A Comparative Study on Microservice Extraction Techniques

Thank you for joining us!

Interview Study: Relationship Types
Think Aloud Study: PartsUnlimitedMRP

Study introduction

We do not collect any company-sensitive information

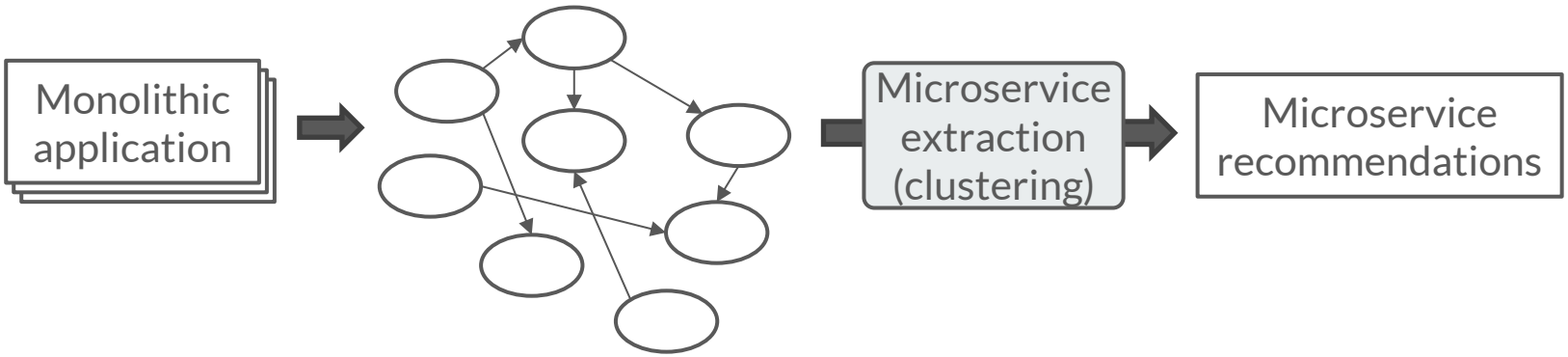
Opt-out policy:
At any point and for any reason

Study topic:
Decomposing monolithic applications into microservices

- Relationship types used for decomposition
- Case study (if time permits)

Decomposition into microservices is difficult

- Automation can help



Graphical representation

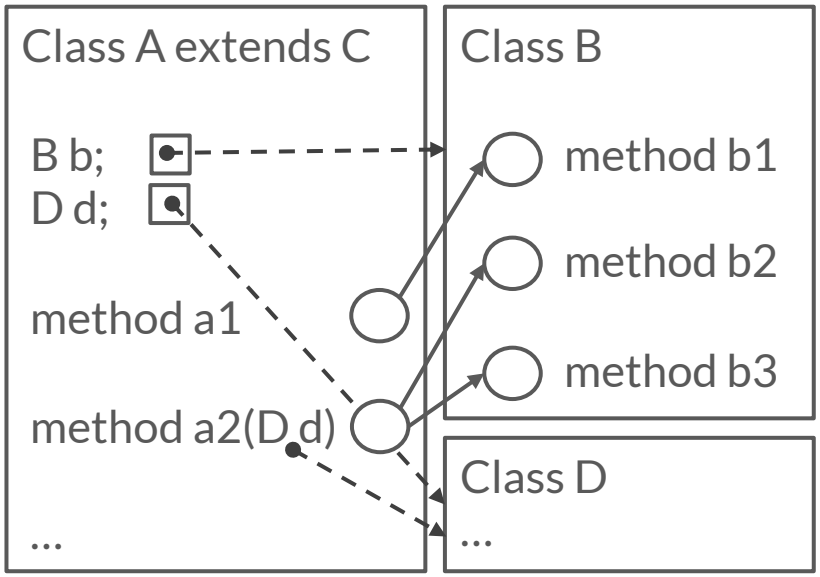
- Automation relies on producing clusters of elements based on their relationships

- We are trying to understand which relationships are useful

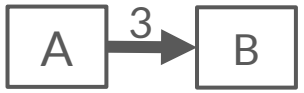
Relationship type 1/3: structural relationships

