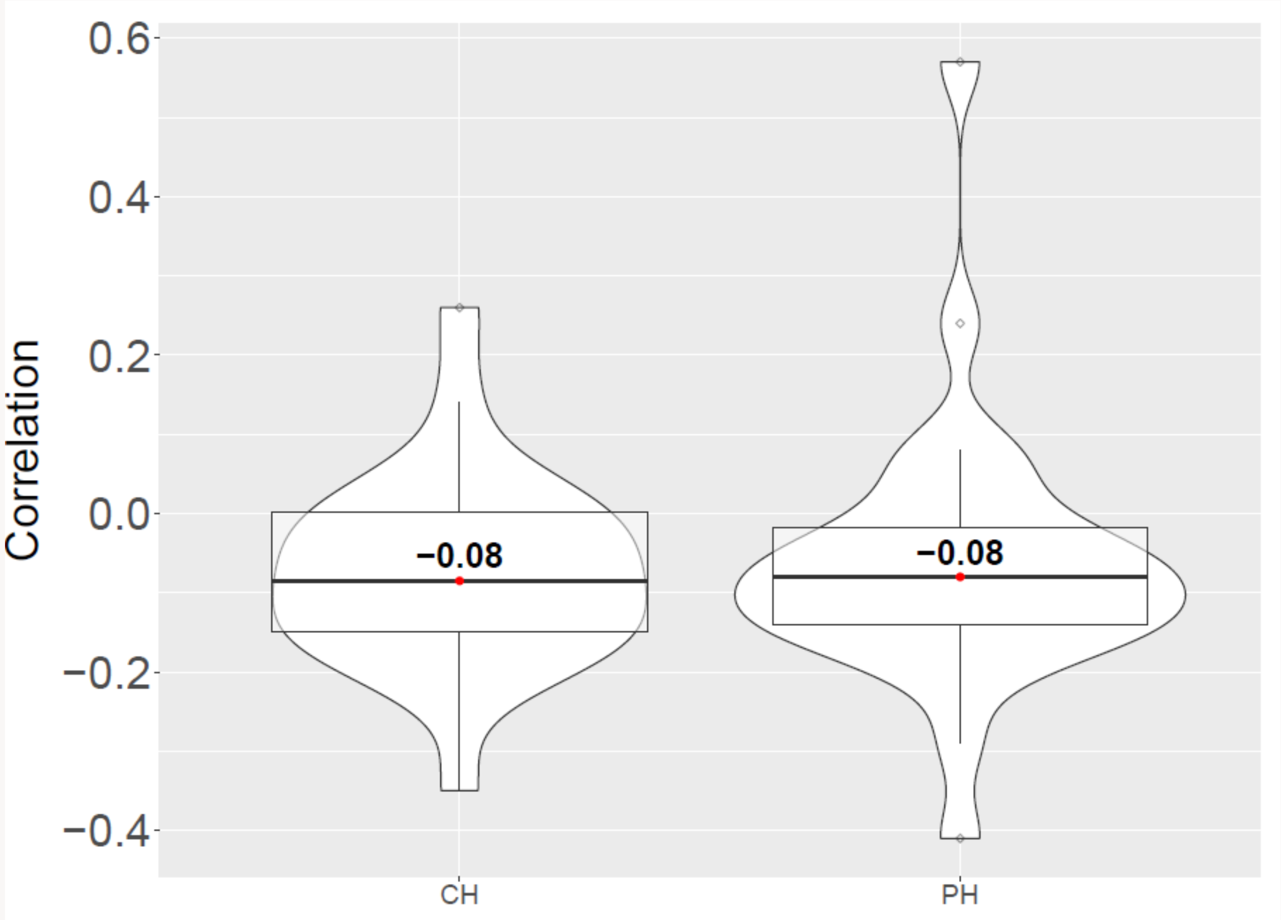
RQ2. Does the amount of commits in a system influence the accuracy of the heuristics?

The precision of the heuristics is not associated with the number of commits analyzed by them.



RQ3. Does the distance between the classes influence their co-change?

