Methods and Tools for the Analysis of Legacy Software Systems

Stana Adelina Diana

Computer Science and Engineering Department
"Politehnica" University of Timisoara

EIACSDR, 2019

Presentation of the research topic

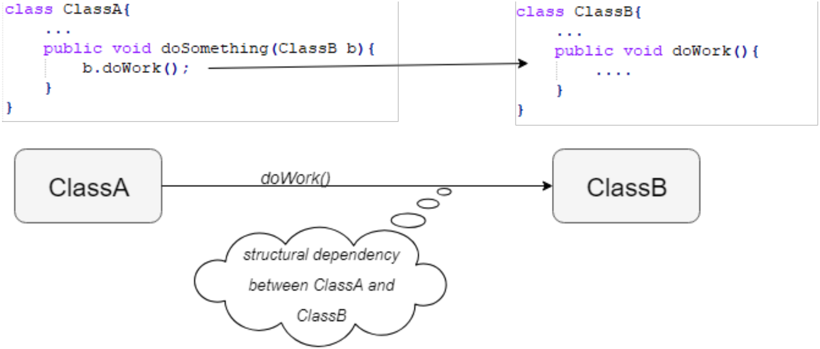
The thesis will develop methods for the analysis of software systems using historical information from the versioning systems[1].

[1] Versioning systems keep track of every change to a file over time so early versions can be restored and used by software teams.

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).

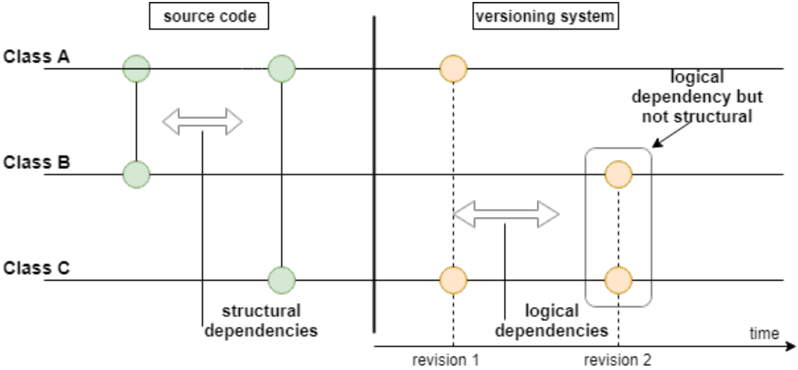


Figure 2: Example of logical and structural dependencies

Current status of research

The current trend recommends that general dependency management methods and tools should also include logical dependencies besides the structural dependencies [2], [3]. But there are no strict rules to *filter co-changes into logical dependencies*, other researches filtered co-changes only in order to decrease their number and not to increase their validity.

[2] Gustavo Ansaldi Oliva and Marco Aurelio Gerosa. On the interplay between structural and logical dependencies in open-source software.

[3] Nemitari Ajienka and Andrea Capiluppi. Understanding the interplay between the logical and structural coupling of software classes.

Research content - filter co-changing classes into logical dependencies

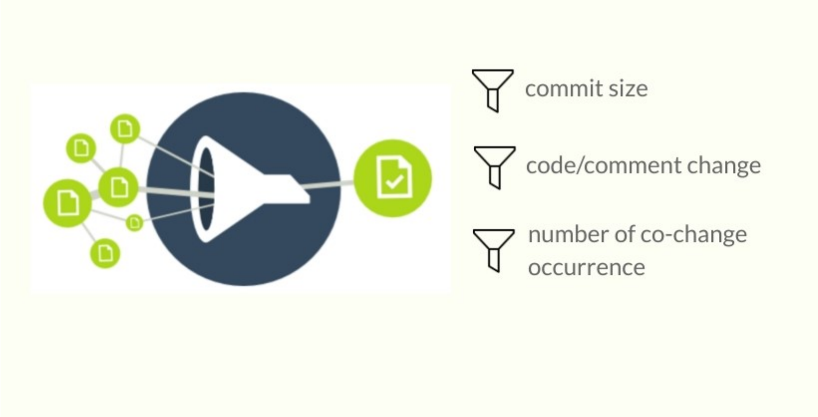


Figure 3: Filters for co-changing classes.