- Class inheritance

- Method calls and data dependencies

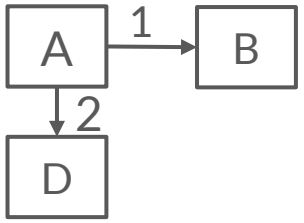


- Method calls



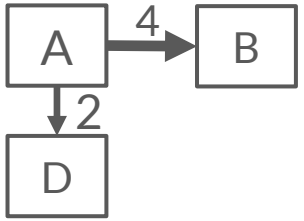
Weighted graph

- Data dependencies



Weighted graph

Combined total



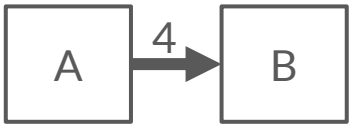
Weighted graph

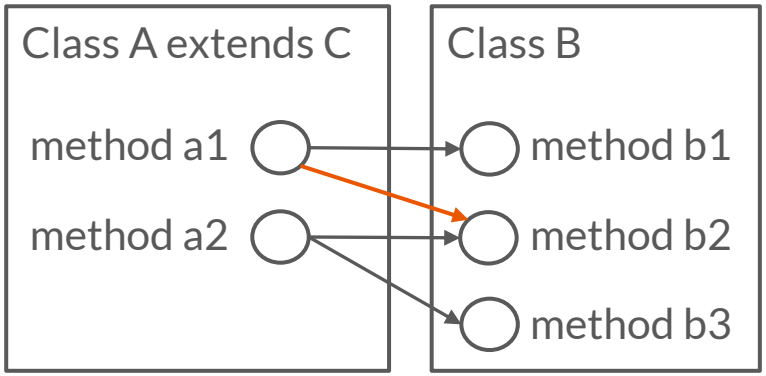
Relationship type 1/3: structural relationships

This information can be collected:

- ***Statically***; using source code
  - Complete, not accurate

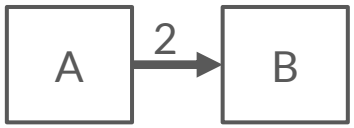
Weighted graph

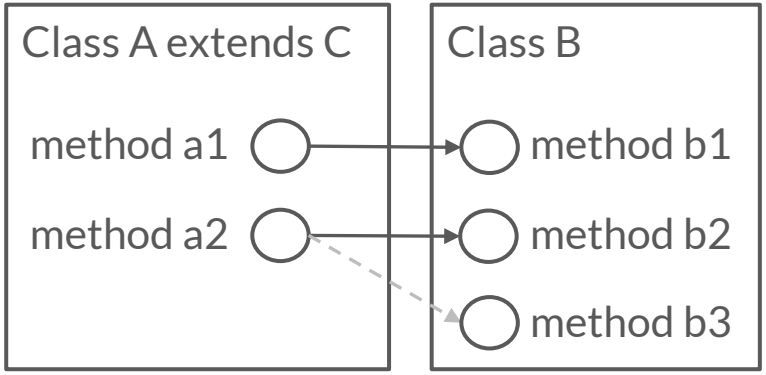




- ***Dynamically***; using execution traces collected at runtime
  - Accurate, not complete

Weighted graph





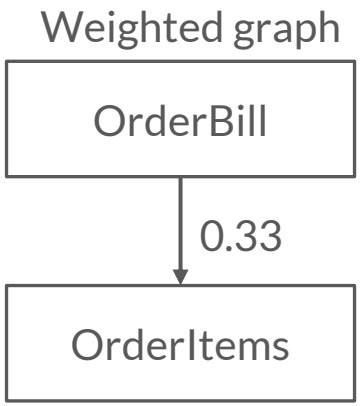
Relationship type 2/3: semantic similarity

Semantic similarity considers the likeness of words used in code

- **Class name similarity:** similarity between names of classes

```
//order summary of a customer
Class OrderBill {
 List<OrderItem> items;
 Date date;
 Customer customer;
 int totalcost;

 //finalize purchase
 purchase(OrderItem item) {…};

 removeItem() {…};
 addItem() {…};

 addCustomer(Customer c) {…}
 changeCustomerDetails(
                Customer c) {…}
 …}
```

```
//item for purchase
Class OrderItem {
 Cost cost;
 Customer customer;
 String description;

 OrderItem(…) {…}

 //getters
 …

 //setters
 …
}
```



