Methods and Tools for the Analysis of Legacy Software Systems
Report 1. Logical dependencies extraction - impact factors.

Stana Adelina Diana

Computer Science and Engineering Department
"Politehnica" University of Timisoara

May, 2021

Presentation of the research topic

The goal of the thesis is to develop methods for analyzing legacy software systems by using historical information extracted from the versioning systems. We divided our work into two main parts

- **historical information collection and filtering**
- usage of the collected information in order to analyze the software systems

Structural dependencies

Definition

Structural dependencies are the result of *source code analysis* and can be extracted from: members, call parameters, local variables.

```
class ClassA{
    ...
    public void doSomething(ClassB b){
        b.doWork();
    }
}
```

```
class ClassB{
    ...
    public void doWork(){
        ....
    }
}
```

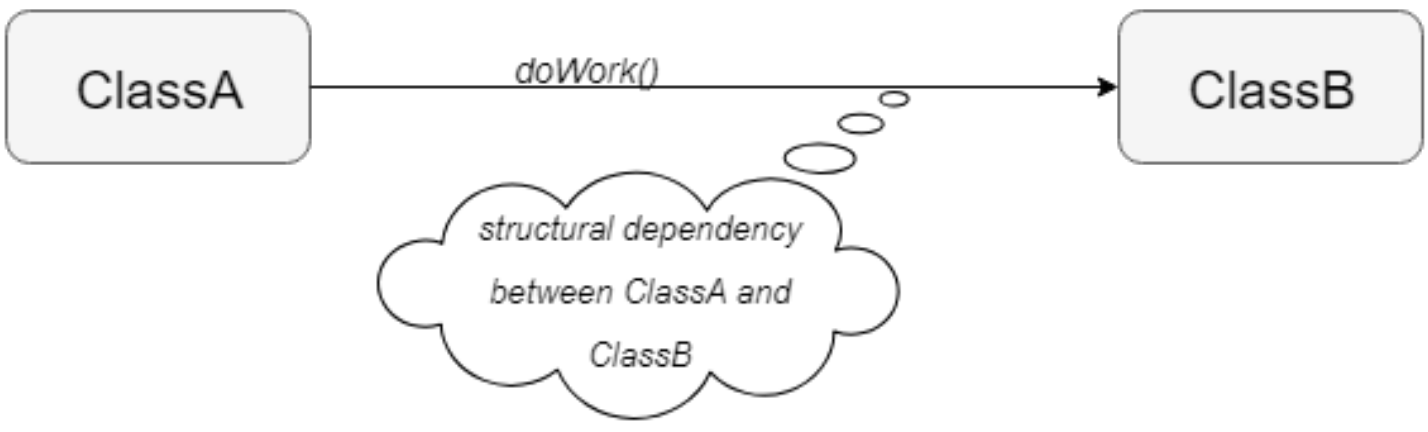


Figure 1: Example of structural dependency between two classes

Logical dependencies

Definition

Logical dependencies are the result of *software history analysis* and can reveal relationships that are not present in the source code code (structural dependencies).

