Component Interface Discovery from Software Execution Data

Cong Liu, Boudewijn van Dongen, Nour Assy, and Wil van der Aalst

c.liu.3@tue.nl

November 16, 2017

Technische Universiteit
Eindhoven
University of Technology

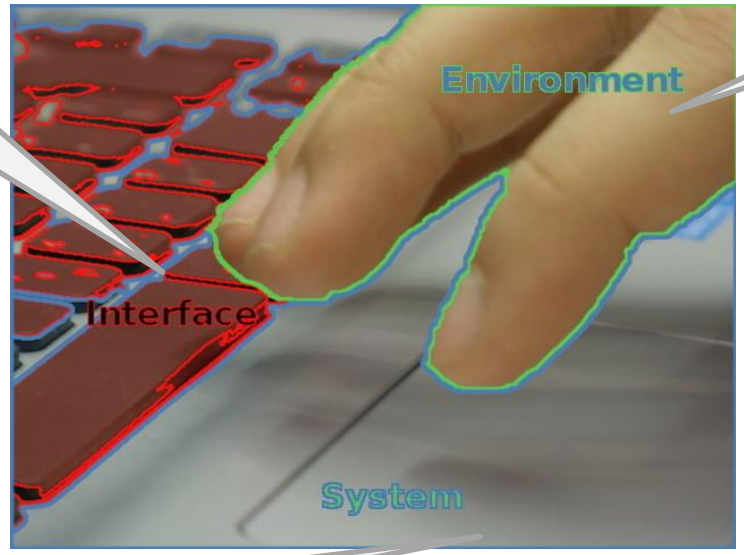
Where innovation starts

Outline

- Background
- Research question
- Our approach
- Experimental evaluation
- Conclusion and future work

What is interface in real life?

Generally speaking, **interface** is the interaction between **system** and its **environment**.



Interface→ keyboard

environment→ hand

System→ laptop

What is an interface in software?

- The **system** can be a component in question, and the environment is the rest of the project (other components).

- A component **interface** is a contract/intersection between the subject component and other components.

For java language:

- The system can be objects/classes, and the environment is other classes/objects.

- A **java interface** is a set of **public methods**, and each method can be invoked independently by the environment.

Why do we need to discover interface?

Software architecture reconstruction: aims to identify a **higher level views** from **low level data** to help understanding.

Low level data: source code, execution data, etc.

Higher level view: component and **interface.**

Component: for oo software, a component is a set of classes that provides a number of **functions**.

Functions are implemented as a set of methods and are usually organized as **interfaces**.

Outline

- **Background**
- **Research question**
- **Our approach**
- **Experimental evaluation**
- **Conclusion**

Research problem

Input 1: the components are clearly defined. E.g., by clustering classes based on coupling and cohesion.

Input 2: execution data record the invocation information among method calls.

Concrete problems:

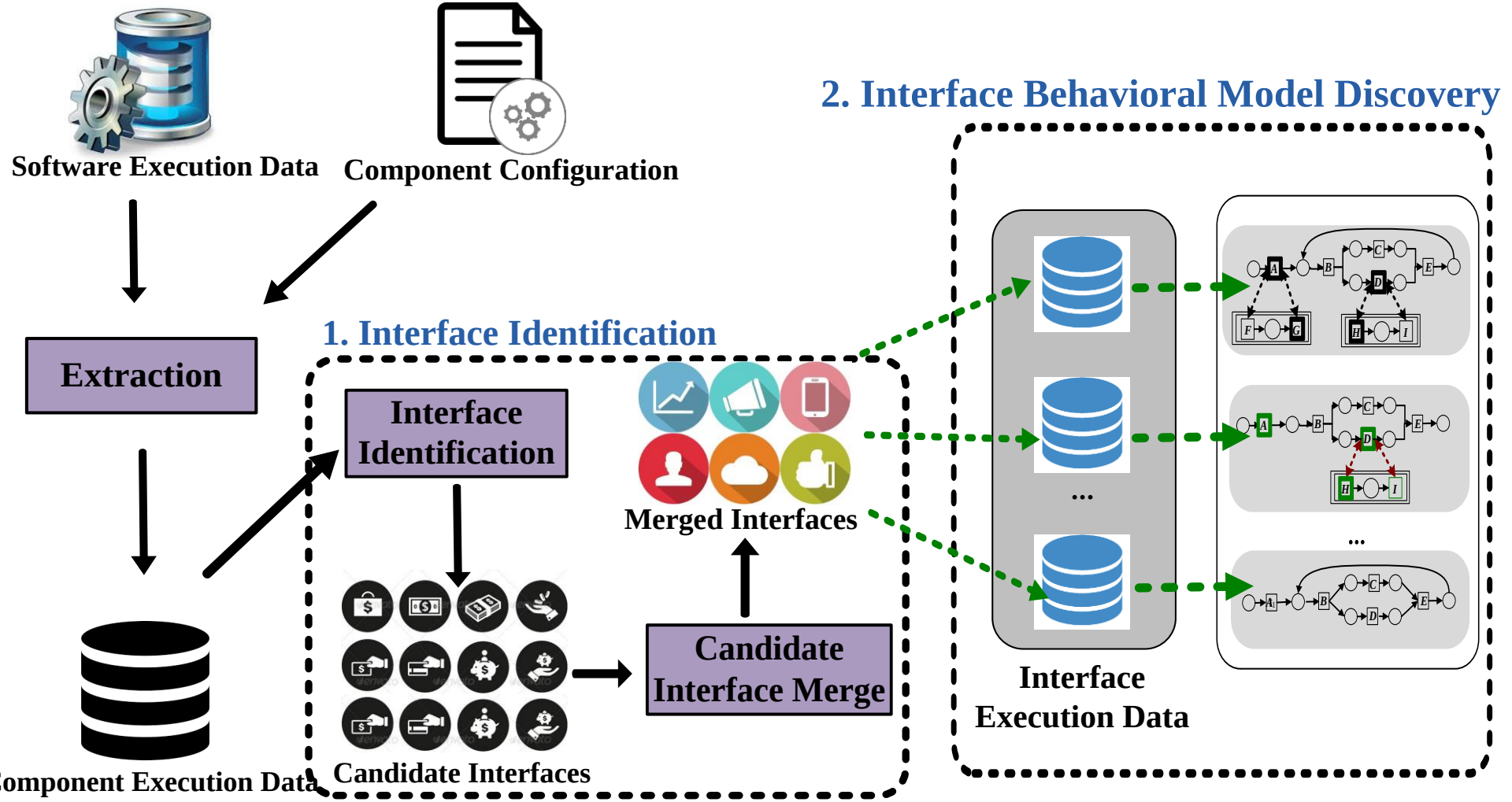
- Identifying interfaces for each component by taking as input execution data and component configuration.

- Discovering behavioral model (contract) of identified interfaces. It explicitly specifies in which order the methods are invoked.

