UFMG

lab-soft
software engineering laboratory

Understanding Code Smell Agglomerations Impact on Source Code: Current Work

Amanda Santana,
Eduardo Figueiredo

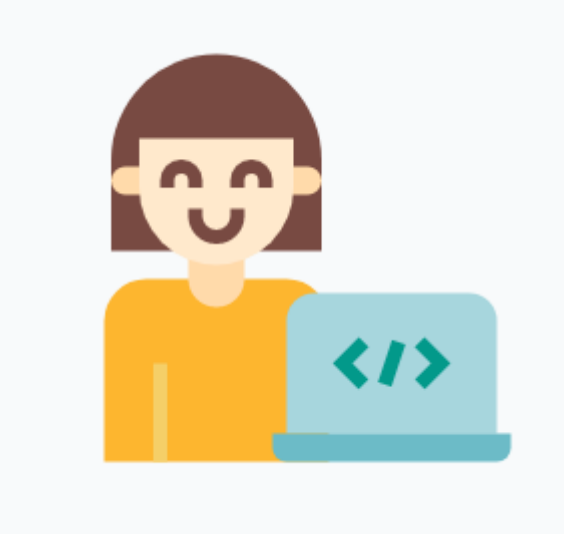
Software Engineering Lab (LabSoft)
http://labsoft.dcc.ufmg.br/

Agenda

- Introduction
- Motivation
- Background
- First Study
- Current Work
- Related Works
- Future Works

System evolution

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



Problems ahead

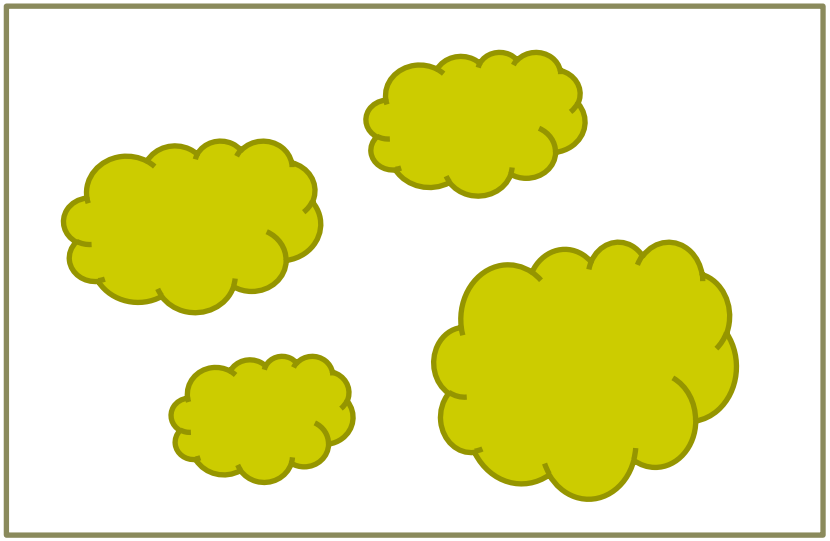
- Change the code is a challenging activity:
  - Understand the code and its complexity
  - Class dependencies and ripple effects

Code Smells

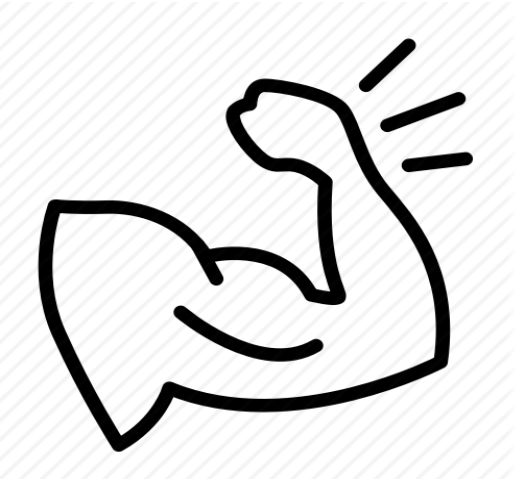
- Code smells are symptoms that the source code quality is degrading, and they should be refactored
  - Too long
  - Too complex
  - Non-cohesive
  - Difficult to understand

Code smell agglomerations

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



Our Motivation



- Lack of studies of how they impact the code and development process
- They did not consider the presence of agglomerations formed by the same code smell

UFMG

lab-soft
software engineering laboratory

Goal

Provide evidences of which agglomeration is more harmful to code quality

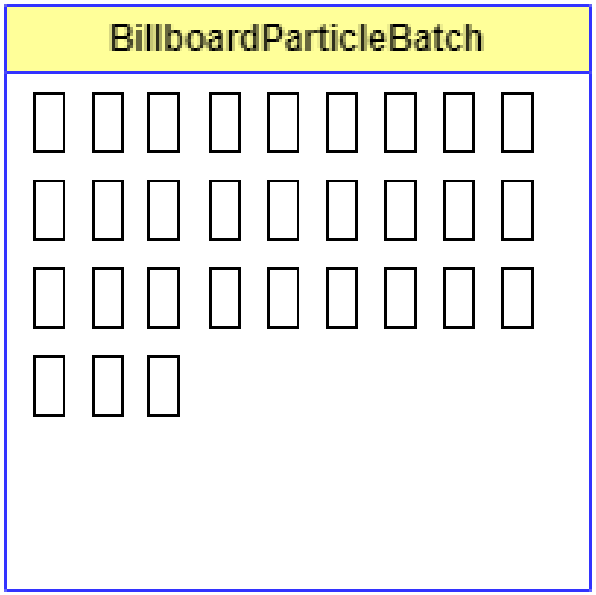
UFMG

lab-soft
software engineering laboratory

But in this work, what are the agglomerations?