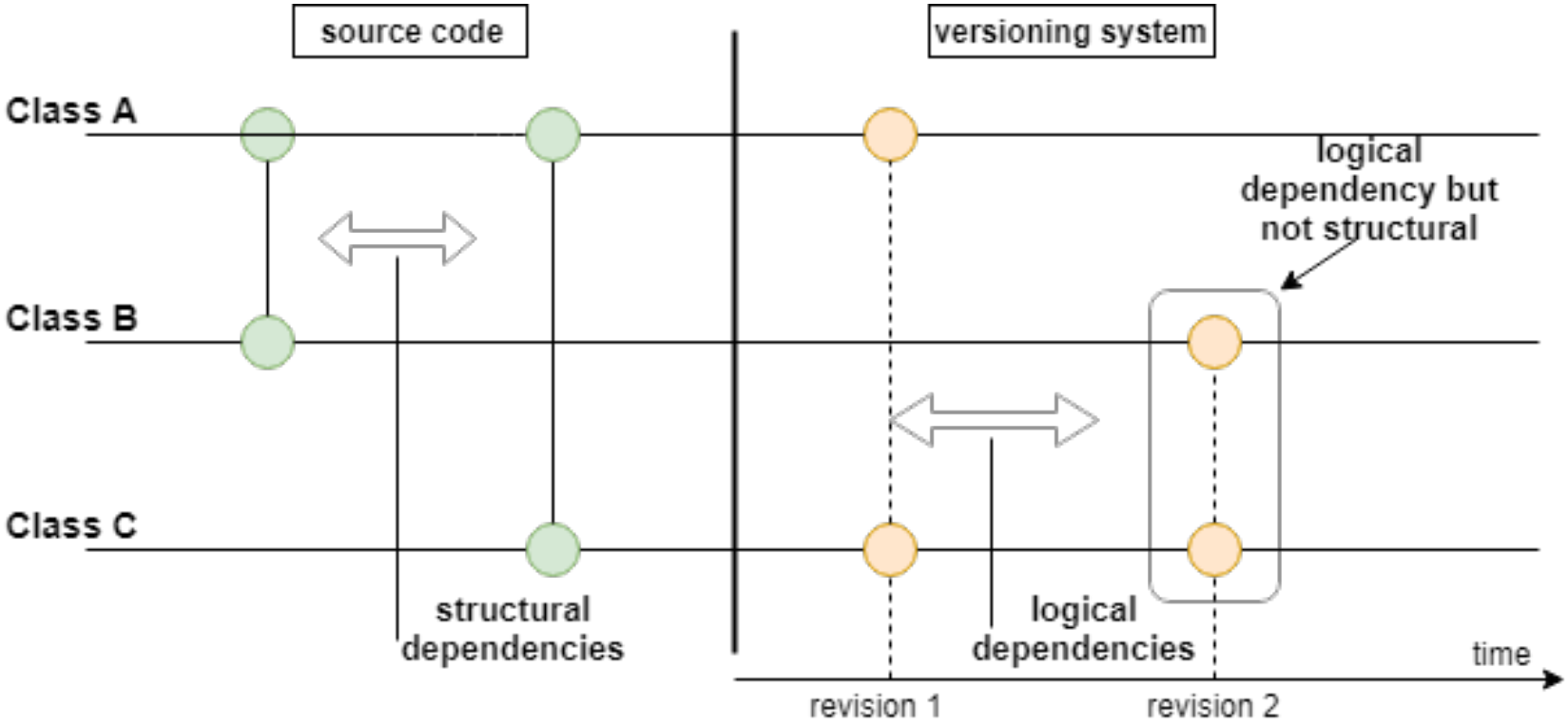


Figure 2: Example of logical and structural dependencies

From co-changing pairs to logical dependencies

The information extracted from the versioning system is under the form of pairs of classes that record co-changes (co-changing pairs). A co-changing pair that passes the filtering steps is considered a logical dependency.

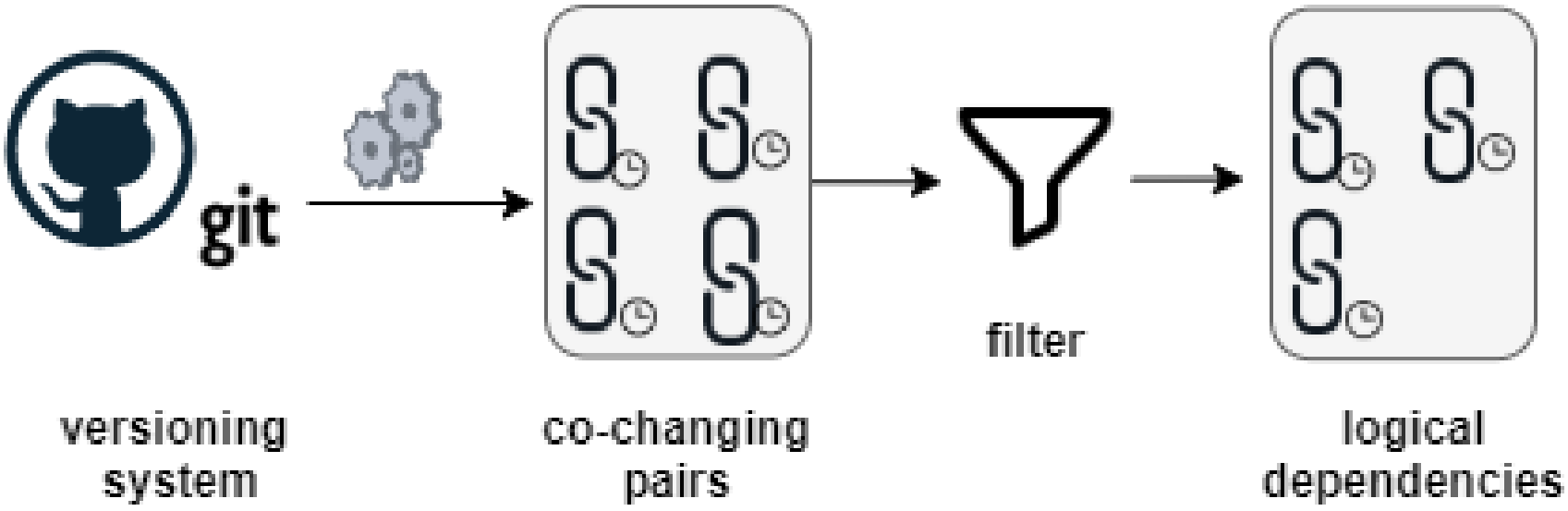


Figure 3: Co-changing pairs extraction and filtering.

Tool for measuring software dependencies

We built a tool that identifies logical and structural dependencies from software systems.