Outline

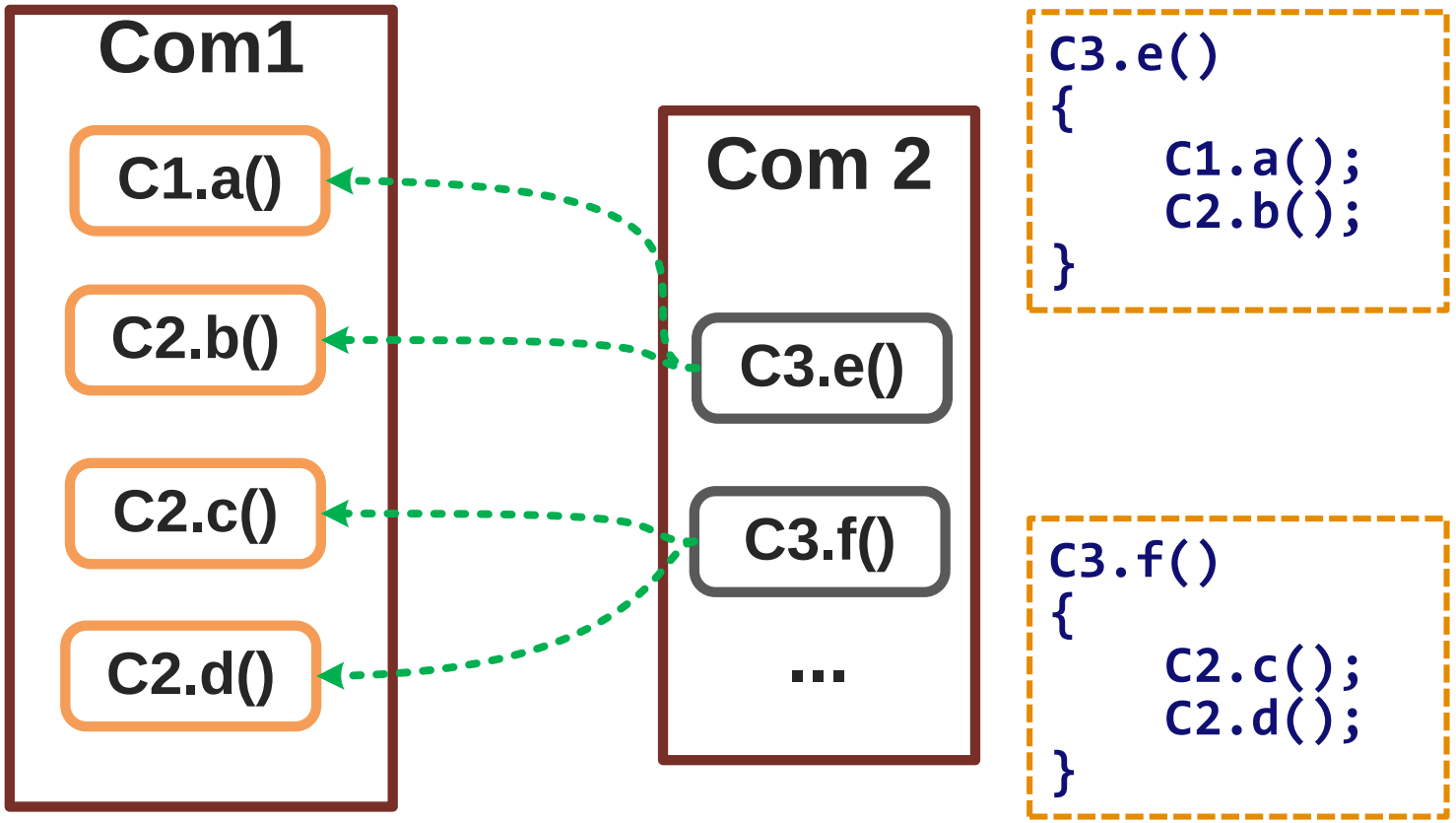
- **Background**
- **Research question**
- **Our approach**
- **Experimental evaluation**
- **Conclusion**

An approach overview



An example

- **Com1={C1, C2} and Com2={C3}.**



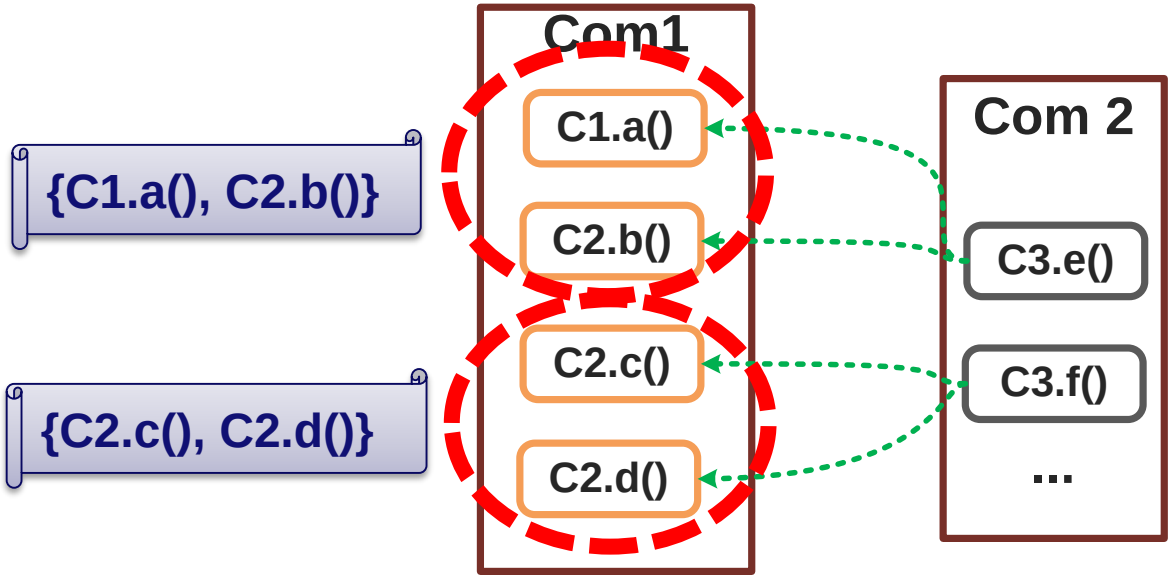
Candidate interface identification

Interfaces are identified based on functional aspects.

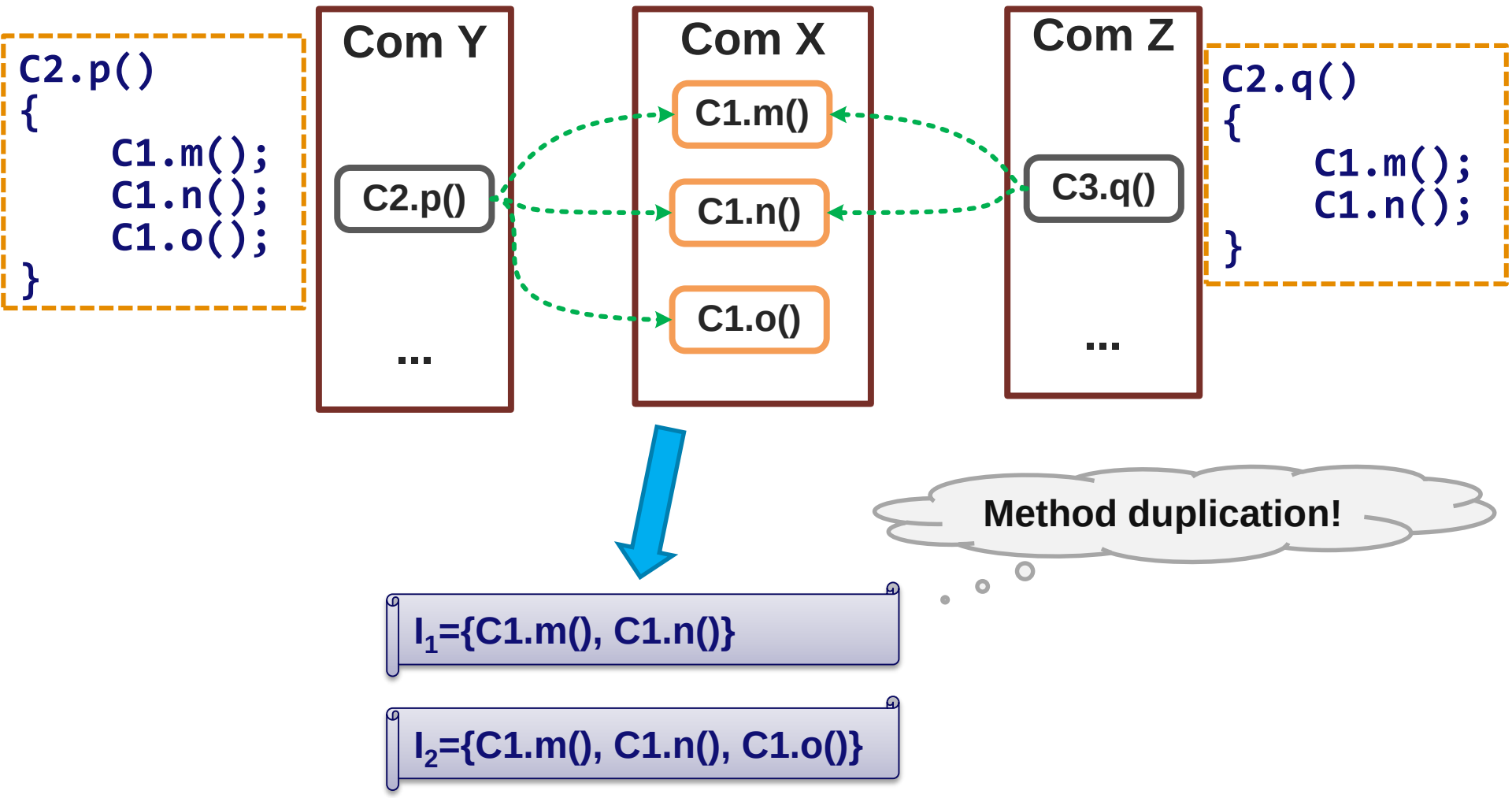
If a set of methods are called by the same method, they probably share the same function.

Functionally related methods

For each component, candidate interfaces are obtained by grouping methods that are called by the same method.



