Evaluating the Impact of Code Smell Agglomerations on Software Systems

AMANDA SANTANA

ADVISOR: EDUARDO FIGUEIREDO

Agenda

- Introduction
- Motivation + Goal
- Background
- Studies Planning Overview
- First Study
- Second Study
- Work in Progress
- Future Works
- Conclusions

System evolution

Systems must evolve to cope with new requirements, to fix existing problems, such as bugs, or to update their dependencies

Problems ahead

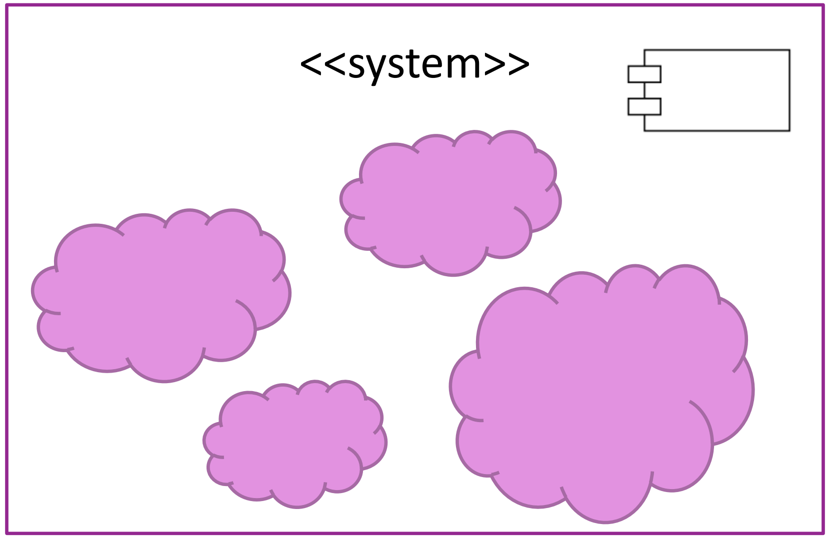
- Changing the code is a challenging activity:
  - Understanding the code and its complexity
  - Class dependencies and ripple effects

Code Smell

- Symptoms of developer's decisions that may lead to code quality degradation:
  - Complexity;
  - Cohesion;
  - Coupling;
  - Modularity;
  - Size;
  - Faults.

Code smell agglomerations

- Evidences in the literature show that when two or more code smells occurs in the same piece of code, forming a code smell agglomeration the code is harder to maintain and to understand.



lab▪soft
software engineering laboratory

Initial Studies on Agglomerations

- Abbes et al. (2011) : Blob + Spaghetti Code ⟶ + Time Consuming

  + Effort

  + Error-prone

- Palomba et al. (2018): Class with ≥ 2 smells ⟶ + Faults

  + Changes

lab·soft
software engineering laboratory

Motivation

- Few studies evaluated the impact of code smell agglomerations;
- Datasets composed of older systems;
- Several studies used small datasets (two~four systems);
- Diverging results.

Goal

Provide evidence of the impact of code smell agglomerations on the code quality, identifying which code smell agglomerations are the most harmful considering different quality perspectives.

Specific Goals (SG)

- Investigate:

SG1: the impact of code smell agglomerations on modularity, using software metrics as our indicator of modularity degradation.

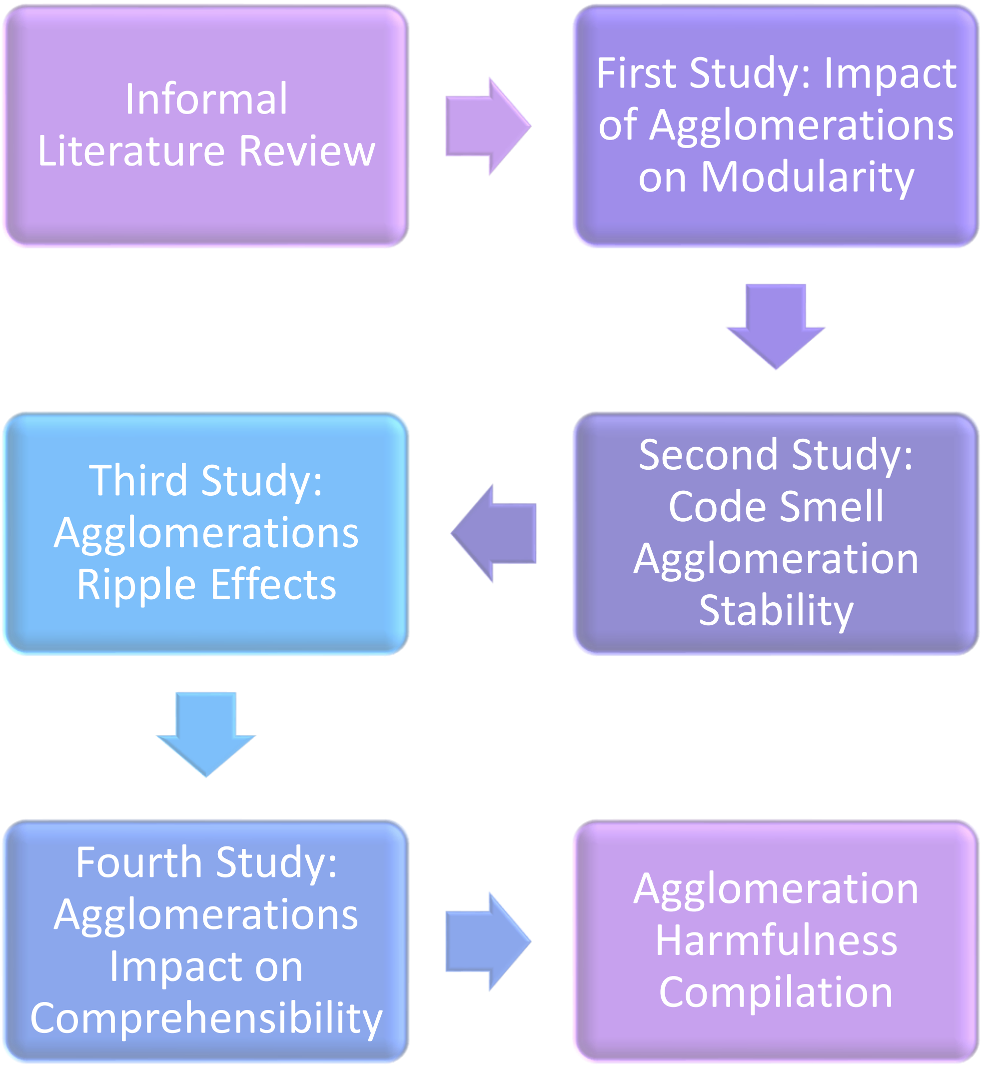
SG2: the stability of code smell agglomerations compared to single smell and no smell classes.

SG3: the propagation of changes due to code smell agglomerations in terms of coupling and ripple-effects.

SG4: how code smell agglomerations impact the code comprehensibility through a user study.

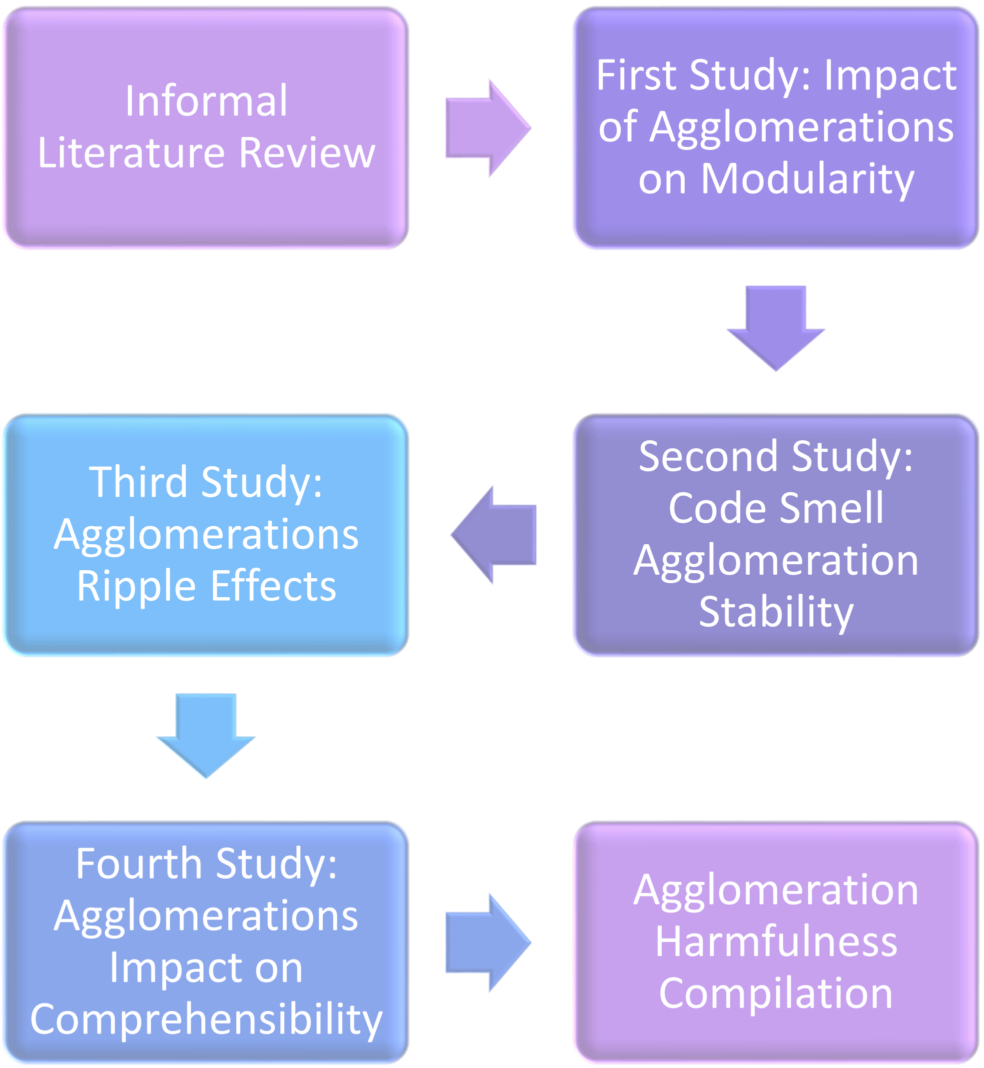
Code Smell Agglomerations

- When two or more code smells occurs on the same piece of code.
- Heterogeneous
  - Two or more smells of different types
- Homogeneous
  - Two or more smells of the same type
- Isolated
  - Only one smell
- Clean
  - No smell



Project Thesis Overview

Project Thesis Overview



Informal Literature Review

GOAL: IDENTIFY RELEVANT STUDIES THAT EVALUATED THE IMPACT OF CODE SMELL AGGLOMERATIONS.

IDENTIFY LITERATURE GAPS AND RESEARCH TOPICS

COMPILE IMPACT OF AGGLOMERATIONS

Informal Literature Review - Steps

Systematic Literature Reviews

Systematic Mappings

Databases:

- IEEE Xplore
- ACM Digital Library
- Scopus

Related Works

Palomba et al. (2018):

- Investigated how the introduction and removal of 13 smells impacts fault and change proneness.
- They found that the presence of a smell increases up to 83% the number of changes that the class suffers.

Palomba, F., Bavota, G., Di Penta, M., Fasano, F., Oliveto, R., and De Lucia, A. (2018). On the diffuseness and the impact on maintainability of code smells: A large scale empirical investigation. In 2018 IEEE/ACM 40th International Conference on Software Engineering. (ICSE), pages 482–482. ISSN 1558-1225.

Related Works

Olbrich et al. (2010):

- Investigated the frequency of faults and changes in smelly classes.
- They found that classes with Brain Class and God Classes changes 4~6x more frequently than non-smelly classes.
- They also found that for Brain Classes and God Classes, the average size of changes were higher than non-smelly classes.

Olbrich, S. M., Cruzes, D. S., and Sjøberg, D. I. K. (2010). Are all code smells harmful? a study of god classes and brain classes in the evolution of three open source systems. In 2010 IEEE International Conference on Software Maintenance, pages 1–10. ISSN 1063-6773

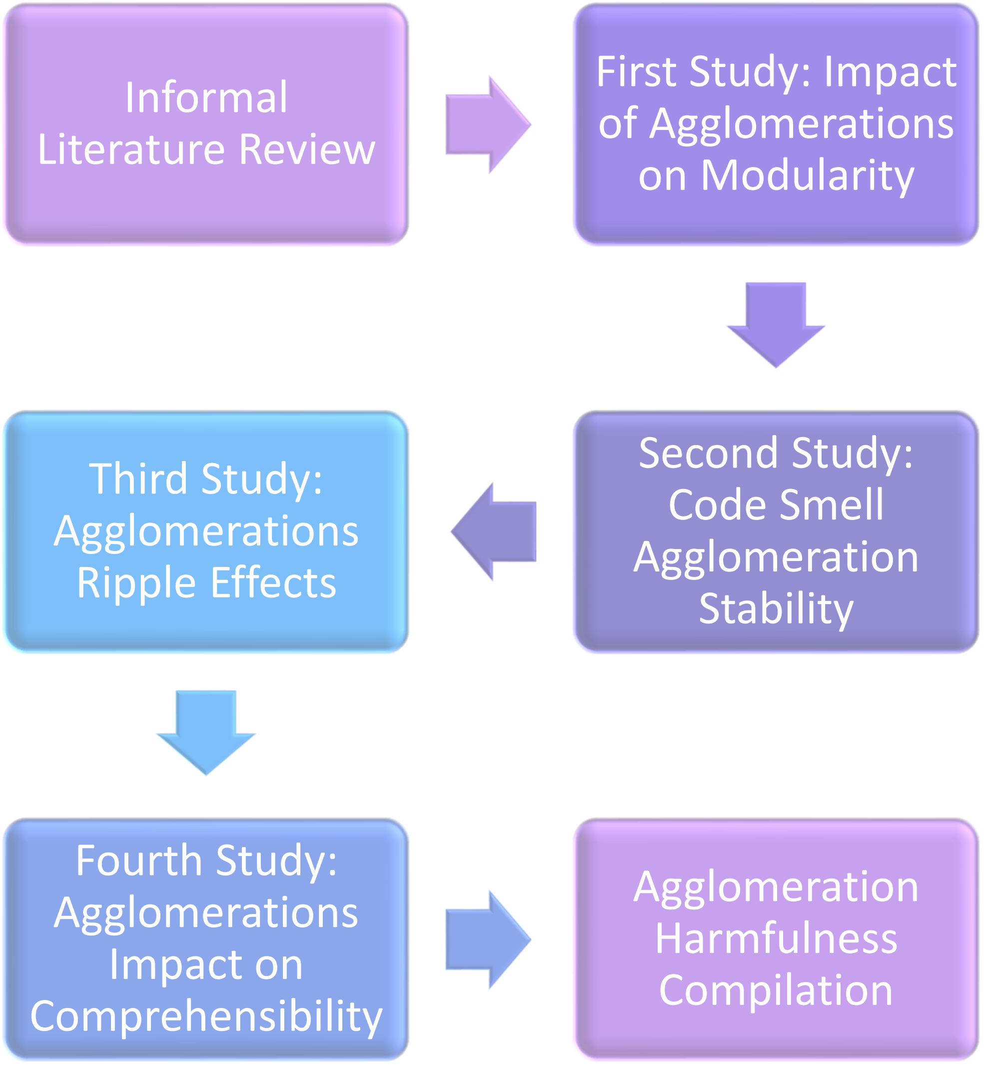
Related Works

Fontana et al. (2015):

- Investigated the relationships between six code smells based on co-occurence and coupling.
- They found that up to 58% of God Classes are called by other smelly classes.
- They found a high interaction between smelly classes.

Fontana, F. A., Ferme, V., and Zanoni, M. (2015). Towards assessing software architecture quality by exploiting code smell relations. In 2015 IEEE/ACM 2nd International Workshop on Software Architecture and Metrics, pages 1–7.

Project Thesis Overview



Amanda Santana, Eduardo Figueiredo, Juliana Alves Pereira, Alessandro Garcia. An exploratory evaluation of code smell agglomerations. Software Quality Journal (2024). https://doi.org/10.1007/s11219-024-09680-6

Goal

Provide evidences of which agglomeration is more harmful to code quality

Research Questions

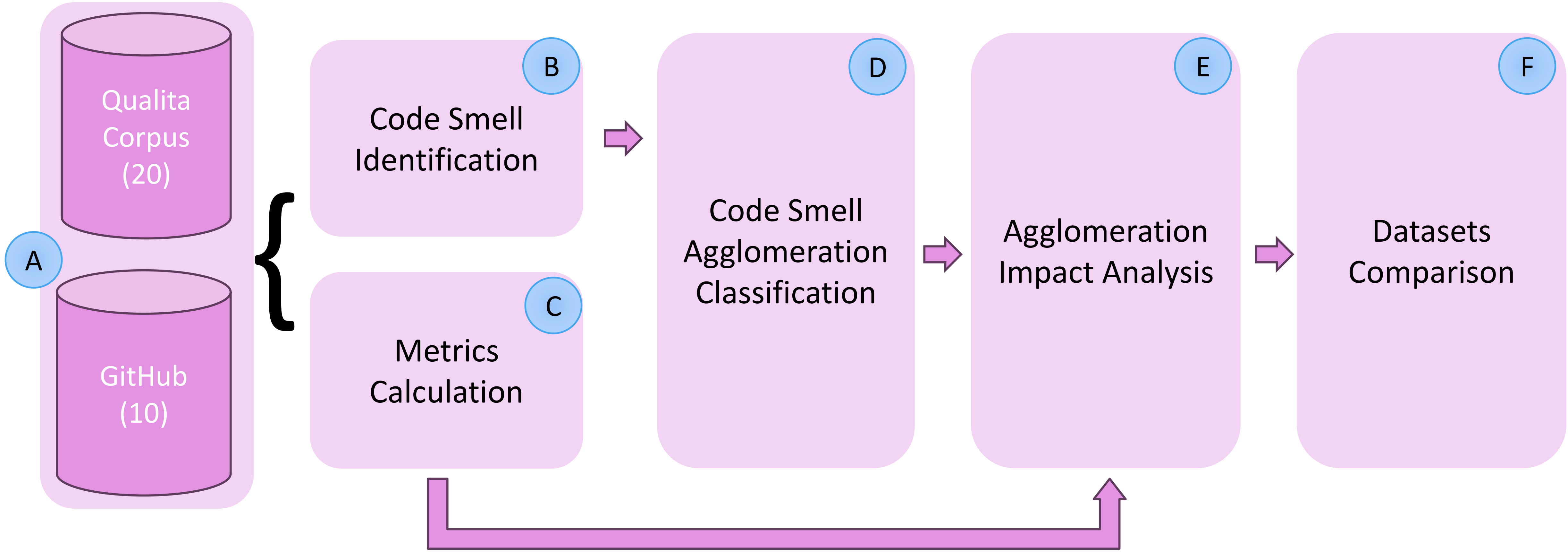
RQ1 - Are Homogeneous Agglomerations frequent in the source code?