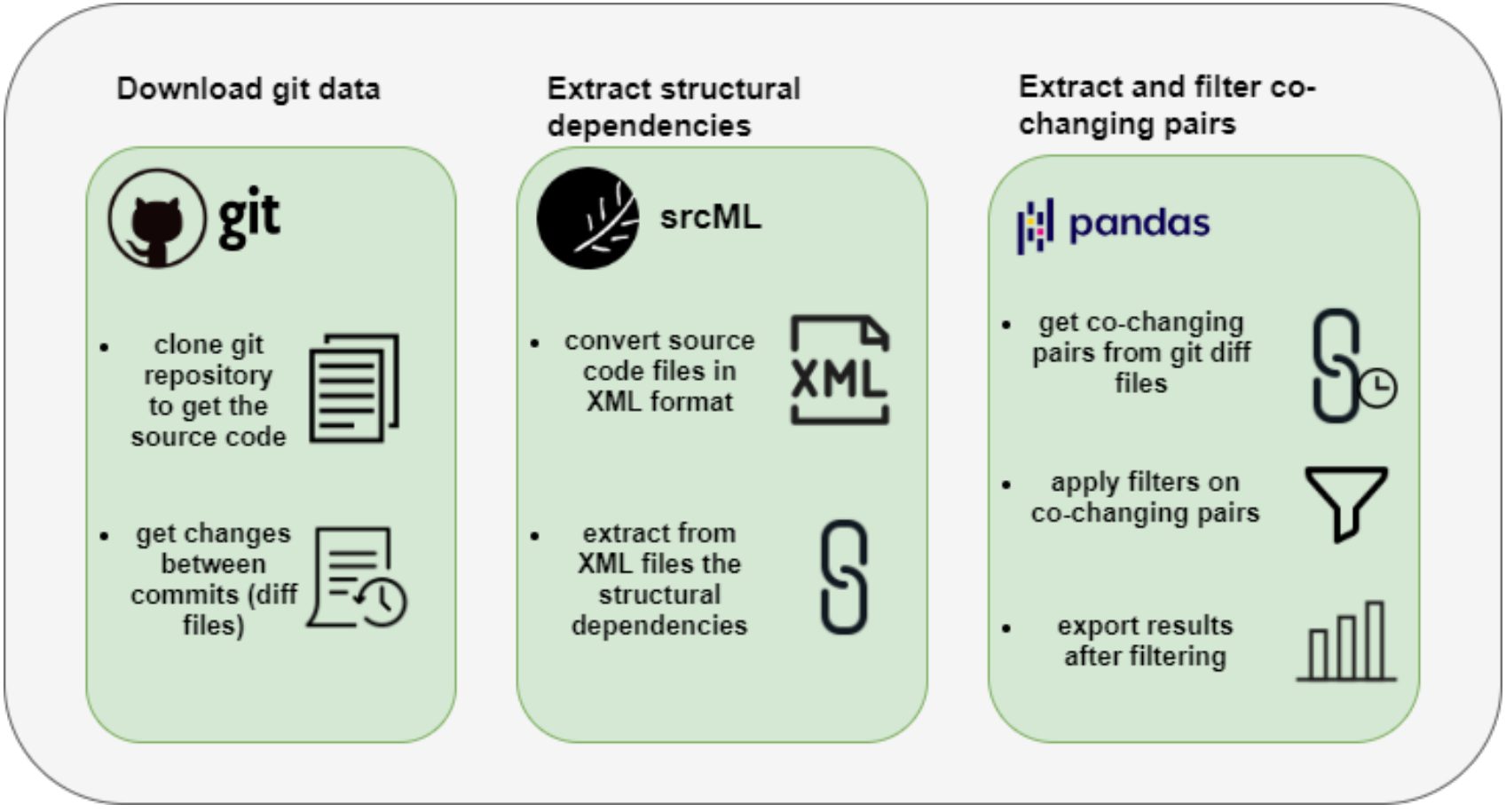


Figure 4: Tool functionalities.

Extracting co-changing pairs

Git clone and diff commands were used to acquire the needed information for co-changing pairs extraction.

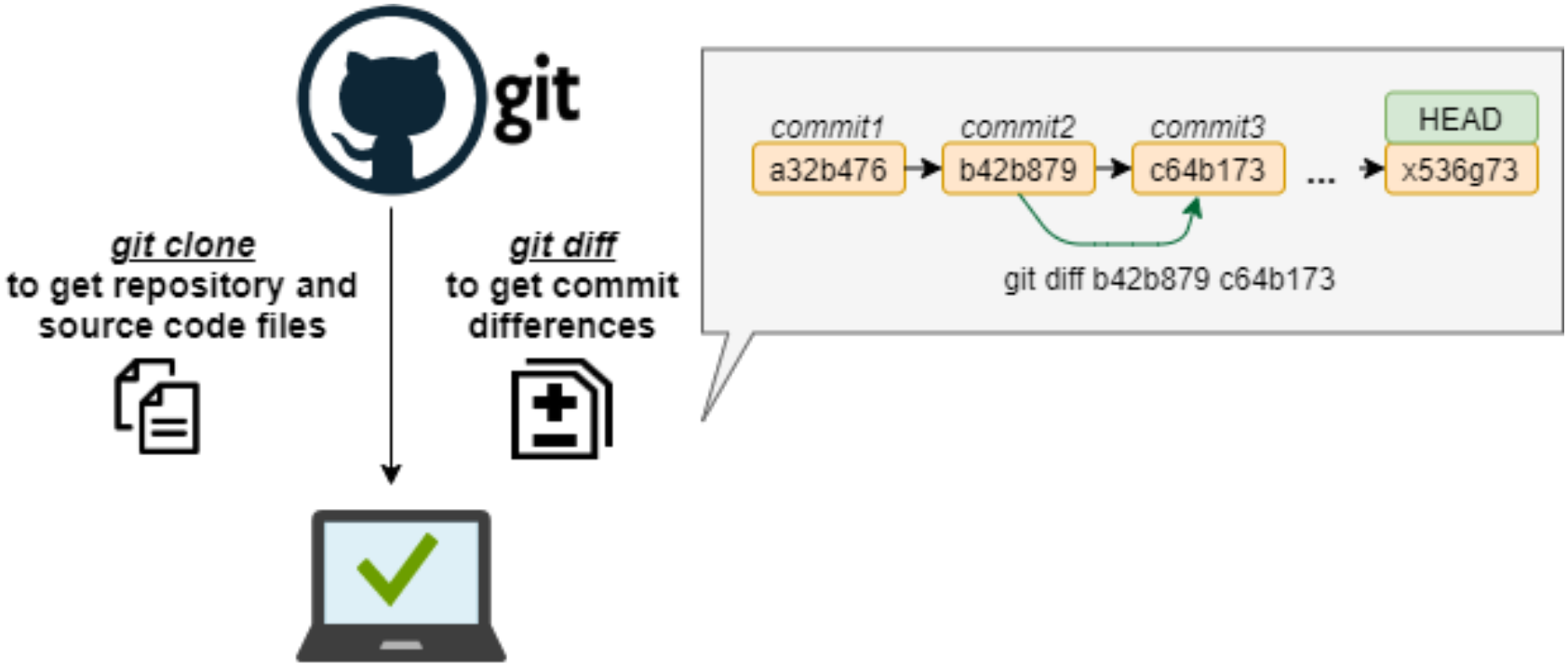


Figure 5: Commands used to download the required data.

