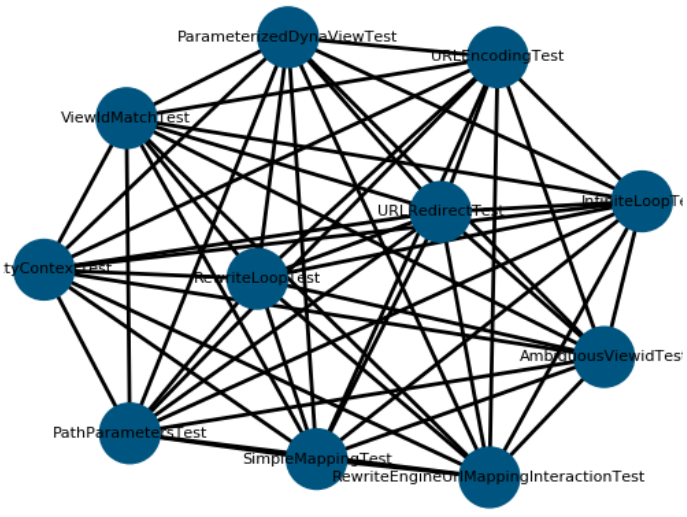
An analysis of the relationship between structural and logical dependencies in software systems

Stana Adelina Diana

Computer Science and Engineering Department
"Politehnica" University of Timisoara

June, 2018

Dependencies



A `dependency` is a relationship that shows that an element, or set of elements, requires other elements for their specification or implementation. [UML Specification]

Figure 1: Dependencies in a project