Another example



Candidate interface merge

- Merge similar candidate interfaces in a way such that the shared methods among interfaces are limited to a reasonable range.
- Similarity between interfaces are measured by ***Jaccard Distance*** of the method set.
- Similarity threshold δ, if δ**<Sim**, then merge; otherwise keep as it is.

Candidate interface merge

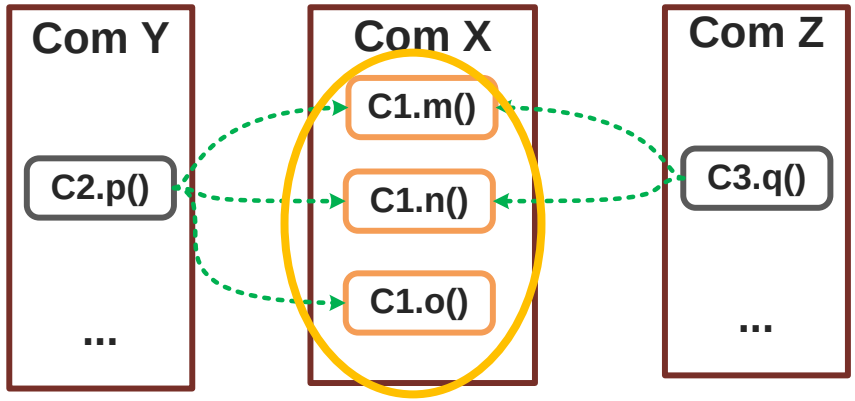
I_1={C1.m(), C1.n()}

I_2={C1.m(), C1.n(), C1.o()}

I_M={C1.m(), C1.n(), C1.o()}

Sim(I1, I2)=2/3.

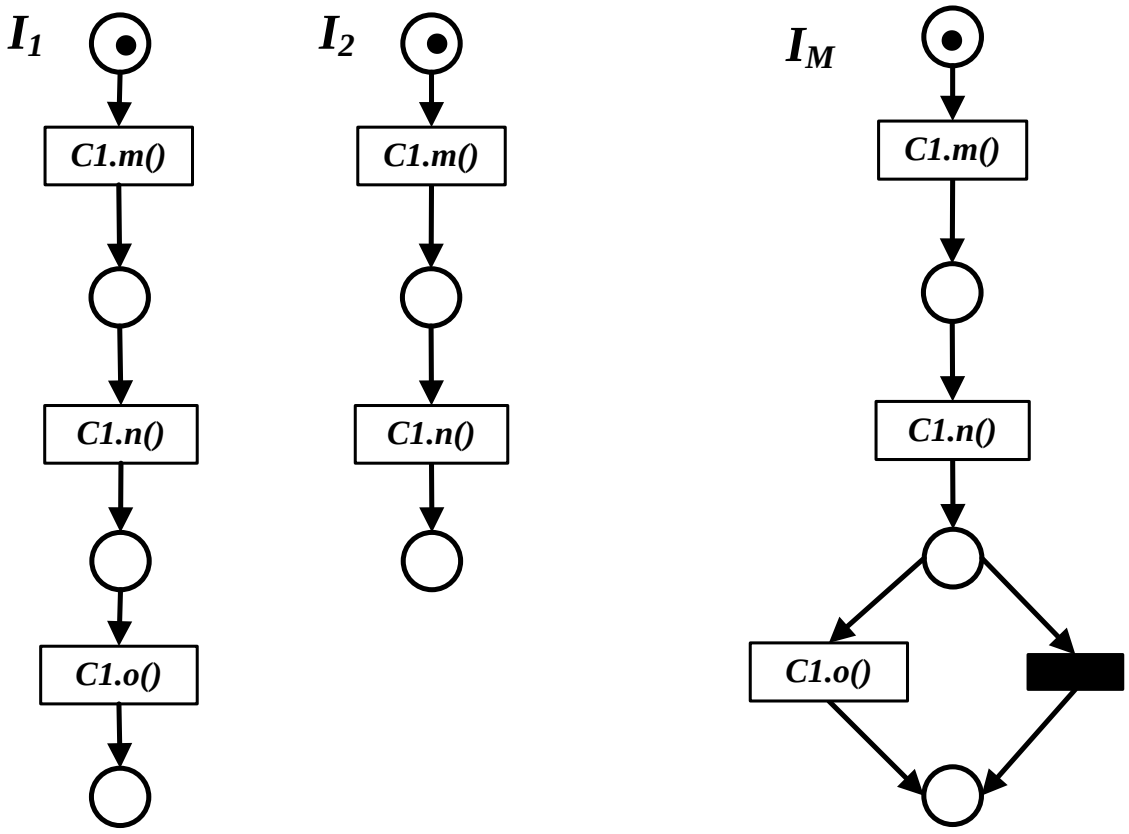
If δ=0.5, then merge them.



Interface behavioral model discovery

Step1: construct an event log for each interface;

Step2: discover a behavioral model using PM.



(a) $\delta = 0.8$

(b) $\delta = 0.5$

Outline

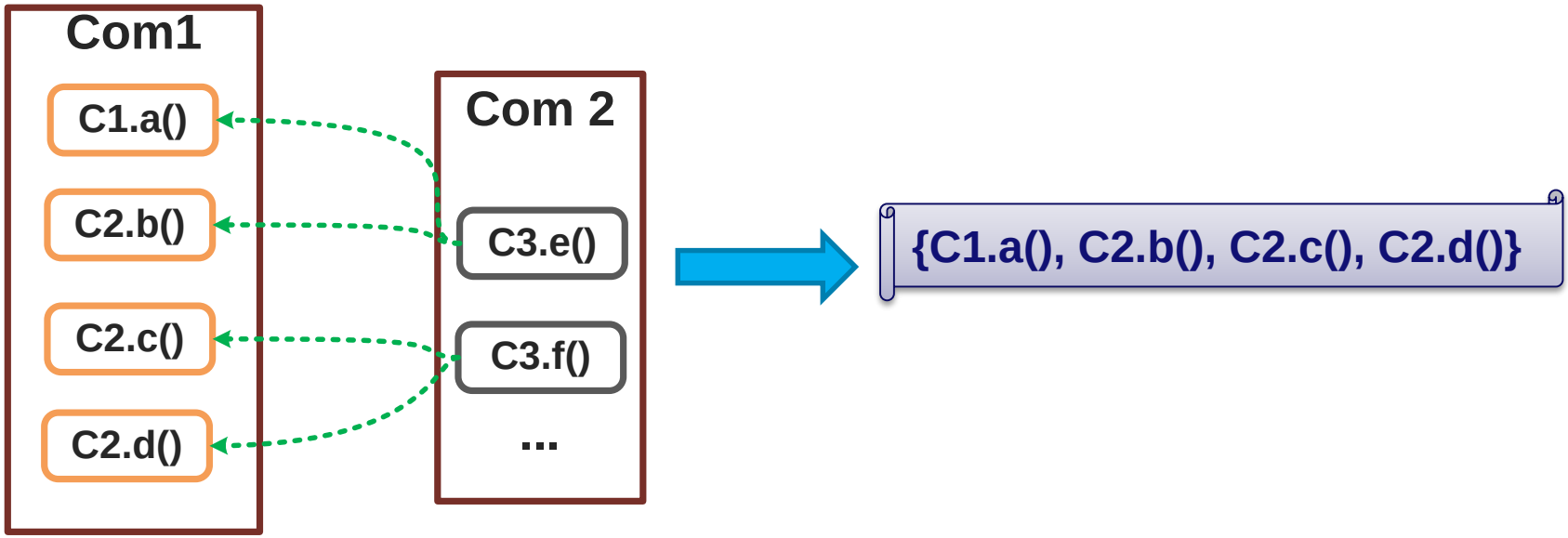
- Background
- Research question
- Our approach
- Experimental evaluation
- Conclusion

Subject Software Systems and Data

Software	#Package	#Class	#Method	#Method Call
Sports Lobby	3	5	15	236
Google Sheet	7	22	57	3180
JGraphx	10	88	712	267627
JHotDraw	5	102	572	392164