RQ2 - Which Heterogeneous Agglomerations are more common in the source code?

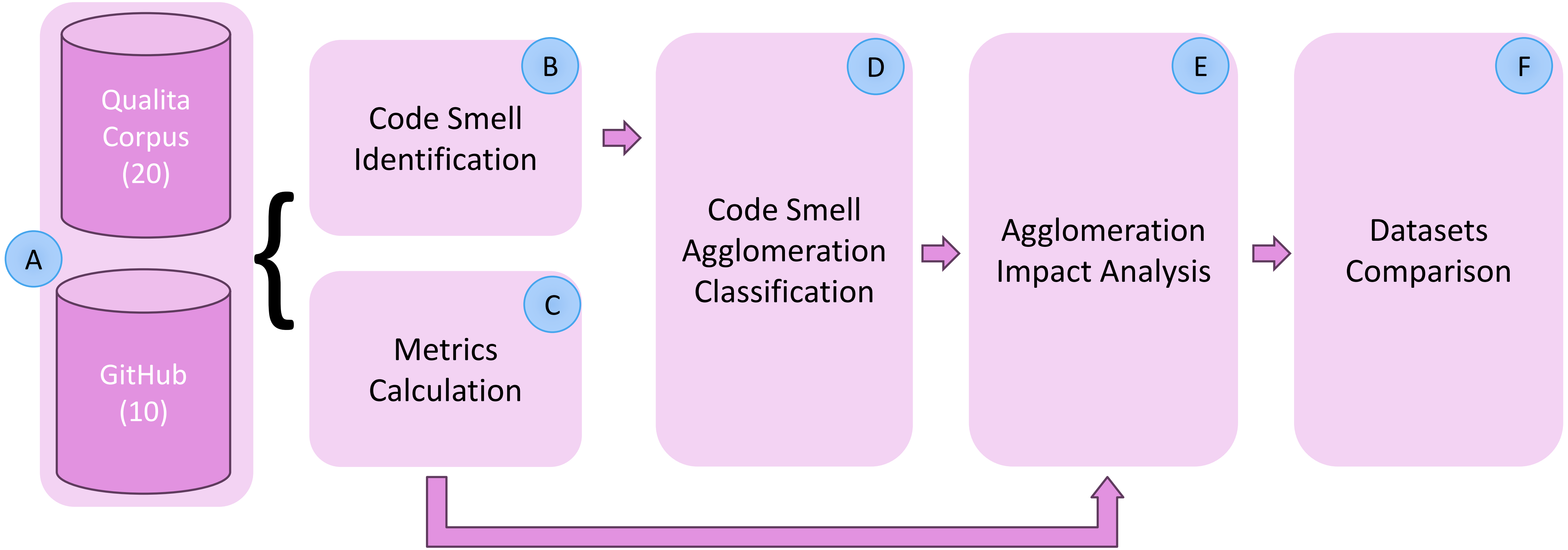
RQ3 - How do the code smell agglomerations impact on the system modularity?

RQ4 - Does the different types of Heterogeneous Agglomerations have an uniform impact on the system modularity?

Study Design



Study Design

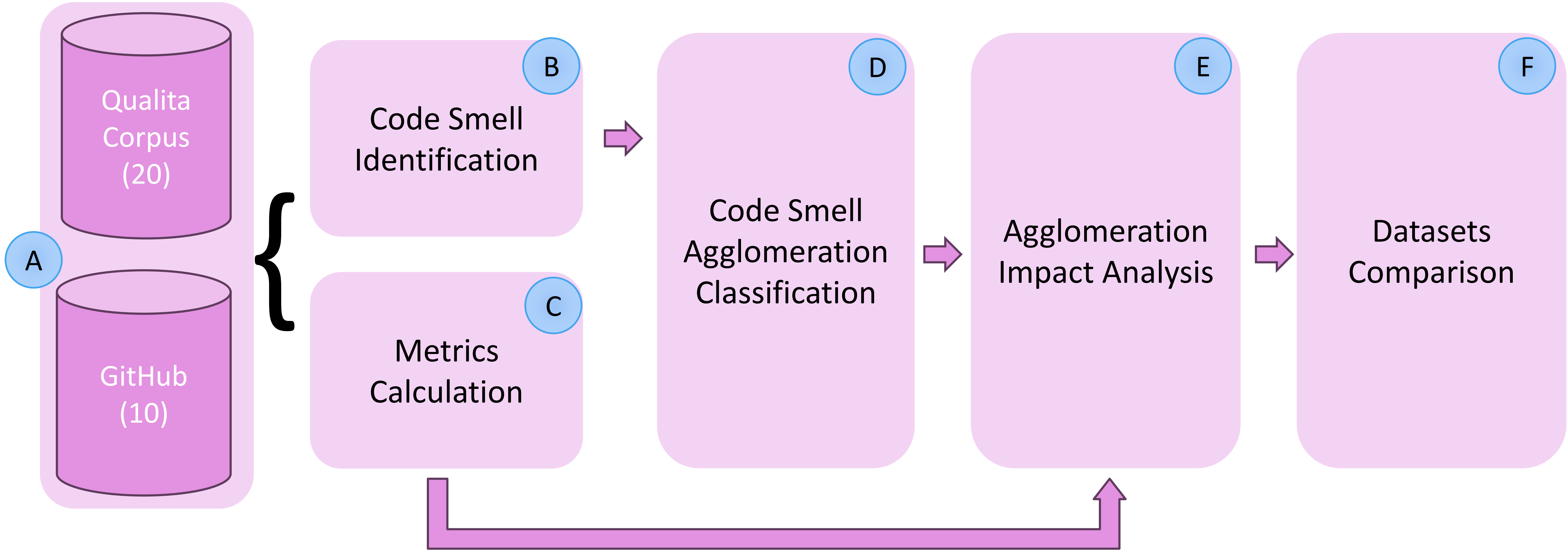


lab·soft
software engineering laboratory

System Selection

- 20 open-source Java systems from the Qualita Corpus
  - Different sizes, domains and support different development stages
- 10 open-source Java systems from the GitHub
  - Filtered by number of stars;
  - Code mostly in Java (more than 90%)
  - Be compilable on command line and Eclipse
  - Removal of educational systems

Study Design



Evaluated Code Smells

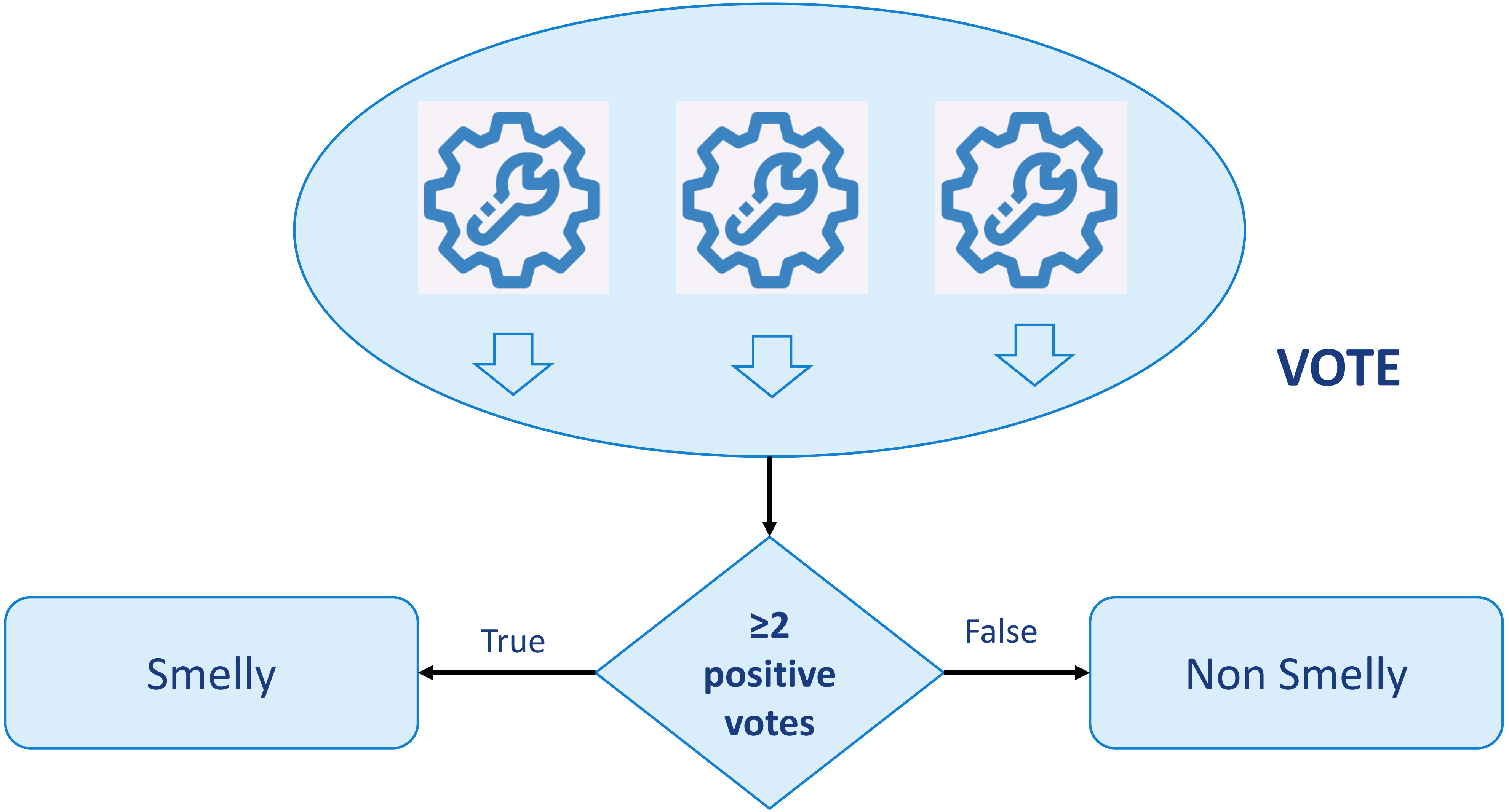
Class Level:

- Large Class
- Refused Bequest

Method Level:

- Feature Envy
- Long Method

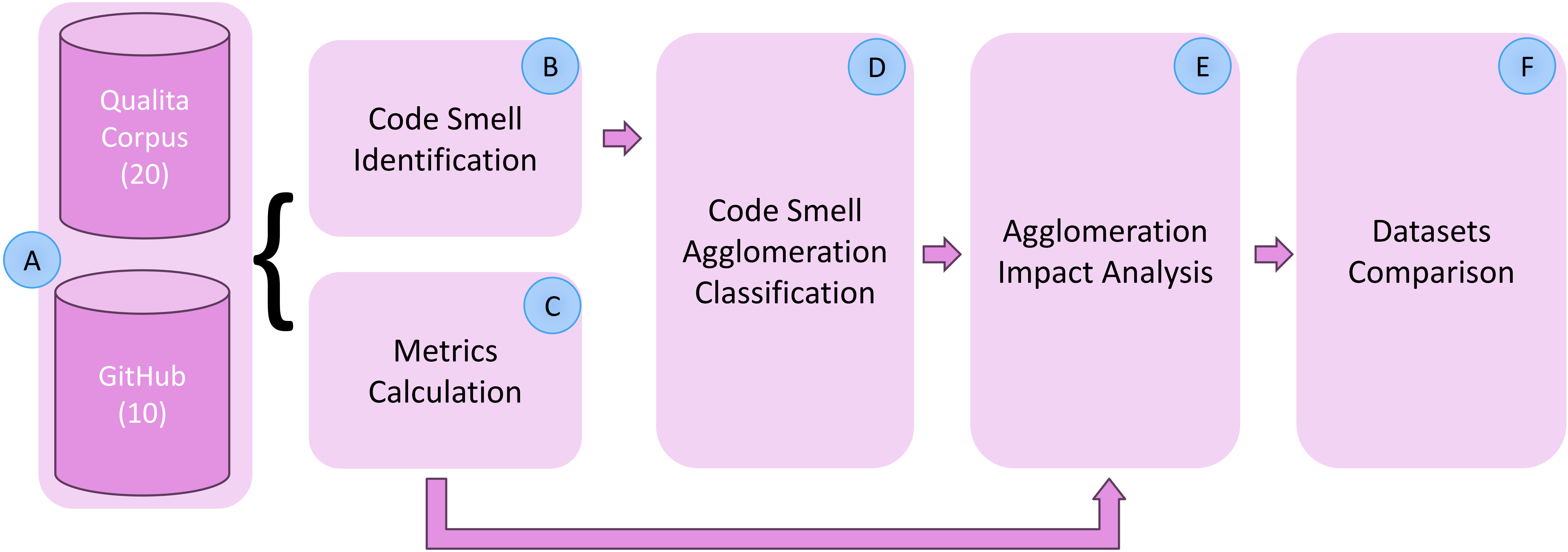
Detection Strategy



Data Cleaning

- Removal of undesired classes from our datasets:
  - Test classes
  - Android classes
  - Demos/Samples/Examples
  - AOP classes

Study Design

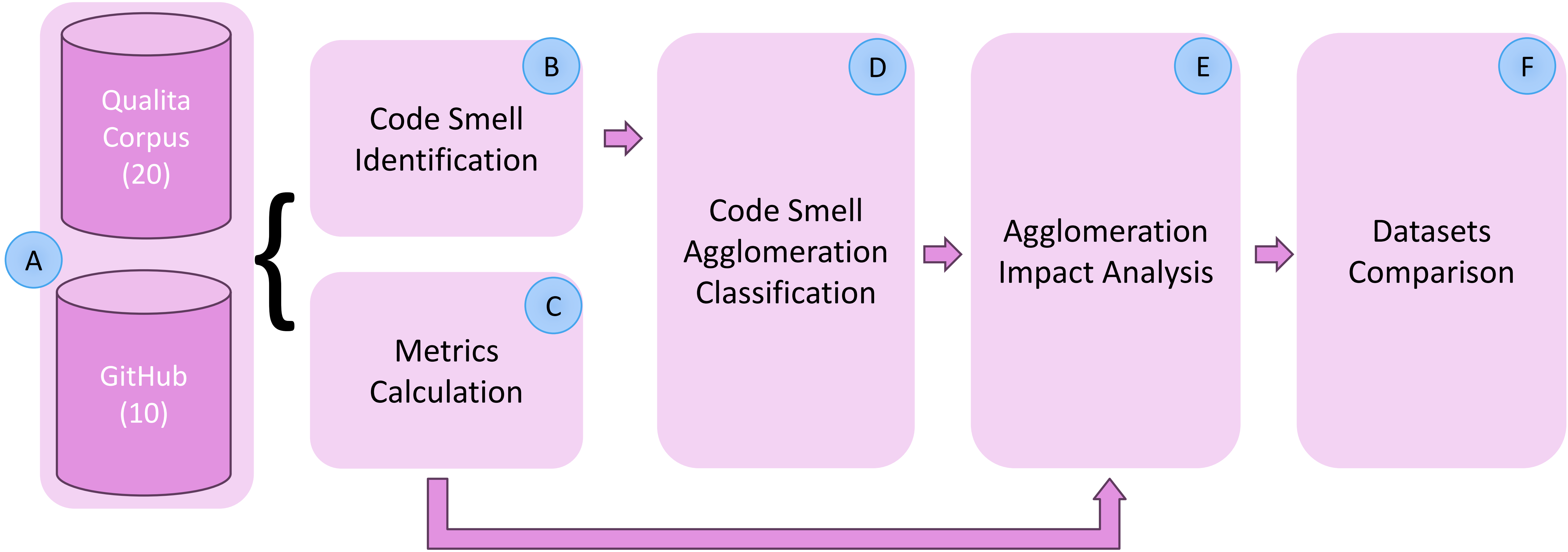


Selected Metrics

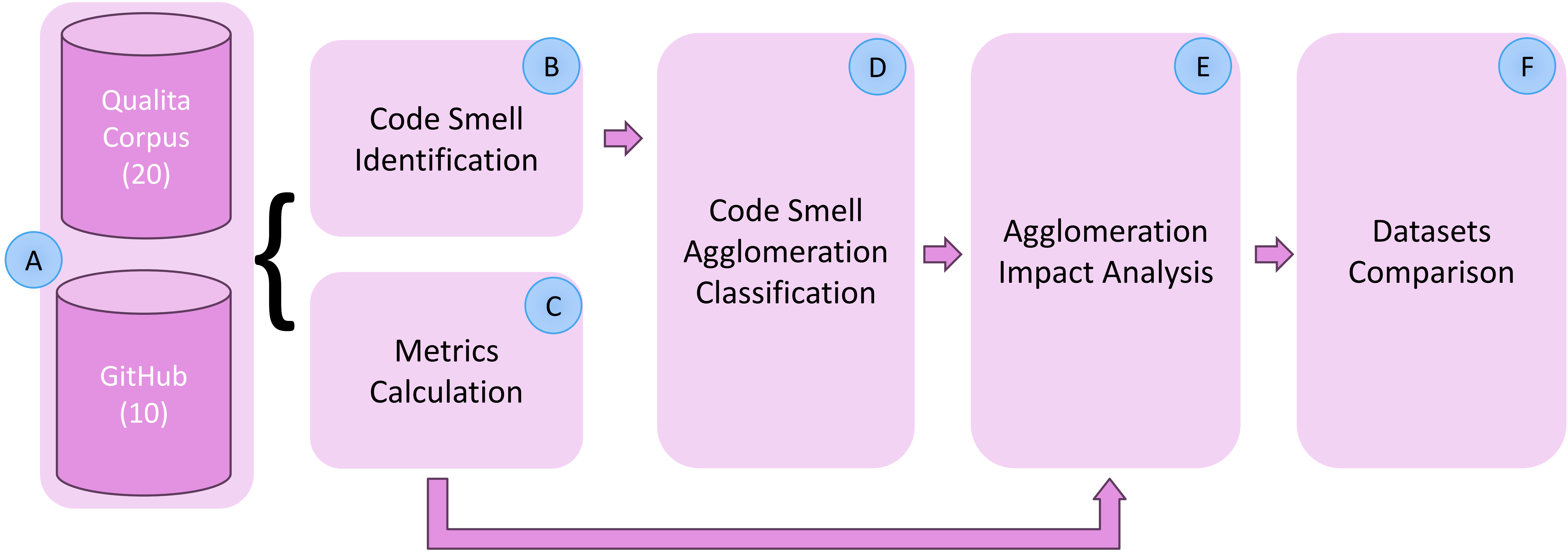
- Coupling Between Objects (CBO)
- Depth of Inheritance Tree (DIT)
- Response set of a class of objects (RFC)
- Weighted Method Class (WMC)
- Max Nest Blocks (maxNest)
- Lack of Cohesion Over Methods (LCOM*)