Heterogeneous Agglomerations

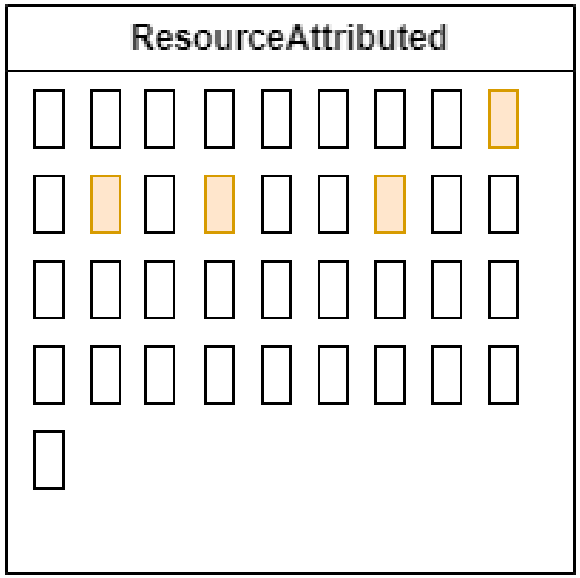
- Heterogeneous:
  - Two or more code smells of **different** types



Large Class
+
Refused Bequest

Homogeneous Agglomerations

- Homogeneous:
  - Two or more code smells of **same** type



Homogeneous Long Method

Isolated Smells

- Isolated:
  - Only one code smell

Clean Classes

- Clean
  - Classes without code smells

First Study

AN EXPLORATORY EVALUATION OF CODE SMELL AGGLOMERATIONS

Goal

- Evaluate the impact of code smell agglomeration presence on six software modularity metrics

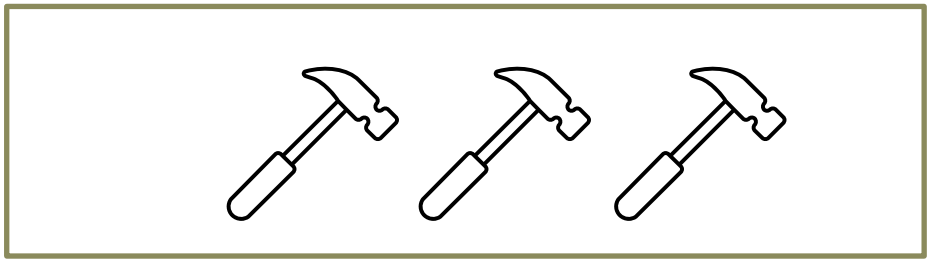
RQ: How do code smell agglomerations impact the system modularity?

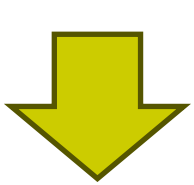
Study Design

- Evaluated dataset:
  - 20 Java systems from the Qualita Corpus (7,974 smells)
  - 10 top-stared Java systems from the GitHub (1,388 smells)
  - 4 code smells (Large Class, Refused Bequest, Long Method and Feature Envy)

Study Design

- Ground truth built using five detection tools (three tools for each smell)