- Pearson Correlation Coefficient
- Classes that change together tend to be closer to each other



Final Remarks

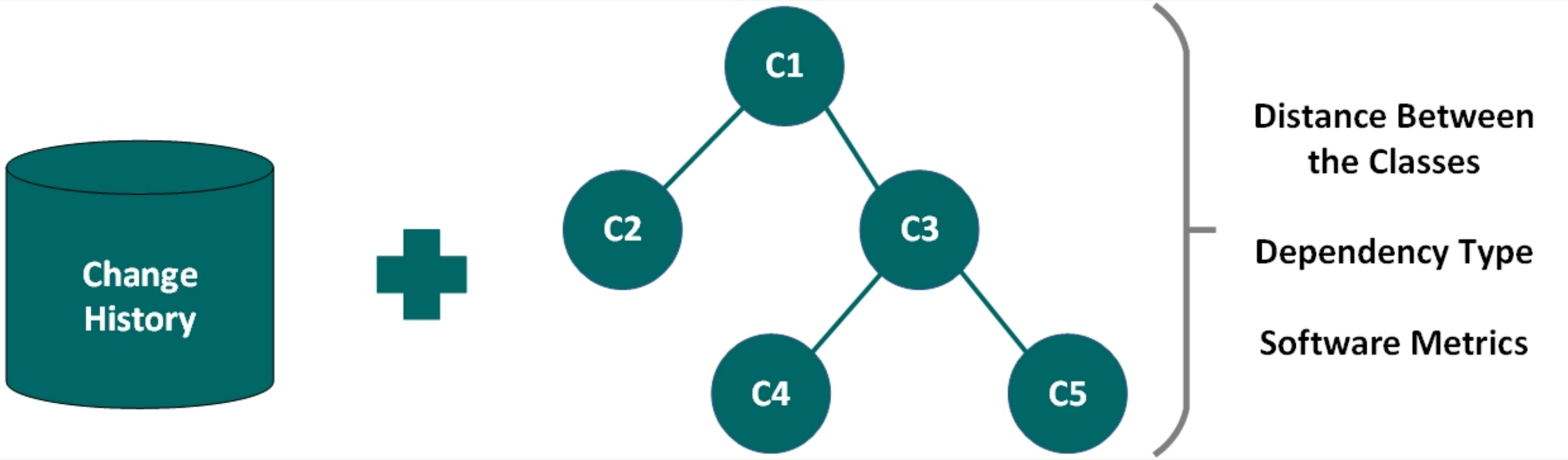
- ✓ Apply commit characteristics improves the sensitivity of commit-based heuristic

- ✓ There is evidence of a relationship between the distance and the number of times two classes changed together

The Proposed Change Impact Model

Proposal Description

- Probabilistic Model
  - Hybrid model based on change history analysis and dependency graph



Proposal Description

- **Change History Analysis**
  - Proposed Heuristic – Co-change dataset
  - Co-change dataset
    - Extract probabilities of change impact
      - ✓ Type of structural dependency (inheritance, use of method or fields)
      - ✓ Distance between classes
      - ✓ Software Metrics (cohesion, coupling...)

Proposal Description

- **Dependency Graph**
  - Edges of the graph weights will be the probability found in the change history analysis
  - Logistic regression to calculate the edges' weights
    - Allows using continuous and categorical predictors

Evaluation Method

- Modification Oracle
  - Mining from GitHub repositories files related to issues labeled as bug
- 90 software systems
- Compare the results of the proposed model to the oracle results

Conclusion

Conclusion

We made

- ✓ A study to understand how software engineering research has evolved and identify the general status of software maintenance research.
- ✓ A survey identifying how software maintenance has been done in practice.
- ✓ A systematic mapping review on change impact analysis.
- ✓ An empirical study on commits' characterization.
- ✓ A new co-change heuristic.