Extracting co-changing pairs (cont.)

The diff file contains all the changes between two commits.

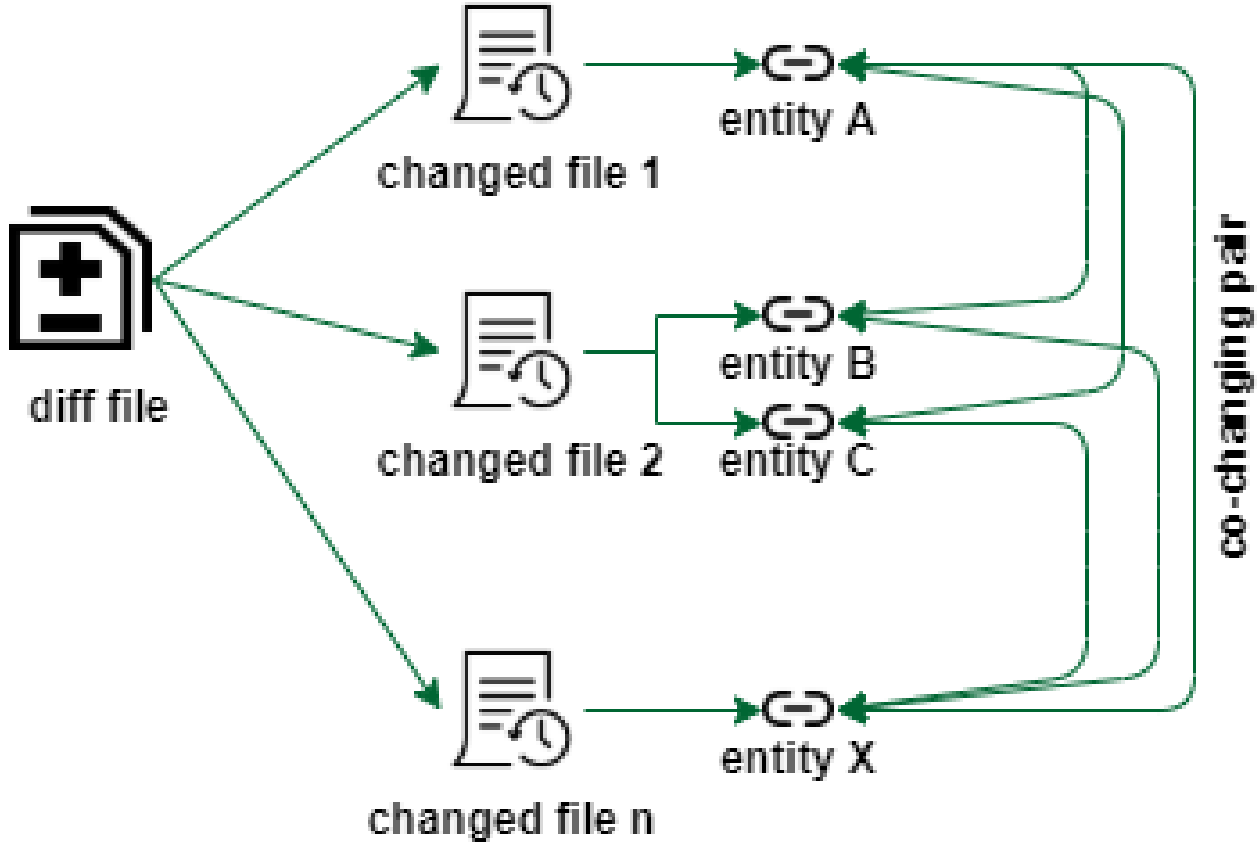


Figure 6: Co-changing pairs extraction from the versioning system.

Co-changing pairs filtering

The filters used are the commit size filter, occurrence filter, and connection strength filter.