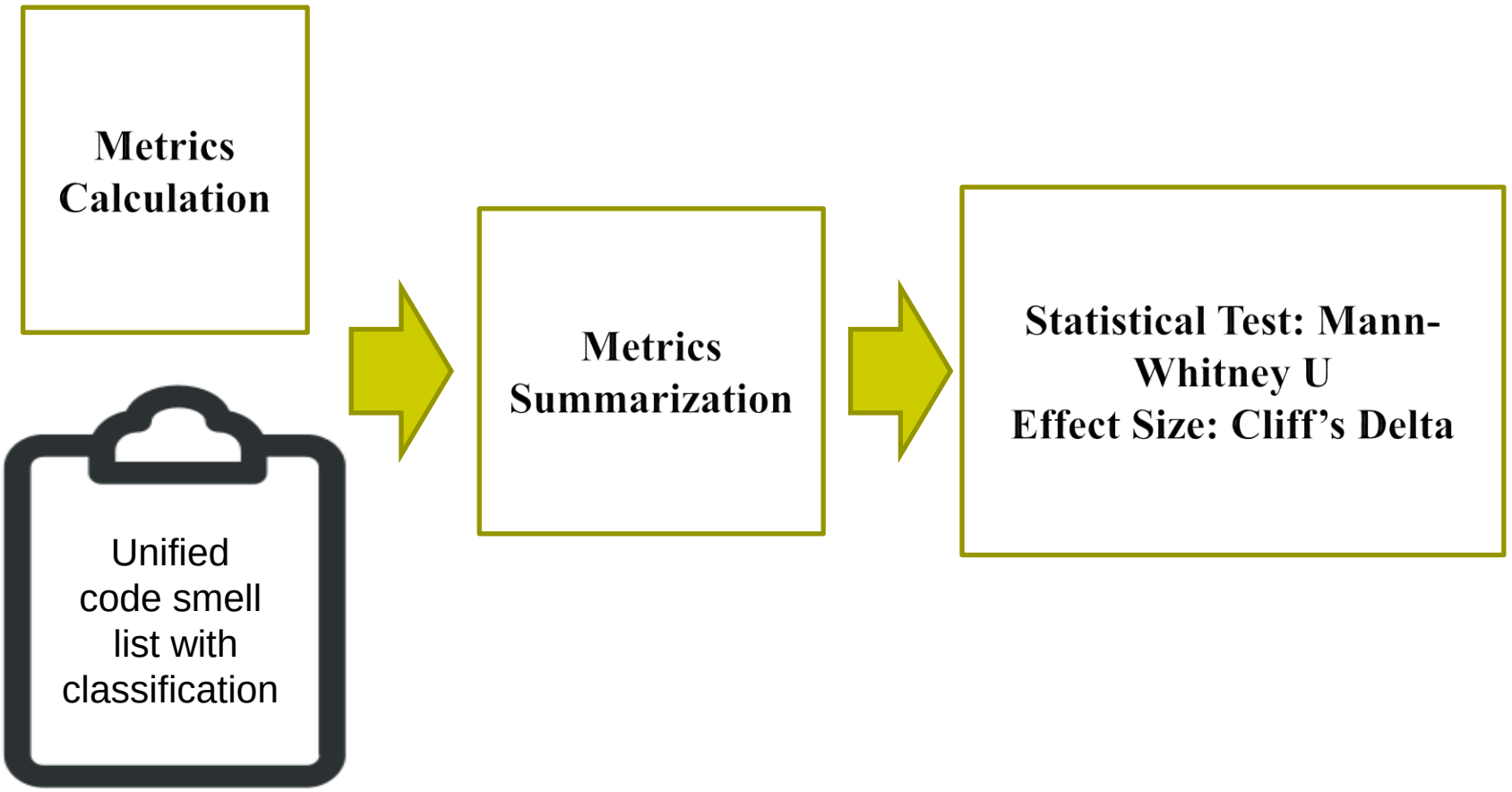
If two or more positive votes:

Code Smell Ground Truth

Data cleaning

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

Steps conducted to calculate impact



UFMG

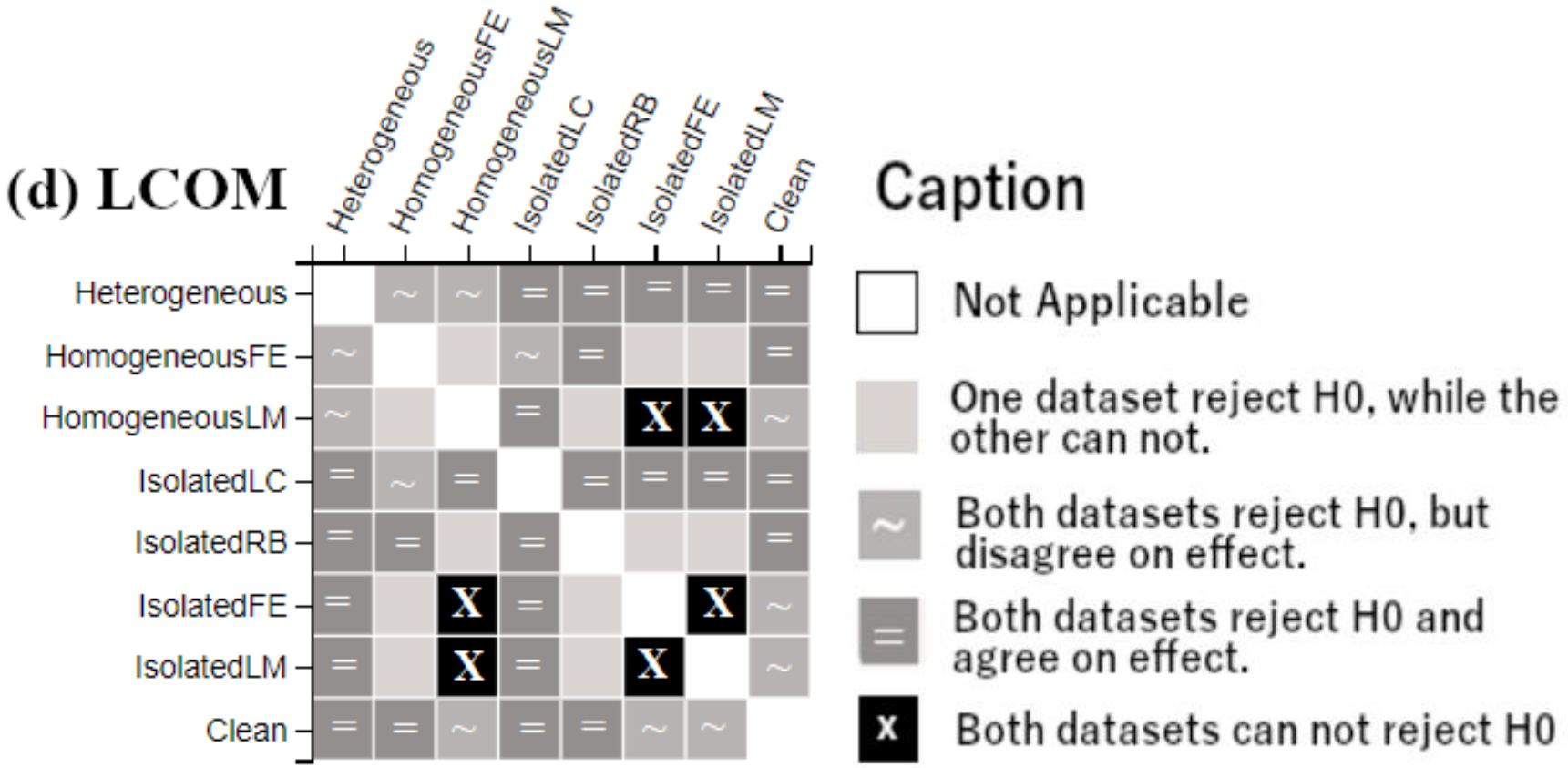
lab-soft
software engineering laboratory