CK Tool

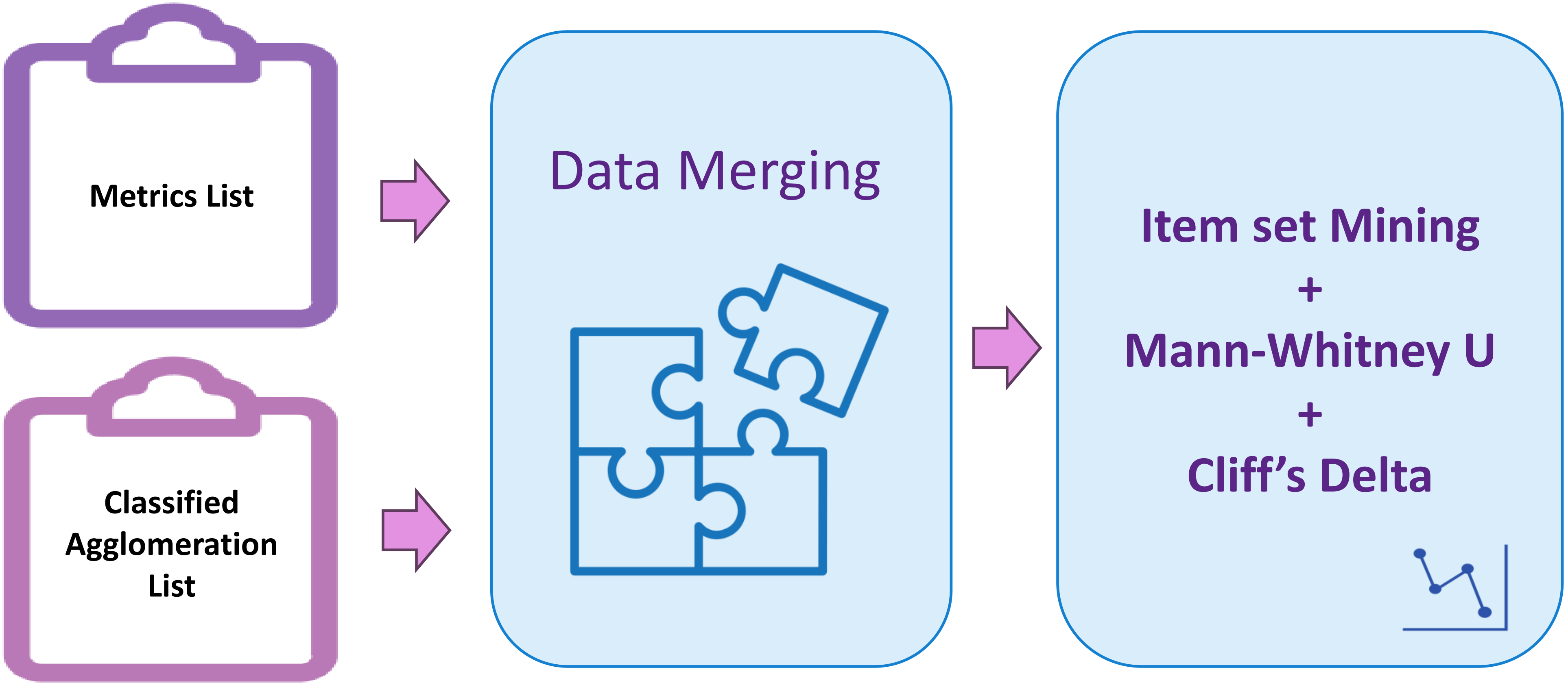
Study Design



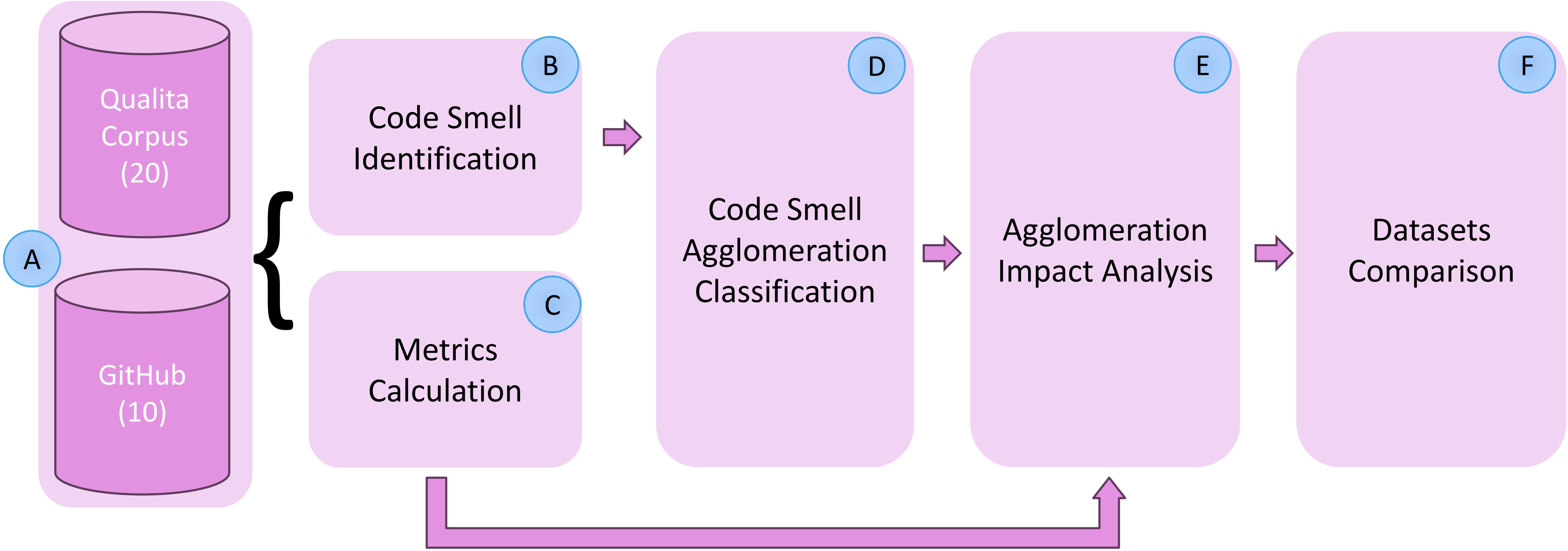
Study Design



Impact Calculation



Study Design



Dataset Comparison

- Comparison of the results found for each RQ by matching the results;
- Analysis of how size and the number of smells found is influencing the results.

Main Findings

lab·soft
software engineering laboratory

Dataset Overview

Code Smell	Qualita Corpus	GitHub
Large Class	580 (7.27%)	194 (14.27%)
Refused Bequest	966 (12.11%)	189 (13.62%)
Feature Envy	5,128 (64.31%)	564 (40.63%)
Long Method	1,300 (16.31%)	441 (31.77%)
	100%	**100%**

lab-soft
software engineering laboratory

Dataset Overview

Code Smell	Qualita Corpus	GitHub
Large Class	580 (7.27%)	194 (14.27%)
Refused Bequest	966 (12.11%)	189 (13.62%)
Feature Envy	5,128 (64.31%)	564 (40.63%)
Long Method	1,300 (16.31%)	441 (31.77%)
	100%	**100%**

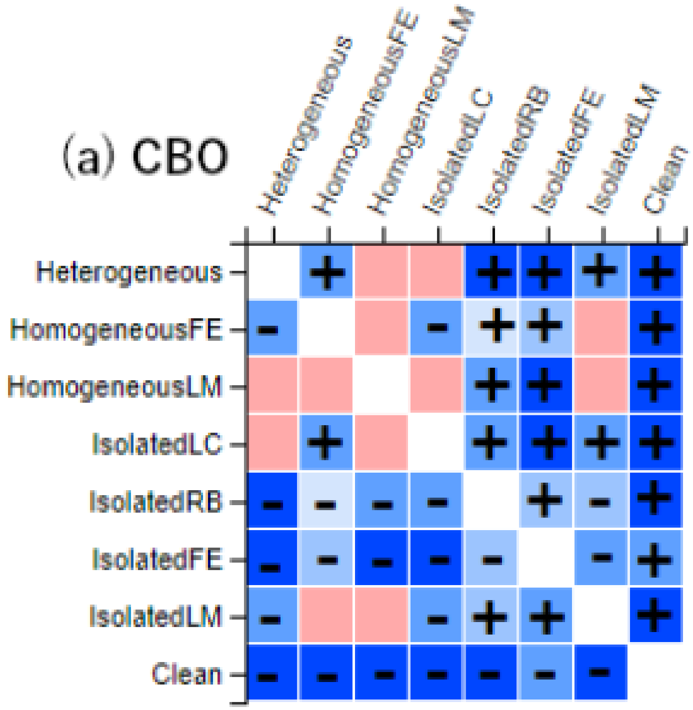
Homogeneous Agglomerations

Smell	Qualita Corpus	GitHub
Feature Envy	528 (0.347)	62 (0.233)
Long Method	25 (0.016)	28 (0.105)

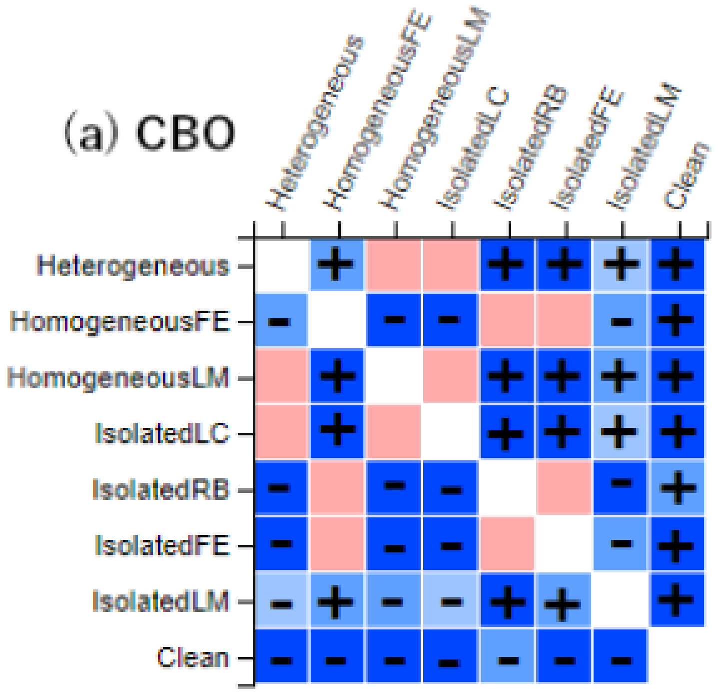
Homogeneous Agglomerations

Smell	Qualita Corpus	GitHub
Feature Envy	528 (0.347)	62 (0.233)
Long Method	25 (0.016)	28 (0.105)

Agglomeration’s Impact

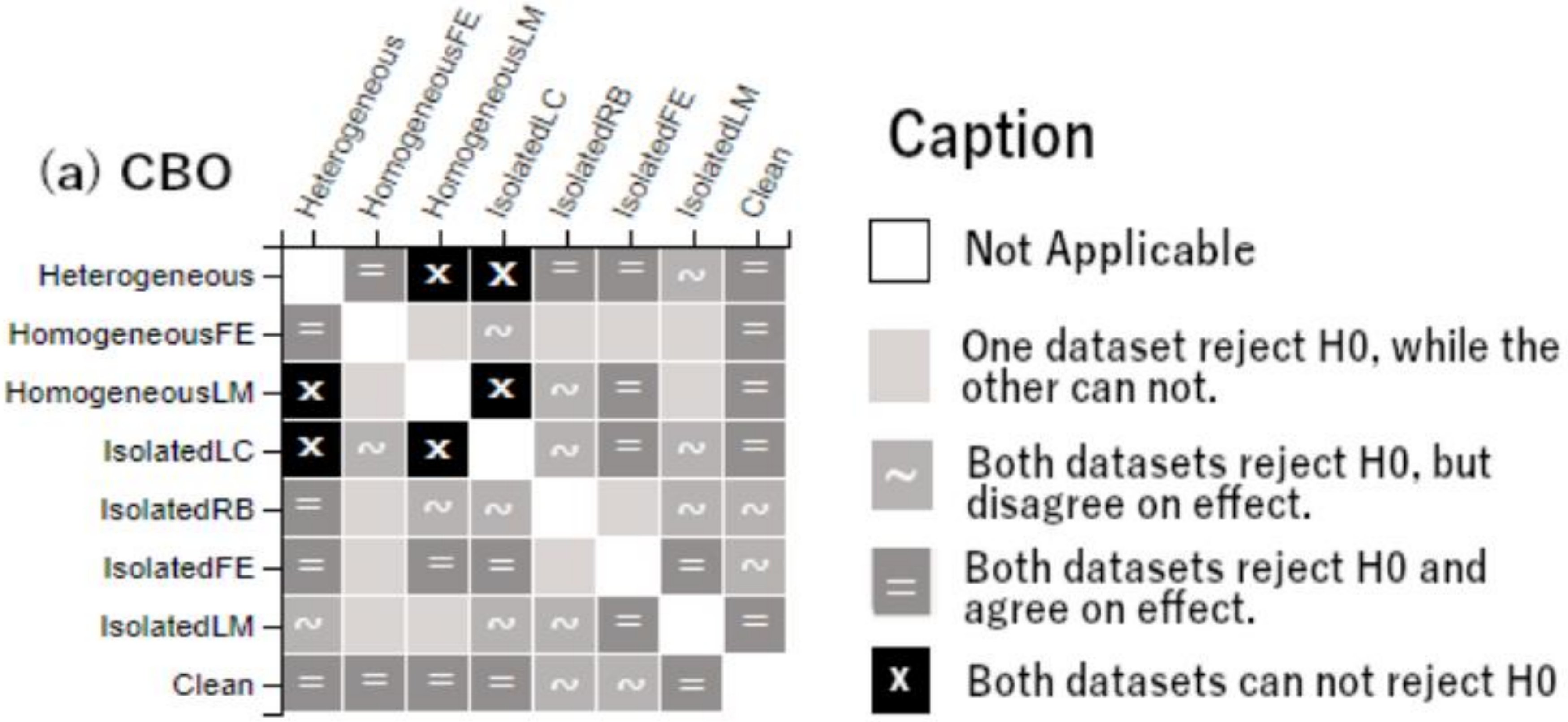


Qualita Corpus



GitHub

Agglomeration's Impact



Summary of Findings

- For the Qualita Corpus:

High Impact for Heterogeneous, Homogeneous and Isolated LC

Isolated RB impacted the most on DIT

NOC: Most positive effects against Isolated FE, Isolated LM and Clean

Agglomerations with LC has higher complexity

Agglomerations with LC has lower cohesion

Summary of Findings

- For the GitHub:

High Impact for Heterogeneous and Isolated LC, except on inheritance

Isolated RB impacted the most on DIT

Agglomerations with LC has higher complexity

Agglomerations with LC has lower cohesion

Summary of Findings

- For the Comparison:

≈	≈	≈	≈
Frequency of Homogeneous FE	LC+LM, RB+FE, LC+FE, FE+LM	Both datasets mostly agree with the hypothesis testing result	Both datasets mostly agree in not rejecting the H_0 for the Heterogeneous Agglomeration analysis

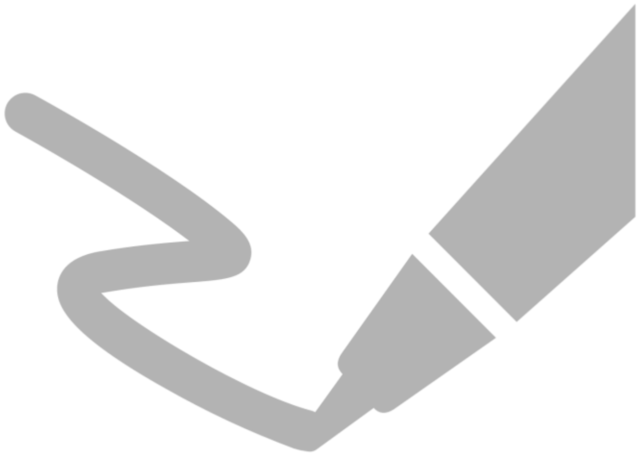
Threats to Validity

Threats to Validity

- Dataset construction
  - Systems used
  - Automatic detection tools
- Selected metrics
  - We rely on studies that evaluated their potential to explain the modularity
- Number of Heterogeneous Agglomerations on the GitHub dataset

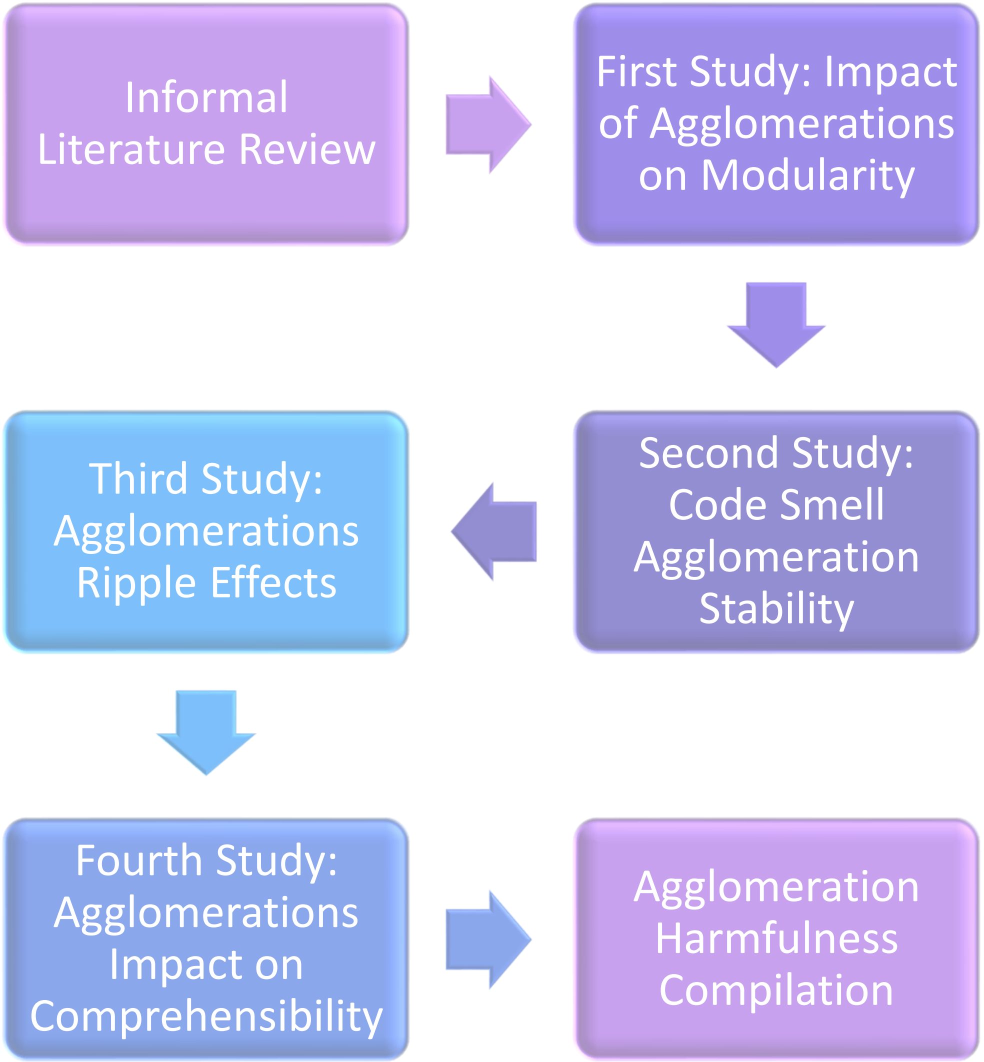
Lessons Learned

1. It is worth to explore in more depth the impact of Homogeneous Agglomerations.
2. Heterogeneous Agglomerations with Large Class smell impacted the most on the modularity.
3. Both datasets agree with the increased impact for the agglomerations.