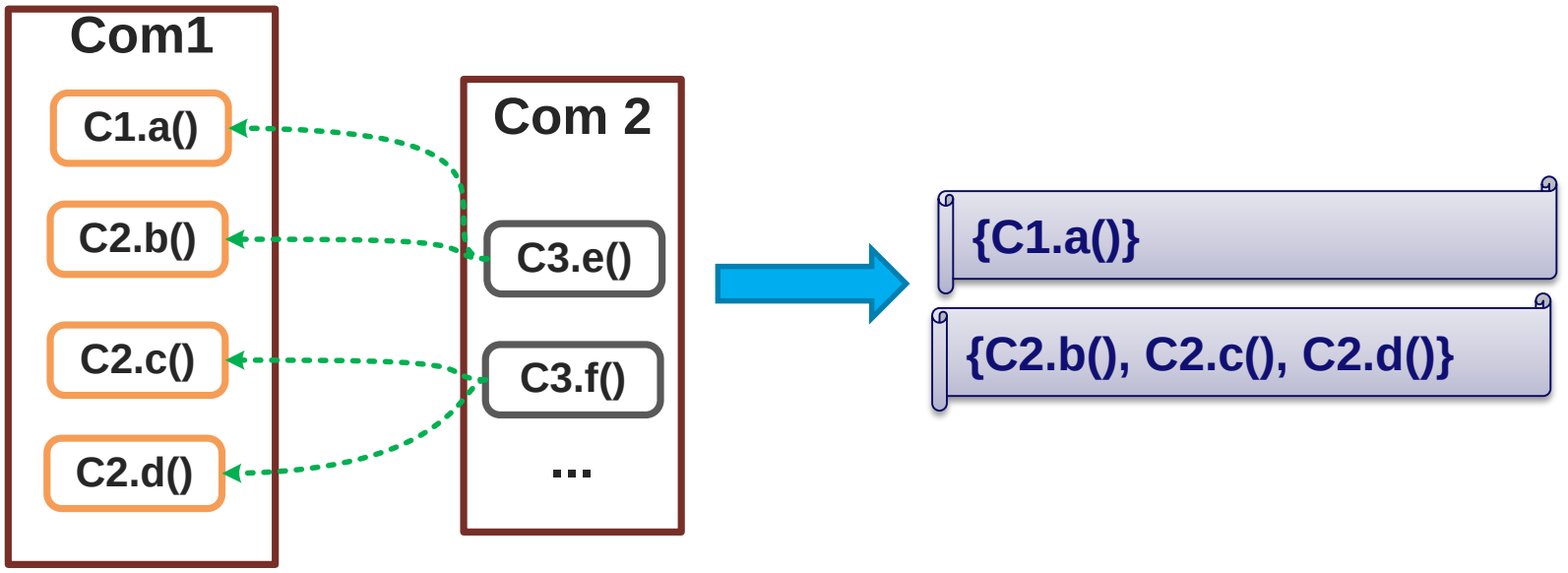
Different approaches

- **A1: interfaces are identified by grouping all methods of the component.**



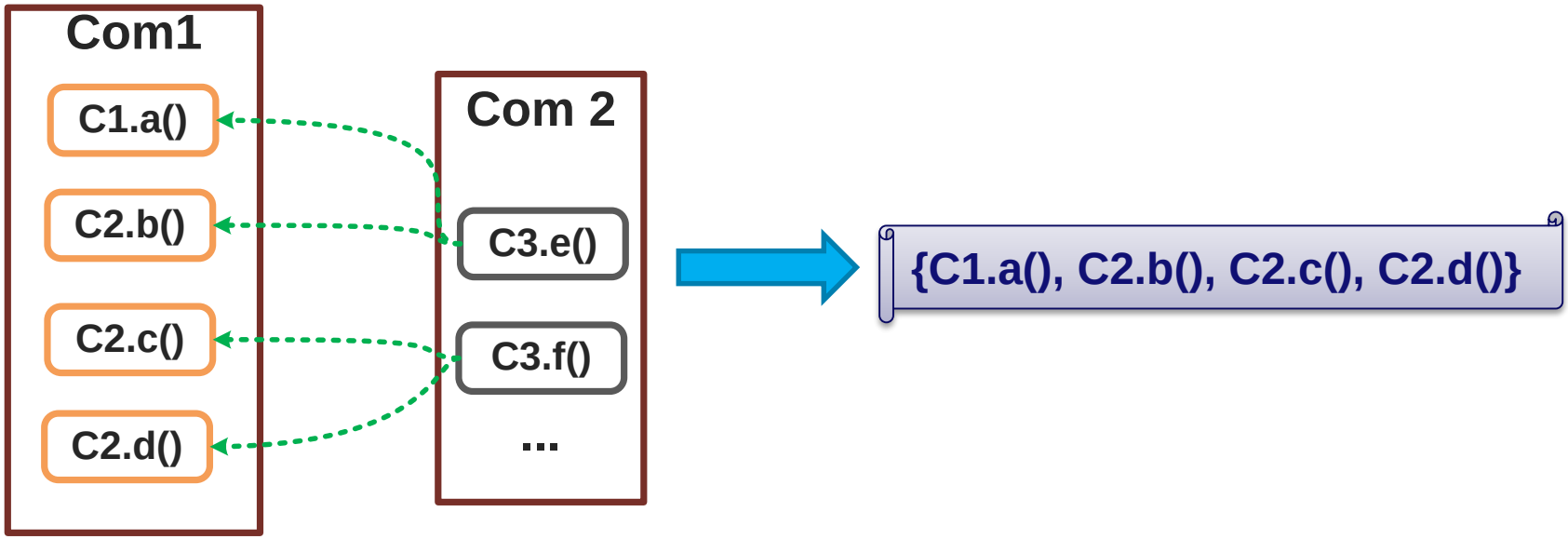
Different approaches

- **A2: interfaces are identified by grouping methods of the same class.**



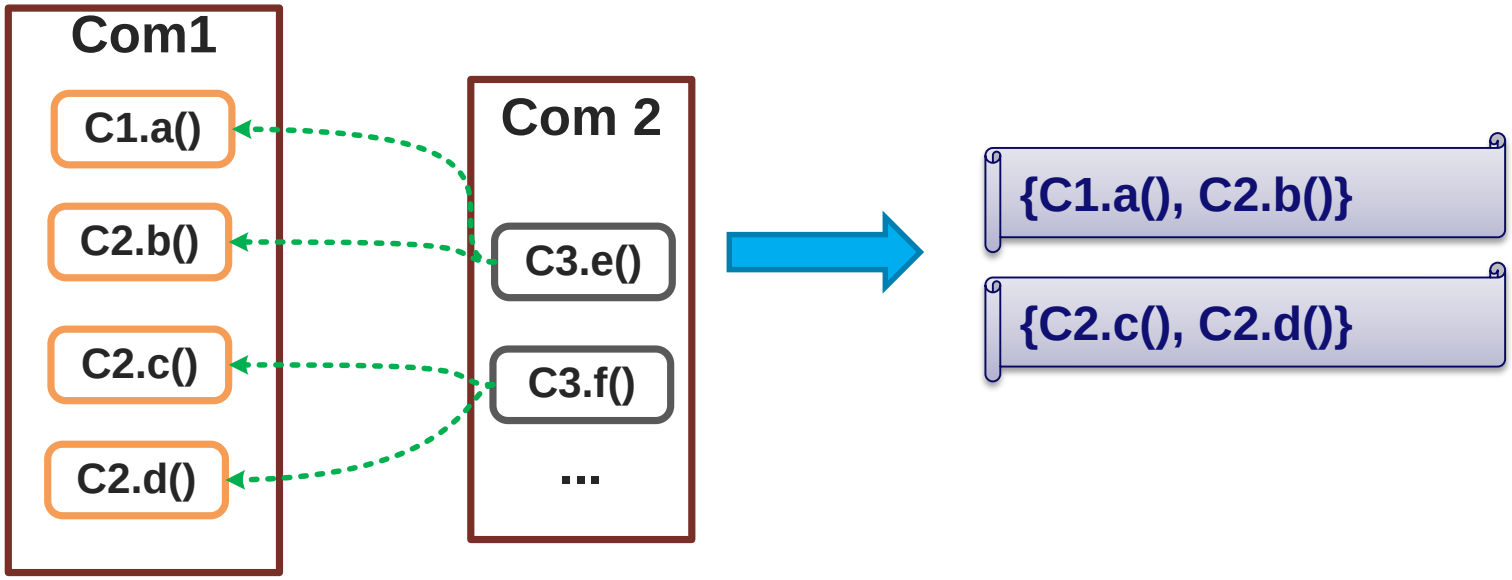
Different approaches

- **A3: interfaces are identified by grouping methods that are called by the same component.**



