

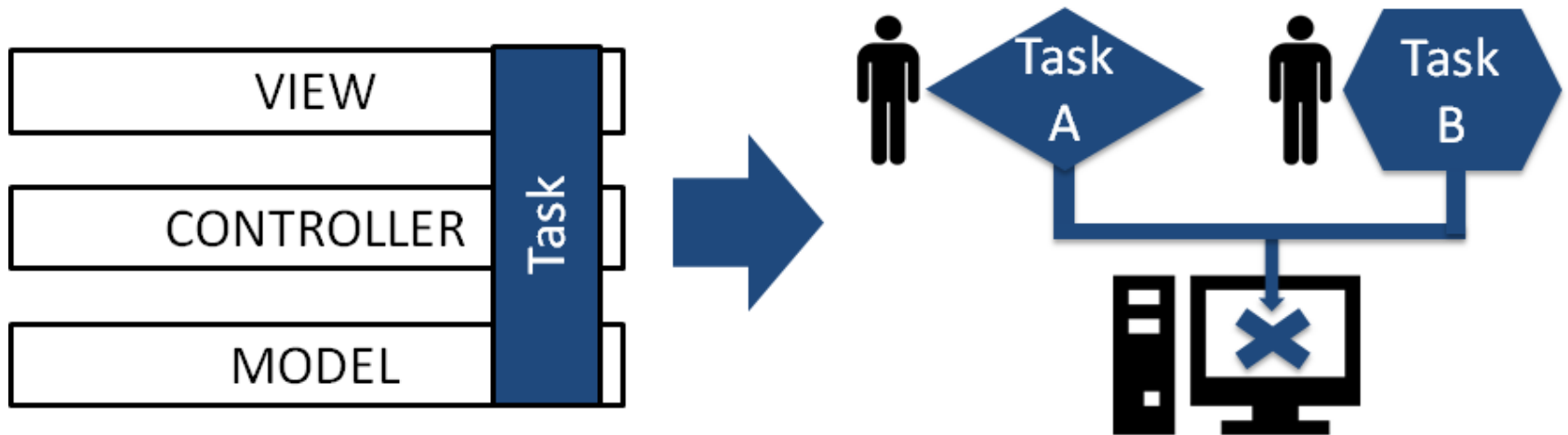
# Comparing **integration effort** and **correctness** of different merge approaches in Version Control Systems

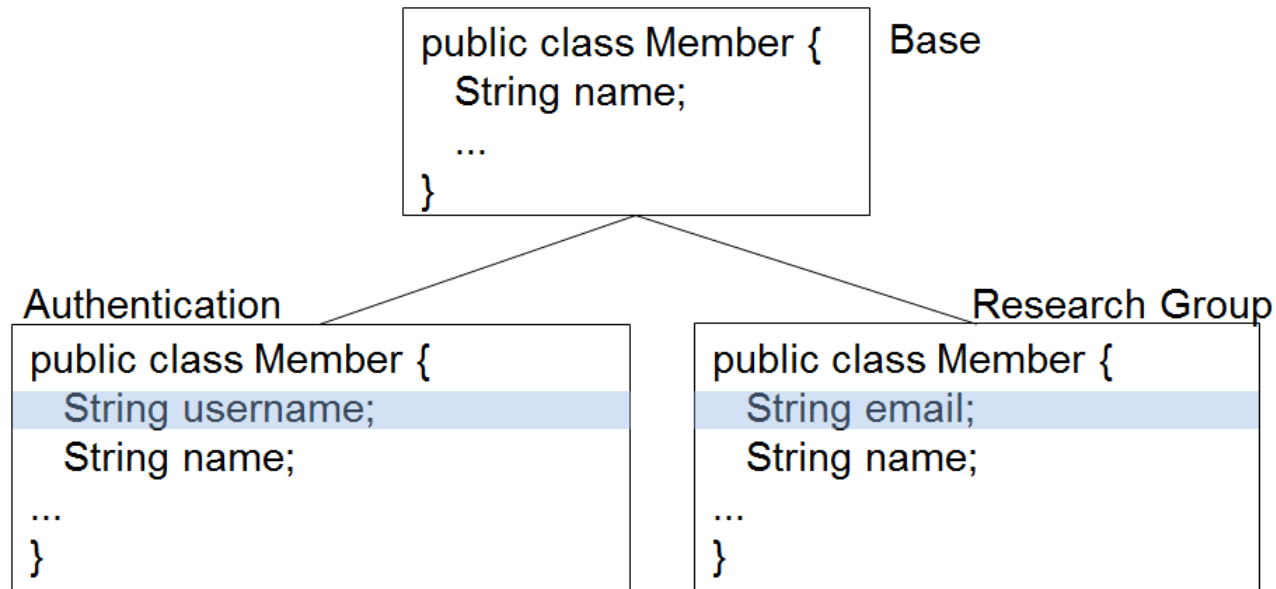
Guilherme Cavalcanti

Advisor: Paulo Borba



Collaborative development is a common characteristic of today's software projects