Results

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)**

Agglomeration's Impact



Findings

- Heterogeneous Agglomerations, Homogeneous Agglomerations and Isolated Large Class impacts the most on the modularity metrics
- For the Heterogeneous Agglomerations, classes that contains Large Class are usually complexer and less cohesive than other agglomerations.
- Homogeneous Agglomerations are indeed frequent in the systems

Threats to validity of our work

- 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

UFMG

lab-soft
software engineering laboratory

Current Work

Goal

Explore if code smell agglomerations are more change-prone than classes with only one smell/no smell in the perspective of pre/pos major release

Dataset expansion

- Expanded our GitHub dataset to include 30 systems
  - Major release in 2021
  - ≥ 90% Java code
  - Currently maintained by the community => updated at data collection, ± 1 Years of commits before/after major release
  - 50.8K classes and 385K methods
  - ElasticSearchAnalysisIK (2KLOC) ~ Guava (2,126KLOC)

Code Smell

- Relaxed voting strategy to two tools for each smell:
  - Large Class, Data Class, Feature Envy, Intensive Coupling, Dispersed Coupling, Long Parameter List, Refused Bequest, Shotgun Surgery, Long Method

Research Questions

- RQ1: When considering the system size, does agglomerations change more frequently than other classes?
  - RQ1.1: How the rate of agglomerations that were changed before/after a major release compares with the ones from other classes?

Research Questions

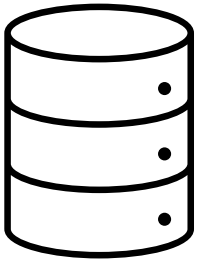
- RQ2: Does agglomerations change more frequently than other classes in absolute terms?
  - R2.1: Does agglomerations change more frequently before or after a major release in terms of commits?
  - RQ2.2: Does agglomerations change more in terms of additions, deletions, changes in line, or chunks?

Research Questions

- RQ3: When normalized to class size, does agglomeration's code suffer more modifications than other classes?
- RQ4: Are agglomerations more prone to change in the future than other classes?

Study Design – Data Collection

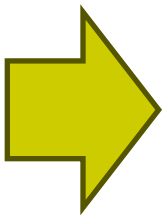
30 Java Systems



- Agglomerations Ground Truth
- Size measurements

GitHub Mining

- Commit
- Pull Request
- Diff
- Patches



- Commits Pushed/Merged
- Java classes
- Cosmetic changes

Study Design – Data Collection

- For each class that was modified and merged, we collected:
  - Is new class or a deleted class
  - # imports
  - # comments
  - # braces
  - # White space
  - # Annotations - # Override
  - # COSMETIC CHANGES
  - #Additions, #Deletions, #Changes

Class Change History Example

commit_sl	modified_	author_na	author_en	author_da	commiter_	commiter_	commiter_	had_comn	was_verifi	verified_re	#package
00faa3c4c	core/src/n	yangxb201	yangxb201	2021-06-0	GitHub	noreply@g	2021-06-0	0	True	valid	0
4d06126b7	core/src/n	gongdewe	kylixs@qq.	2020-09-2	GitHub	noreply@g	2020-09-2	0	True	valid	0
7a01f23a7	core/src/n	gongdewe	kylixs@qq.	2020-08-0	GitHub	noreply@g	2020-08-0	0	True	valid	0
be1247e3(	core/src/n	hengyunak	hengyunak	2020-06-1	GitHub	noreply@g	2020-06-1	0	True	valid	1
fe8aba65c	core/src/n	gongdewe	kylixs@qq.	2020-05-2	gongdewe	kylixs@qq.	2020-05-2	0	False	unsigned	0
907b5c8b(	core/src/n	gongdewe	kylixs@qq.	2020-05-2	gongdewe	kylixs@qq.	2020-05-2	0	False	unsigned	1

deleted_cl	#+braces	#-braces	#cosmetic	#+commer	#-commer	#+white_s	#-white_sp	#+import	#-import
0	3	0	3	0	3	1	1	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	2	0	0
1	25	0	21	1	25	0	22	0	10
0	0	0	0	0	0	0	0	0	0
1	25	0	21	1	25	0	22	0	10

