Utilization of Method Graphs to Measure Cohesion in Object Oriented Software

Atakan Aral, Tolga Ovatman

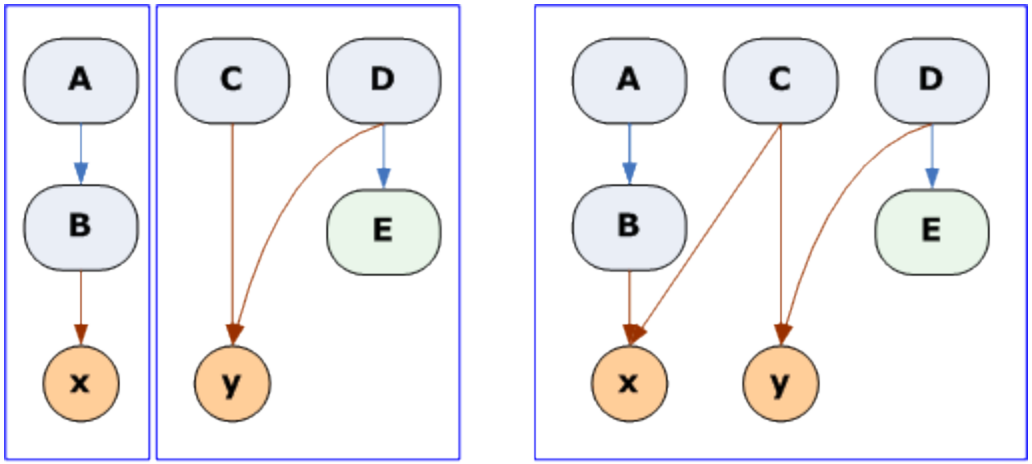
Istanbul Technical Universtiy
Faculty of Computer and Informatics

Outline

- Introduction
- Motivation
- Contribution
- Method Graphs
- Clustering & Similarity Measurement
- Results & Discussion
- Conclusion
- Future Work

Introduction

- Cohesion is the *harmony* of software components in performing responsibilities
- Hard to measure quantitatively
- Cohesion of a class is usually measured by examining *common data usage of its methods*



Motivation

- Conventional techniques analyze *object code* and require *compilation*
- Hard to detect data usage from *source code*
  - Indirect ways to access a variable
- Prevents evaluating incomplete code

- Our aim is to derive *a pure source code based cohesion measure*

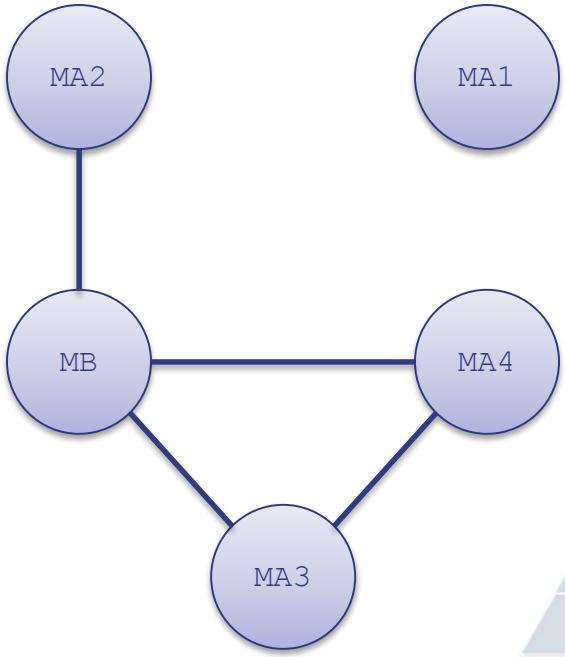
Contribution

- A cohesion measure that
  - depends purely on *static analysis*
  - considers only *method-to-method relations*
  - is *graph based* and *software-wide*
  - is highly correlated to *LCOM metric*

Method Graphs

- Cooperating Methods (CM) relation
  - involves method couples that are *called together* from another *host method*.