Code  
integration  
(three-way merge)

```
public class Member {  
    <<<<<<<  
        String email;  
    =====  
        String username;  
    >>>>>>>  
        String name;  
    ...  
}
```

Merge  
conflict

#### Mining Workspace Updates in CVS

Thomas Zimmermann

23% to 46% of code integration leads to merge conflicts

Zimmermann 03

#### Proactive Detection of Collaboration Conflicts

16% of merges resulted in conflicts.  
33% of clean merges resulted in build and test conflicts.

Brun et al 11

#### Cassandra: Proactive Conflict Minimization through Optimized Task Scheduling

Merge conflicts ranged from 7% to 19%; build conflicts from 2% to 15%; and test conflicts from 6% to 35%

Kasi and Sarma 13

#### Assessing the Value of Branches with What-if Analysis

Christian Bird

Thomas Zimmermann

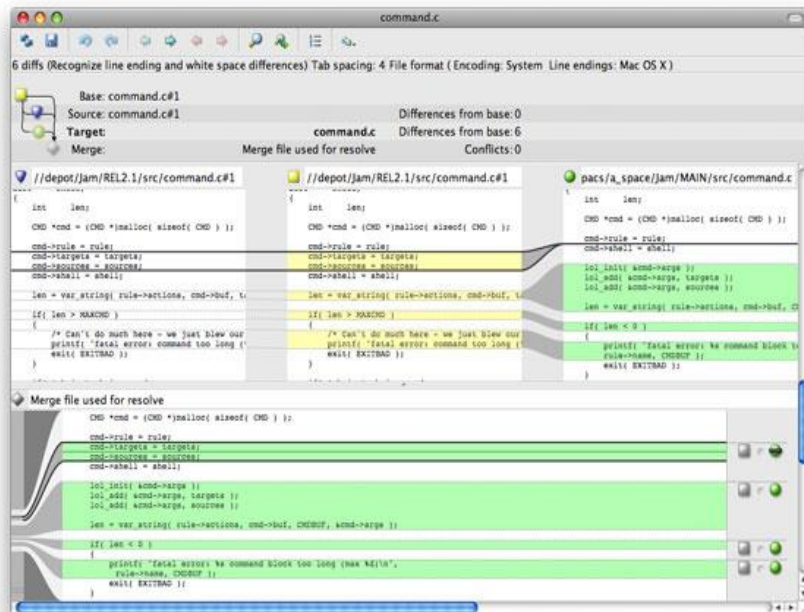
Developers spend significant time dealing with conflicts

Bird and Zimmermann 11

Conflicts are frequent and time-consuming!

# Merge Approaches

*unstructured merge*



*structured merge*

JAVA

GRAMMAR

$stmt \rightarrow identifier = exp$

$stmt \rightarrow return\ exp$

$stmt \rightarrow if\ exp\ compoundstmt$

$stmt \rightarrow if\ exp\ compoundstmt\ else\ compoundstmt$

$compoundstmt \rightarrow \{ stmt\ S \}$

$stmt\ S \rightarrow stmt\ ;\ stmt\ S$

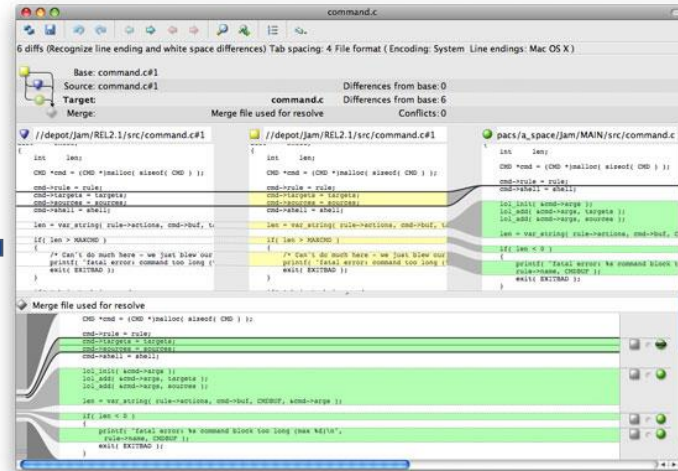
$stmt\ S \rightarrow \epsilon$

# semistructured merge

JAVA

GRAMMAR

stmt  $\rightarrow$  identifier = exp  
stmt  $\rightarrow$  return exp  
stmt  $\rightarrow$  if exp compoundstmt  
stmt  $\rightarrow$  if exp compoundstmt else compoundstmt  
compoundstmt  $\rightarrow$  { stmtS }  
stmtS  $\rightarrow$  stmt ; stmtS  
stmtS  $\rightarrow$   $\epsilon$