Is the modularity impact felt by developers?

Project Thesis Overview



Amanda Santana, Eduardo Figueiredo, Juliana Alves Pereira. Unraveling the Impact of Code Smell Agglomerations on Code Stability. In Proceedings of the 40th International Conference on Software Maintenance and Evolution. 2024. IEEE Computer Society. Flagstaff, AZ, USA.

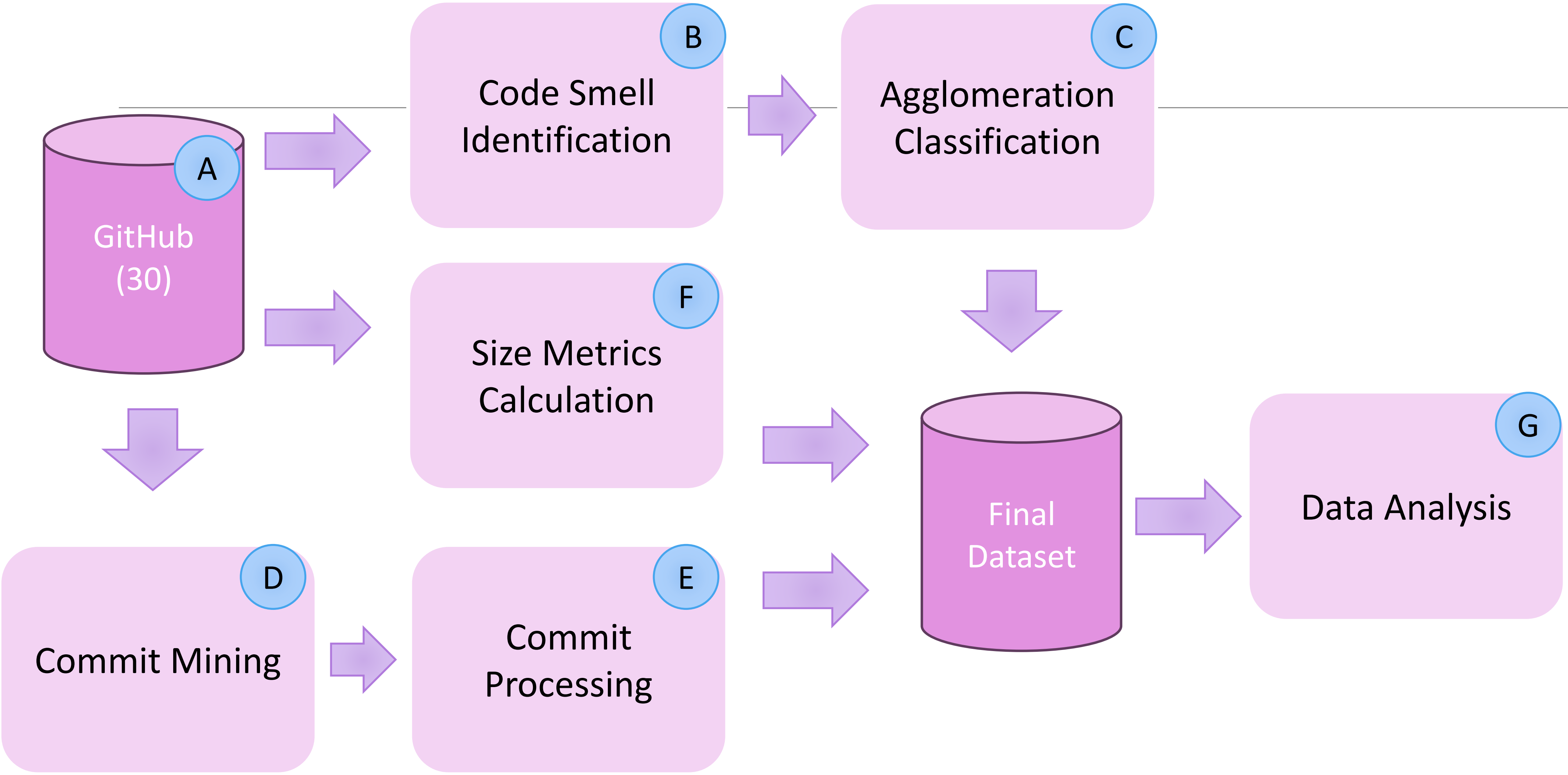
Goal

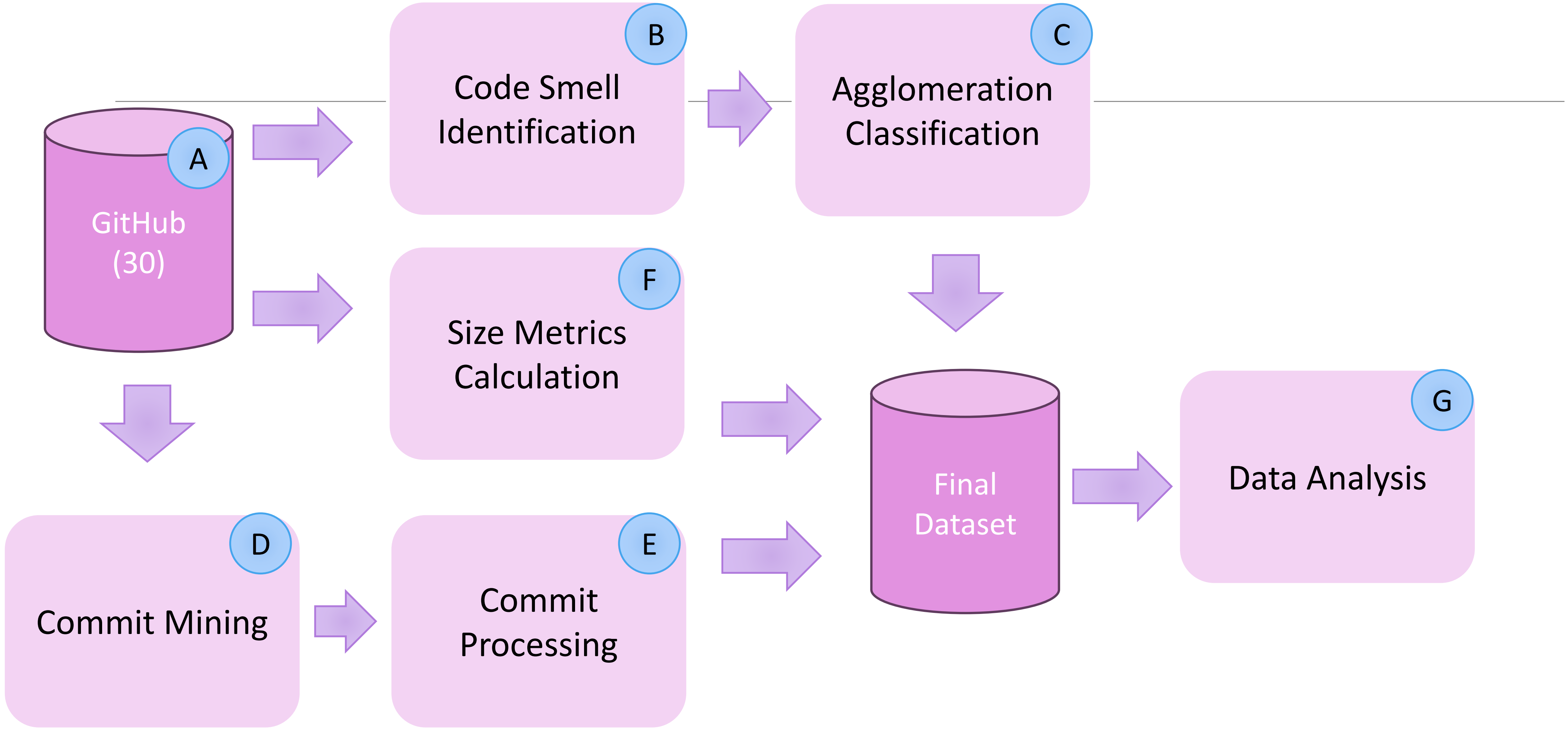
Provide Evidences of Code Smell Agglomeration Stability

Research Questions

- **RQ1:** *Do Heterogeneous and Homogeneous Agglomerations undergo changes in more frequency than Isolated and Clean types?*
- **RQ2:** Do Heterogeneous and Homogeneous Agglomerations undergo changes in more intensity than Isolated and Clean types?

Study Design


GitHub
(30)
A
Code Smell Identification
B
Agglomeration Classification
C
Size Metrics Calculation
F
Commit Mining
D
Commit Processing
E
Final Dataset
Data Analysis
G


A
GitHub (30)
B
Code Smell Identification
C
Agglomeration Classification
F
Size Metrics Calculation
D
Commit Mining
E
Commit Processing
Final Dataset
G
Data Analysis

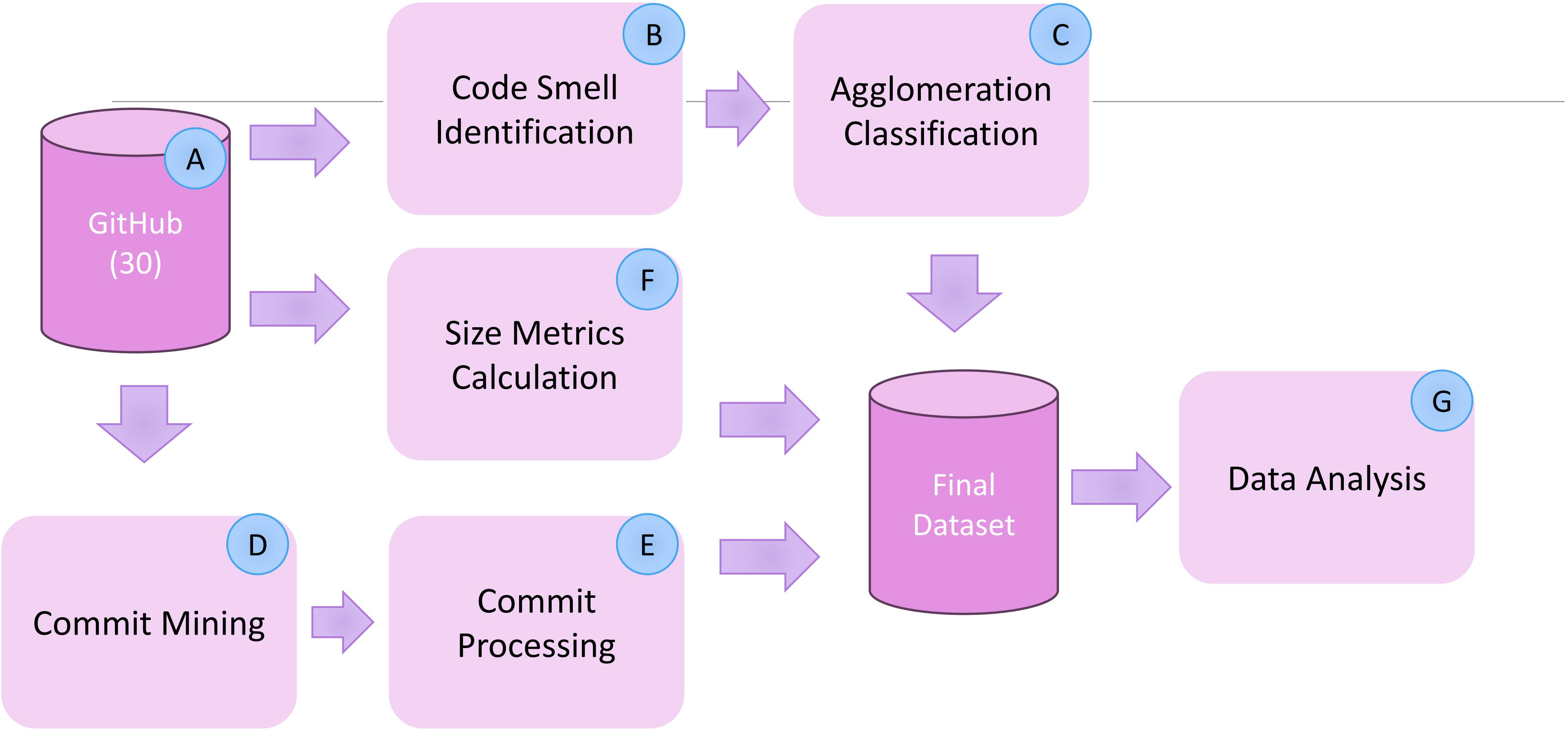
Systems Selection

Top-star GitHub Java systems

≥ 2 years of commits

Updated between 2021~2022

90% of its code in Java


A
GitHub
(30)
B
Code Smell
Identification
C
Agglomeration
Classification
F
Size Metrics
Calculation
Final
Dataset
G
Data Analysis
D
Commit Mining
E
Commit
Processing

Code Smells Identified

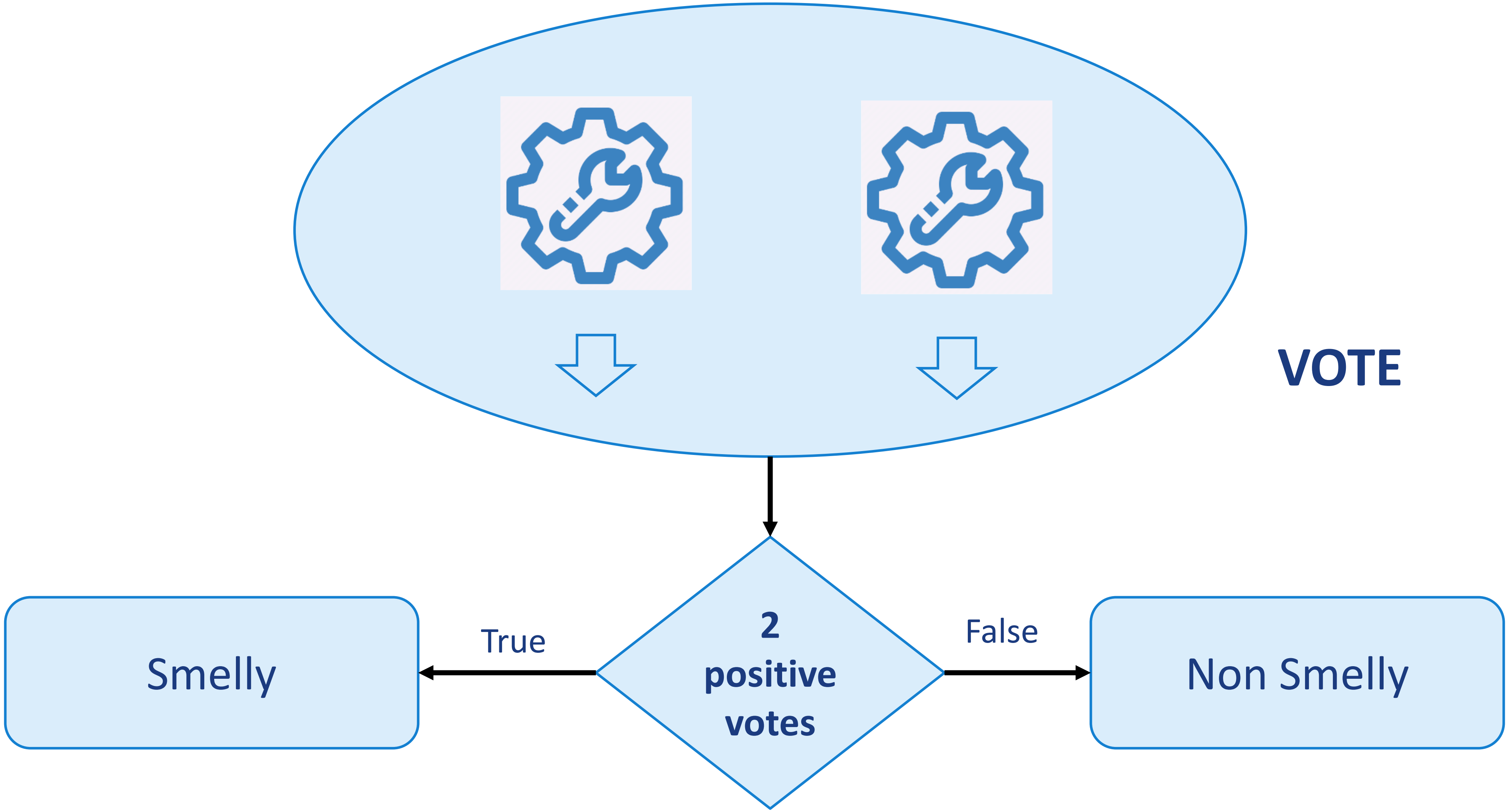
Class Level:

- Large Class (LC)
- Data Class (DC)
- Refused Bequest (RB)