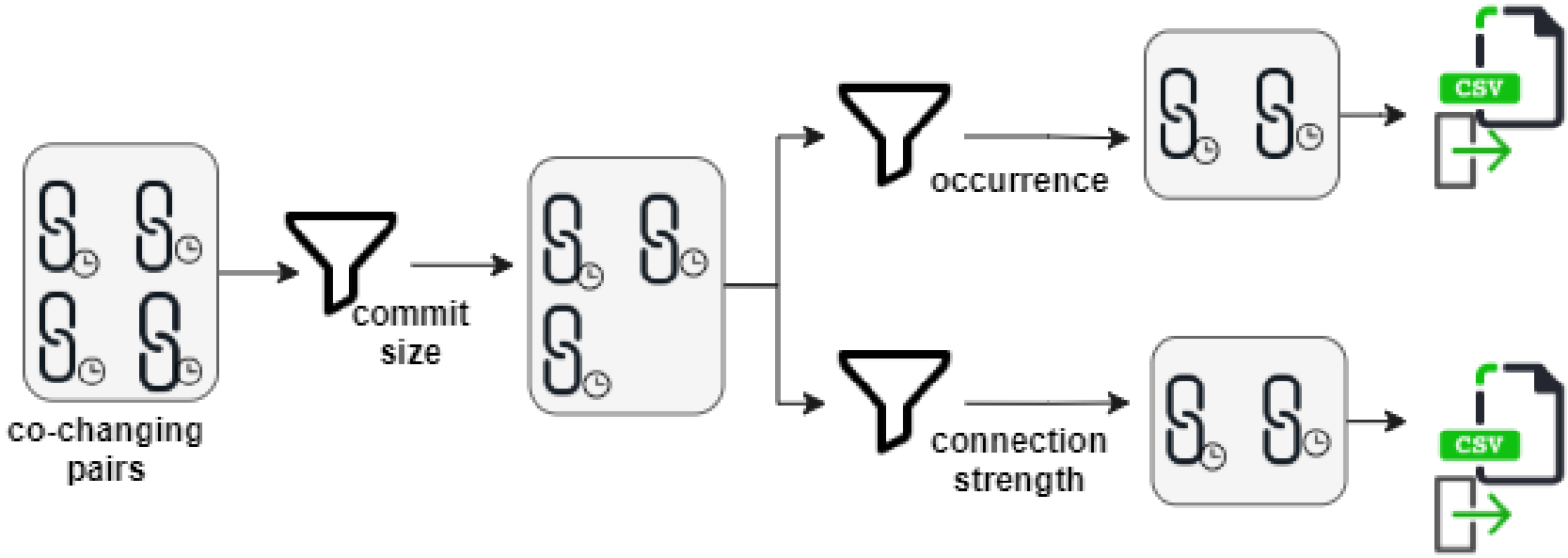


Figure 7: Co-changing pairs filtering.

To see how the filters impact the results obtained, we studied 27 open-source projects and reported the results.

Open source projects studied

ID	Project	Nr. of entites	Nr. of commits	Type
1	bluecove	2685	894	java
2	aima-java	5232	1006	java
3	powermock	2801	949	java
4	restfb	3350	1391	java
5	rxjava	21097	4398	java
6	metro-jax-ws	6482	2927	java
7	mockito	5189	3330	java
8	grizzly	10687	3113	java
9	shipkit	639	1563	java
10	OpenClinica	9655	3276	java
11	robolectric	8922	5912	java
12	aeron	4159	5977	java
13	antlr4	4747	4431	java
14	mcidasv	3272	4136	java
15	ShareX	4289	5485	csharp
16	aspnetboilerplate	9712	4323	csharp
17	orleans	16963	3995	csharp
18	cli	2063	4488	csharp
19	cake	12260	2518	csharp
20	Avalonia	16732	5264	csharp
21	EntityFrameworkCore	50179	5210	csharp
22	jellyfin	8764	5433	csharp
23	PowerShell	2405	3250	csharp
24	WeiXinMPSDK	7075	5729	csharp
25	ArchiSteamFarm	702	2497	csharp
26	VisualStudio	4869	5039	csharp
27	CppSharp	17060	4522	csharp

Filtering based on the size of commit transactions

size of commit transaction = number of files changed between two commits

The commits were divided into 4 categories:

- commits with maximum 5 files changed
- commits with maximum 10 files changed
- commits with maximum 20 files changed
- commits with more than 20 files changed

Filtering based on the size of commit transactions

We analyzed the overall transaction size trend for all the open-source systems with a total of 74 332 commits.

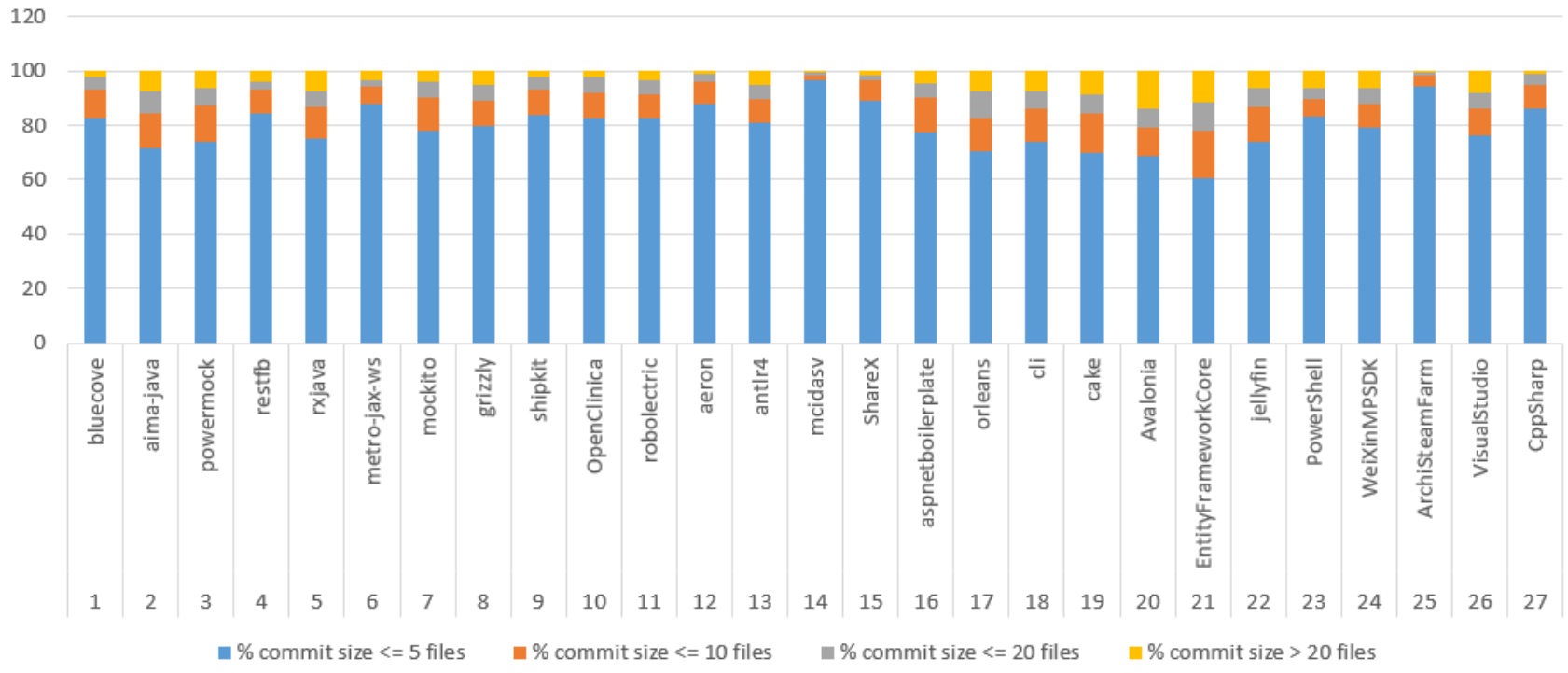


Figure 8: Commit transaction size(cs) trend in percentages.