Conflict Handlers



## Semistructured Merge: Rethinking Merge in Revision Control Systems

Sven Apel, Jörg Liebig,  
Benjamin Brandl, Christian Longauer  
University of Passau, Germany

Christian Kästner  
Philipps University Marburg, Germany

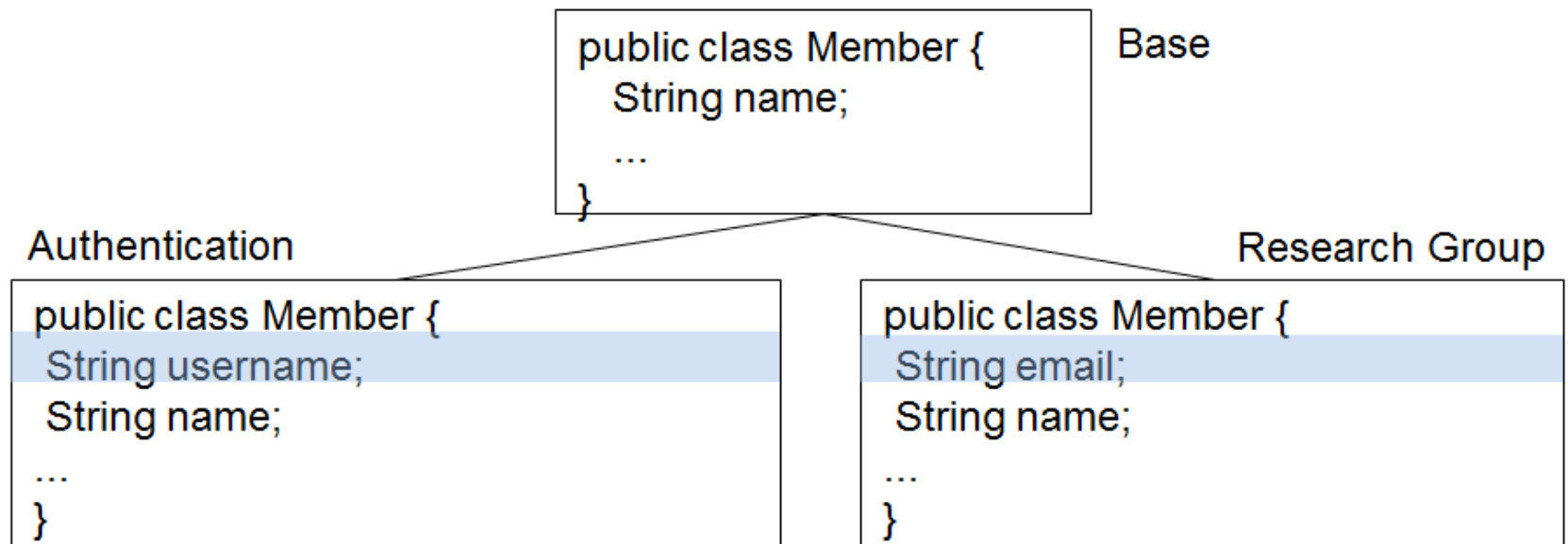
### ABSTRACT

An ongoing problem in revision control systems is how to resolve conflicts in a merge of independently developed revisions. Un-

### 1. INTRODUCTION

Revision control systems (a.k.a. version control systems) have a long tradition in software engineering and are a major means for managing versions and contents of today's software systems. [\[1\]](#)

Apel et al. 11



```
SEMI OR STRUCTURED MERGE
public class Member {
String username;
String email;
String name;
...
}
```

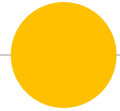
```
UNSTRUCTURED MERGE
public class Member {
<<<<<<<
String username;
=====
String email;
>>>>>>>
String name;
...
}
```

