



Research on Identification and Refactoring Approach of Event-driven Architecture Based on Ontology

Li WANG, Xiang-long KONG, Xiao-fei WANG, Bi-xin LI

2022-07-01

Contents

01 Introduction

02 Related Work

03 Approach

04 Experiments and Results Analysis

05 Conclusion



01
PART ONE

Introduction

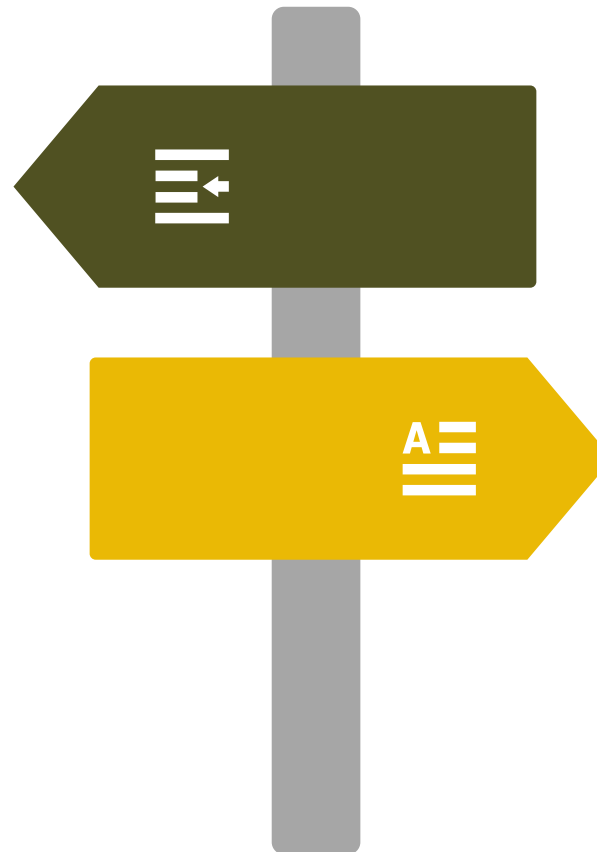
Introduction

Problems

In the process of software evolution, the deviation and corrosion often occur to architecture, which leads to larger deviation between actual software architecture and design architecture.



There're many factors may make the software architecture deviate from the original design, such as the change of requirements, the improvement of functions, et al.



Solutions

Identification and Refactoring
Approach of Event-driven Architecture Based on Ontology



02
PART TWO

Related Work

Related Work

Related work



Architecture can be represented by logic and architecture style can be described by configuration



Use logical predicates to model anti-patterns, and build an engine based on these logical predicates to detect anti-patterns in the target system



a method of knowledge retrieval to identify instances of architecture patterns in software systems



an automatic analysis architecture model based on knowledge representation and information extraction, and then reconstructed the system according to the analysis results

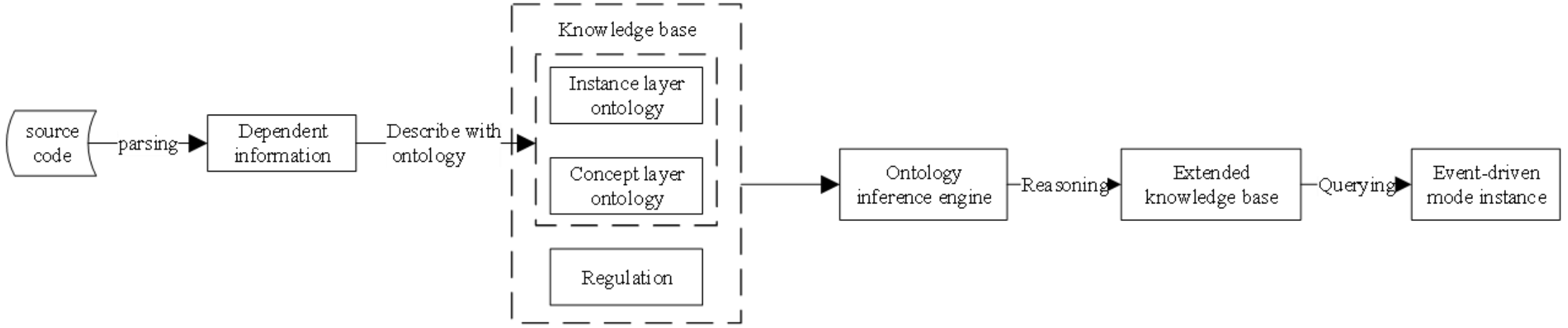
.....