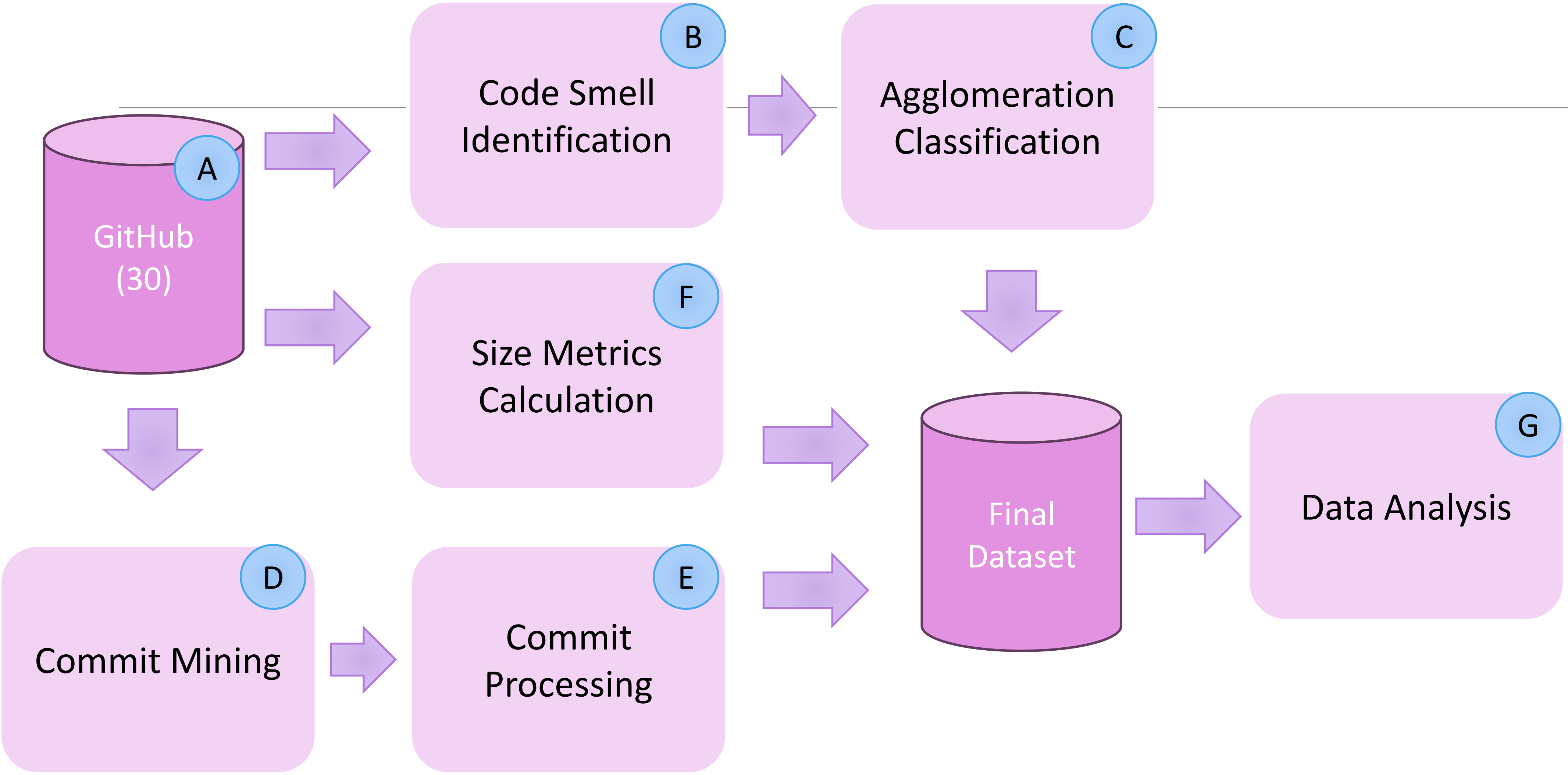
Method Level:

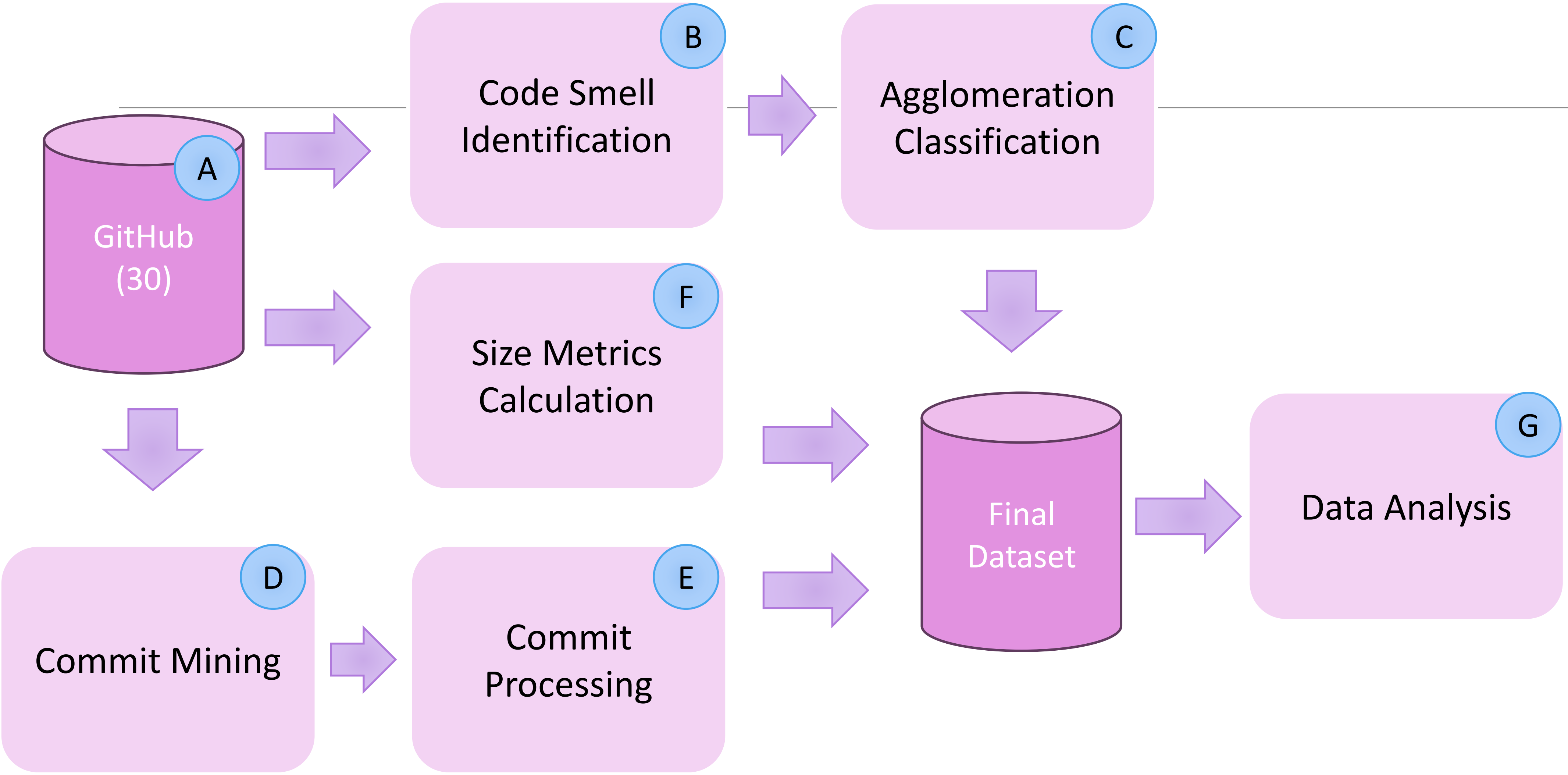
- Feature Envy (FE)
- Intensive Coupling (IC)
- Dispersed Coupling (DiCo)
- Long Parameter List (LPL)
- Shotgun Surgery (SS)
- Long Method (LM)

Detection Strategy

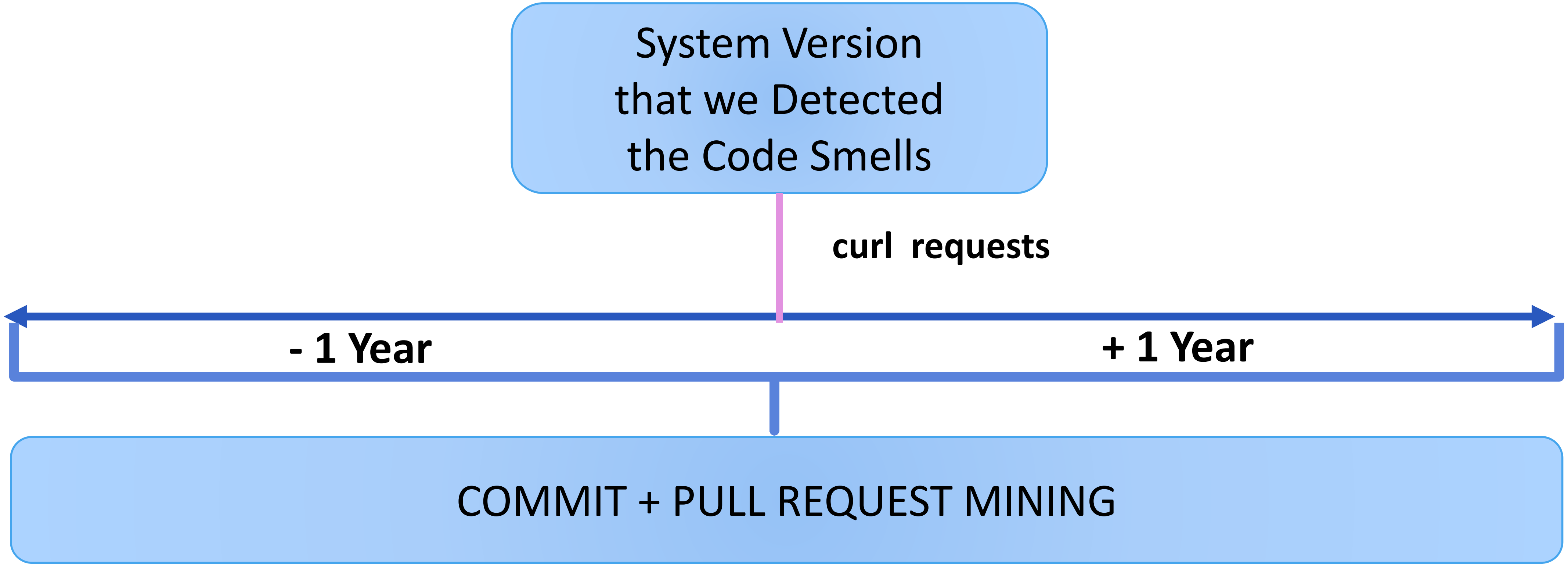


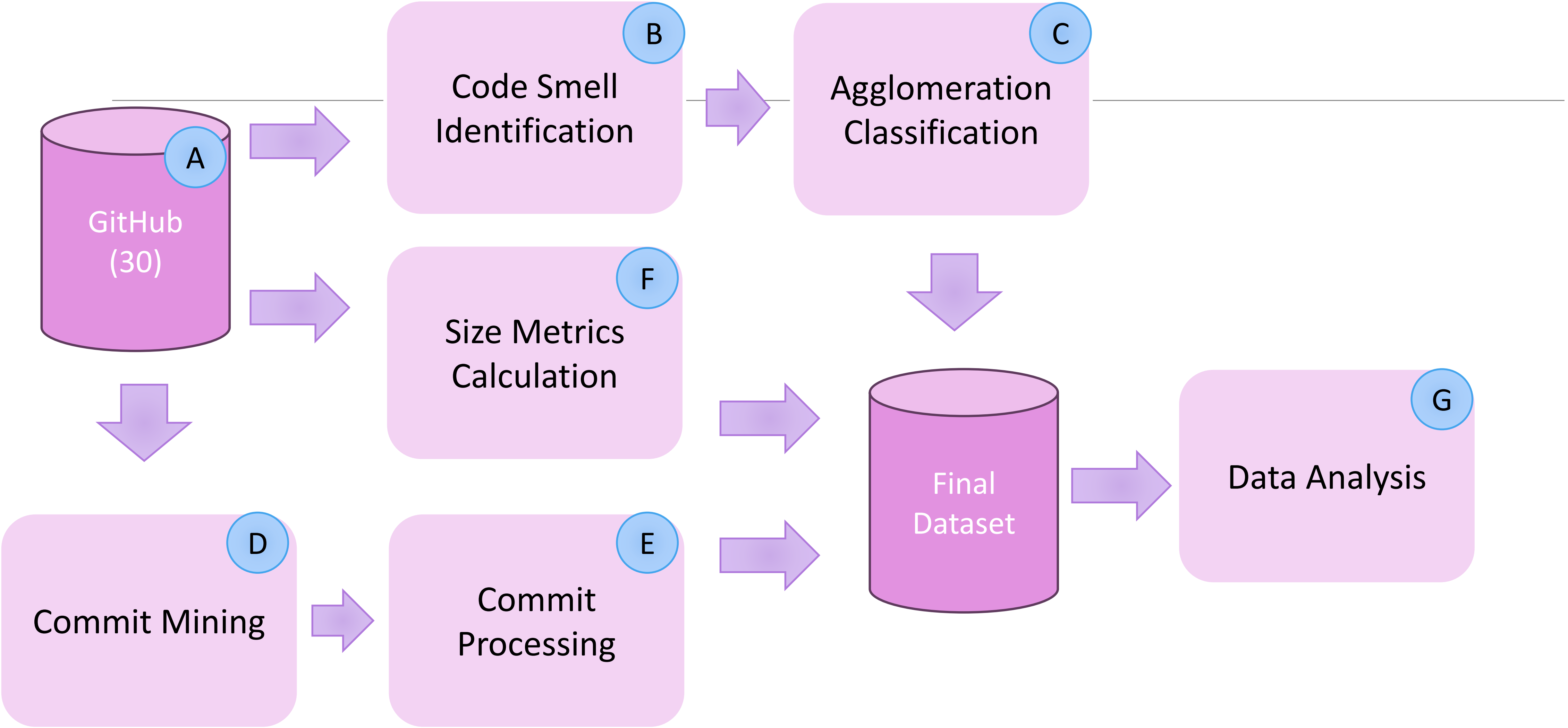
lab-soft
software engineering laboratory




GitHub
(30)
A
Code Smell
Identification
B
Agglomeration
Classification
C
Size Metrics
Calculation
F
Final
Dataset
Data Analysis
G
Commit Mining
D
Commit
Processing
E