Merge result

# To understand impact on productivity and quality...

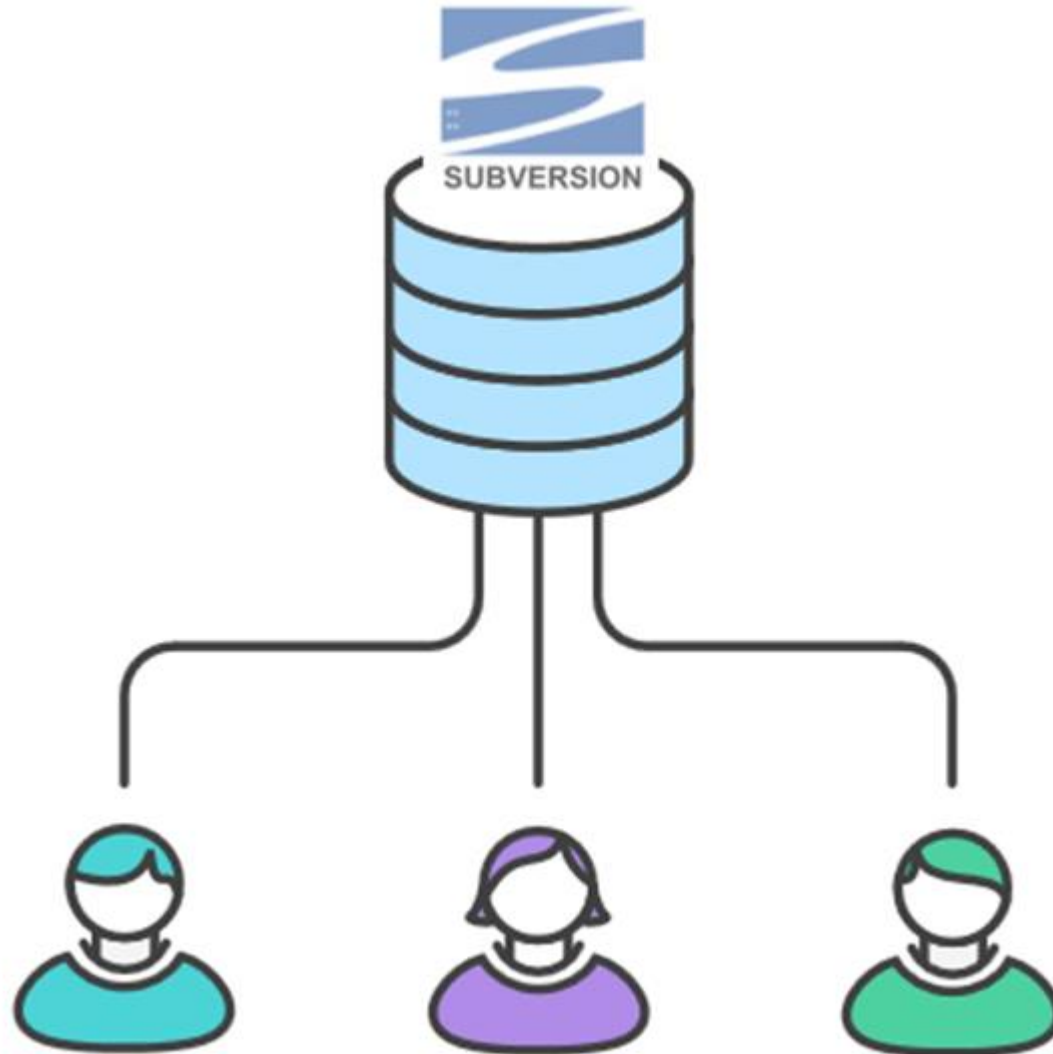
- We compare the number of reported conflicts by the semistructured and unstructured merge approaches
- We compare the number of false positives and false negatives resulting from these merge approaches





# Replication Study

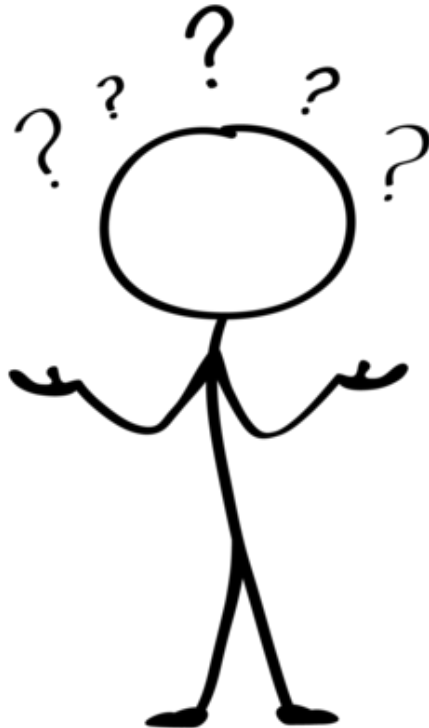
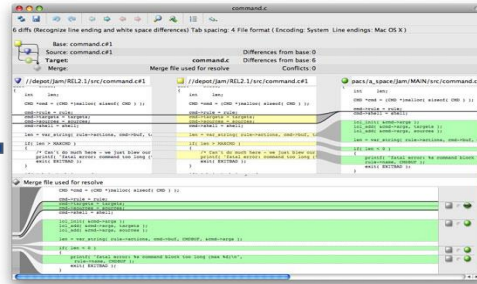
Apel et al. evaluated *semistructured merge* on **180** merge scenarios from **24** projects that use Subversion, a CVCS