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(){
        ....
    }
}
```

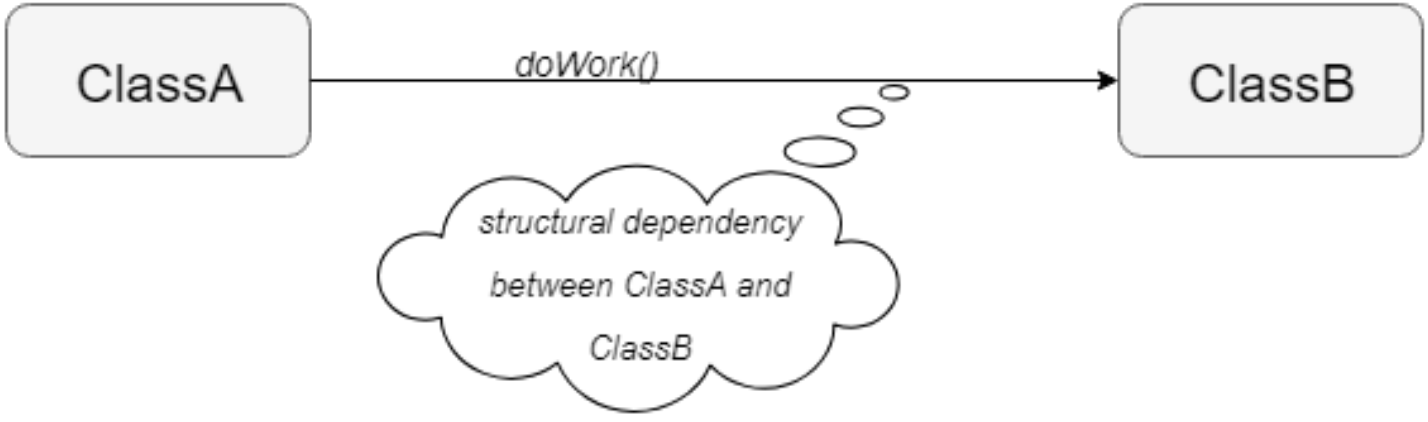


Figure 2: Example of structural dependency between two classes

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).*

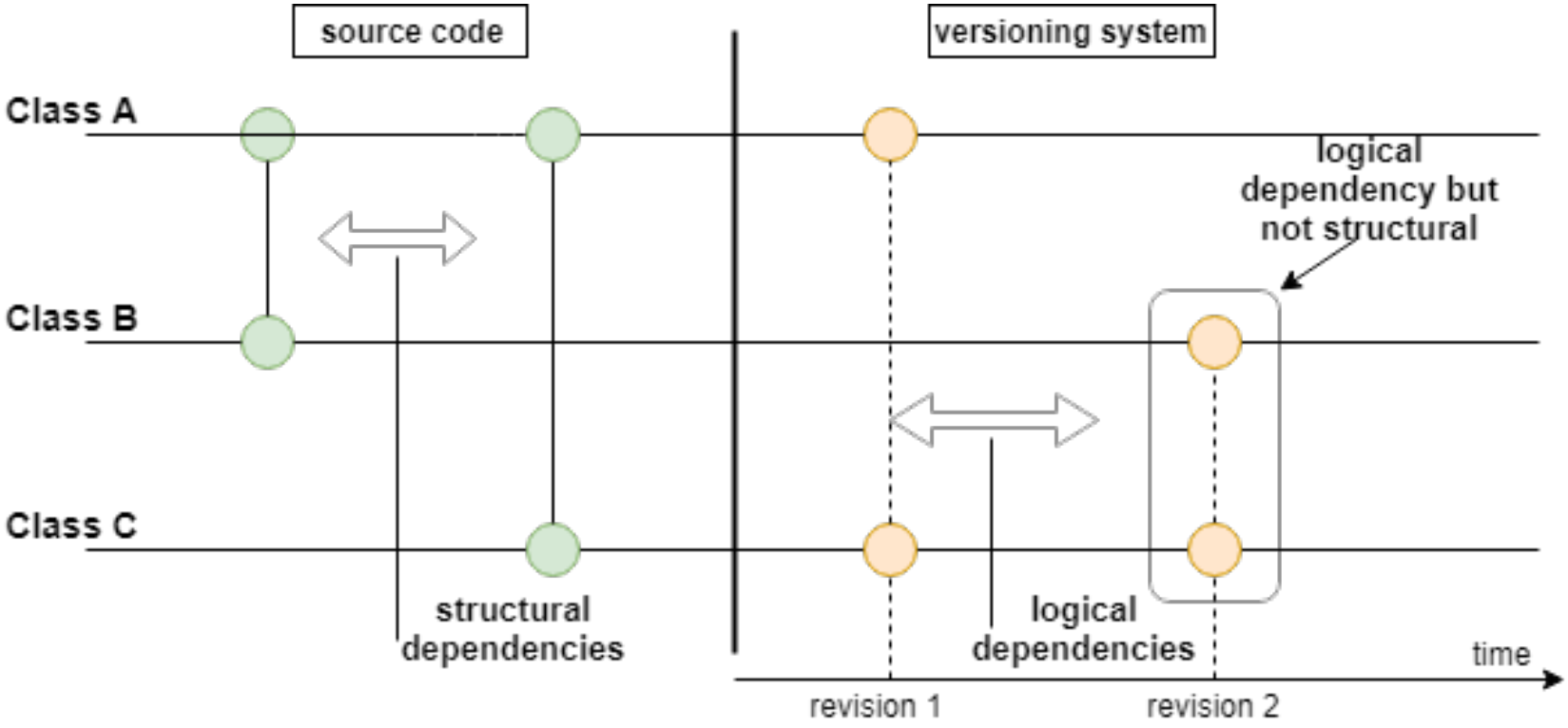


Figure 3: Example of logical and structural dependencies

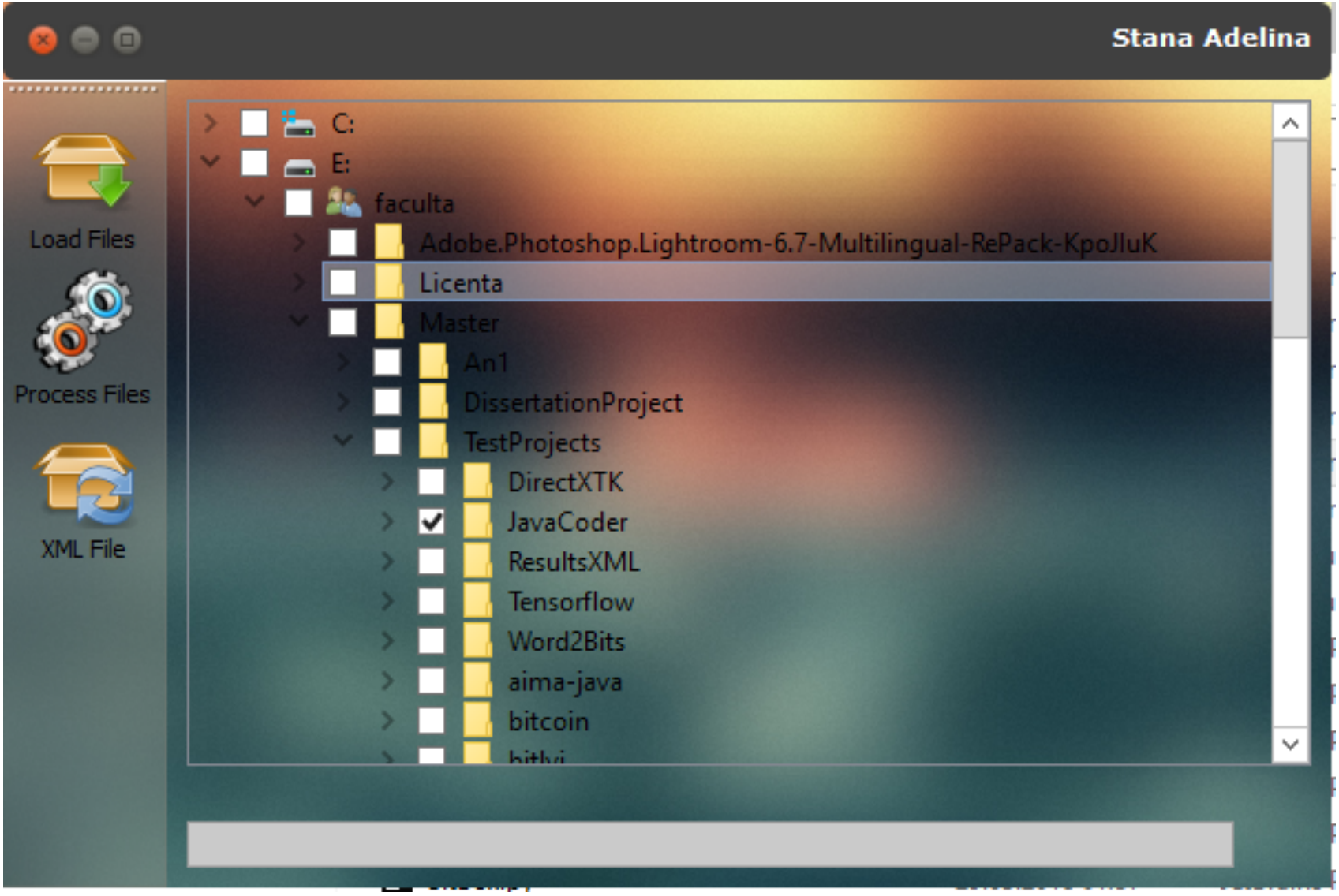
Logical dependencies

Research questions

We build logical dependencies based on three questions :
Question 1: *How the number of files changed in a commit can influence the logical dependencies of the system?*
Question 2: *Considering comment changes as valid changes can lead to additional logical dependencies ?*
Question 3: *One occurrence of a logical dependency is enough to consider it as valid ?*

Tool for measuring software dependencies

In order to answer these research questions, we have built a tool that extracts structural and logical dependencies.