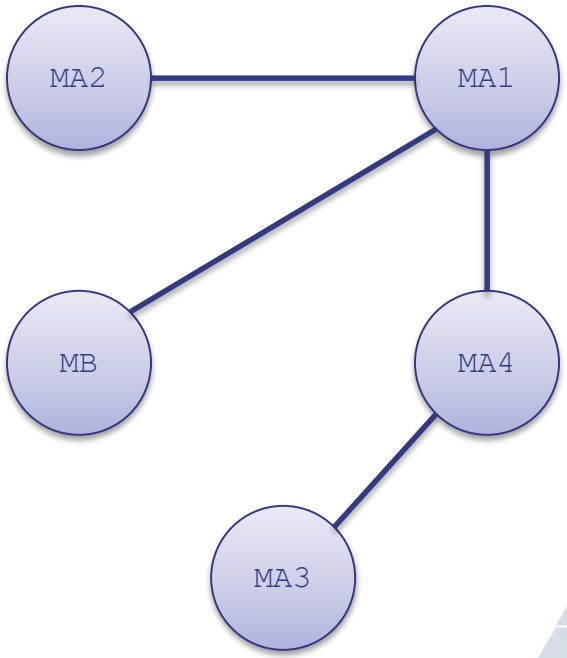
```
Class A{
    MA1(){
        MA2();
        B.MB();
    }
    MA2(){
        B.MB();
        MA3();
        MA4();
    }
    MA3(){}
    MA4(){}
}
```



Method Graphs

- Method Call (MC) relation
  - involves method couples that *call one another*.

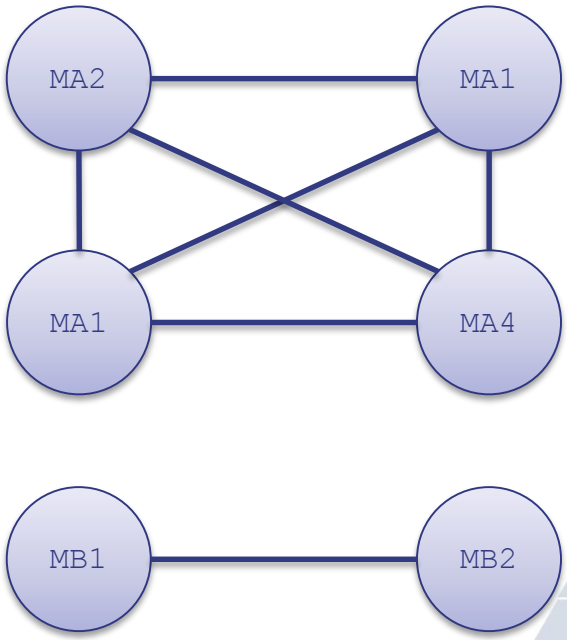
```
Class A{
    MA1(){
        MA2();
        B.MB();
    }
    MA2(){}
    MA3(){
        MA4();
    }
    MA4(){
        MA1();
    }
}
```



Method Graphs

- Method Layout (ML) relation
  - involves method couples that are in the *same class*.

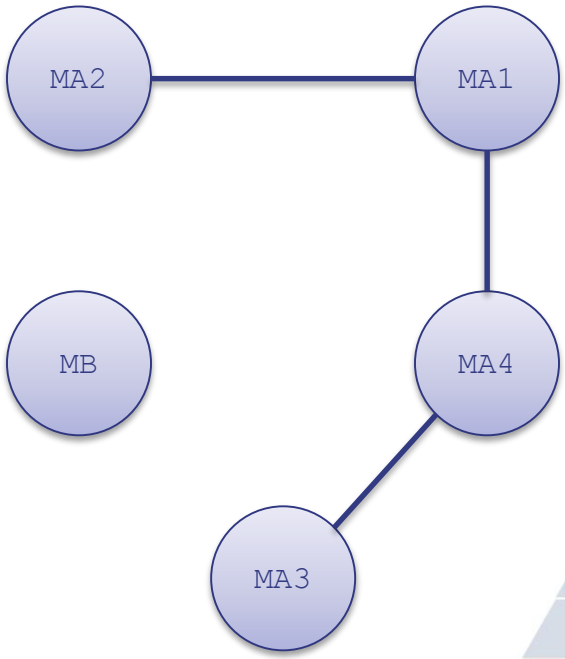
```
Class A{
    MA1(){}
    MA2(){}
    MA3(){}
    MA4(){}
}

Class B{
    MB1(){}
    MB2(){}
}
```



Method Graphs

- Internal Call (IC) relation
  - involves method couples that *call one another* and are in the *same class*.

```
Class A{
    MA1(){
        MA2();
        B.MB();
    }
    MA2(){}
    MA3(){
        MA4();
    }
    MA4(){
        MA1();
    }
}
```



Clustering & Similarity

- 4 graphs are clustered using *Weak Component Clustering*
- 6 clustering couples are compared using *adjusted Rand index*
  - CM—MC: Are the methods calling each other, also called together from another body?
  - IC—ML: What is the level of fragmentation within classes?