Filtering based on the size of commit transactions (cont.)

We also analyzed the number of co-changing pairs extracted from each commit transaction size(cs) group

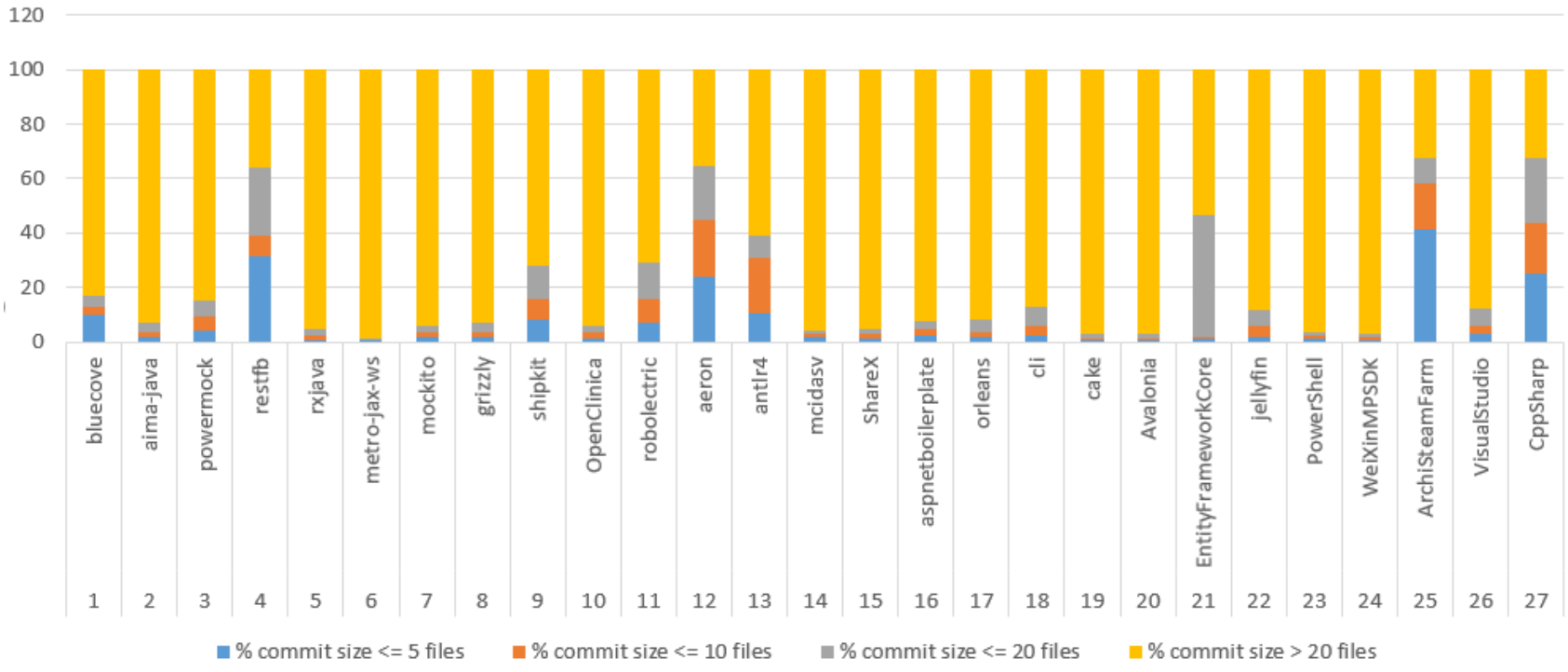


Figure 9: Percentages of pairs extracted from each commit transaction size(cs) group

Filtering based on the size of commit transactions (cont.)

One single commit transaction can lead to a large amount of co-changing pairs.

For example in RxJava we have commit transactions with 1030 source code files, this means that those commits can generate ${}^{n}C_{k} = \frac{n!}{k!(n-k)!} = \frac{1030!}{2!(1028)!} = 529935$ logical dependencies.

By setting a threshold on the commit transaction size we can avoid the introduction of those co-changing pairs into the system.

Filtering based on number of occurrences

One occurrence of a co-change pair can be a coincidence. To avoid this kind of coincidences, we set a minimum number of repeated occurrences for a co-change to be counted as logical dependency. The values for this threshold are 1, 2, 3 and 4.

Filtering based on number of occurrences (cont.)