03
PART THREE

Approach

Identifying Event-driven Architecture Based on Ontology



STEP 1

Construction instance
layer ontology



STEP 2

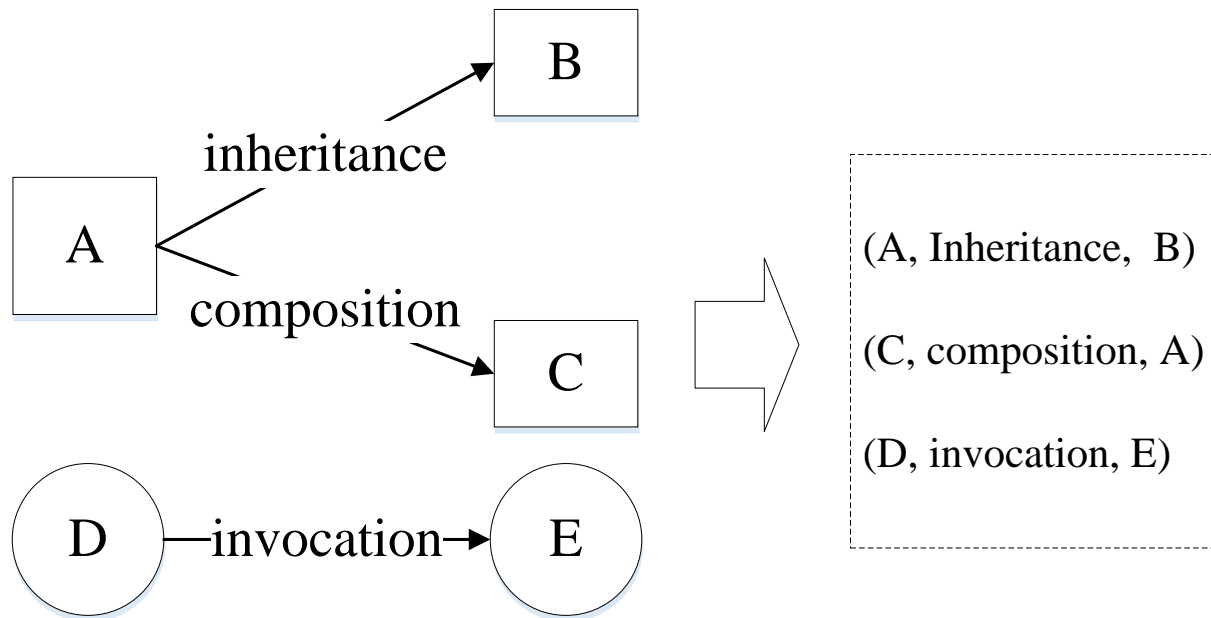
Construction concept
layer ontology



STEP 3

Reasoning and inquiry

STEP1: Construction instance layer ontology

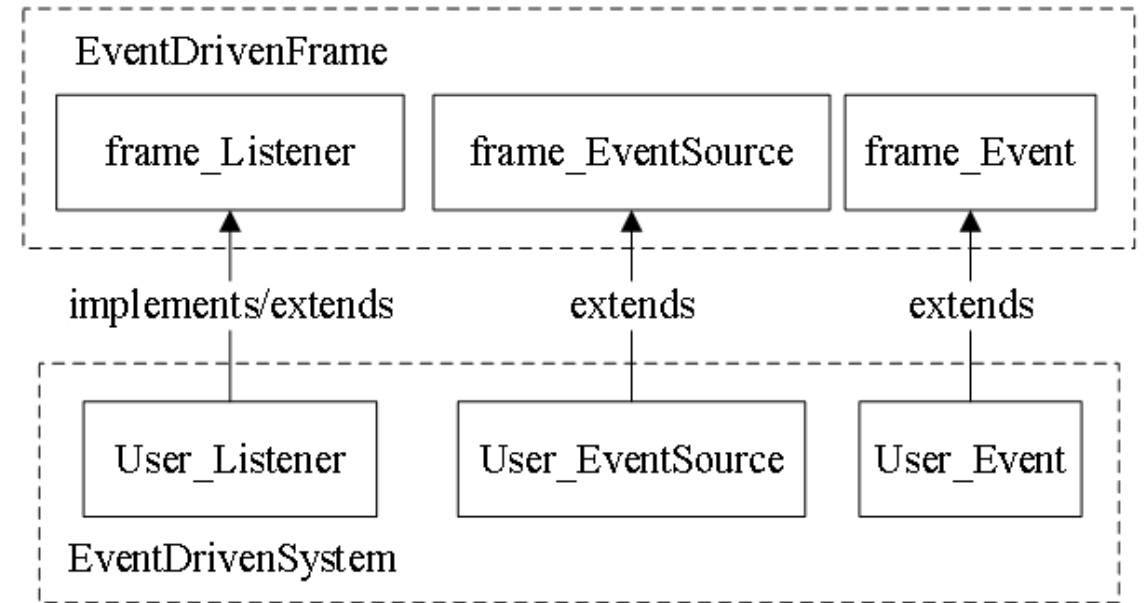