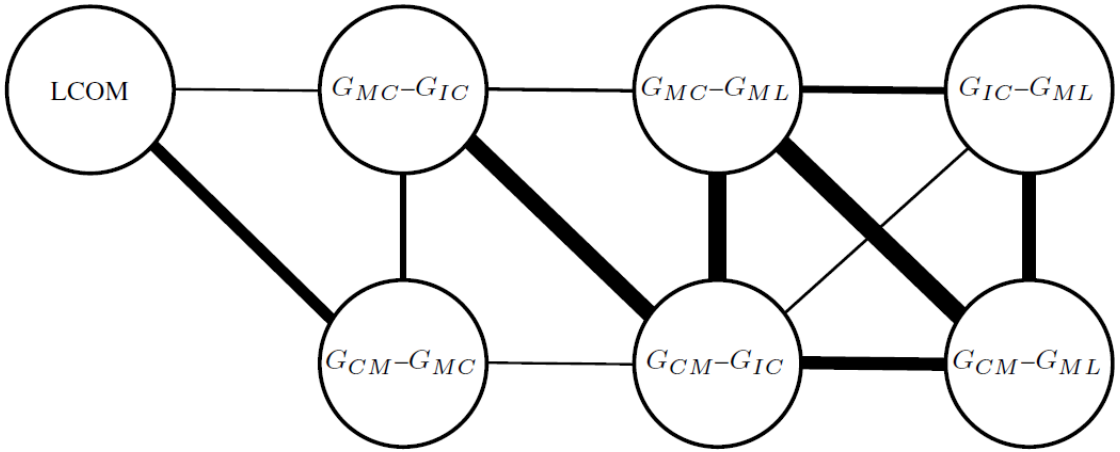
Results & Discussion

- 14 most popular open-source Java projects from *GitHub* and *SourceForge*

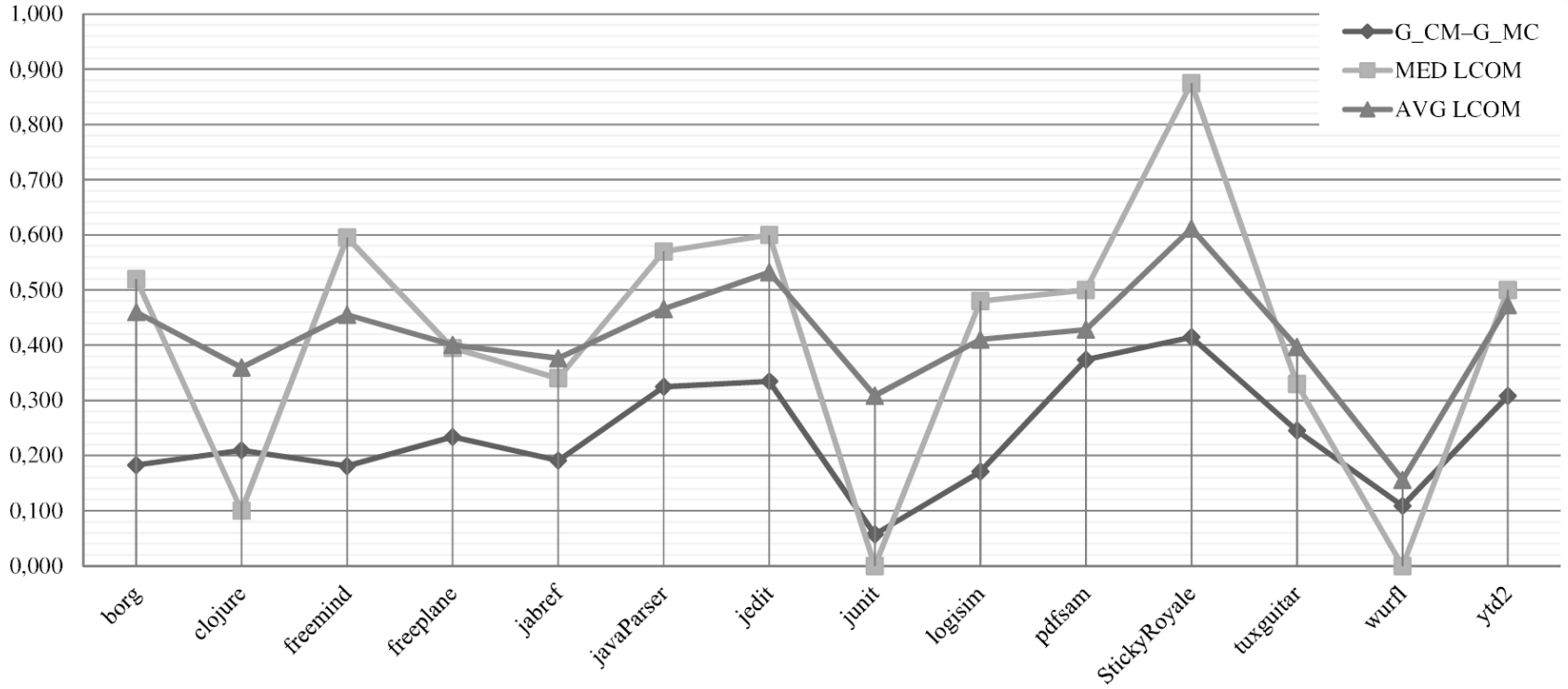
PROJECT	RAND INDICES						LCOM HS	
	G_{CM}–G_{MC}	G_{CM}–G_{IC}	G_{CM}–G_{ML}	G_{MC}–G_{IC}	G_{MC}–G_{ML}	G_{IC}–G_{ML}	Median	Average
borg	0.183	0.012	0.017	0.004	0.003	0.438	0.520	0.460
clojure	0.210	0.077	0.057	0.019	0.009	0.484	0.100	0.360
freemind	0.181	0.010	0.020	0.003	0.003	0.291	0.595	0.455
freeplane	0.234	0.006	0.009	0.002	0.002	0.360	0.395	0.400
jabref	0.191	0.005	0.011	0.001	0.002	0.325	0.340	0.376
javaParser	0.325	0.507	0.476	0.480	0.466	0.952	0.570	0.466
jedit	0.334	0.124	0.074	0.068	0.036	0.570	0.600	0.532
junit	0.057	0.018	0.021	0.002	0.004	0.212	0.000	0.309
logisim	0.171	0.004	0.005	0.001	0.001	0.294	0.480	0.410
pdfsam	0.374	0.007	0.045	0.005	0.011	0.110	0.500	0.429
StickyRoyale	0.414	0.364	0.088	0.675	0.234	0.278	0.875	0.612
tuxguitar	0.245	0.007	0.010	0.003	0.002	0.542	0.330	0.397
wurfl	0.109	0.049	0.055	0.007	0.010	0.249	0.000	0.156