Tool for measuring software dependencies

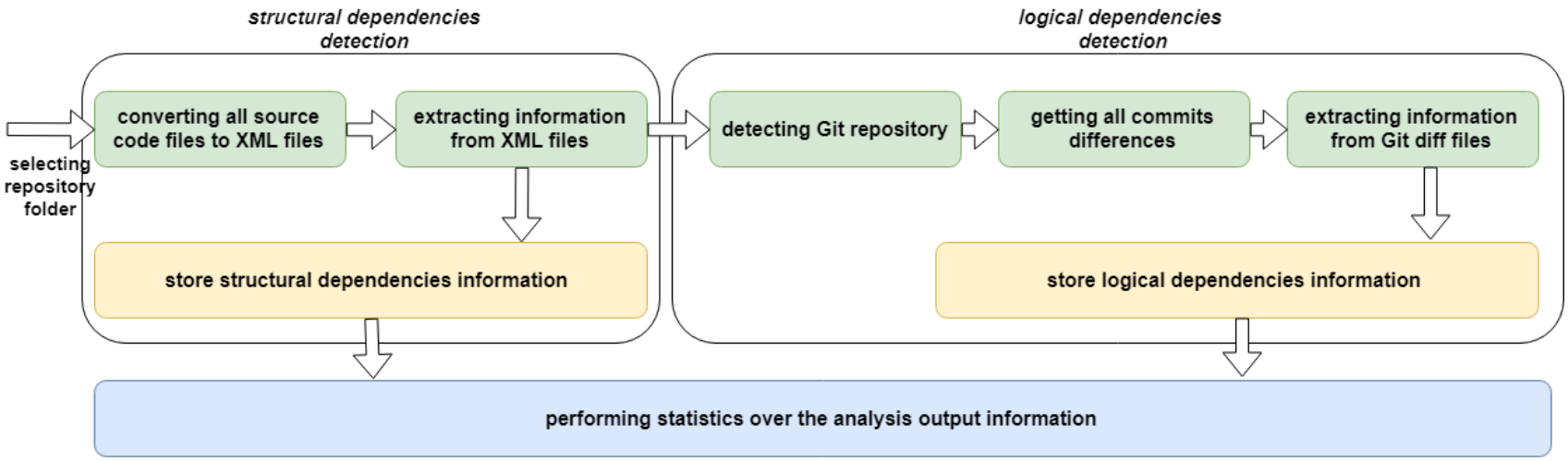


Figure 5: Workflow diagram of the tool

The workflow can be delimited by three major steps as it follows:

- Extracting structural dependencies.
- Extracting logical dependencies.
- Processing the information extracted.

Open source projects studied

ID	Project	Nr. of classes	Nr. of commits	
1	urSQL	41	89	java
2	JavaCoder	5	11	java
3	jbandwidthlog	14	54	java
4	sjava-logging	18	62	java
5	daedalum	66	29	java
6	prettyfaces	236	207	java