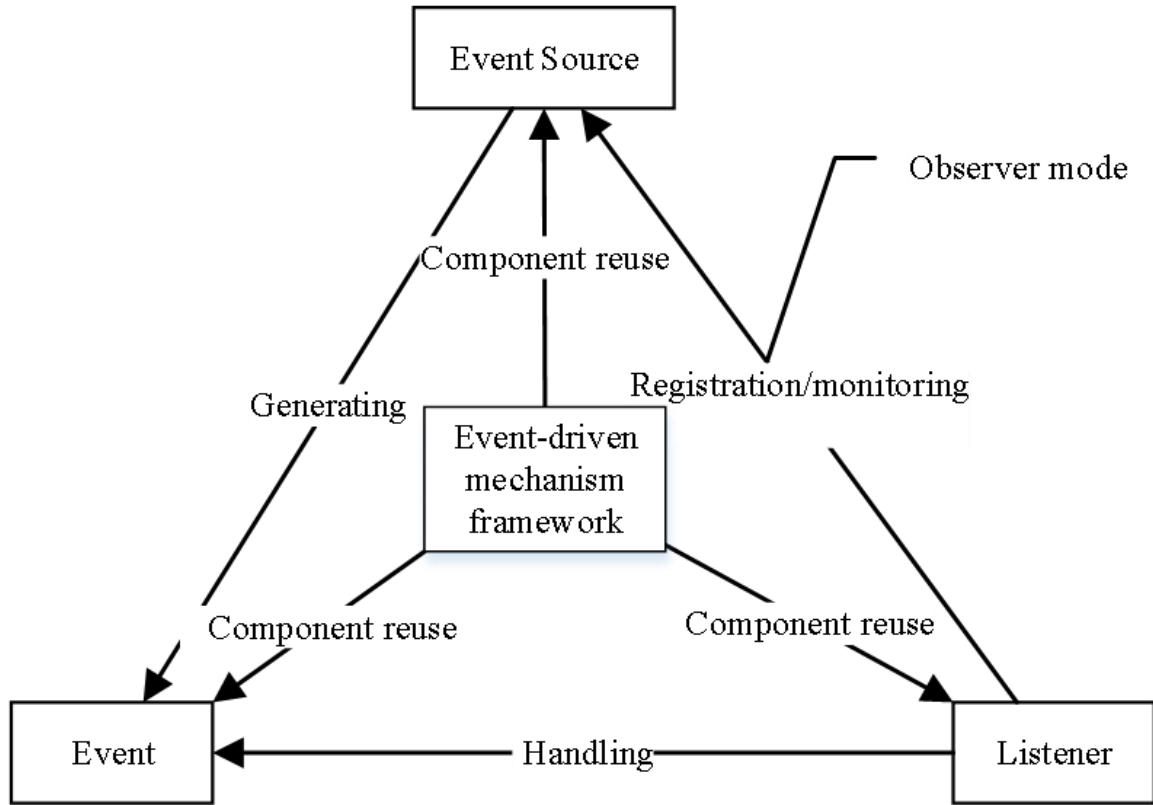



Construction instance

layer ontology

- We extract the dependency information by traversing the ADT to build a dependency graph.
- the node represents the program entities, and the directed edge represents the dependency between program entities.
- We convert the nodes and directed edges into RDF triples set.