ytd2	0.308	0.181	-0.033	0.513	-0.080	0.274	0.500	0.473

Results & Discussion

	G_{CM} – G_{MC}	G_{CM} – G_{IC}	G_{CM} – G_{ML}	G_{MC} – G_{IC}	G_{MC} – G_{ML}	G_{IC} – G_{ML}
G_{CM}–G_{IC}	**0.585**					
G_{CM}–G_{ML}	0.320	**0.808**				
G_{MC}–G_{IC}	**0.644**	**0.876**	0.445			
G_{MC}–G_{ML}	0.430	**0.882**	**0.938**	**0.602**		
G_{IC}–G_{ML}	0.225	**0.593**	**0.778**	0.245	**0.677**	
MED LCOM	**0.767**	0.474	0.211	**0.572**	0.393	0.168
AVG LCOM	**0.770**	0.476	0.168	**0.580**	0.393	0.233



Results & Discussion



- Calling two methods, that call each other, from the body of another method, decreases the overall cohesion of software

Conclusion

- We suggest a novel technique that allows to measure *software-wide* cohesion.
- It uses *static source code analysis*.
- Results indicate that its correlation to LCOM HS is 77%.

Future Work

- **to adapt our measure to class level**
- to use direction—weight data to improve accuracy
- to build a mathematical basis for the relation between our measure and the LCOM metric
- to study on the other correlated groups

İTÜ

Thank you!