7	jbal	102	113	java
8	guavatools	237	85	java
9	monome-pages	240	280	java
10	kryo	309	743	java
11	bitlyj	21	81	java
12	slema	276	368	java
13	bluecove	435	1679	java
14	gp-net-radius	25	28	java
15	aima-java	833	1181	java
16	powermock	966	1512	java
17	restfb	757	1545	java
18	Tensorflow	1104	2386	cpp
19	mangnum	143	1728	cpp

Experimental results I

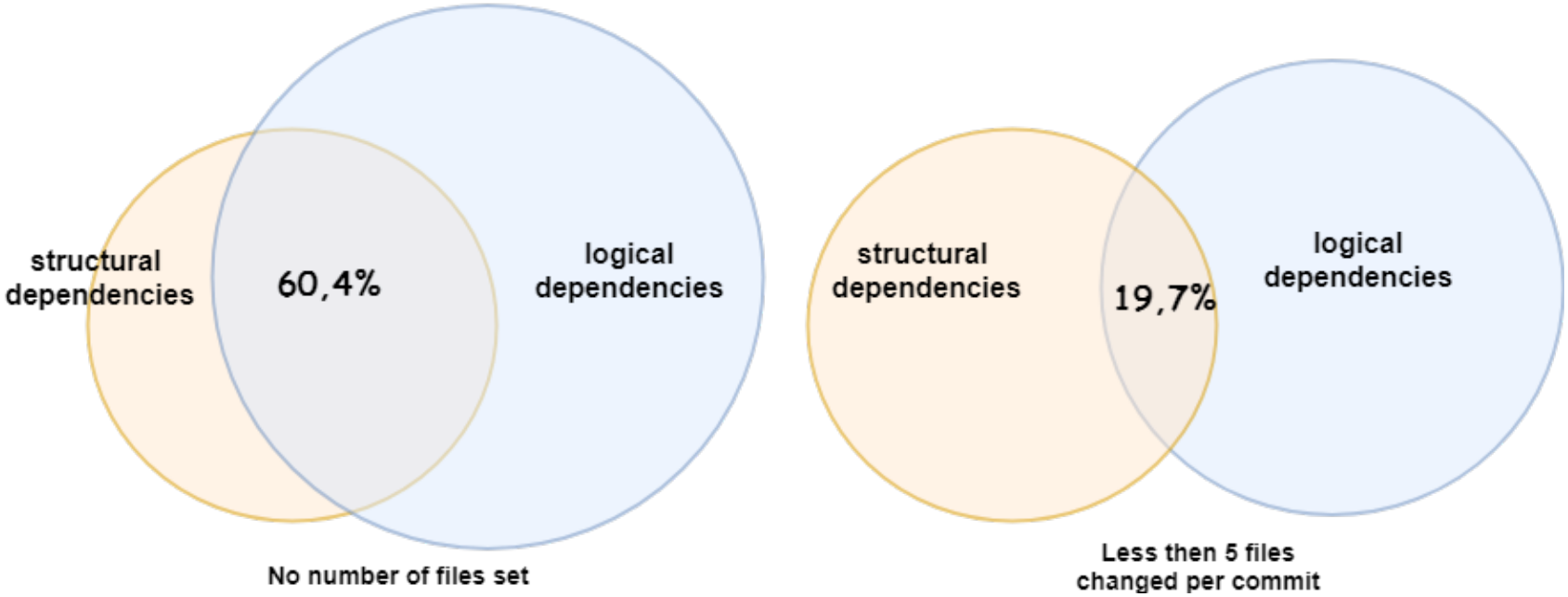
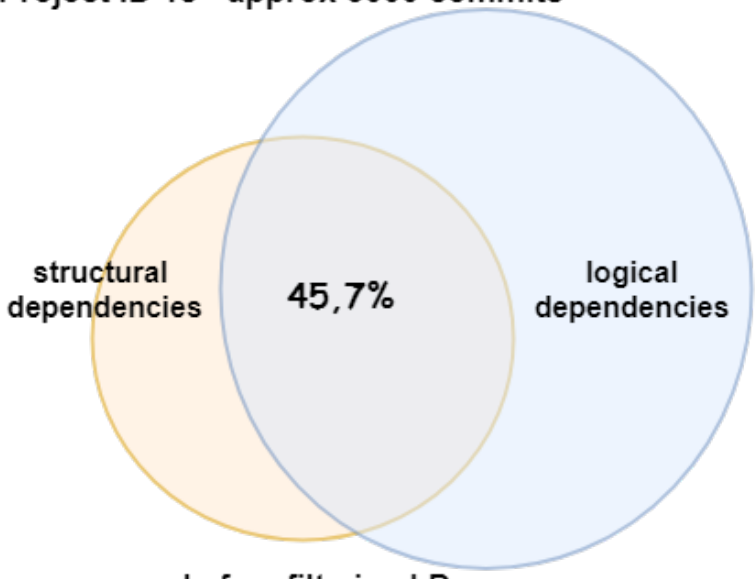


Figure 6: Venn diagrams of the overlapping rates with comments taken into consideration as change.