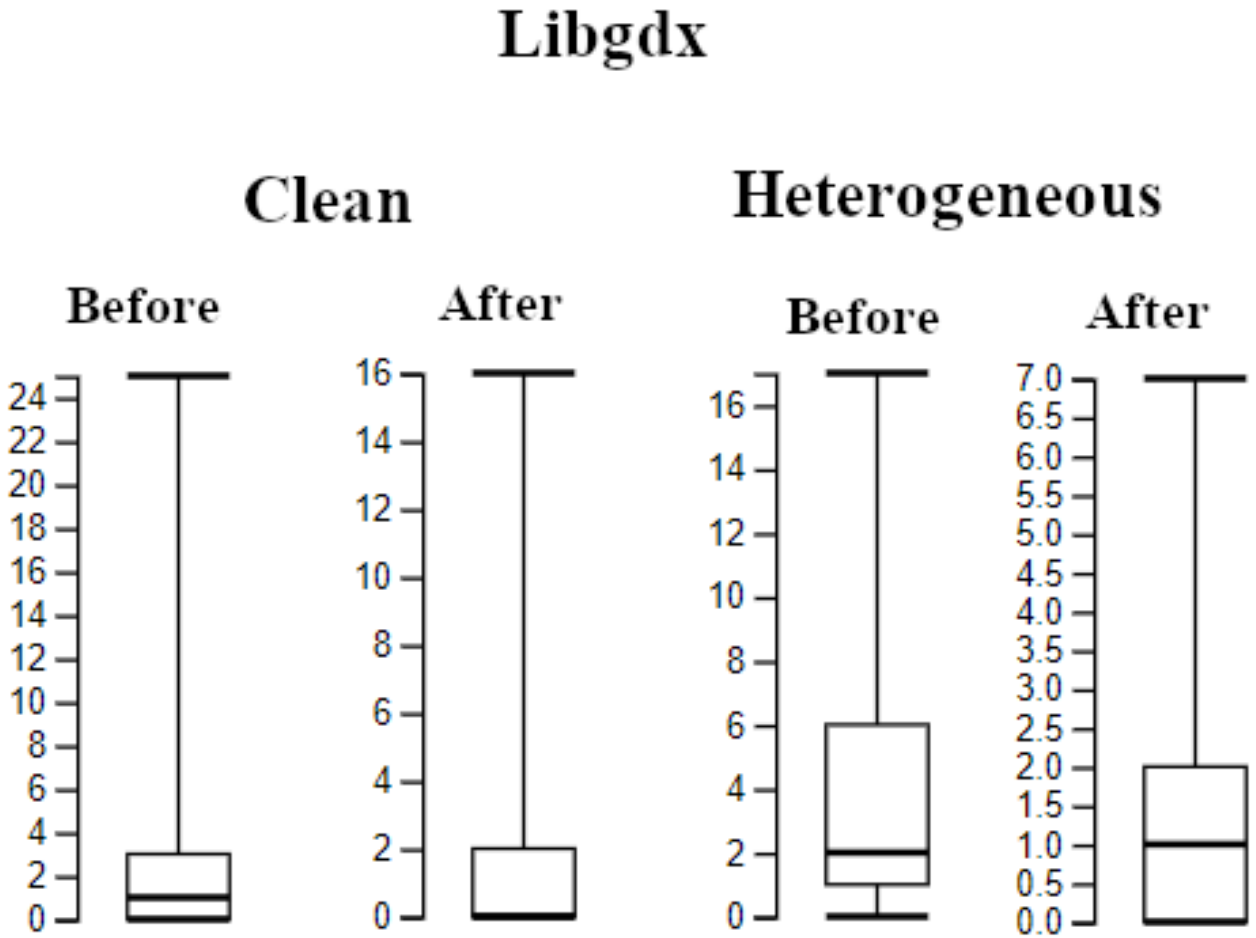
Classes modified in commits

System	#Heterogeneous	#Homogeneous	Isolated	Clean
Dbeaver	173	5	246	1412
HikariCP	2	0	2	6
Libgdx	37	7	53	516
Retrofit	0	0	0	0

Smelly Class Summary

- Dbeaver (714 smelly classes): 229 Heterogeneous, 1 Homogeneous FE, 2 Homogeneous IC, 10 Homogeneous LM, 100 Isolated
- Sa-Token (4 smelly classes): 4 Isolated
- Nanohttpd (4 smelly classes): 1 Heterogeneous, 3 Isolated

Class change by number of commit



Current Analysis

- Compare distribution of changes between agglomerations, isolated and clean classes considering:
  - System size, number of commits
  - How many times the class was changed before/after major release

Current Analysis

- Quantify and compare for each system:
  - Distribution of additions, deletions, changes for each class, and chunks
    - Full total
    - Subtracting cosmetic additions/deletions/changes
- Is there an agglomeration type that changes more frequently in general? (consider data of all systems)
- Calculate how prone agglomerations are to change in the future

UFMG

lab-soft
software engineering laboratory

Related Works

Related Works

- Lozano et al. (2015)
  - Investigated agglomerations lifecycle
- Martins et al. (2020)
  - Studied how code smell agglomeration refactoring impacts metric values of three industrial systems
- Walter et al. (2018)
  - Evaluated 14 smells on the Qualita Corpus

UFMG

lab-soft
software engineering laboratory

Future Works

What can we do with all this data?

- How many developers change these agglomerations classes? (type of user)
- Acceptance analysis
- Use RefDiff tool in order to verify if agglomerations are more refactored than other classes
- Mine commit messages/ASTs in order to identify why the developer did the change

UFMG

lab-soft
software engineering laboratory

Thank you!