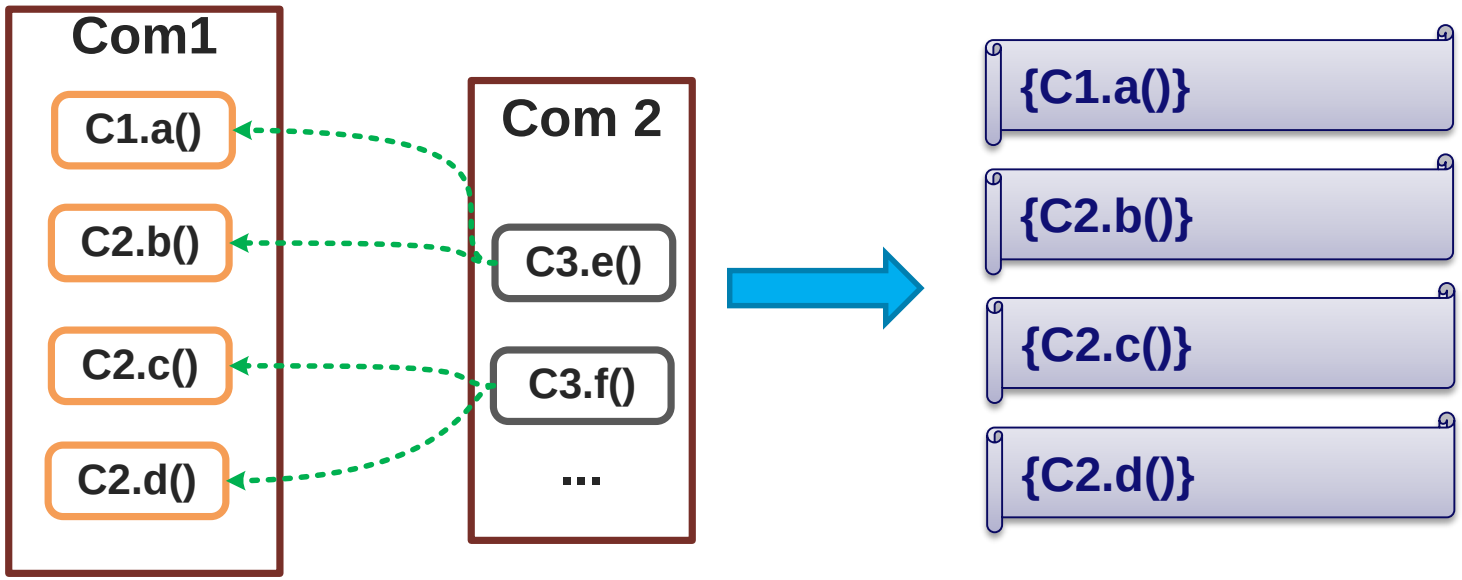
Different approaches

- **A4: interfaces are identified by grouping methods that are called by the same method of another component.**



Different approaches

- **A5: interfaces are identified for each method of the component.**



Quality criteria for evaluation

- **Number and size of identified interfaces.**
  --a reasonable number of interfaces of manageable size.

- **Functional consistency of identified interfaces.**
  --wrongly group functionally unrelated methods?
  --wrongly separate functionally related methods?

- **Complexity of the discovered interface models.**
  --interfaces that are easy to understand.

Number and size of identified interfaces

- **NOI:** quantifies the number of interfaces identified for each component.

- **ANM:** measures the size of an interface by the number of methods it contains. We take the average for each component.

Functional consistency measure

Functionally related methods

- **PMM:** measures the proportion of methods within the same interface that are functionally related.
- **PCM:** measures the proportion of methods that are functionally related and that are grouped in the same interface.

Complexity

- **Complexity of discovered interface behavioral models, i.e., workflow net.**

- **ECaM (Extended Cardoso metric):** uses a syntactical view of process complexity by counting the various splits (XOR,OR and AND) in the net structure.

- **ECyM (Extended Cyclomatic metric):** measures the behavior complexity of the model by analyzing the reachability graph.

Evaluating the similarity threshold

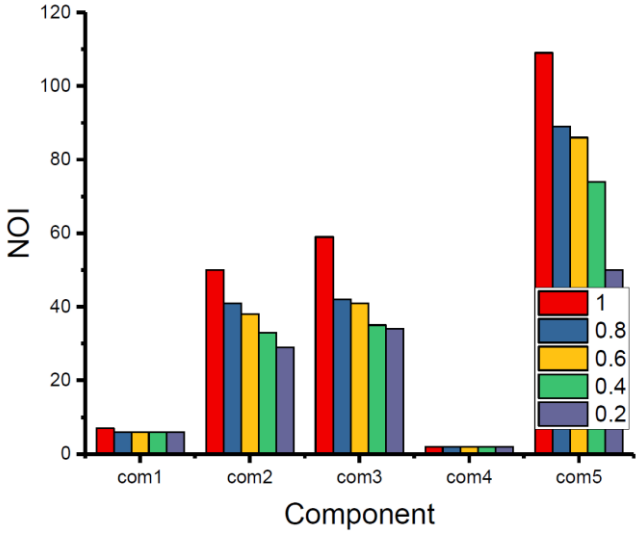
- **Target 1:** how similarity threshold affects the quality of the discovered of interface for A4.
- **Target 2:** find the optimal threshold value for A4.

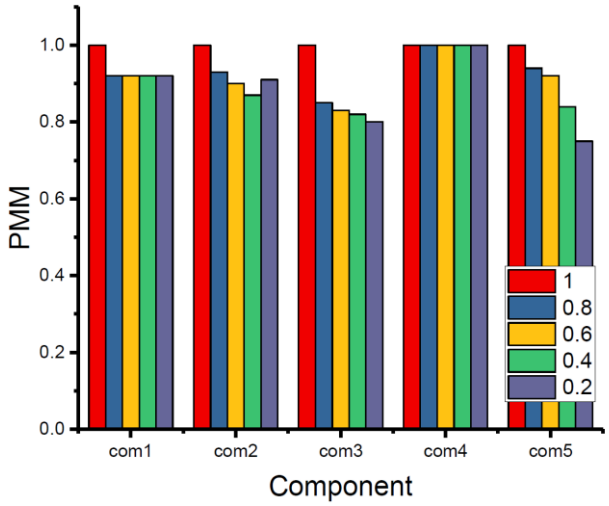
- **General affect:** (1) reduce the number of interfaces per component; (2) limit method duplication.

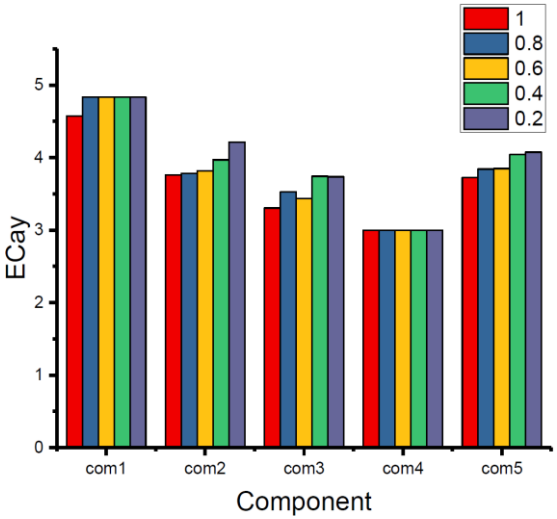
- **Threshold values used:** 1.0, 0.8, 0.6, 0.4, and 0.2.

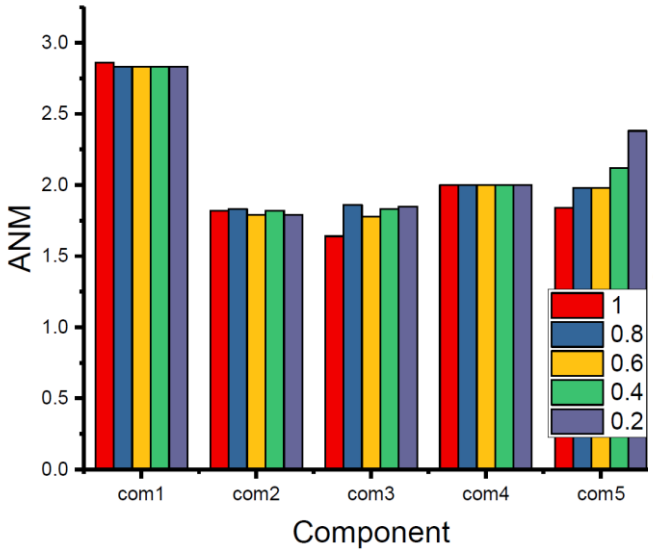
- **Software systems:** JHotDraw and JGraphx.