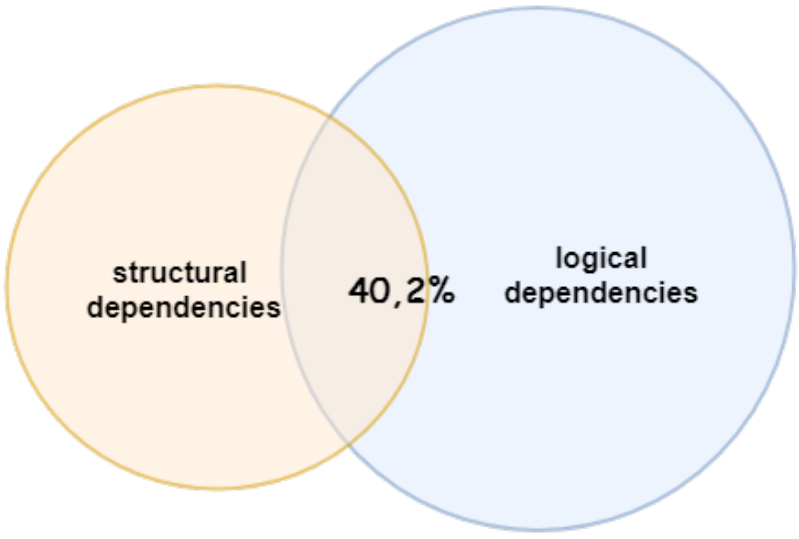
Category	With comments	Without comments
less 5	19,7%	18,9%
more 5 less 20	31,19%	28,7%
more 20	31,47%	29,43%
total	60,4%	57,28%

Experimental results II

Project ID 18 - approx 3000 commits

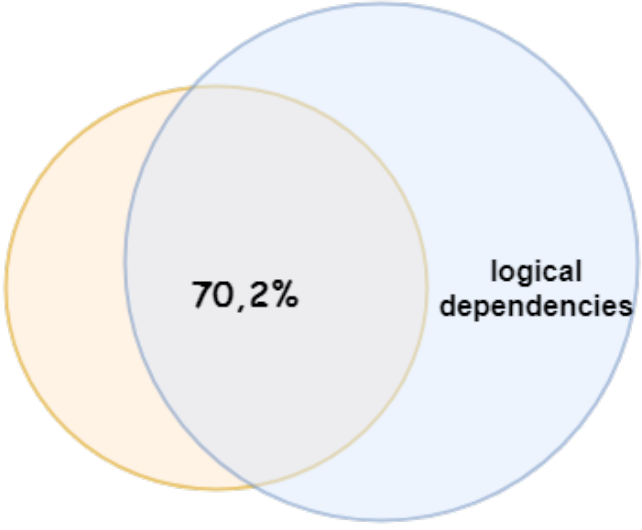


before filtering LD occurrences

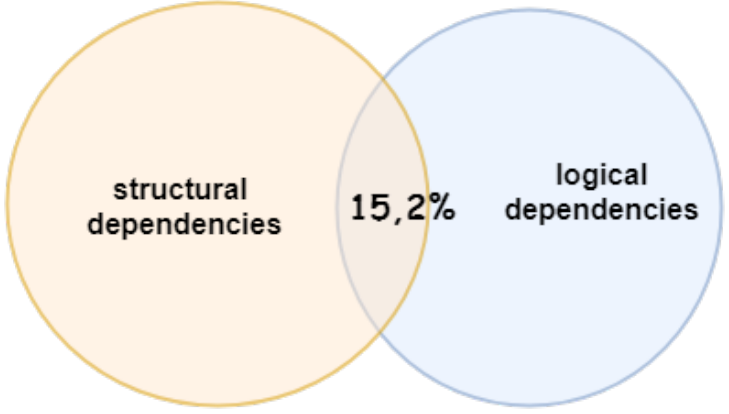


after filtering LD occurrences

Project ID 8 - approx 90 commits



before filtering LD occurrences



after filtering LD occurrences

Conclusions

- Large number of structural dependencies are not doubled by logical ➔ systems partially stable
- + -3% for comments as a change
- The number of changed files taken into consideration influence the results
  - big threshold ➔ not so relevant logical dependencies
  - small threshold (5 10) ➔ more accurate results
- Filtering the logical dependencies after occurrences is good only for projects with a significant number of commits.

Future work

Investigate the cause for the large number of logical dependencies which are not overlapping with structural dependencies.