STEP2: Construction concept layer ontology



 Behavior characteristics of event-driven architecture

 Event-driven pattern framework reuses behavior

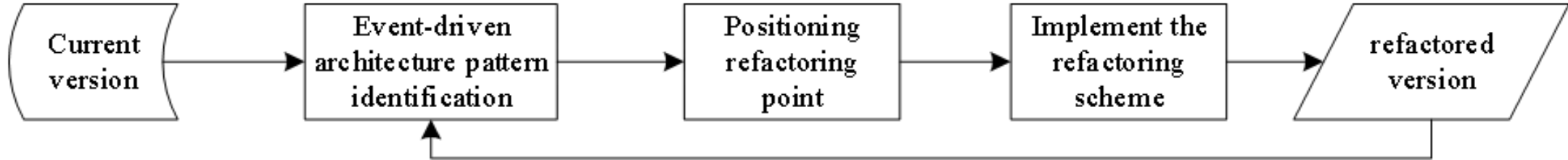
STEP3: Reasoning and inquiry



```
Procedure AcqEventDrivenInstance(){
    OntModel ontModel =ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);
        ontModel.read("file:/ architecture_pattern/java/EventDriven.owl");
        ontModel.read("file:/ architecture_pattern/java/Instance.rdf");
        ontModel.rulereason(EventDriven.rules)

    String queryString = "PREFIX EV: http://www.semanticweb.org/wang/EventDriven.owl# SELECT ?Listener
WHERE {? listener rdf:type EV: Listener }";
    Query query = QueryFactory.create(queryString);
    QueryExecution queryExecution = QueryExecutionFactory.create(query, ontModel);
    ResultSet resultSet = queryExecution.execSelect();
    ResultSetFormatter.out(System.out, resultSet, query);
    queryExecution.close()
}
```