Amanda Santana - amandads@dcc.ufmg.br

References

- A. Lozano, K. Mens, J. Portugal, Analyzing code evolution to uncover relations between bad smells, 2015
- B. Walter, F. A. Fontana, V. Ferme, Code smells and their collocations: A large-scale experiment on open-source systems, J. Syst. Soft. 144 (2018) 1–21.
- J. Martins, C. Bezerra, A. Uchôa, A. Garcia, Are code smell co-occurrences harmful to internal quality attributes? a mixed-method study, in: Proceedings of the 34th Brazilian Symposium on Software Engineering, SBES ’20, Association for Computing Machinery, New York, NY, USA, 2020, p.52–61.

UFMG

lab-soft
software engineering laboratory

Appendix

Code smell impact

- They can affect different aspects of the quality:
  - Understandability
  - Extensibility
  - Reusability

- They are also more change prone and fault prone

Qualita Corpus Systems

Table 2: Selected Systems from Qualita Corpus

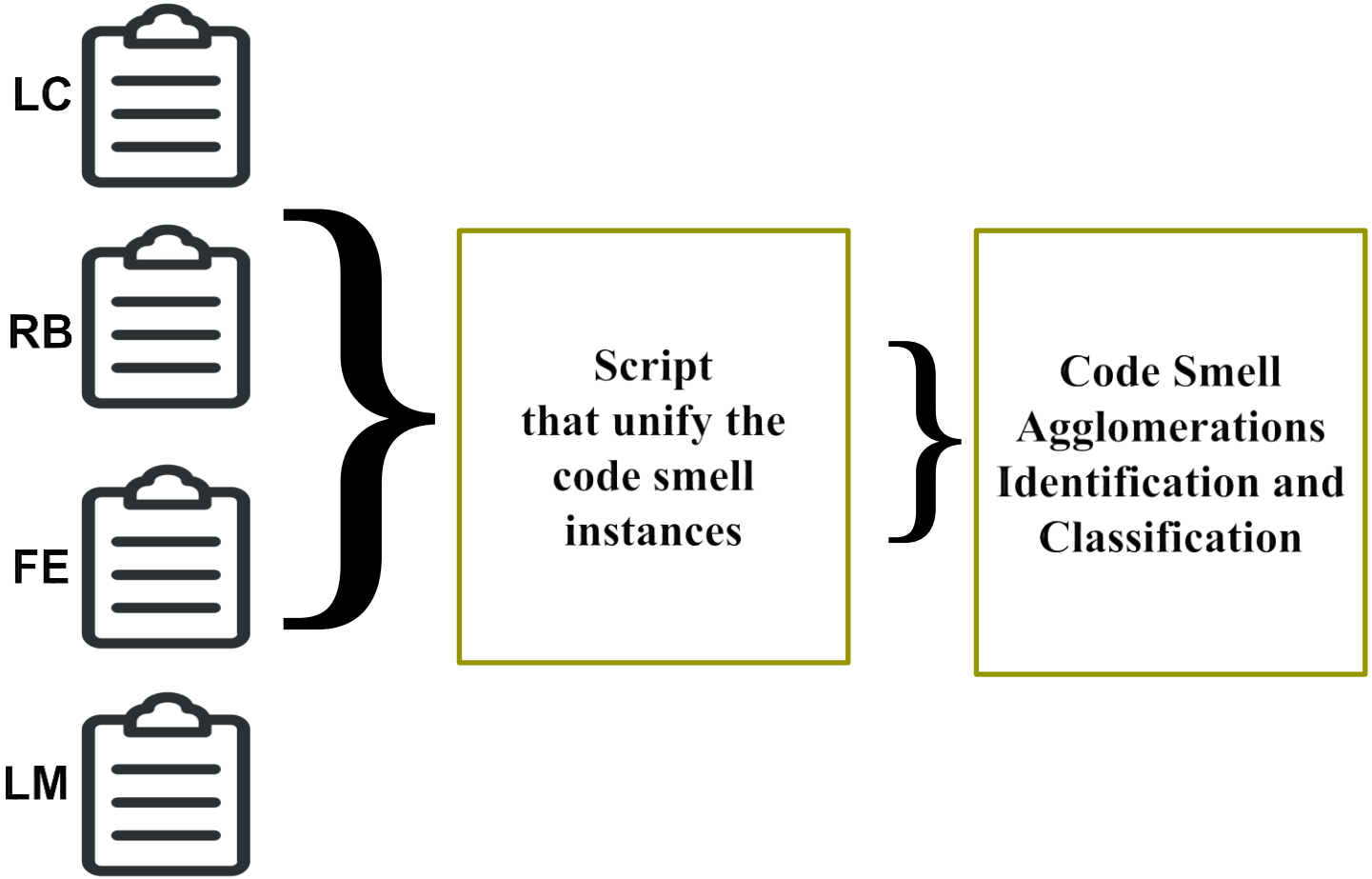
Name	TLOC	#Class	NOM	Domain	BS
checkstyle-5.6	22,877	453	2,782	IDE	231
commons-codec	7,314	96	772	tool	4
commons-io	9,309	134	1,280	tool	7
commons-lang	28,167	269	3,223	tool	19
commons-logging	2,712	28	302	tool	3
hadoop-1.1.2	211,436	2,747	17,566	middleware	173
hibernate-4.2.0	176,607	3,264	25,062	database	148
htmlunit-2.8	37,135	545	5,040	testing	273
jasperreports-3.7.4	126,793	1,779	16,527	visualization	1,109
jfreechart-1.0.13	80,619	638	8,645	tool	540
jhotdraw-7.5.1	76,983	638	7,584	graphic	470
jmeter-2.5.1	70,961	955	7,613	testing	680
lucene-4.2.0	223,880	3,261	17,550	tool	524
quartz-1.8.3	22,609	248	2,697	middleware	186
spring-3.0.5	133,713	2,733	17,838	middleware	40
squirrelsql-3.1.2	6,011	169	689	database	1
struts-2.2.1	98,396	1,580	10,691	middleware	608
tapestry-5.1.0.5	37,565	1,395	5,739	middleware	307
tomcat-7.0.2	162,621	1,700	15,634	middleware	952
weka-3.6.9	254,947	2,095	18,965	tool	1,699