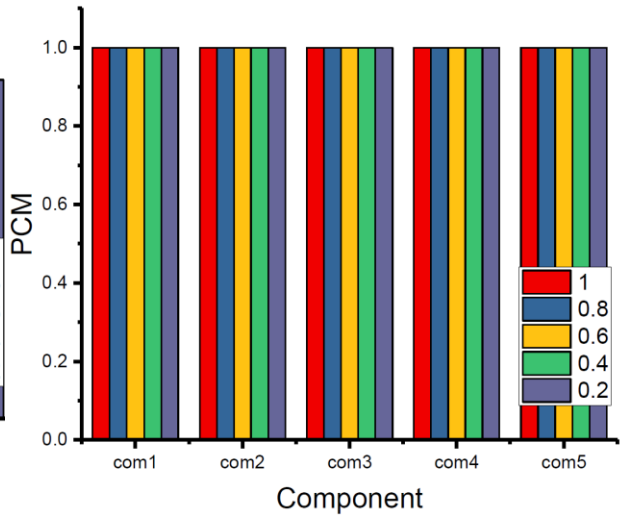
Evaluating the similarity threshold (JHotDraw)

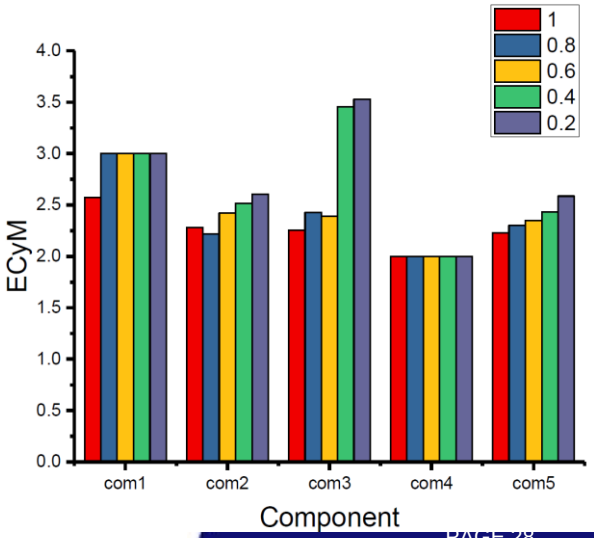




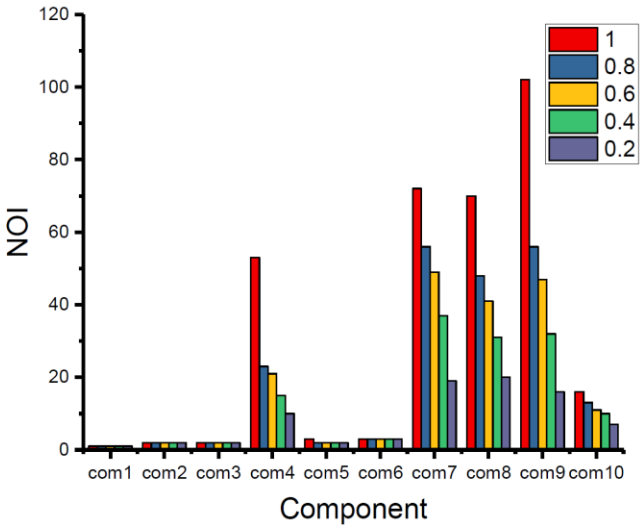


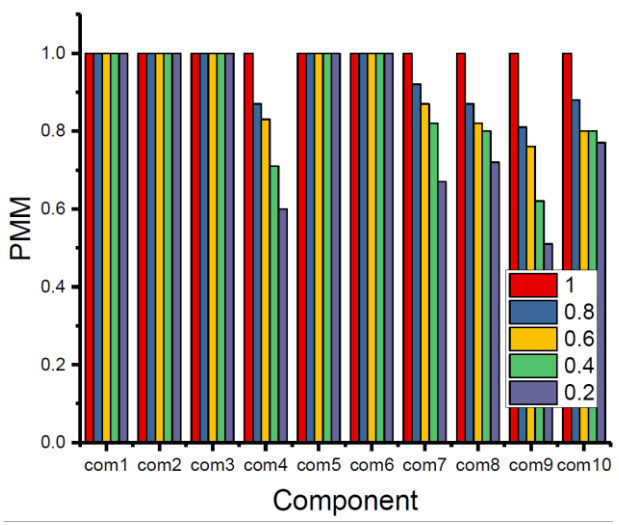


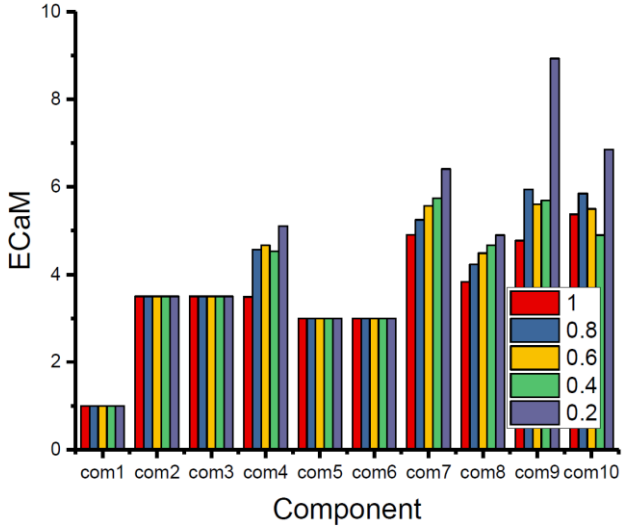


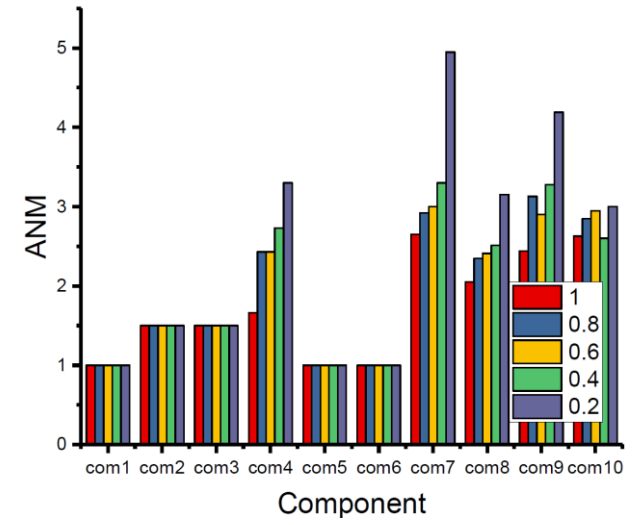


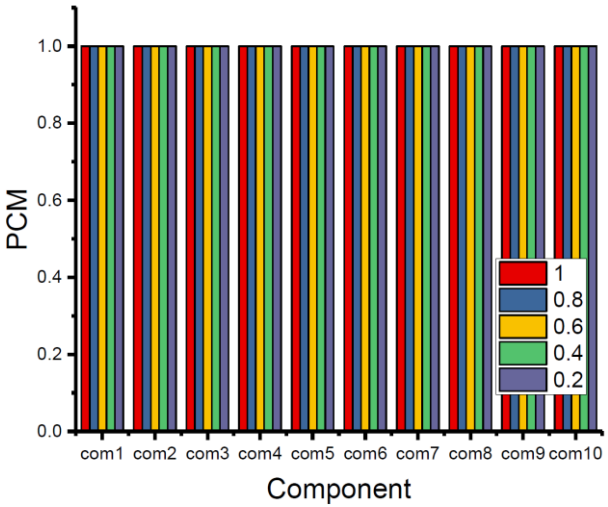
Evaluating the similarity threshold (JGraphx)