Refactor event-driven architecture



STEP 1

Locate the
refactoring point



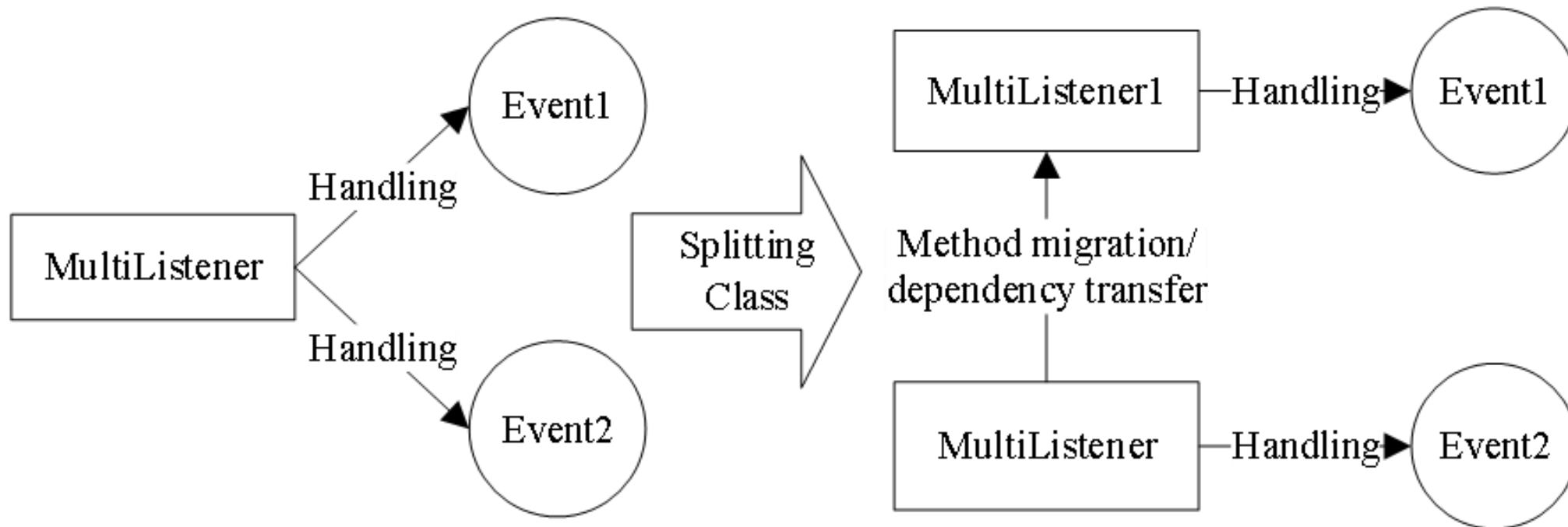
STEP 2

Implement the scheme
for the refactoring points

Refactor event-driven architecture



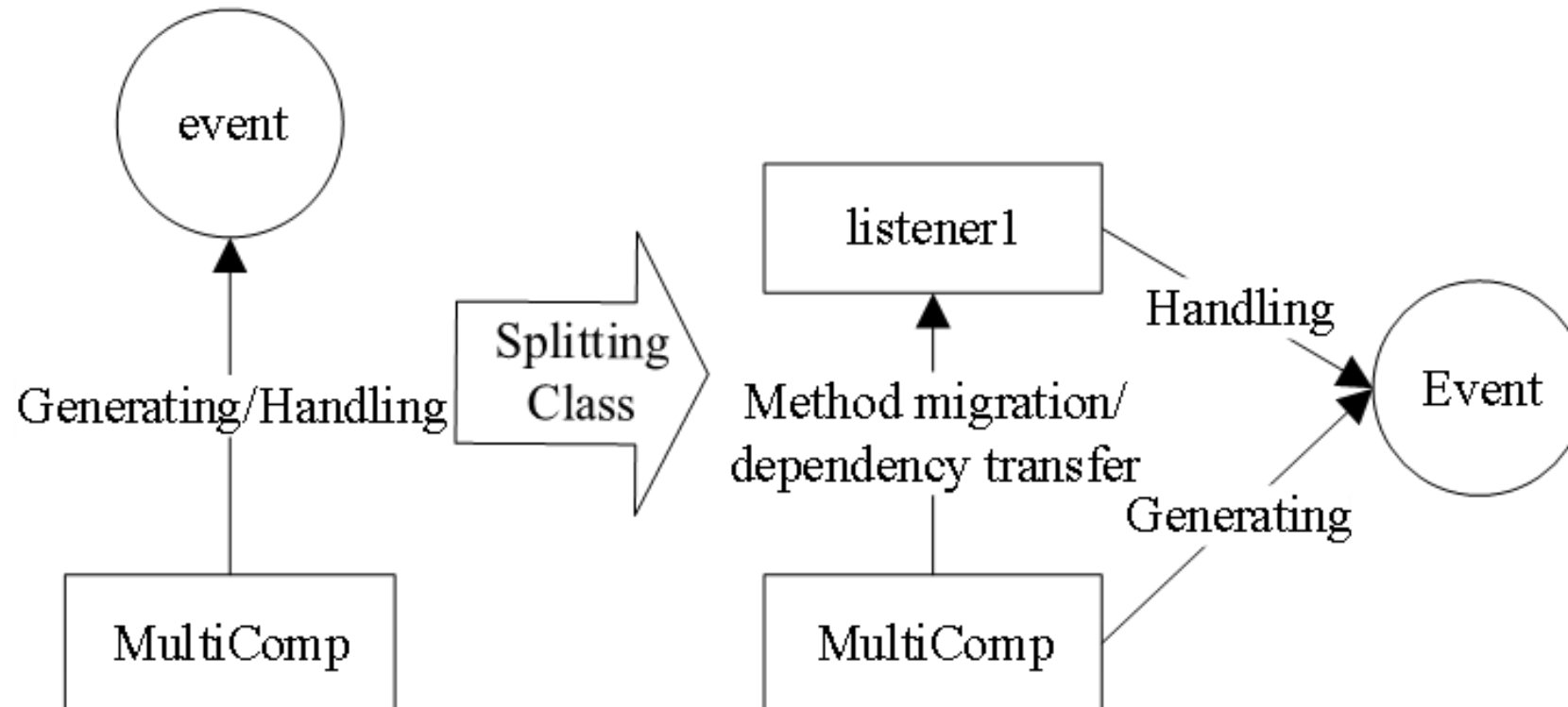
RS 1: The refactoring scheme for the single responsibility of the listener.



Refactor event-driven architecture



RS 2: The refactoring scheme for the distributed processing.