Publications

1. Software Engineering Evolution: The History Told by ICSE, (shortpaper) - (SBES), 2019.
2. The Software Engineering Observatory Portal. (ISSI), 2021
3. On The Gap Between Software Maintenance Theory and Practitioners' Approaches. (SER&IP), 202
4. Inside Commits: An Empirical Study on Commits in Open-Source Software, (short-paper). (SBES), 2021
5. Characterizing Commits in Open-Source Software. (SBQS), 2022

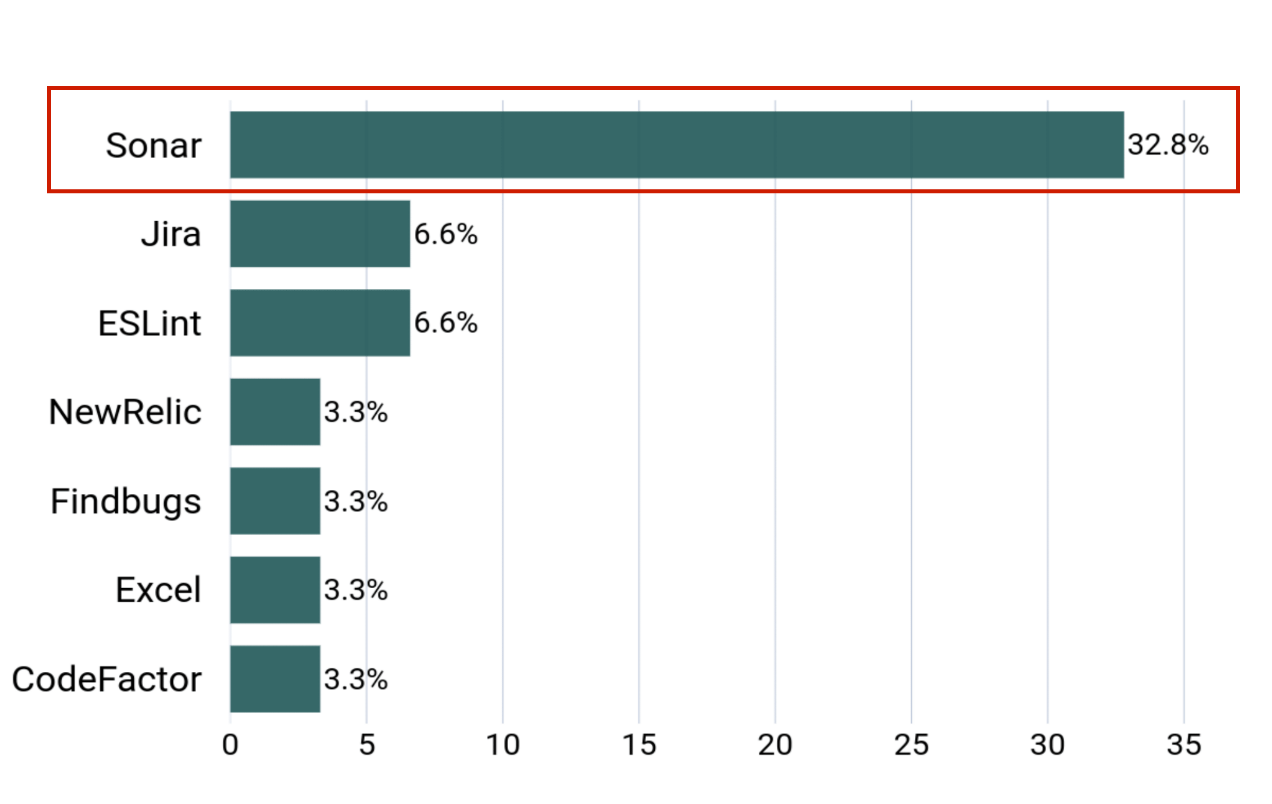
Next Steps

1. Modify the CK tool to obtain data on types of dependencies between classes the systems’ classes.
2. Run the co-change heuristic in the remaining data set.
3. Perform an empirical analysis to find the probabilities of change impact considering the structural dependency type, the distance between the classes, and the software metrics.
4. Define and implement the change impact analysis method.
5. Evaluate the proposed method
6. Write the chapters of the Ph.D. dissertation describing the proposed change impact analysis method and its evaluation.
7. Write a paper about the proposed change impact analysis method.
8. Present the Ph.D. dissertation.

Thanks for your attention!