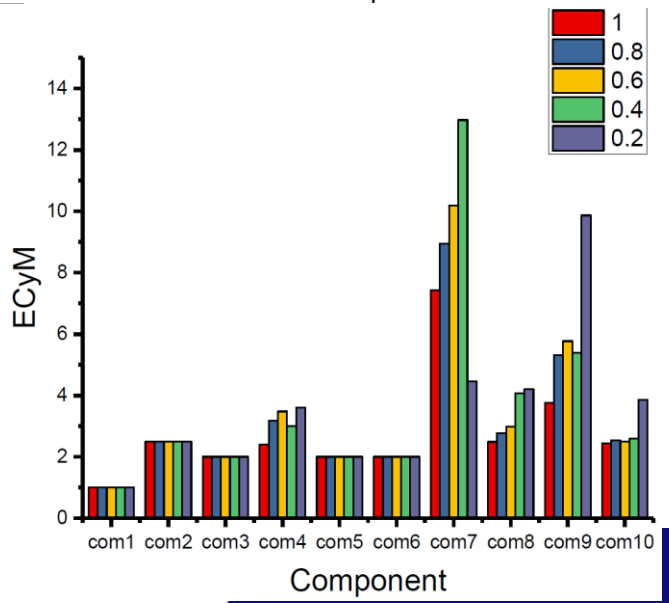




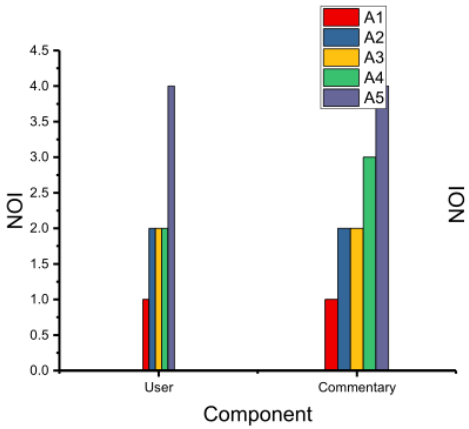




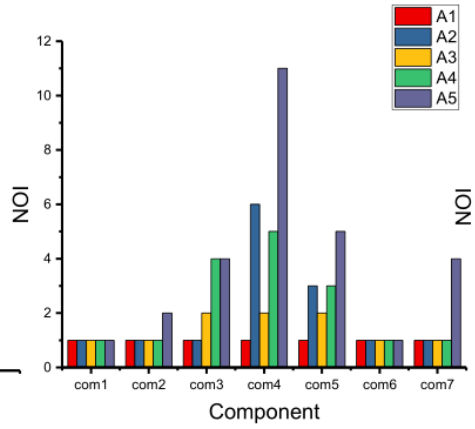




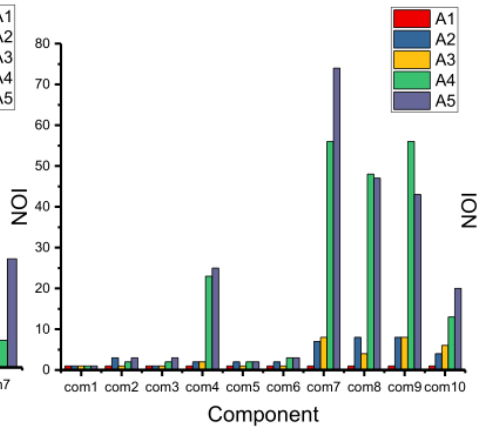
Evaluating different approaches



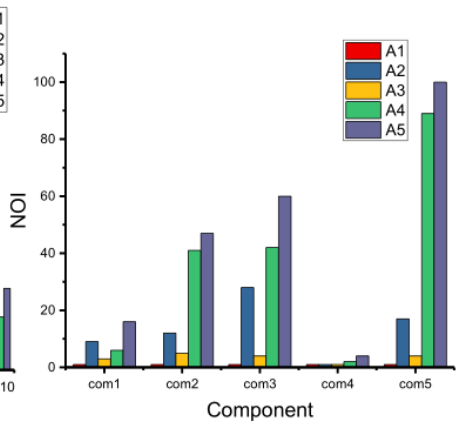
(a) SportLobby NOI



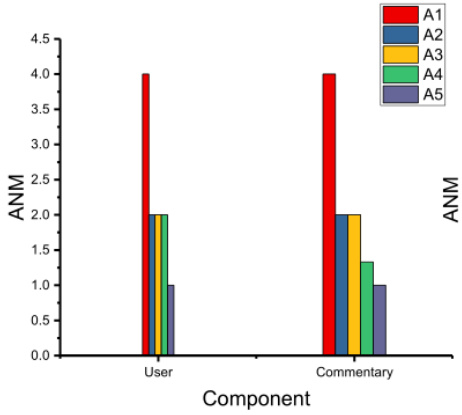
(b) GoogleSheet NOI



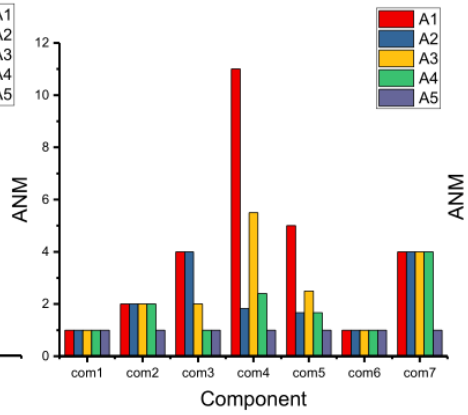
(c) JGraphx NOI



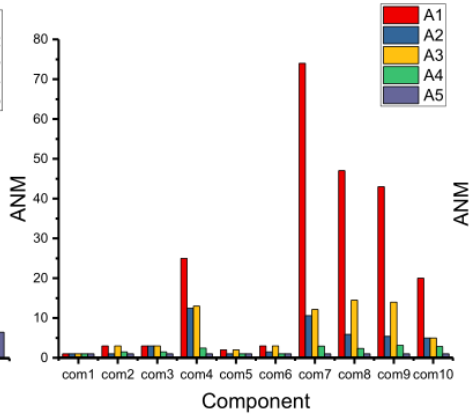
(d) JHotDraw NOI



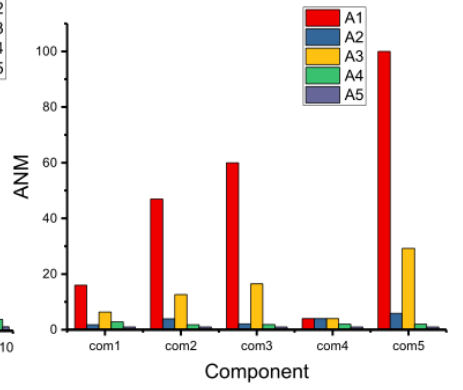
(a) SportLobby ANM



(b) GoogleSheet ANM

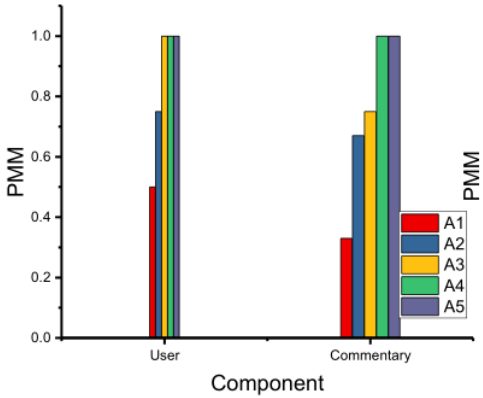


(c) JGraphx ANM

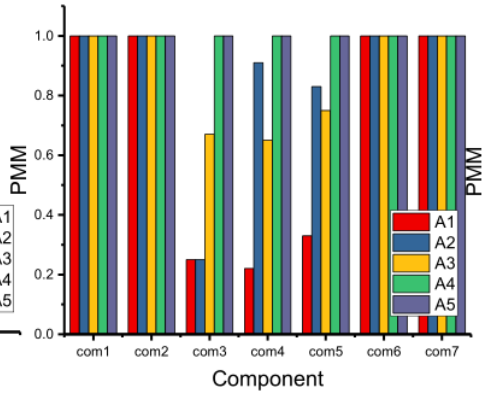


(d) JHotDraw ANM

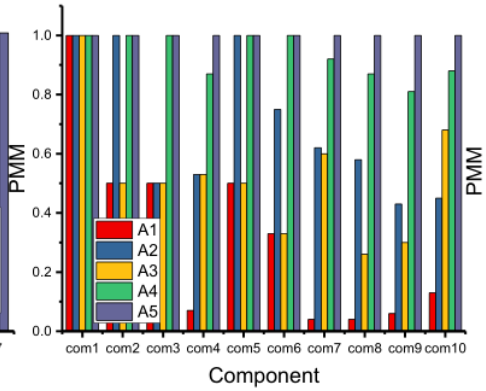
Evaluating different approaches



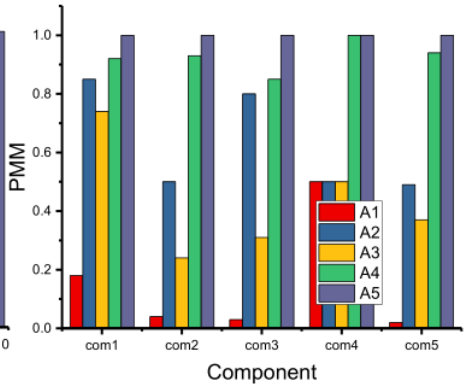
(a) SportLobby PMM



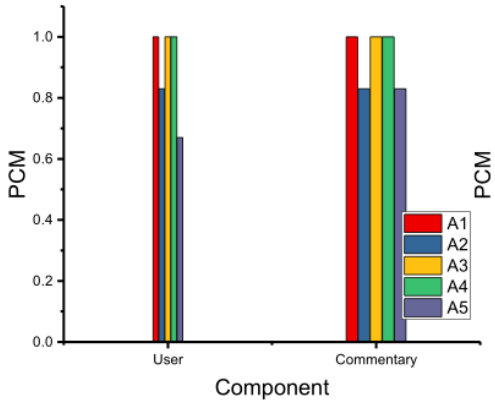
(b) GoogleSheet PMM



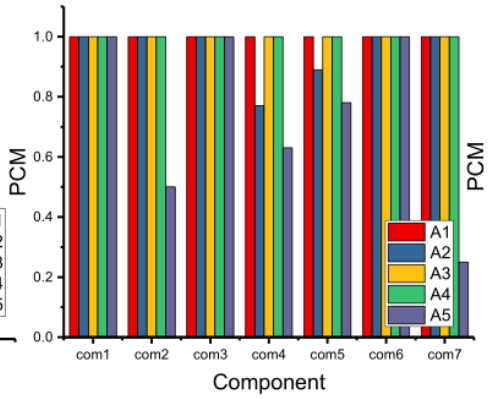
(c) JGraphx PMM



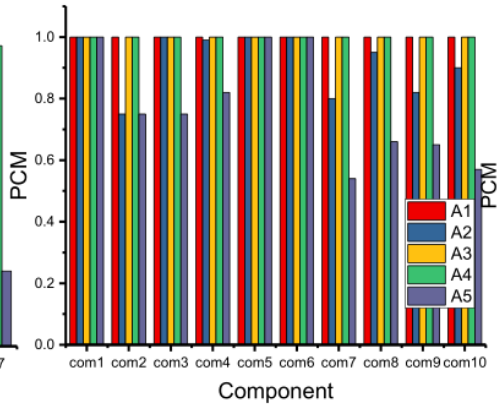
(d) JHotDraw PMM



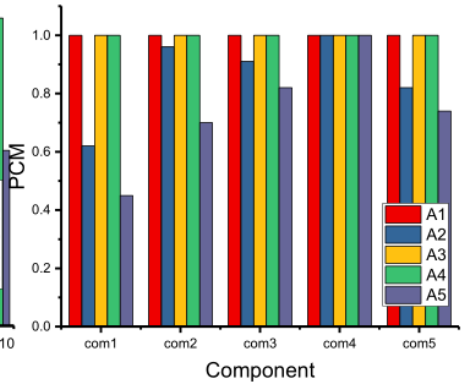
(a) SportLobby PCM



(b) GoogleSheet PCM

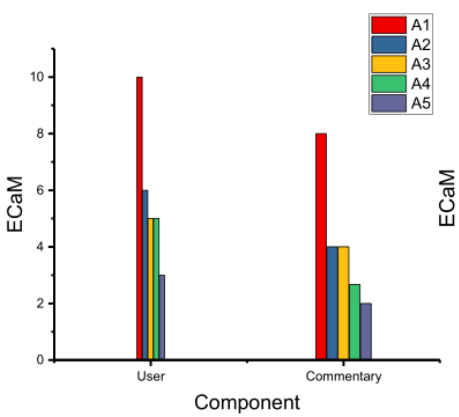


(c) JGraphx PCM

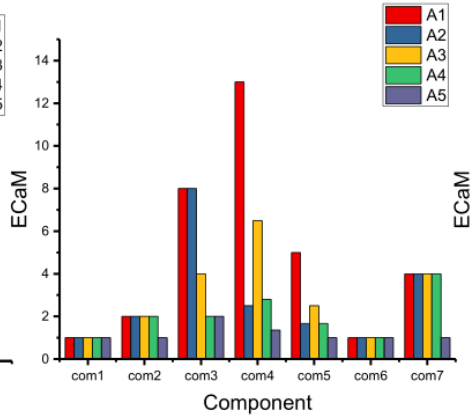


(d) JHotDraw PCM

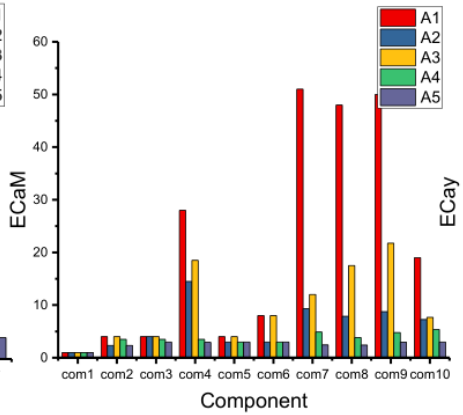
Evaluating different approaches