GitHub Systems

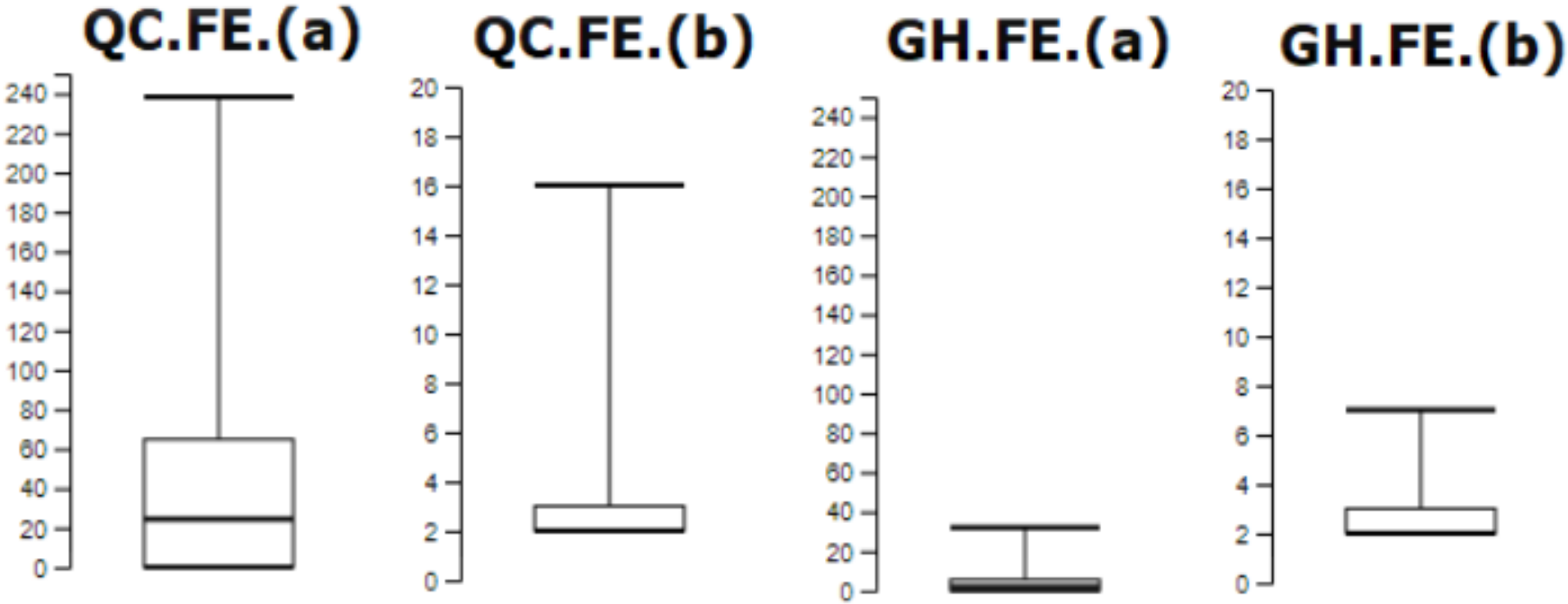
Table 3: Selected Systems from GitHub

Name	TLOC	#Class	NOM	Domain	BS
DBeaver-21.0.2	348,609	6,449	36,575	database	705
FastJSON-1.2.76	43,536	249	1,996	tool	66
GSON	11,641	228	920	tool	26
jsoup1-1.14.2	17,290	242	1,540	tool	19
JUnit-4.13.2	10,769	310	1,541	testing	11
libgdx-1.9.14	208,028	2,714	39,338	game development	512
NanoHTTPD-2.3.1	3,566	67	380	middleware	6
Netty-1.7.18	5,217	138	712	middleware	10
RetroFit-1.6.0	4,706	115	403	middleware	8
WebMagic-0.7.3	5,857	163	819	tool	25