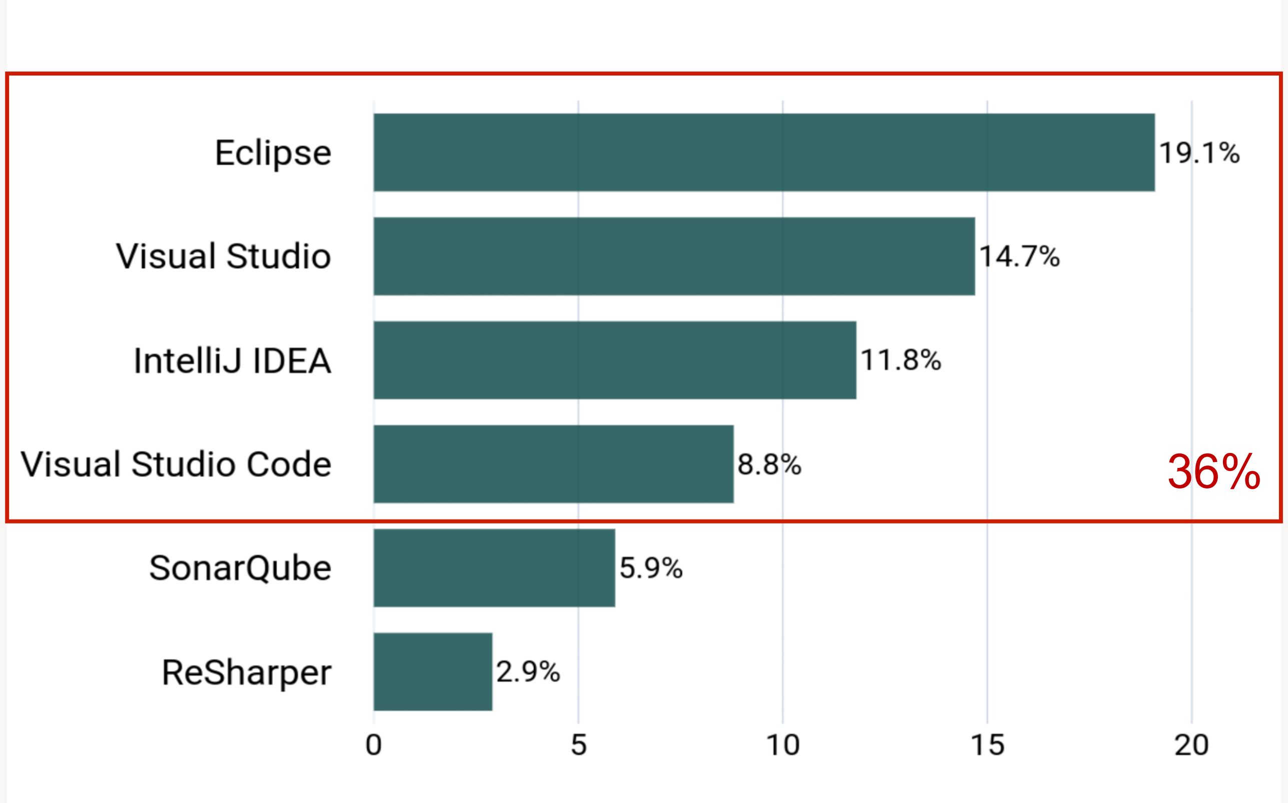
RQ3. Which are the tools most used by practitioners in software maintenance?

Software Metric Tools



RQ3. Which are the tools most used by practitioners in software maintenance?