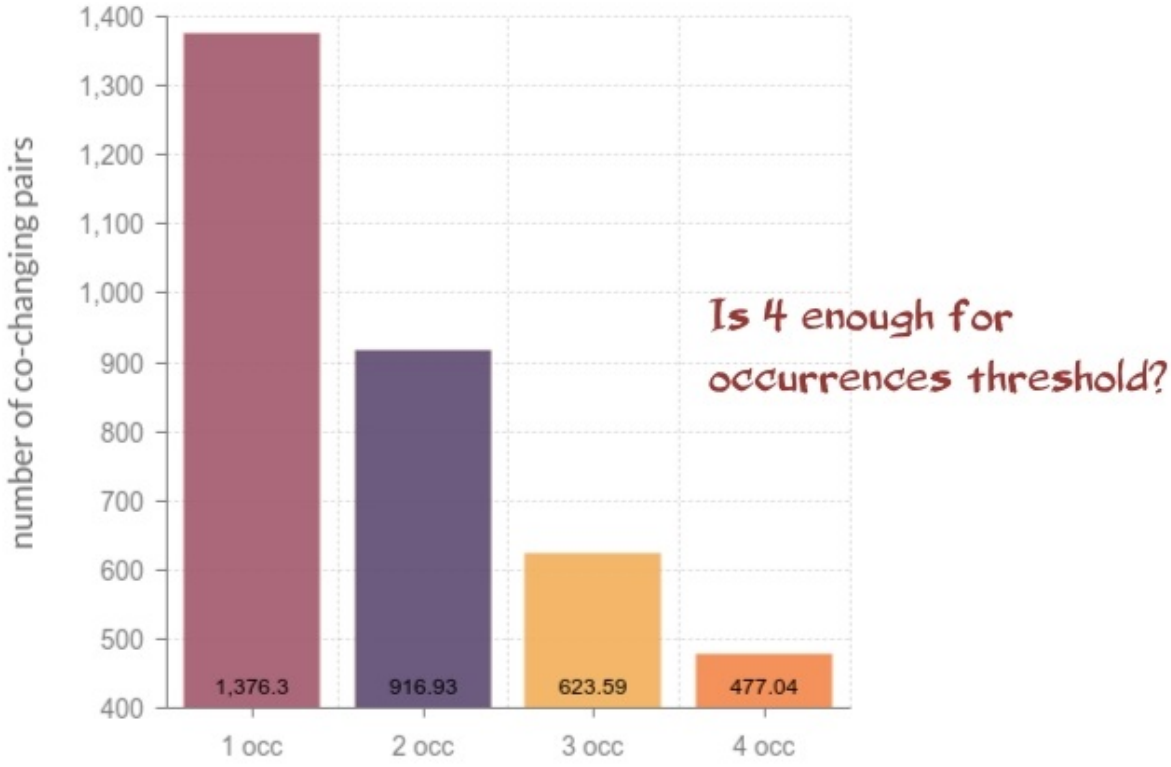


Figure 10: The average number of co-changing pairs after occurrence filtering.

If we try to go higher with the occurrences threshold, we will risk filtering all the existing co-changing pairs for some systems.

Filtering based on connection strength

This filter focuses on the connection strength of a co-changing pair (how strongly connected are the entities that form a co-changing pair). Assuming that we have a co-changing pair formed by entities A and B:

$$connection\ factor\ for\ A = \frac{100 * commits\ involving\ A\ and\ B}{total\ nr\ of\ commits\ involving\ A} \quad (1)$$

$$connection\ factor\ for\ B = \frac{100 * commits\ involving\ A\ and\ B}{total\ nr\ of\ commits\ involving\ B} \quad (2)$$

Filtering based on connection strength (cont.)

The co-changing pairs are filtered out based on two scenarios:

- factor A and factor B $\geq$ *threshold*%
- factor A or factor B $\geq$ *threshold*%

We begin with a threshold value of 10 and increment it by 10 until we reach 100.

Filtering based on connection strength (cont.)