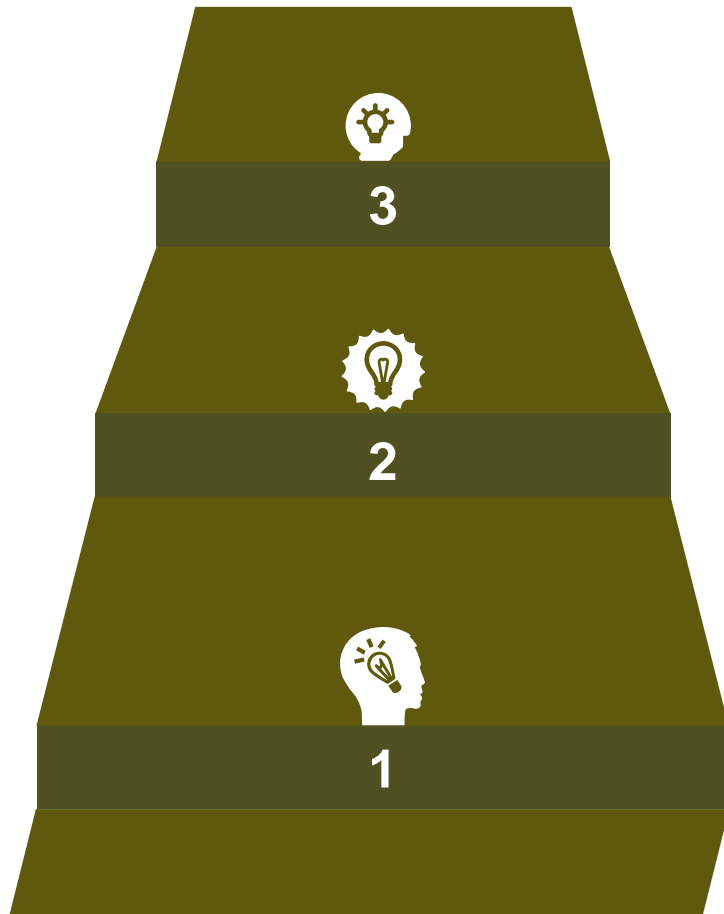


04
PART FOUR

Experiments and Results Analysis



Experiments and Results Analysis



RQ1: How about the accuracy of the software architecture identification technique?



RQ2: How about the accuracy of the software architecture refactoring technique?



RQ3: How about the efficiency of the software architecture refactoring technique?

Experimental Steps



Collect 50 Java projects from GitHub and SourceForge with the key words.



Identify their architecture pattern manually to confirm whether they are event-driven architecture project.



We obtain their ground-truth architecture manually to build the comparative experiments.



We refactor their architecture base on ontology, and we collect all the results to analyze the accuracy and effectiveness.



We obtain the refactoring points and process the refactoring .



05
PART FIVE

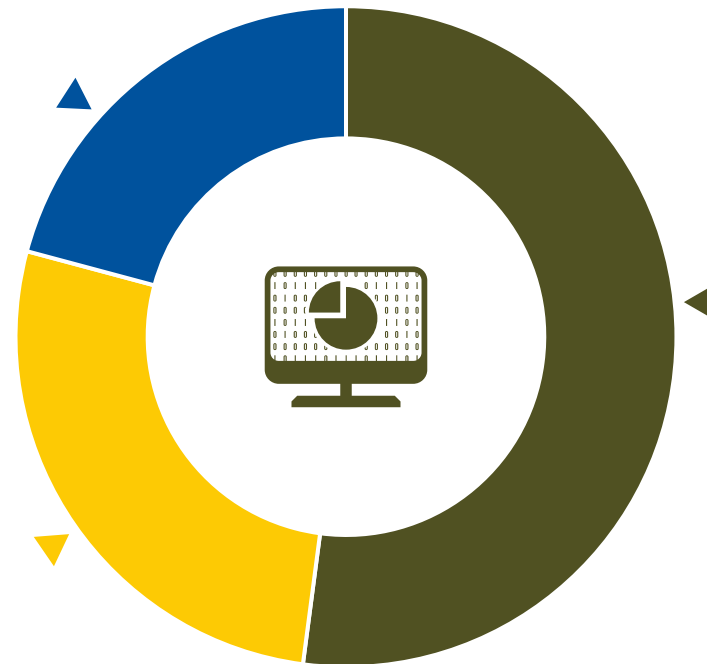
Conclusion

Conclusion

The recognition result of our approach has a high coincidence degree with the ground-truth architecture.

It's highly effective for typical event-driven architecture projects with high document quality.

Novel Contributions



The refactoring schemes had a significant effect on reducing violations and improving the quality of the software.



Thanks for your listening!

Q & A