GitHub Mining

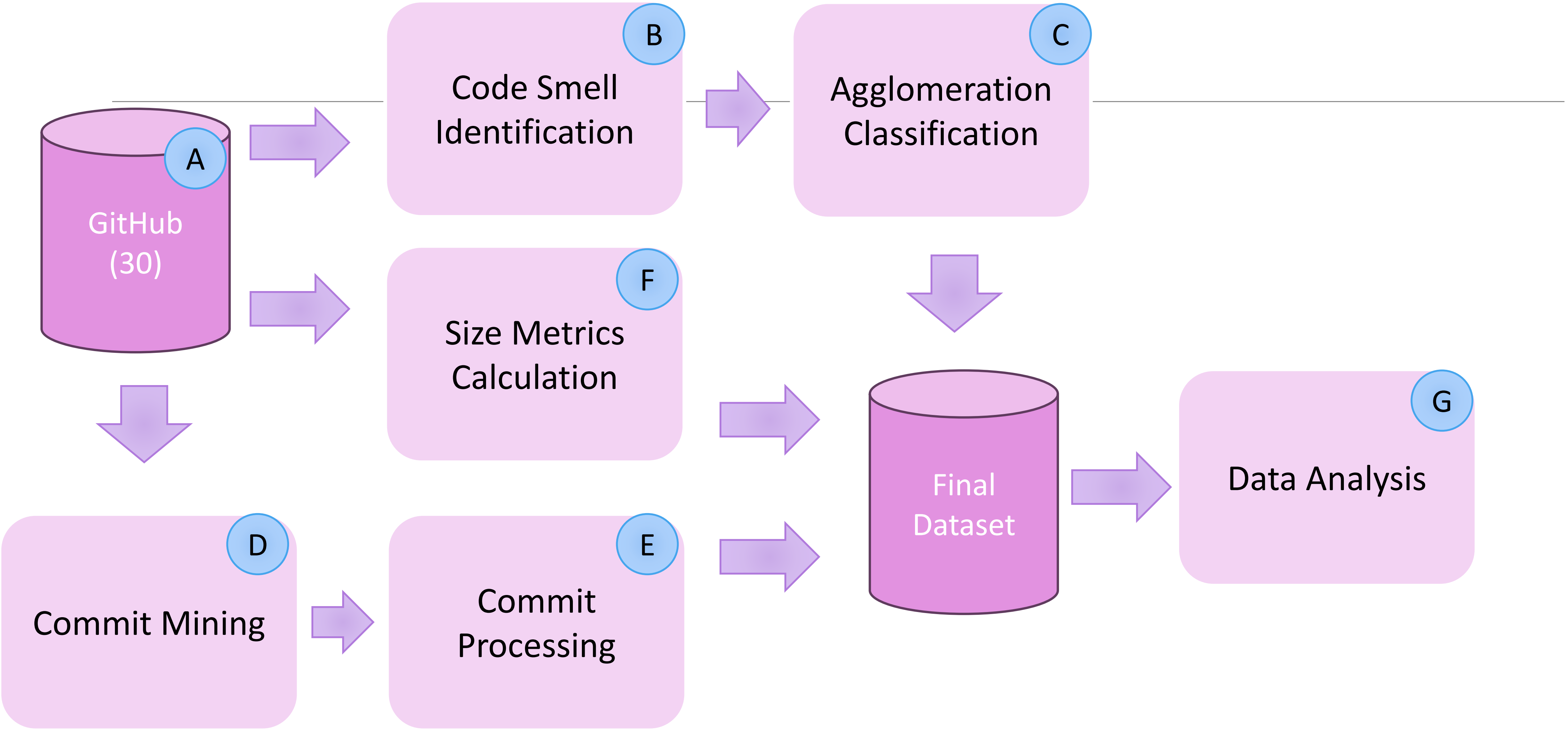


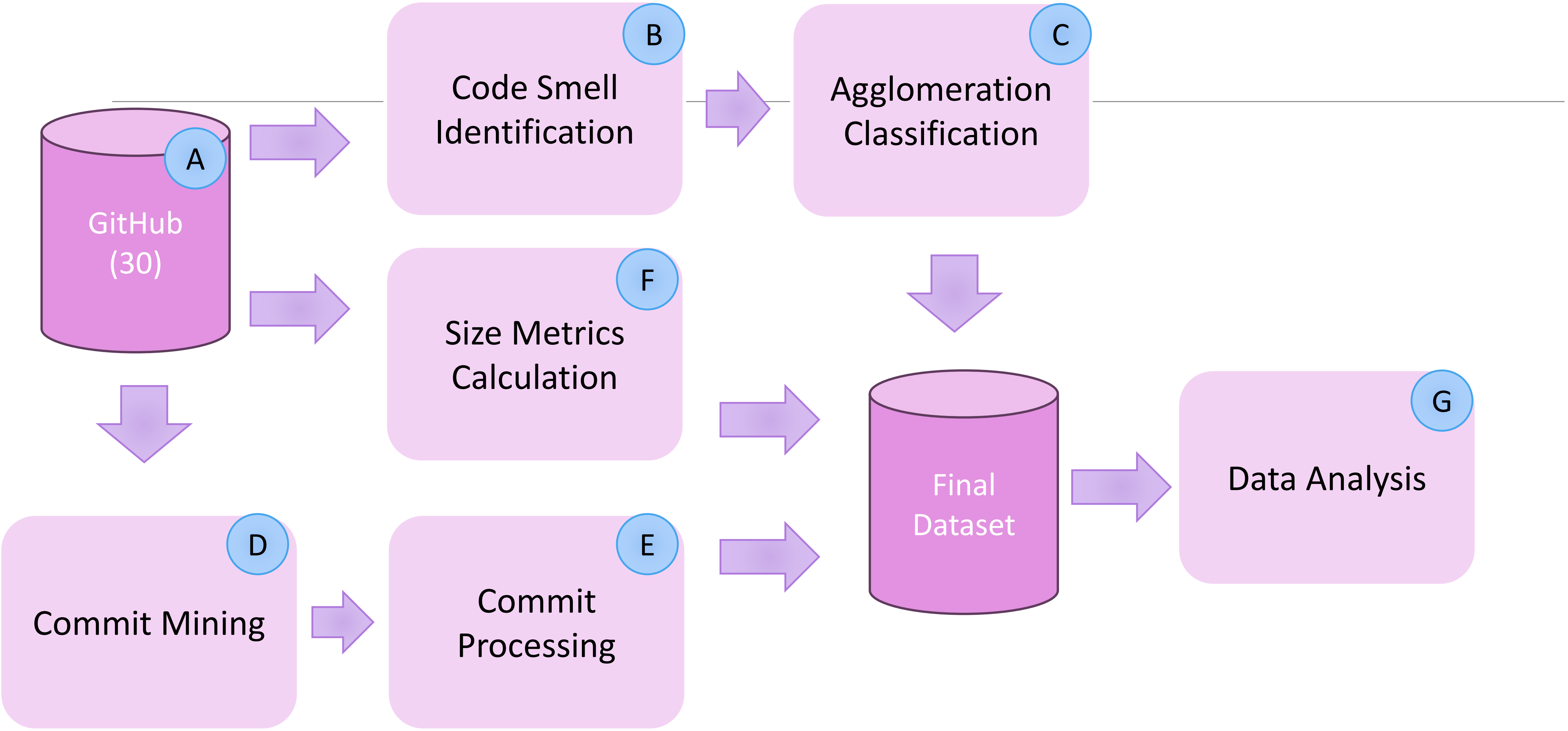


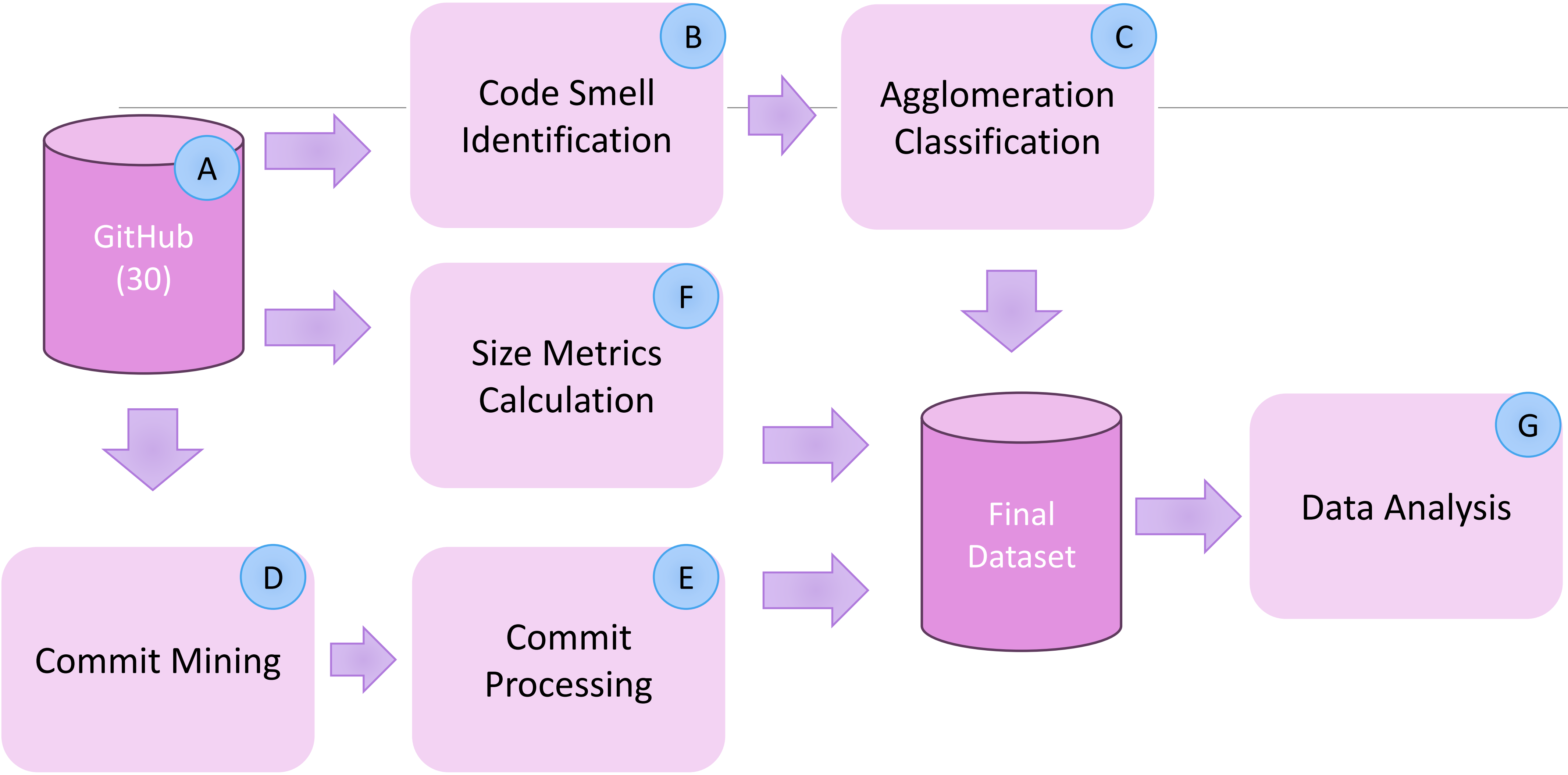
A
GitHub
(30)
B
Code Smell Identification
C
Agglomeration Classification
F
Size Metrics Calculation
Final Dataset
G
Data Analysis
D
Commit Mining
E
Commit Processing

Commit and Pull Request Processing

1. Associate a commit with an accepted pull request
2. Remove commits that are not associated with Java classes
3. Remove commits that are associated with test
4. Parse diff files to identify lines of code added/deleted
5. Parse diff files to identify and remove non-functional changes

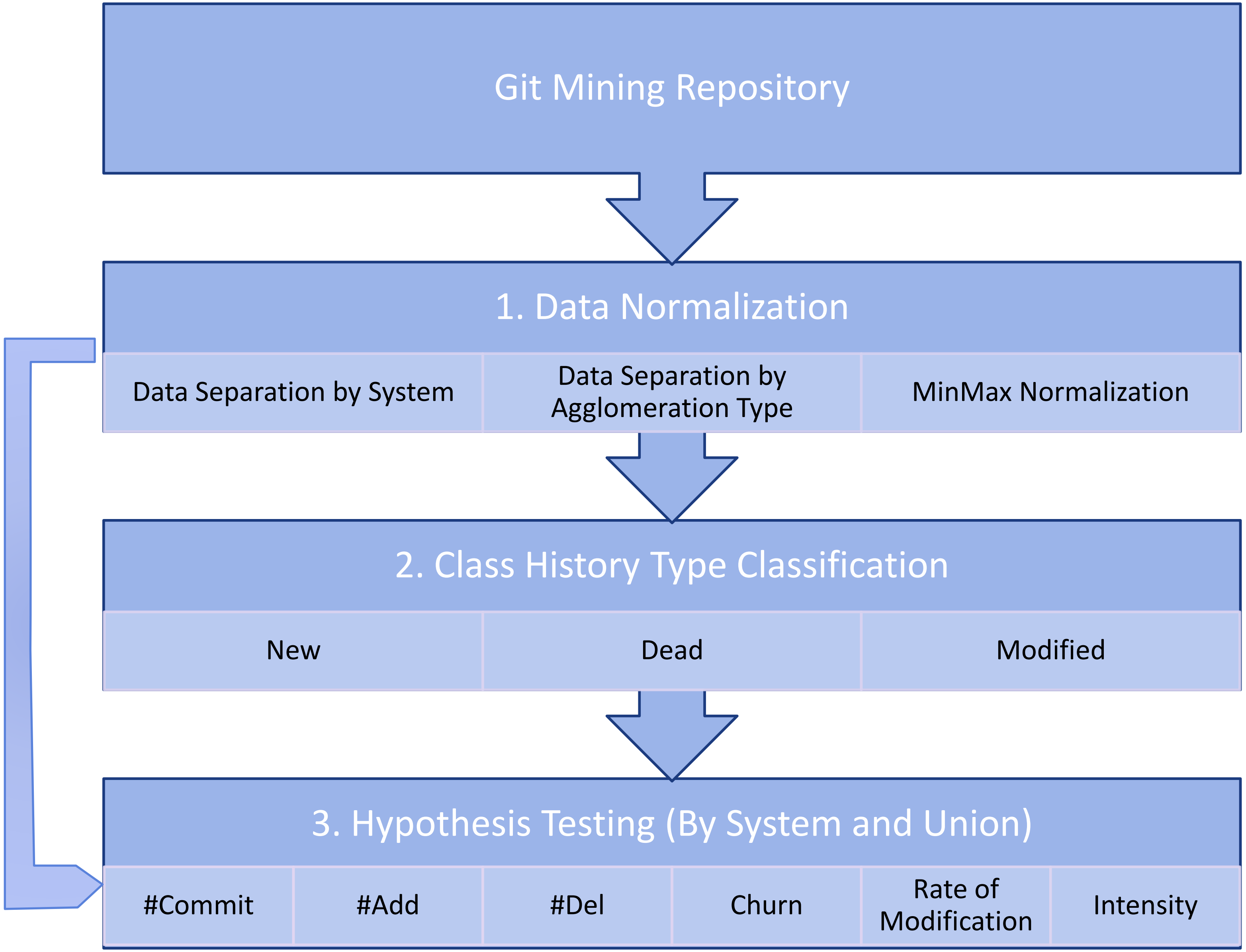

A
GitHub
(30)
B
Code Smell
Identification
C
Agglomeration
Classification
F
Size Metrics
Calculation
Final
Dataset
G
Data Analysis
D
Commit Mining
E
Commit
Processing


GitHub (30)
A
Code Smell Identification
B
Agglomeration Classification
C
Size Metrics Calculation
F
Final Dataset
Data Analysis
G
Commit Mining
D
Commit Processing
E



lab·soft
software engineering laboratory

Data Analysis Steps



Main Findings

Lines of Modified Code – Union Perspective – Class Type History

Dataset	Mod. Type	Agg. Type	Cliff's Delta
Dead	**#Add**	**Het-Isol**	**0.73**
		Het-Clean	**0.81**
		Isol-Clean	0.47
	Churn	**Het-Isol**	**0.73**
		Het-Clean	**0.81**
		Isol-Clean	0.47

Changes by rate of agglomerations found per system

System	Het.	Hom.	Isol.	Clean
arthas	0.94 (17)	0.6 (3)	0.77 (75)	0.38 (270)
easyexcel	1.0 (4)	0.0 (0)	0.95 (37)	0.68 (138)
hutool	1.0 (7)	1.0 (2)	0.70 (32)	0.59 (682)
Mean	0.63	0.35	0.42	0.21
Union	0.7 (344)	0.21 (38)	0.51 (719)	0.1 (5,073)

Intensity of Changes – Union – No Separation

	Cliff's Delta		
Agg. Types	**#Add**	**#Del**	**Churn**
Het-Hom	- 0.27		- 0.08
Het-Isol	0.26	- 0.11	
Het-Clean	0.99	0.89	0.95
Hom-Isol	0.55		
Hom-Clean	1.0	0.75	0.9
Isol-Clean	1.0	0.75	0.9

Intensity of Changes – Union – No Separation

	Cliff's Delta		
Agg. Types	**#Add**	**#Del**	**Churn**
Het-Hom	- 0.27		- 0.08
Het-Isol	0.26	- 0.11	
Het-Clean	**0.99**	**0.89**	**0.95**
Hom-Isol	0.55		
Hom-Clean	**1.0**	**0.75**	**0.9**
Isol-Clean	**1.0**	**0.75**	**0.9**

Intensity of Changes – Union - Separated

	Dead			Modified		
Agg. Type	#Add	#Del	Churn	#Add	#Del	Churn
Het-Hom				- 0.27		
Het-Isol				0.26	- 0.11	
Het-Clean	0.99		0.98	0.99	0.89	0.95
Hom-Isol				0.55		
Hom-Clean				1.0	0.75	0.9
Isol-Clean	0.99	0.98	0.99	1.0	0.85	0.97

Intensity of Changes – Union - Separated

Agg. Type	Dead #Add	Dead #Del	Dead Churn	Modified #Add	Modified #Del	Modified Churn
Het-Hom				- 0.27		
Het-Isol				0.26	- 0.11	
Het-Clean	**0.99**		**0.98**	**0.99**	**0.89**	**0.95**
Hom-Isol				0.55		
Hom-Clean				**1.0**	**0.75**	**0.9**
Isol-Clean	**0.99**	**0.98**	**0.99**	**1.0**	**0.85**	**0.97**

Summary of Findings

Agglomerations receives less commits

Union: Heterogeneous impacts on #add and Churn, mostly for Modified and Dead

By System: #Add, #Del and Churn for Het-Clean, Het-Isol, Isol-Clean, but Small~Negligible

Removal of non-functional code impacts the results

For most systems that had Heterogeneous Agg., they were modified in the 2-year time-span

Summary of Findings

Union: Compared to Clean classes, smelly classes change in more intensity

Union: Heterogeneous and Homogeneous had Large effects for intensity

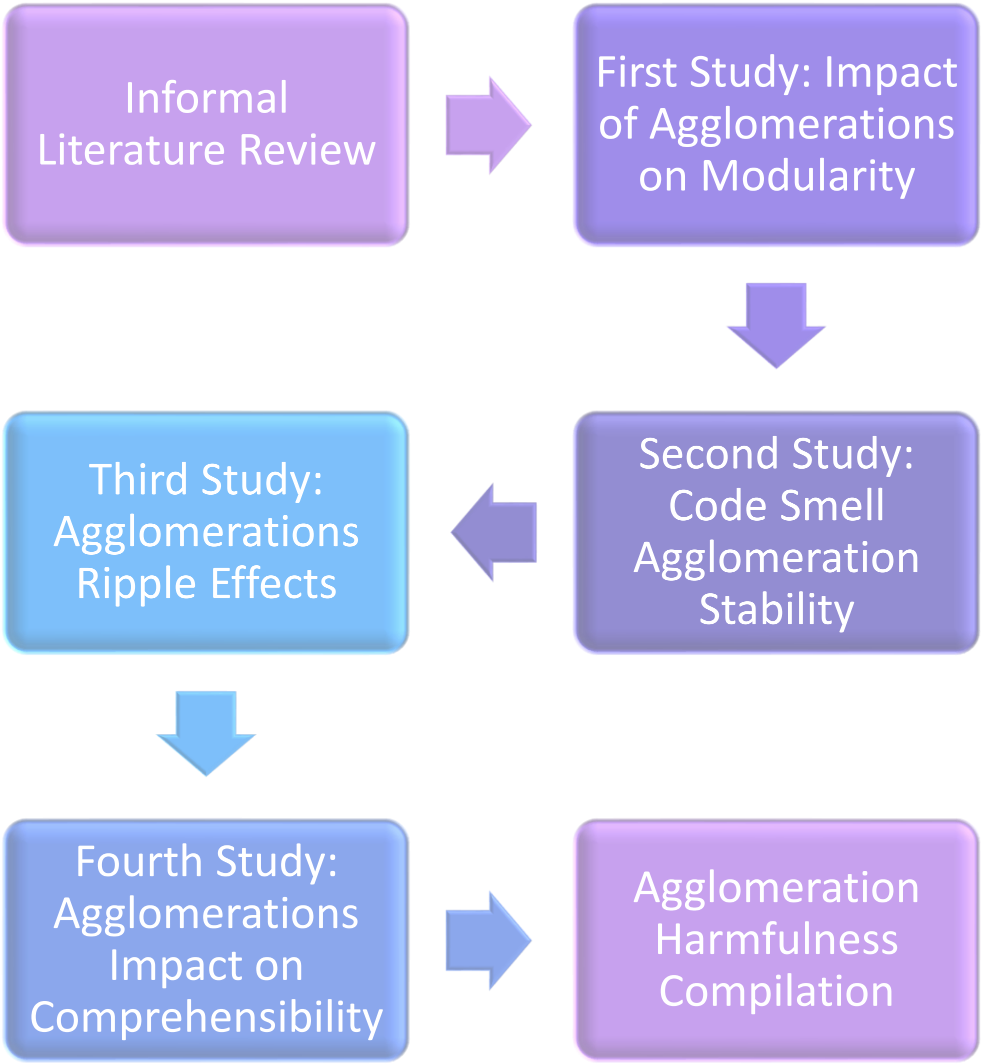
Modified dataset impact the most the Union intensity

By System: smelly classes change in more intensity than Clean classes for #Add and Churn

Threats to Validity

- Detection of smells at only one system release;
- Two year mining timespan;
- Systems selected;
- Effect of system size.

Evidences that agglomerations are changed frequently. We should keep exploring their impact!