Refactoring Tools



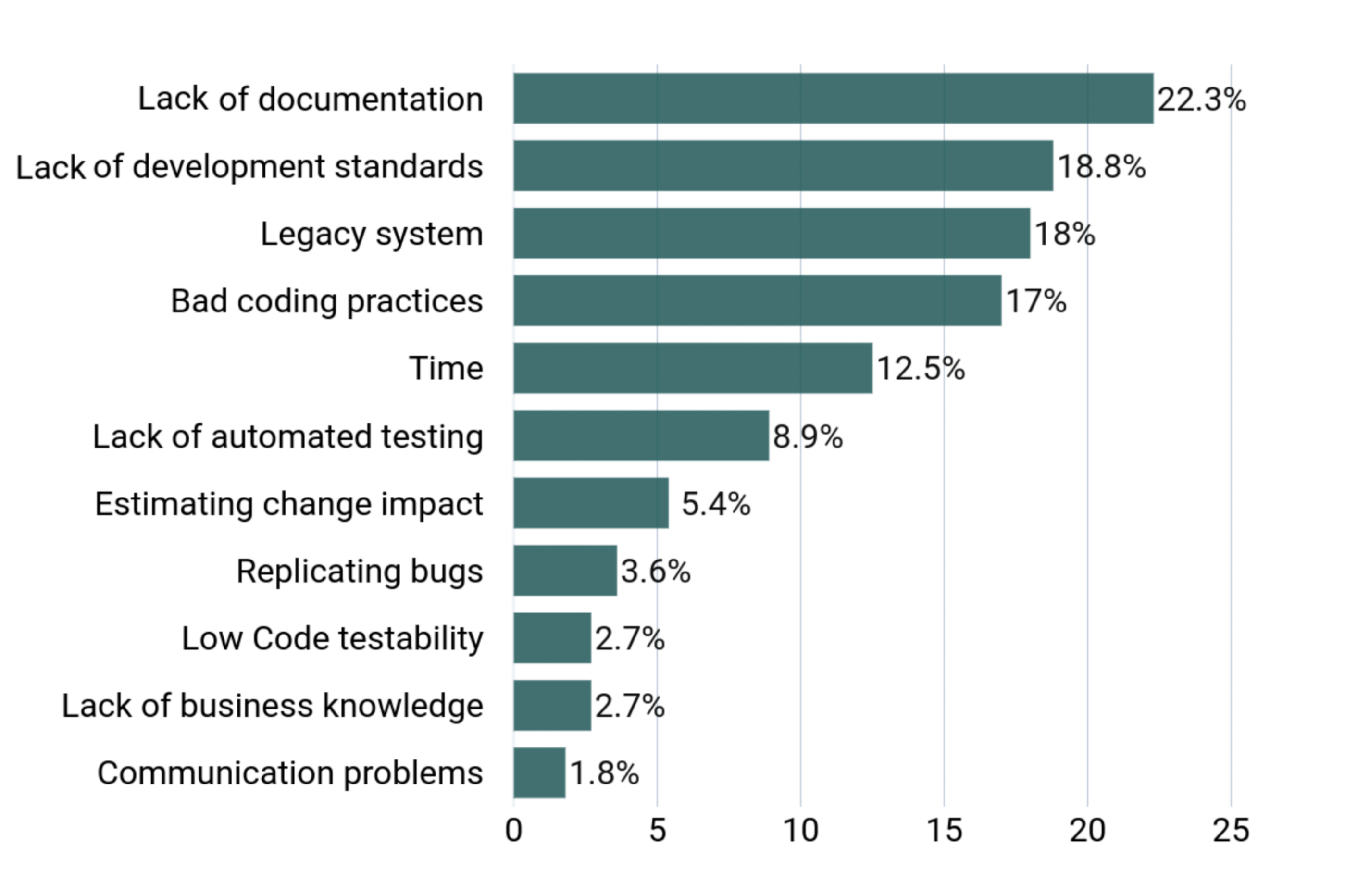
RQ5. Which metrics, refactoring techniques, and bad smells practitioners apply in their activities?

Metrics	Number of Bugs (9.9%)
	Test Coverage (8.91%)
	Cyclomatic Complexity (7.92%)

Refactoring	Extract Method (21.43%)
	Rename Method (13.39%)
	Extract Class (12.5%)

Bad Smell	Duplicate Code (23.21%)
	Duplicate Code (23.21%)
	Long Class (9.82%)

RQ6. What are the biggest challenges faced by practitioners when carrying out software maintenance?



RQ2. Which are the characteristics of these approaches and tools?

- **Scientific Method**
    - Empirical method is the most applied by researches (114 out of 141)