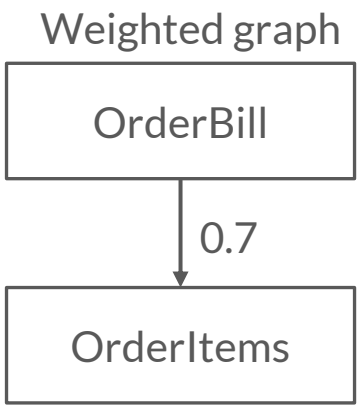
Relationship type 2/3: semantic similarity

Semantic similarity considers the likeness of words used in code

- **Term similarity:** similarity between all terms within classes

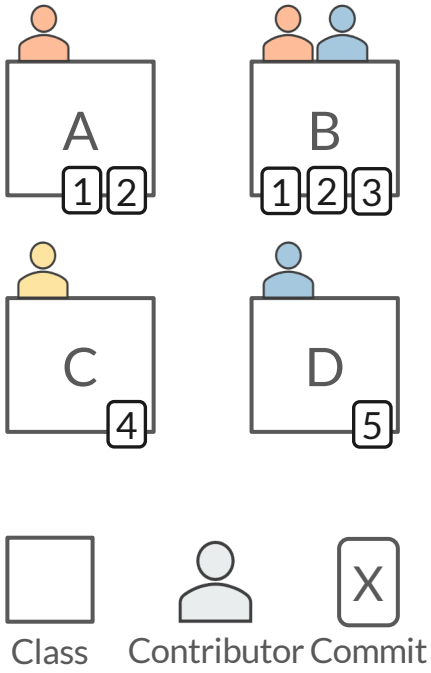
```
//order summary of a customer
Class OrderBill {
 List<OrderItem> items;
 Date date;
 Customer customer;
 int totalcost;

 //finalize purchase
 purchase(OrderItem item) {…};

 removeItem() {…};
 addItem() {…};

 addCustomer(Customer c) {…}
 changeCustomerDetails(
              Customer c) {…}
…}
```

```
//item for purchase
Class OrderItem {
 Cost cost;
 Customer customer;
 String description;

 OrderItem(…) {…}

 //getters
 …

 //setters
 …
}
```

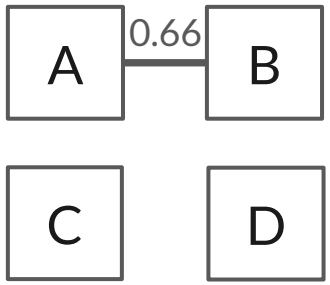


Relationship type 3/3: evolutionary similarity

Evolutionary data considers the evolution of an application



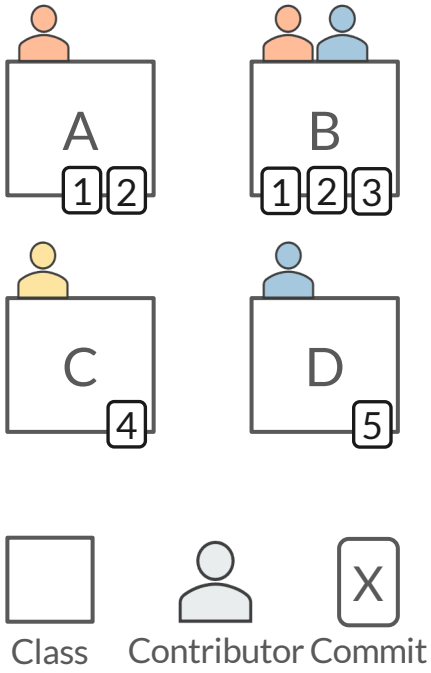
- ***Commit similarity:*** the fraction of commits simultaneously changing both classes out of all commits changing the classes



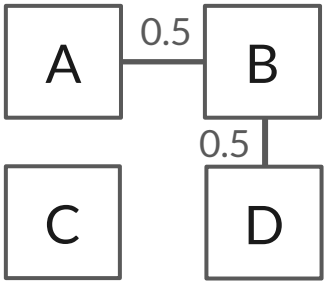
Weighted similarity graph

Relationship type 3/3: evolutionary similarity

Evolutionary data considers the evolution of an application



- ***Contributor similarity:*** the fraction of developers changing both classes out of all developers changing the classes



Weighted similarity graph

Relationship Types Questions

1. Out of 100, how many points would you assign to the importance of each relationship type for identifying clusters of related elements in a monolithic application? Explain.