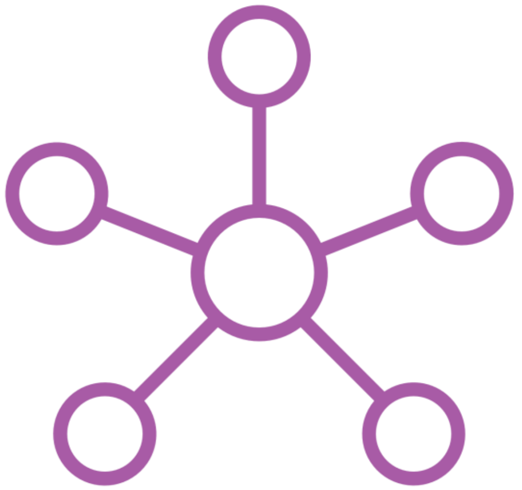
Project Thesis Overview

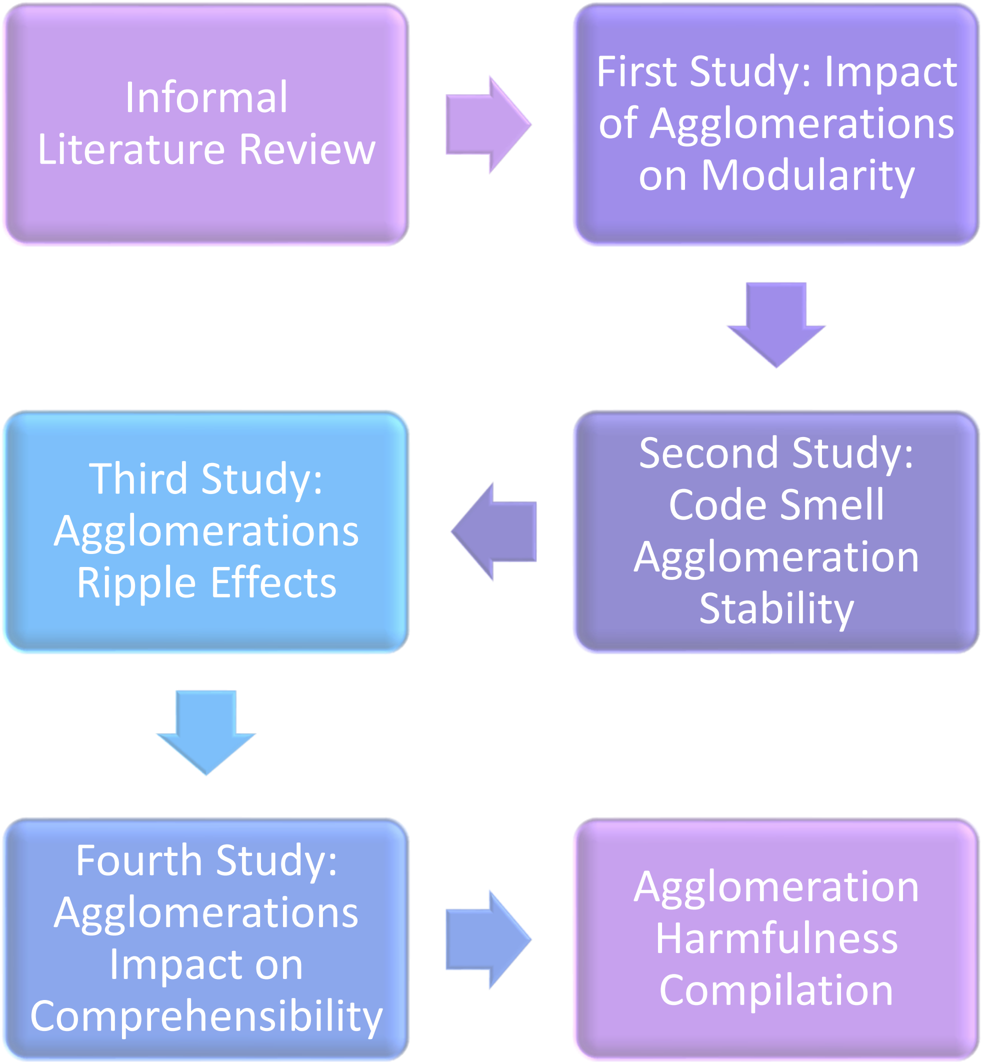
Third Study Planning

1. As on the First Study, extend the Second Study to a more in depth analysis:
    1. Break the Heterogeneous, Homogeneous, and Isolated types according to its smells.
    2. Hypothesis testing of the difference in change frequency and intensity.

Third Study Planning

2. Evaluate the proneness of ripple effects:
    1. Build a graph of class coupling.
    2. Evaluate the frequency in which agglomeration types changes with coupled/non-coupled classes.
    3. Evaluate in qualitative fashion a sample of “reasons” that an agglomeration changed another class.



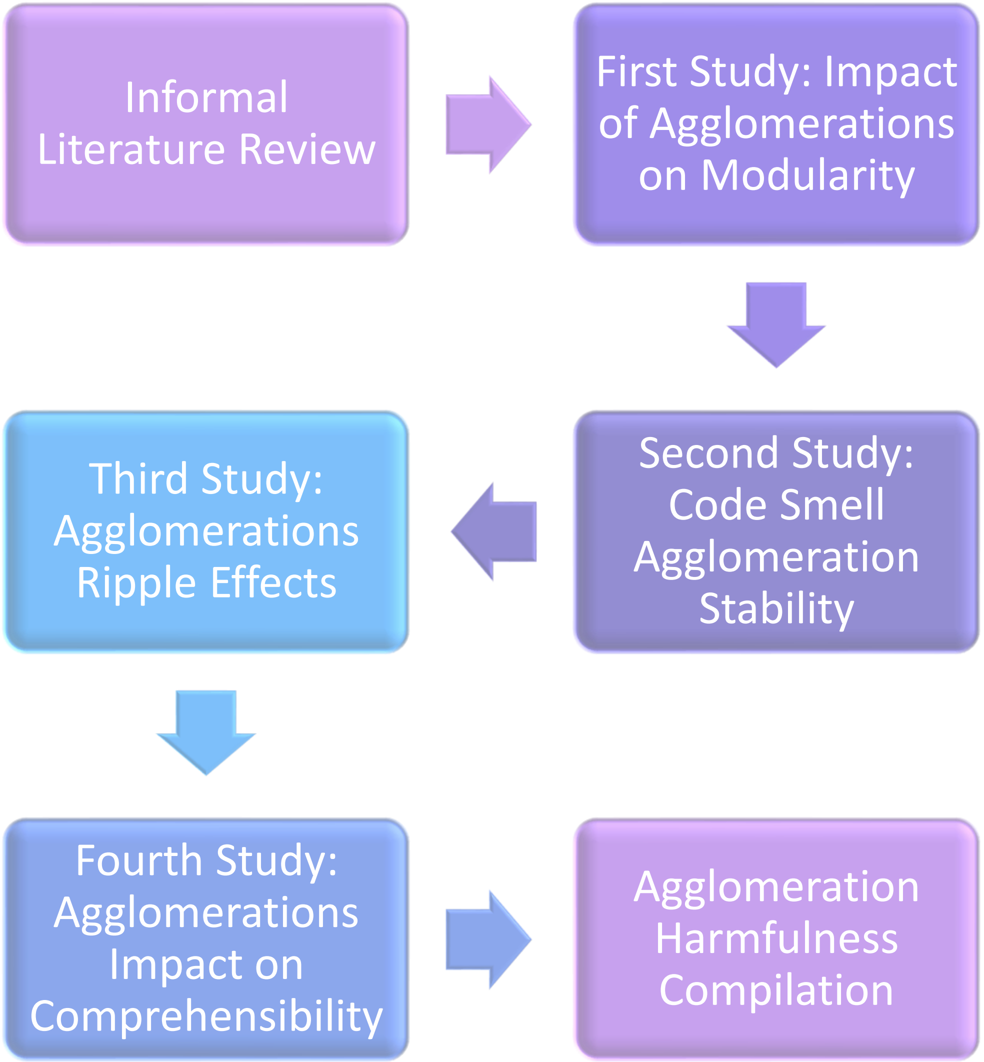


Project Thesis Overview

Fourth Study Planning

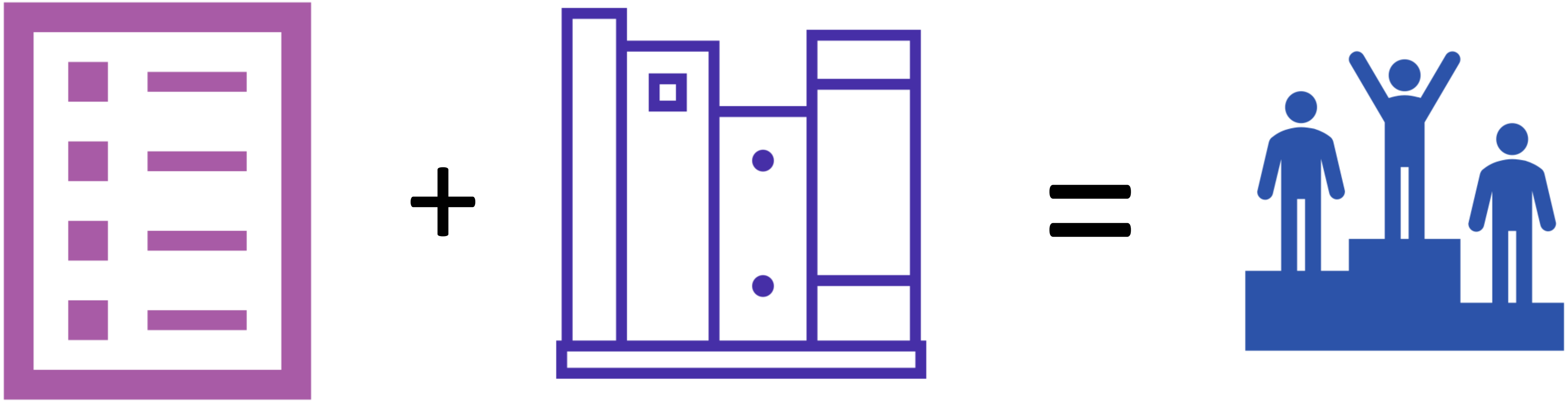
An qualitative/quantitative experiment with undergraduate/graduate students to assess agglomeration comprehensibility:

1. Students will conduct a series of comprehensibility tasks.
2. We will measure the time consumed, the correct answers, and students perceived effort.



Project Thesis Overview

Agglomeration Harmfulness Compilation



Literature Evidences + Our findings = Most harmful agglomerations

Contributions so far…

Contributions

- Evidences of agglomerations impact on modularity and their frequency/intensity of changes.
- A dataset of 30 Java systems:
  - 2-year of commits;
  - Detection output of 4 tools;
  - Metrics;
  - .class

Other Contributions

- Daniel Cruz, Amanda Santana, Eduardo Figueiredo. An Exploratory Evaluation of Continuous Feedback to Enhance Machine Learning Code Smell Detection. In: Congresso IberoAmericano em Engenharia de Software, 2024, Brasil. Anais do XXVII Congresso Ibero-Americano em Engenharia de Software (CIbSE 2024), 2024. p. 76.
- Henrique Gomes Nunes, Amanda Santana, Eduardo Figueiredo, Heitor Costa. Tuning Code Smell Prediction Models: A Replication Study. In: ICPC '24: 32nd IEEE/ACM International Conference on Program Comprehension, 2024, Lisbon Portugal. Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension. New York: ACM, 2024. p. 316.
- Geanderson Santos, Amanda Santana, Gustavo Vale, and Eduardo Figueiredo. Yet Another Model! A Study on Model's Similarities for Defect and Code Smells. In proceedings of the 26th International Conference on Fundamental Approaches to Software Engineering (FASE), LNCS, volume 13991. Paris, 2023.

Expected Contributions

Expected Contributions

Evidences of agglomeration harmfulness considering different quality aspects.

Help developers in prioritizing code refactoring considering the aspects that are more valuable to their context

Conclusions

Conclusion

- Homogeneous Agglomerations should be further explored.
- Heterogeneous and Homogeneous agglomerations change with more frequency, in more intensity and impacts the most the modularity.
- The agglomeration harmfulness can be used by developers to prioritize the refactoring.

Thank you! :)

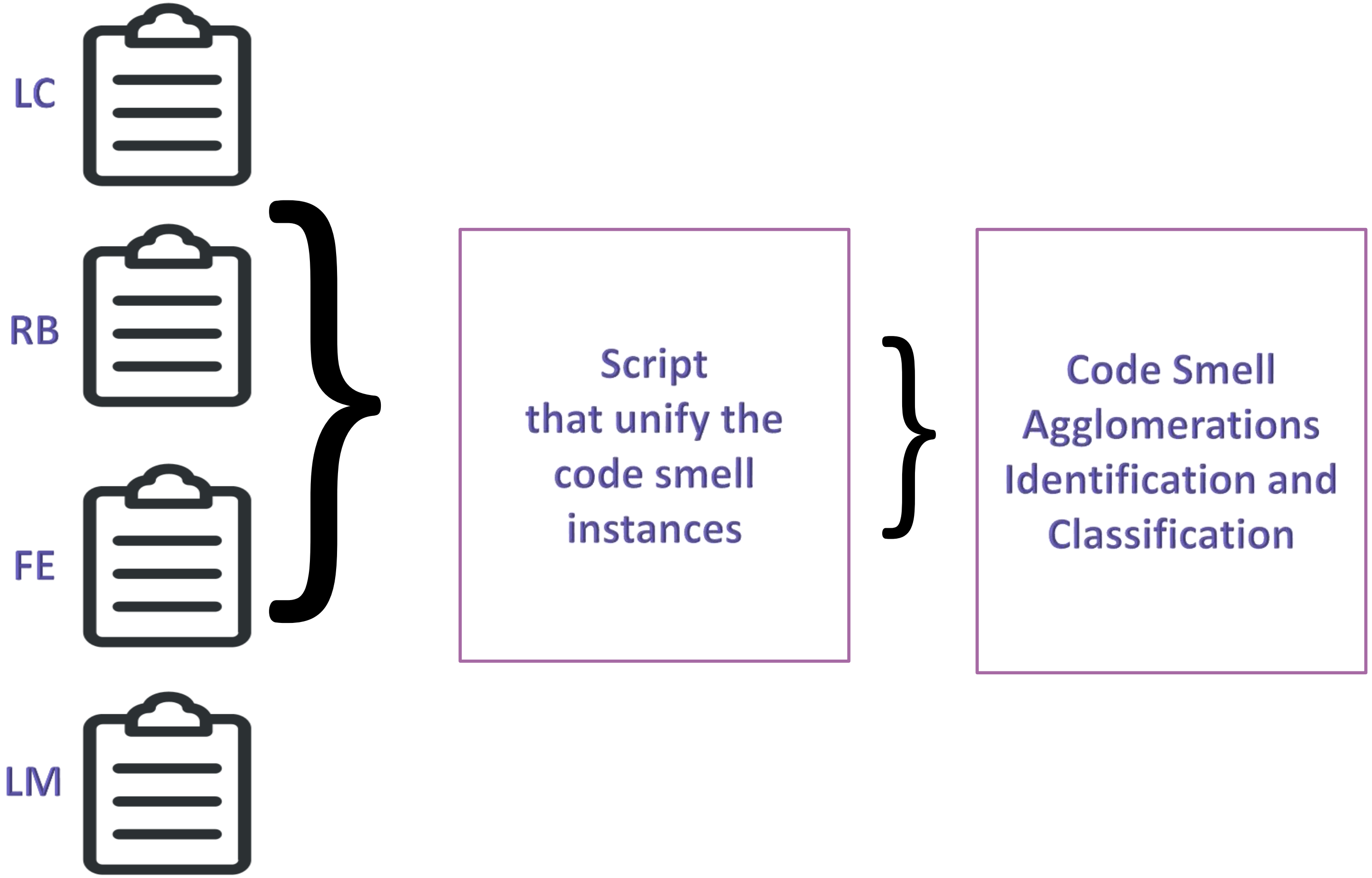
AMANDADS@DCC.UFMG.BR

Bibliography

- Abbes, M., Khomh, F., Guéhéneuc, Y., and Antoniol, G. (2011). An empirical study of the impact of two antipatterns, blob and spaghetti code, on program comprehension. In 2011 15th European Conference on Software Maintenance and Reengineering, pages 181–190. ISSN 1534-5351.

- Palomba, F., Bavota, G., Di Penta, M., Fasano, F., Oliveto, R., and De Lucia, A. (2018). On the diffuseness and the impact on maintainability of code smells: A large scale empirical investigation. In 2018 IEEE/ACM 40th International Conference on Software Engineering. (ICSE), pages 482–482. ISSN 1558-1225.

Step C - Identification of agglomerations



Step C –Item set Mining

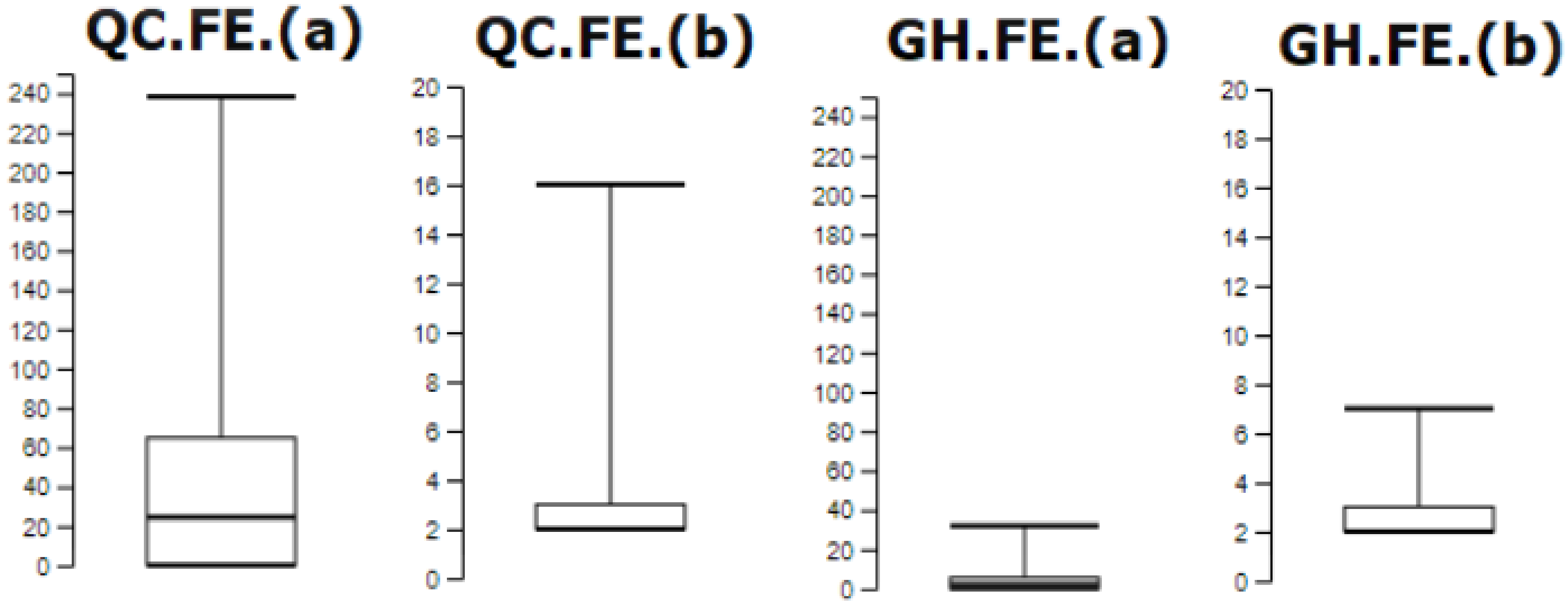
- Use of Support to identify which combinations of smells are more frequent and relevant (redundancy removal) on both datasets
- Support > 0.15

Step C - Homogeneous Agglomerations identification

- Restriction of the Association Rule Algorithm that items in the Antecedent cannot appear on Consequent

Frequency Statistics and Variation Measurements

FE Homogeneous Agglomerations



Heterogeneous Agglomerations

	Qualita Corpus	GitHub
Itemset	Support (Count)	Support (Count)
(RB, FE)	0.27 (264)	0.165 (29)
(LC, FE)	0.307 (298)	0.3522 (62)
(LM, FE)	0.47 (458)	0.3466 (61)
(LC, LM)	0.169 (164)	0.335 (59)