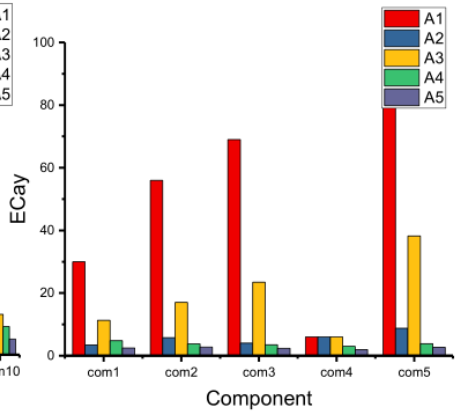
(a) SportLobby ECaM



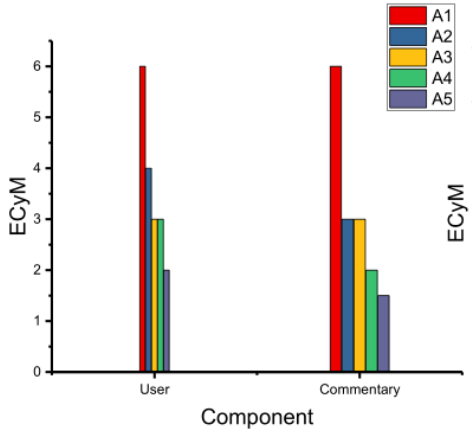
(b) GoogleSheet ECaM



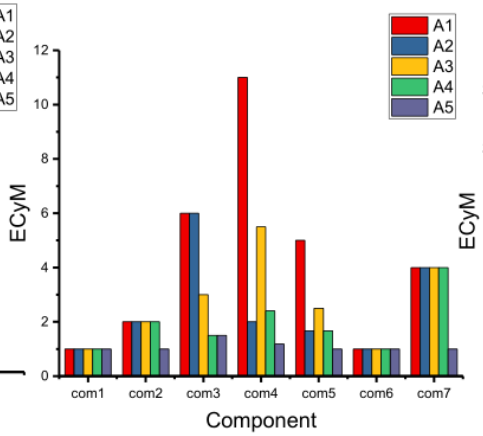
(c) JGraphx ECaM



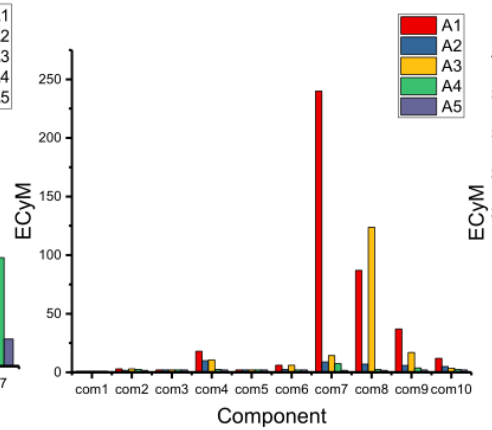
(d) JHotDraw ECaM



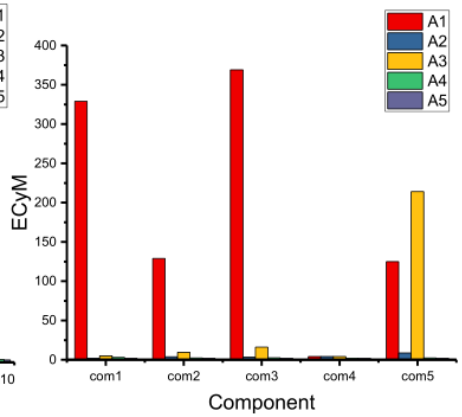
(a) SportLobby ECyM



(b) GoogleSheet ECyM



(c) JGraphx ECyM



(d) JHotDraw ECyM

Comparison results

A1 and A2 are not function-oriented, and they always wrongly group functionally unrelated methods to the same interface, i.e. low PMM, high ECaM and ECyM.

A3 focuses on coarse-grained function notion, group functionally unrelated methods to the same interface.

A5 simply creates one interface for each method will aggressively separate functionally related methods to different interfaces, i.e., low PCM.

A4 can discover interfaces that neither group unrelated methods nor separate functionally related methods.

Outline

- Background
- Research question
- Our approach
- Experimental evaluation
- Conclusion

Conclusion and future

Conclusion:

- Functional oriented approach for interface identification;
- Discovering interface behavioral model.

Future work:

- Discover interaction among component.
- Construct the architectural view with multiple abstractions.

Thank you!