2. Please further divide the points given to structural relationships and explain.

3. Please further divide the points given to semantic similarity and explain.

4. Please further divide the points given to evolutionary similarity and explain.

5. Are there **additional relationship types** that you find useful? Explain.

6. Would the weightings change if you were to consider a different application? Explain.

A. **Structural relationships**,
   a. Static: class inheritance, method calls, and data dependencies between classes collected by analyzing source code
   b. Dynamic: method calls and data dependencies between classes collected by analyzing runtime execution traces

B. **Semantic relationships,**
   a. Class name similarity: similarity between names of classes
   b. Term similarity: similarity between all terms (method names, parameters, comments, etc.) within classes

C. **Evolutionary relationships,**
   a. Commit similarity: the frequency of classes being changed in the same commit
   b. Contributor similarity: the frequency of classes being changed by the same developer

PartsUnlimitedMRP

Benchmark application

PartsUnlimitedMRP

Web application that uses java (backend), javascript (frontend), mongodb (database)

Parts Unlimited MRP is a fictional Manufacturing Resource Planning (MRP) application

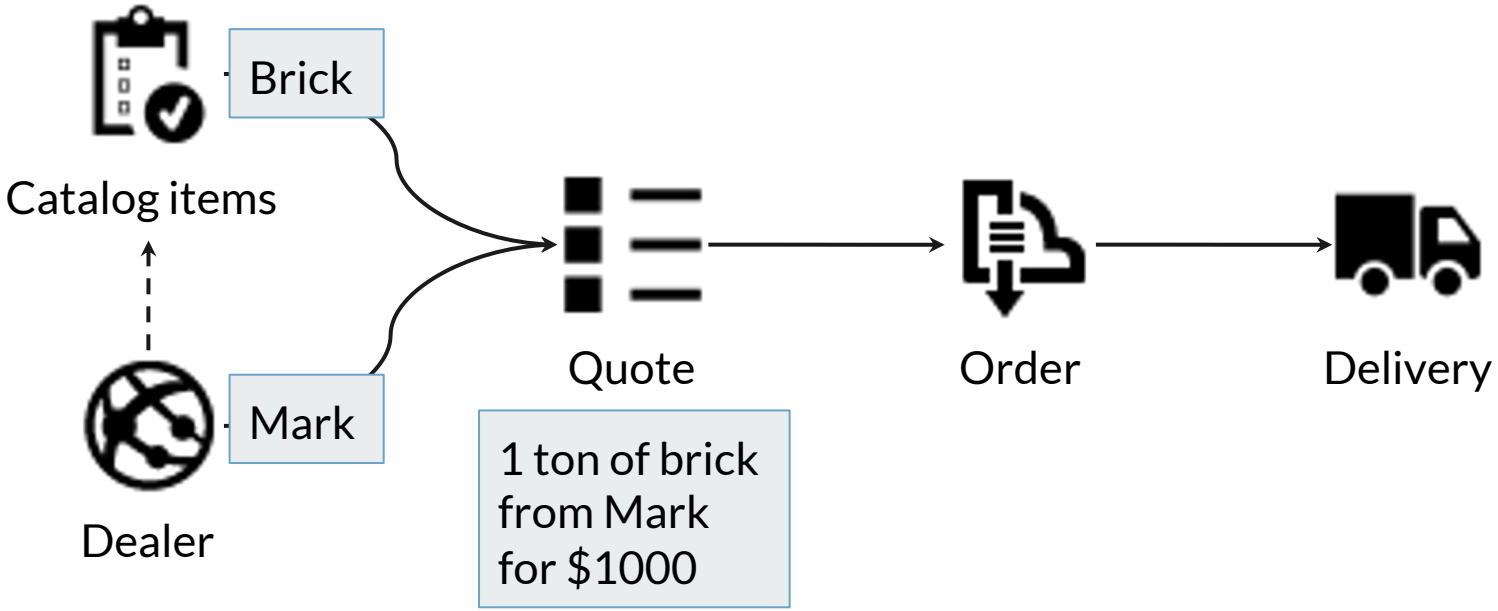
It is an inventory control system

PartsUnlimitedMRP - example

You manage a construction company that builds brick houses

To build, you need resources/supplies from external suppliers

PartsUnlimitedMRP helps you keep track of your supplies



Our Clustering Framework GUI

Thank you for participating!

We will use your inputs in our framework