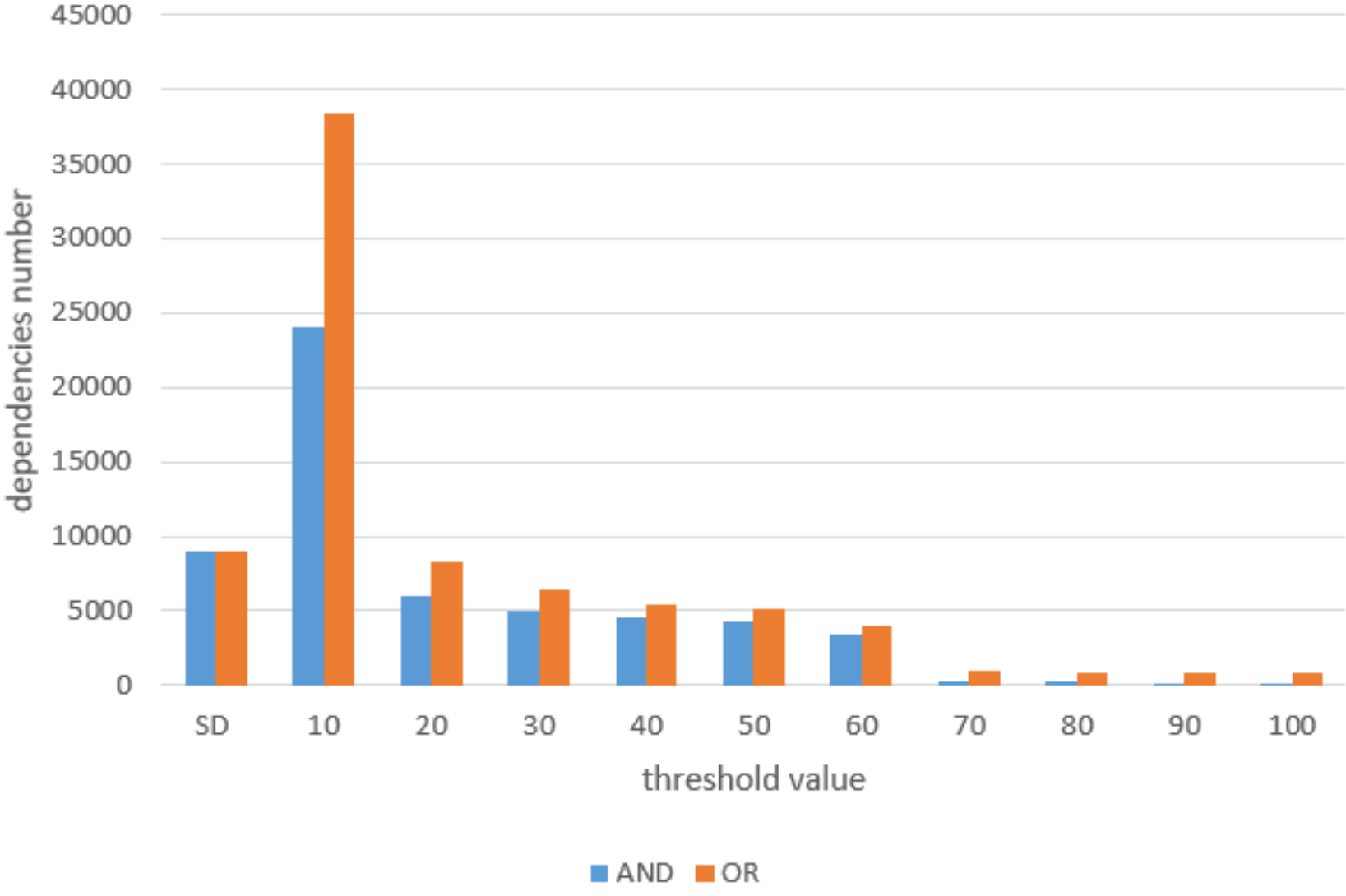
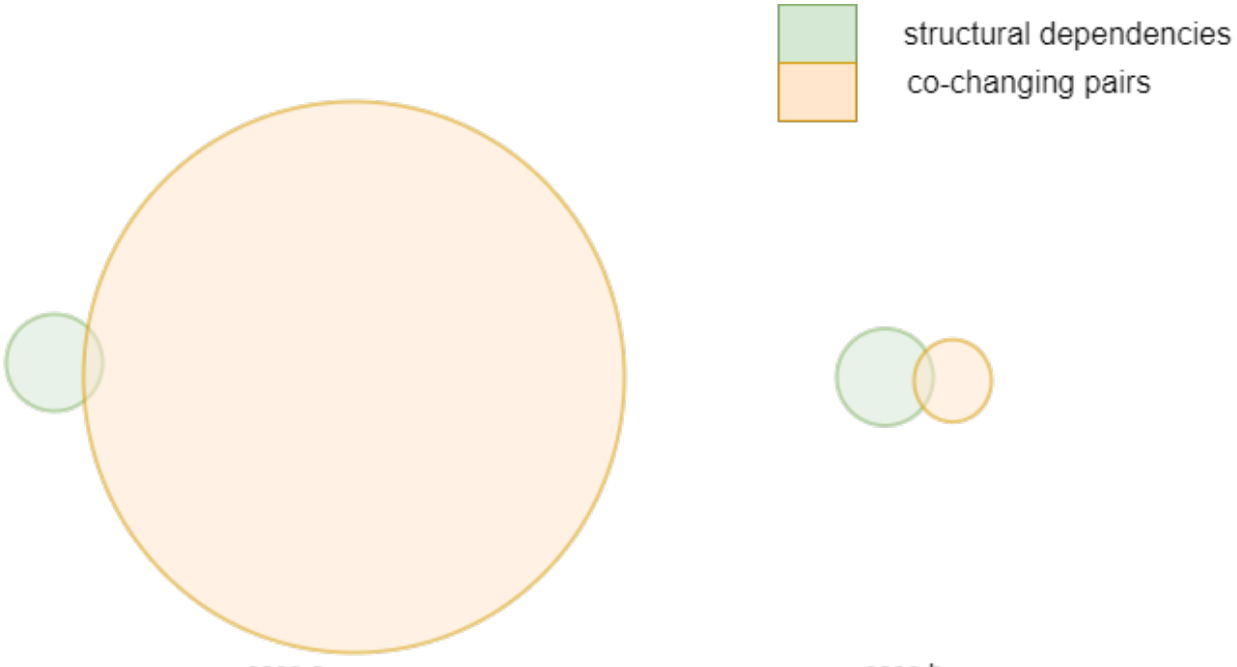


Figure 11: The average number of co-changing pairs after connection strength filtering.

Overlaps between structural and co-changing pairs



case a.
cs <= inf
occ >=1
Avg co-changing pairs/SD = 80.31
Percentage of co-changes that are also SD = 1.79

case b.
cs <= 5
occ >=4
Avg co-changing pairs/SD = 0,90
Percentage of co-changes that are also SD = 18,96

case c.
cs <= 10
connection strength = 50 % (AND)
Avg co-changing pairs/SD = 0,53
Percentage of co-changes that are also SD = 28.57

Conclusions

In order to identify logical dependencies from co-changing pairs we apply 3 sorts of filters: the commit size filter, occurrence filter, and connection strength filter.

- the commit size filter is used to reduce the quantity of co-changing pairs extracted
- the occurrence and connection strength filter are used to increase the confidence that the co-changing pairs are logical connected
  - the occurrence filter is not suited for systems of medium and small size
  - the connection strength filter is better suited for all systems sizes and we remain with a decent quantity of co-changing pairs after filtering

Future work

For future investigations regarding the usage of the dependencies extracted we will use the commit size filter with threshold 10 combined with the connection strength filter.

Articles published

Based on the research presented we published two articles:

- Adelina Diana Stana. and Ioana Sora. *Identifying logical dependencies from co-changing classes.* In Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: ENASE,, pages 486–493. INSTICC, SciTePress, 2019.
- Stana Adelina and Sora Ioana. *Analyzing information from versioning systems to detect logical dependencies in software systems.* In International Symposium on Applied Computational Intelligence and Informatics (SACI), May 2019.