4

[4] Adelina Diana Stana and Ioana Sora. Identifying logical dependencies from co-changing classes.

Research content - refine filter for occurrences of co-changing classes

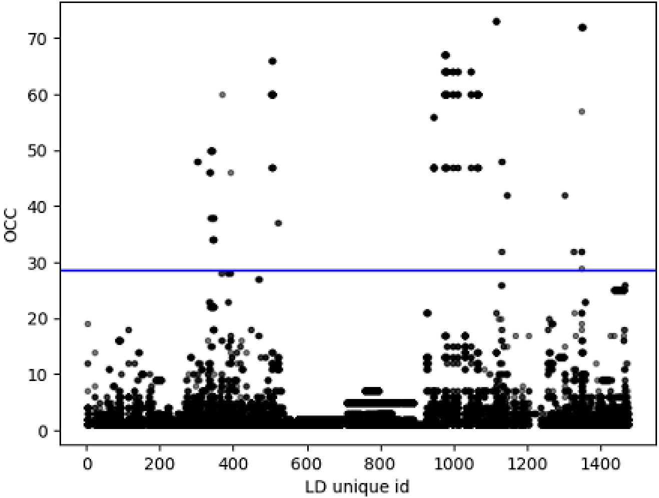


Figure 4: Occurrences rates of co-changing classes extracted from one system. Number of total commits: 6000.

Research content - architectural reconstruction

Use the logical dependencies extracted among structural dependencies in tools for architectural reconstruction to evaluate the improvement.

Research content - software metrics

Compare the number of logical dependencies with metrics and study their connections. Metrics:

- Fan Out - number of other classes referenced by a class.
- Fan In - number of other classes that reference a class.

Paper: Identifying logical dependencies from co-changing classes

We studied 20 open source systems written in Java and CSharp.
Filters and Thresholds:

- commit size (cs): the maximum size of commit transactions which are accepted to generate logical dependencies. The values for this threshold were 5, 10, 20 and no threshold (infinity).
- number of occurrences (occ): the minimum number of repeated occurrences for a co-change to be counted as logical dependency. The values for this threshold were 1, 2, 3 and 4.
- with/without taking comments into consideration as valid change.

Paper: Identifying logical dependencies from co-changing classes

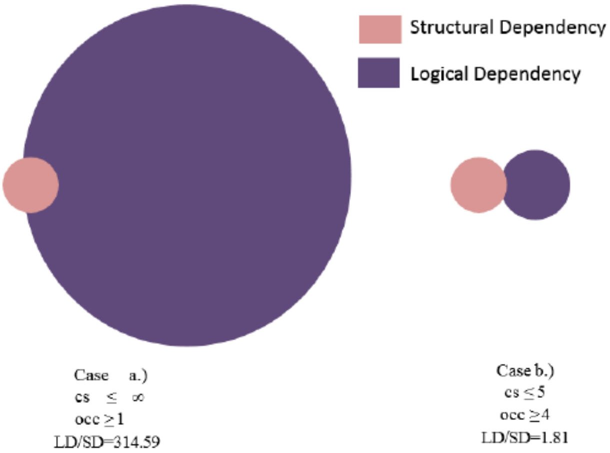


Figure 5: Logical and structural dependencies overlapping.

Paper: Conclusions

- Large number of structural dependencies are not doubled by logical dependencies.
- The most important factors in co-changing classes filtering: commit size (cs) and number of occurrences (occ).
- The commit size threshold(cs) influence the size of the extracted co-changes but not their relevance.
- Filtering the logical dependencies after occurrences must be made using a dynamically calculated threshold.