Step C - Identification of agglomerations



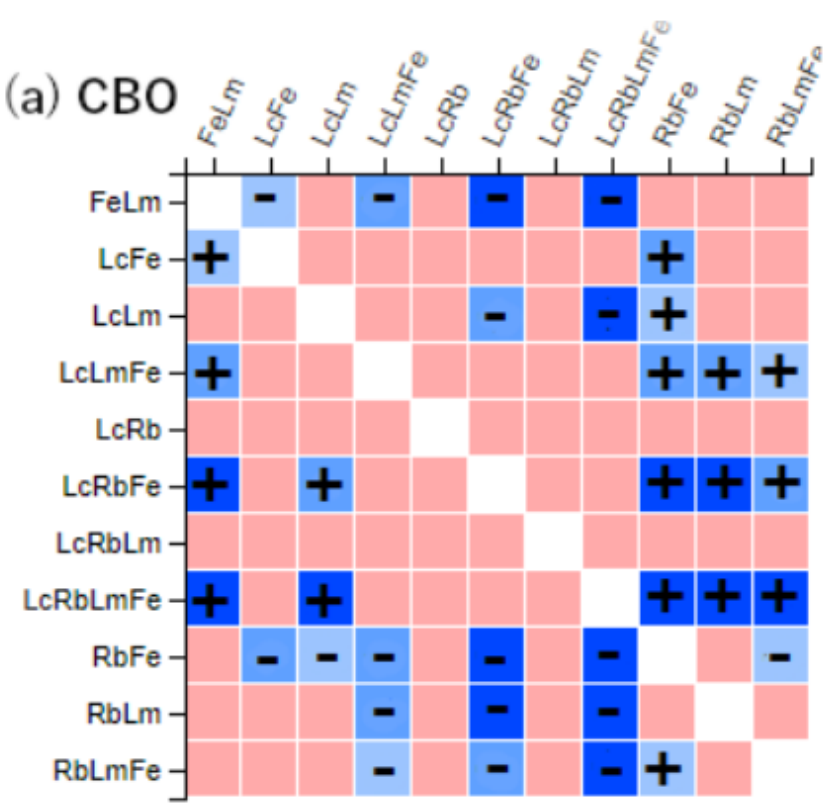
FE Homogeneous Agglomerations



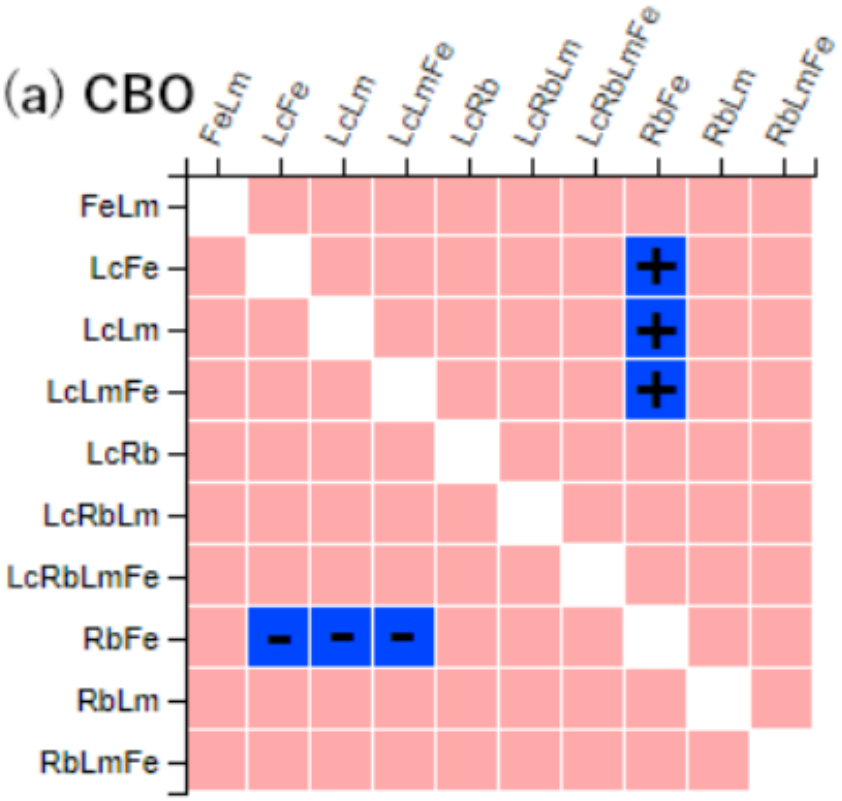
Research Quest Answers

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*

Heterogeneous's Impact



Qualita Corpus



GitHub

Future Works

- Expand the dataset by adding new systems, new metrics and evaluating new bad smells
- Mine the historical data from each system in order to verify how the agglomerations behave along the lifecycle of the system
- Investigate the existence and impact of agglomerations that are at method level.

Current steps for the future

- Expanding the current dataset to include the following smells: **Data Class, Intensive Coupling, Tradition Breaker, Long Parameter List, Type Checking, Message Chains***
- Method-level Agglomerations Analysis -> collect more metrics (Understand tool)
- Expand the GitHub dataset to include new systems (20~30 systems)

Next steps for the future

- Mine system's information (6 months after and 6 months before the current version in the dataset)
  - Change Proneness Analysis
  - Agglomeration Evolution (?)
  - Fault Proneness Analysis
  - Qualitative analysis of commits/pull-requests
- Add other metrics of quality, mainly of cohesion

Next steps for the future

- Agglomeration's density in packages
- Experiment with students about code comprehension and maintenance of modules with agglomerations (?)

- Other suggestions of analysis or data to collect?