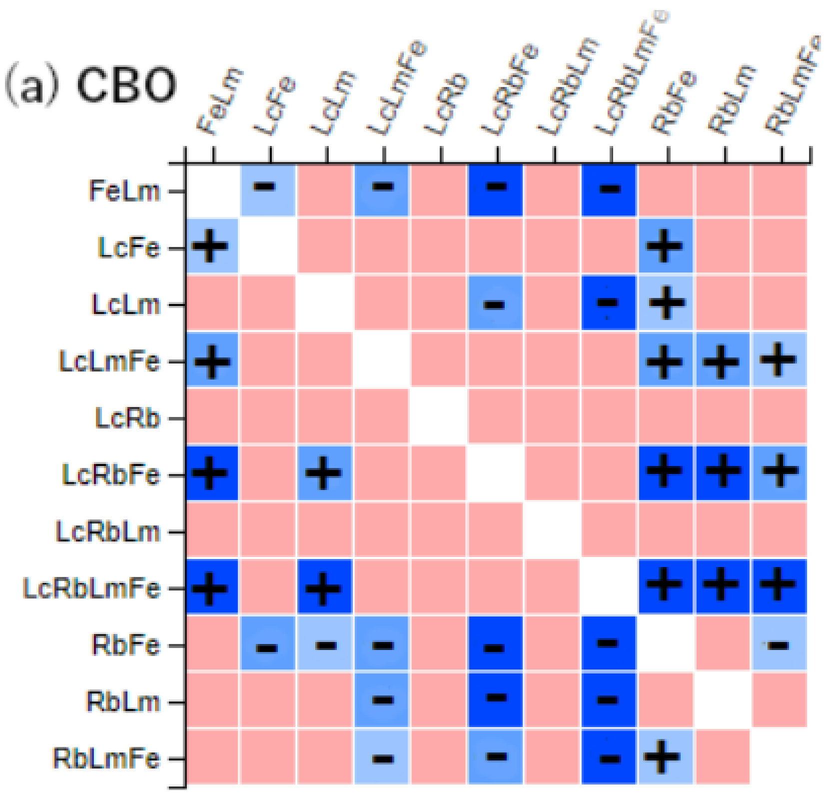
Research Quest Answers

RQ1. Are Homogeneous Agglomerations frequent in the source code? *For both datasets, we have found that they are indeed frequent, mainly the Homogeneous Feature Envy agglomeration, with more than 20% of participation.*

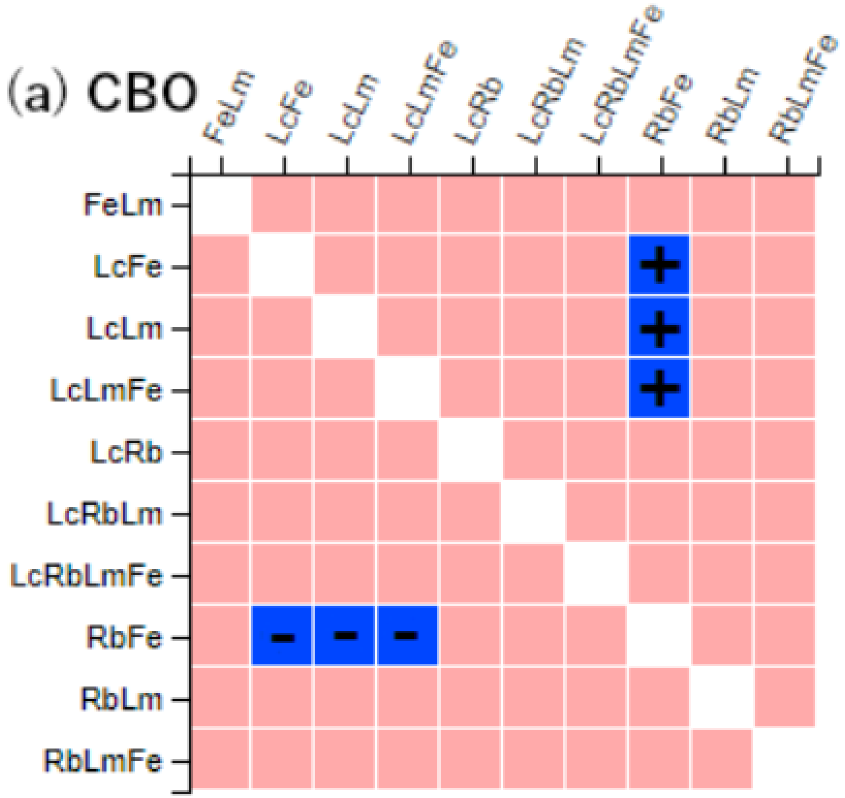
Research Quest Answers

RQ2. Which Heterogeneous Agglomerations are more common in the source code? *For both datasets, we have found that all the heterogeneous item sets were composed of two smells: (LC,LM), (RB,FE), (LC,FE), and (FE,LM).*

Heterogeneous's Impact



Qualita Corpus



GitHub

RQ3 Answer

RQ3. How do code smell agglomerations impact on the system modularity?

With inheritance being an exception (DIT metric), for most metrics, we could observe that both datasets agree with rejecting the null hypothesis. We could also observe that most of these agreements are in relation to Heterogeneous Agglomerations and Clean classes, with Heterogeneous usually with large and positive effects.

RQ4 Answer

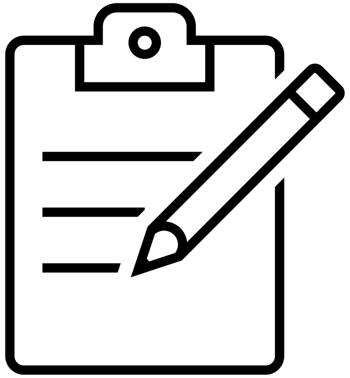
RQ4. Does the different types of Heterogeneous Agglomerations have an uniform impact on the system modularity?

For both datasets, we have not found statistical differences between the different agglomerations for the CBO, DIT and maxNest metrics. Although we did not find significant difference in the metric behavior of Heterogeneous Agglomerations, we provide initial evidences that some agglomerations behave differently on the Qualita Corpus dataset.

lab-soft
software engineering laboratory

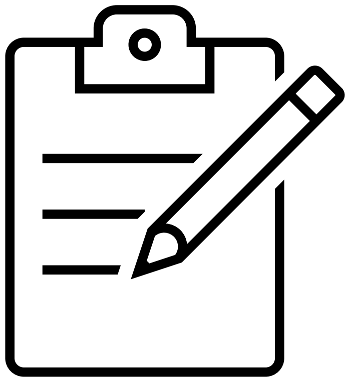
Some important concepts

- Perspective:
  - Union: all 30 systems are considered as our dataset;
  - By System: each system is considered individually.
- Modification Type:
  - #Add: Number of lines added in the commit;
  - #Del: Number of lines deleted in the commit;
  - Churn: #Add + #Del



Some concepts

- Class History Type:
  - Modified: classes that were modified in our two year time-span
  - New: classes created in our two-year time-span
  - Dead: classes deleted in our two-year time-span



Commits

- Het=Clean (-0.58)
- Hom=Clean (-0.58)
- Isol=Clean (-0.584).

Clean classes receive statistically more commits than smelly classes, and the difference is Large.

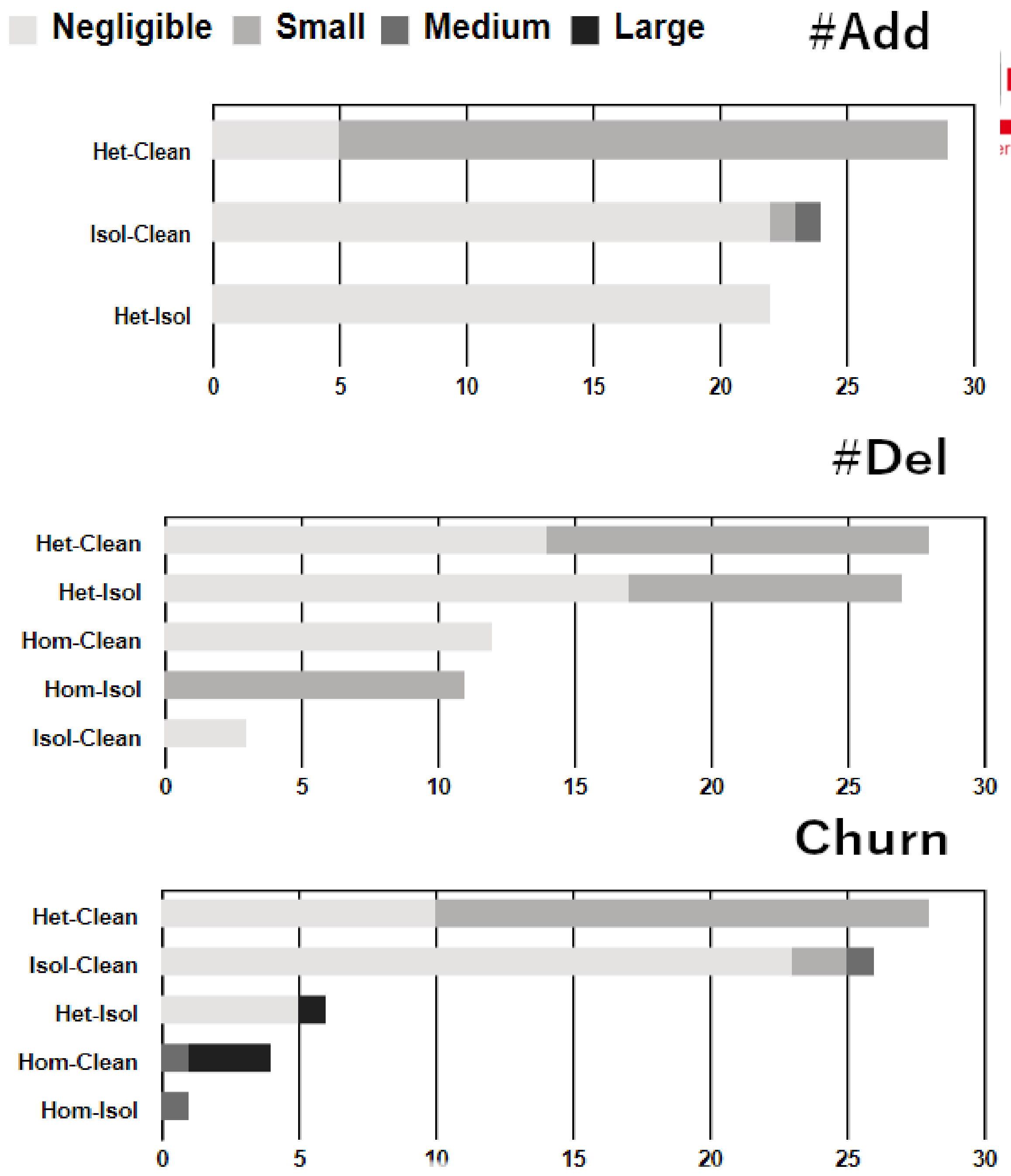
Lines of Modified Code – Union Perspective – Class Type History

Dataset	Mod. Type	Agg. Type	Cliff's Delta
No Separation	#Add	Het-Hom	0.37
		Hom-Clean	-0.47
Modified	#Add	Het-Isol	0.11
		Het-Clean	0.27
		Isol-Clean	0.13
	#Del	Het-Isol	0.12
		Het-Clean	0.17
		Hom-Isol	0.20
		Hom-Clean	0.26
	Churn	Het-Isol	0.12
		Het-Clean	0.27
		Isol-Clean	0.13

Lines of Modified Code – Union Perspective – Class Type History

Dataset	Mod. Type	Agg. Type	Cliff's Delta
No Separation	**#Add**	**Het-Hom**	**0.37**
		Hom-Clean	**-0.47**
Modified	**#Add**	Het-Isol	0.11
		Het-Clean	**0.27**
		Isol-Clean	0.13
	#Del	Het-Isol	0.12
		Het-Clean	0.17
		Hom-Isol	0.20
		Hom-Clean	**0.26**
	Churn	Het-Isol	0.12
		Het-Clean	**0.27**
		Isol-Clean	0.13

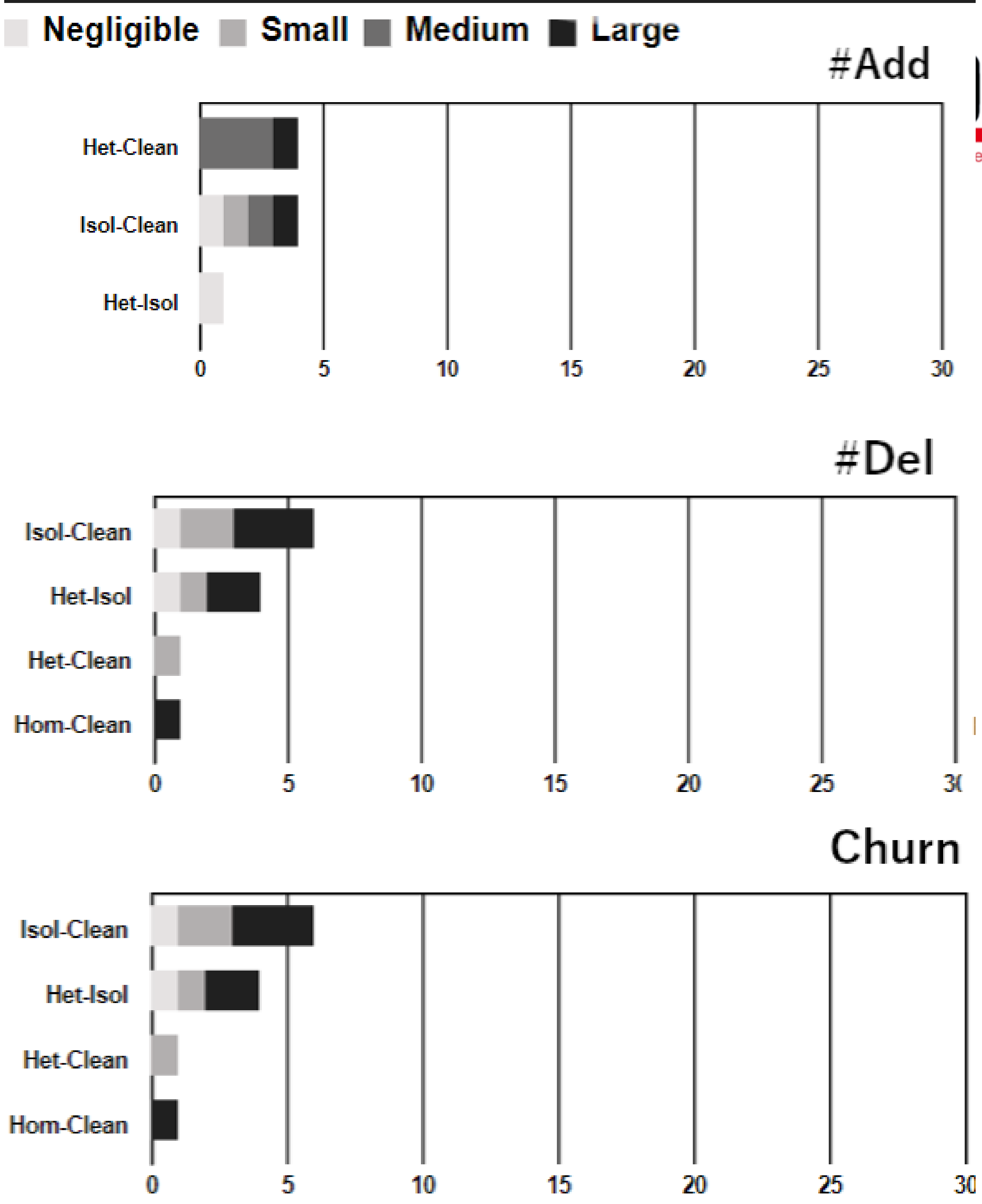
Intensity of Changes - No Separation – By System



System	Het.	Hom.	Isol.	Clean
arthas	**0.94** (17)	0.6 (3)	0.77 (75)	0.38 (270)
cryptomator	**1.0** (1)		0.89 (8)	0.27 (158)
dbeaver	**0.73** (169)	0.06 (5)	0.59 (237)	0.17 (950)
easyexcel	**1.0** (4)	0.0 (0)	0.95 (37)	0.68 (138)
elasticsearch	0.0 (0)		**0.17** (1)	0.0
fastjson	**0.73** (11)	0.67 (4)	0.44 (12)	0.09 (18)
gson		**1.0** (2)	0.5 (2)	0.19 (42)
guava	**0.57** (4)	0.5 (2)	0.48 (23)	0.01 (274)
HikariCP	**0.67** (2)		0.29 (2)	0.09 (42)
hutool	**1.0** (7)	**1.0** (2)	0.70 (32)	0.59 (682)
java-faker			0.5 (1)	**0.50** (52)
jedis	0.0 (0)	0.0 (0)	**0.47** (7)	0.19 (136)
jenkins	0.78 (18)	0.1 (1)	**0.86** (62)	0.35 (825)
jitwatch	**0.05** (1)	0.0 (0)	0.0 (0)	0.01 (3)
jsoup	**1.0** (4)		0.43 (3)	0.11 (26)
junit4			0.0 (0)	0.06 (17)
libgdx	**0.70** (31)	0.55 (6)	0.26 (42)	0.13 (320)
mall	0.0 (0)	0.0 (0)	**0.14** (1)	0.03 (21)
mybatis-3	0.63 (5)	**0.8** (4)	0.24 (19)	0.72 (204)
nanohttpd	**1.0** (1)		0.75 (3)	0.47 (33)
netty-socketio	**1.0** (1)		0.33 (2)	0.08 (11)
redisson	**0.65** (13)	0.2 (2)	0.62 (21)	0.12 (184)
retrofit	0.0 (0)	0.0 (0)	0.0 (0)	0.0 (0)
rocketmq	**0.88** (38)	0.18 (2)	0.50 (64)	0.28 (232)
Sa-Token			0.0 (0)	**0.12** (22)
Sentinel	0.33 (3)	0.0 (0)	**0.35** (23)	0.23 (216)
spring-cloud	**1.0** (3)		**1.0** (27)	0.53 (202)
webmagic	**1.0** (1)	0.5 (1)	0.18 (2)	0.05 (10)
xxl-job	**0.33** (1)	0.0 (0)	0.10 (2)	0.05 (6)
zxing	0.43 (9)	**0.8** (4)	0.22 (11)	0.07 (16)
Mean	**0.63**	0.35	0.42	0.21
Union	**0.7** (344)	0.21 (38)	0.51 (719)	0.10 (5073)

lab·soft
software engineering laboratory

Intensity of Changes – By System - Modified Dataset



Answering RQ1

- ***RQ1:*** *Do Heterogeneous and Homogeneous Agglomerations undergo changes in more frequency than Isolated and Clean types?*

For the Union dataset, smelly classes change more frequently than Clean ones. We also found that Heterogeneous agglomerations change more frequently than other agglomeration types. For number of commits, we found evidence favorable to the Clean classes being unstable.

Answering RQ2

- **RQ2:** *Do Heterogeneous and Homogeneous Agglomerations undergo changes in more intensity than Isolated and Clean types?*

In the Union perspective, we could observe that smelly classes tend to change in more intensity in three modification types, with the presence of Large Effects. We observe similar results for the Modified and No Separation dataset. When observing the results by system, for all modification types we could not reject H0 for more than 50% of the systems. We provide evidence that, smelly classes change in more intensity than clean ones, mainly the Heterogeneous Agglomeration.