✓ Which makes sense given the nature of the problem

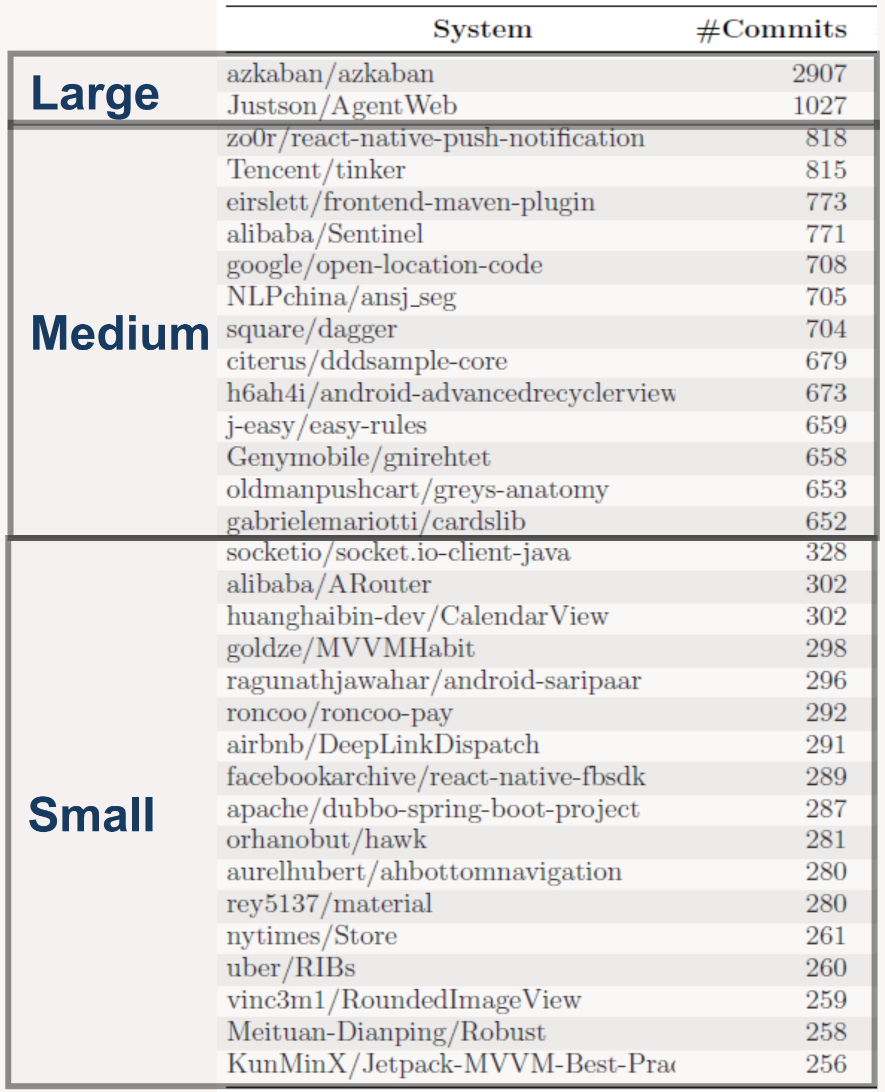
A

- Research publications might not be accessible to the industry, and their results might not be easily implemented in the practice
- Software engineering researchers may face challenges when collaborating with practitioners

The target of this study was to understand the gap between the research and the practice of software maintenance.

RQ2. Does the amount of commits in a system influence the accuracy of the heuristics?

Precision according to the number of commits of the systems



	System	#Commits
Large	azkaban/azkaban	2907
	Justson/AgentWeb	1027
Medium	zo0r/react-native-push-notification	818
	Tencent/tinker	815
	eirslett/frontend-maven-plugin	773
	alibaba/Sentinel	771
	google/open-location-code	708
	NLPchina/ansj_seg	705
	square/dagger	704
	citerus/dddsample-core	679
	h6ah4i/android-advancedrecyclerview	673
	j-easy/easy-rules	659
	Genymobile/gnirehtet	658
	oldmanpushcart/greys-anatomy	653
	gabrielemariotti/cardslib	652
Small	socketio/socket.io-client-java	328
	alibaba/ARouter	302
	huanghaibin-dev/CalendarView	302
	goldze/MVVMHabit	298
	ragunathjawahar/android-saripaar	296
	roncoo/roncoo-pay	292
	airbnb/DeepLinkDispatch	291
	facebookarchive/react-native-fbsdk	289
	apache/dubbo-spring-boot-project	287
	orhanobut/hawk	281
	aurelhubert/ahbottomnavigation	280
	rey5137/material	280
	nytimes/Store	261
	uber/RIBs	260
	vinc3m1/RoundedImageView	259
	Meituan-Dianping/Robust	258
	KunMinX/Jetpack-MVVM-Best-Pra	256