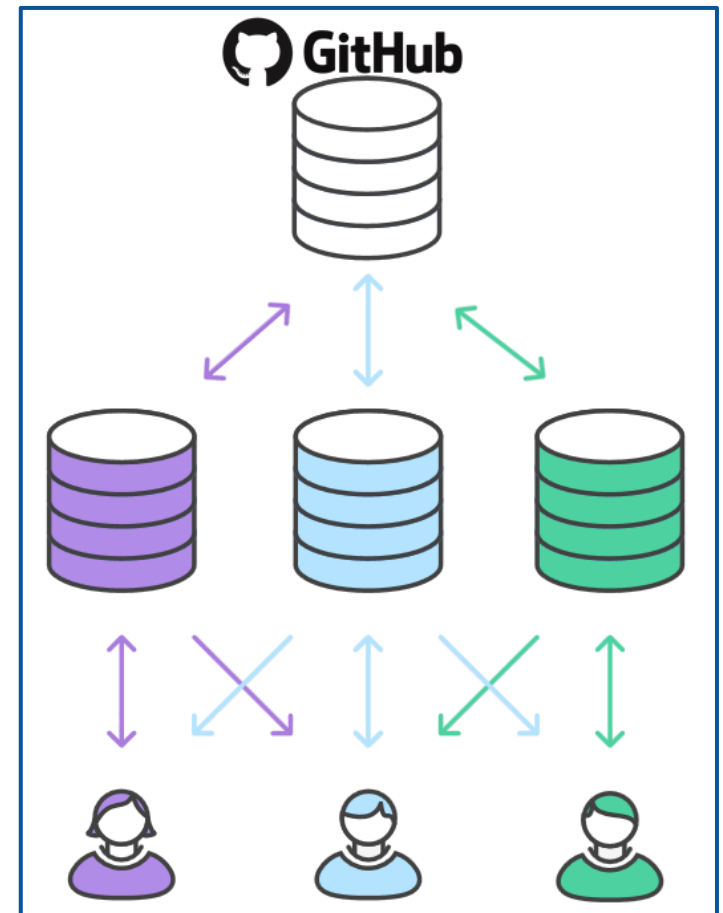
# semistructured merge

## JAVASCRIPT GRAMMAR

```
stmt → ident:por = exp  
stmt → return exp  
stmt → if exp compoundstmt  
stmt → if exp compoundstmt else compoundstmt  
compoundstmt → { stmtS }  
stmtS → stmt ; stmtS  
stmtS → ε
```



DVCS



# *unstructured merge*

## Cassandra: Proactive Conflict Minimization through Optimized Task Scheduling

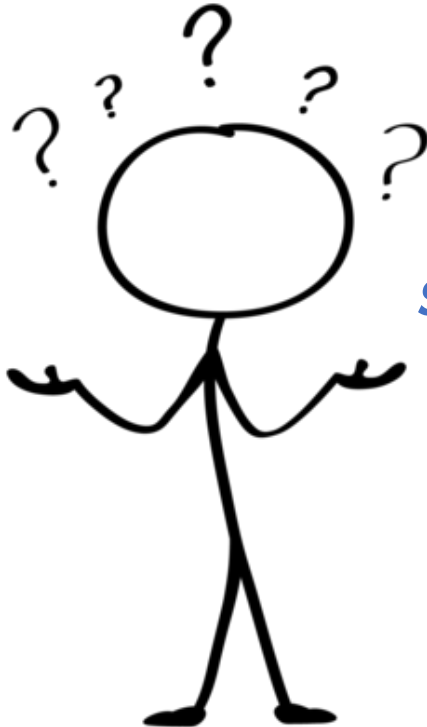
Merge conflicts ranged from 7% to 19%; build conflicts from 2% to 15%; and test conflicts from 6% to 35%

Kasi and Sarma 13

## Proactive Detection of Collaboration Conflicts

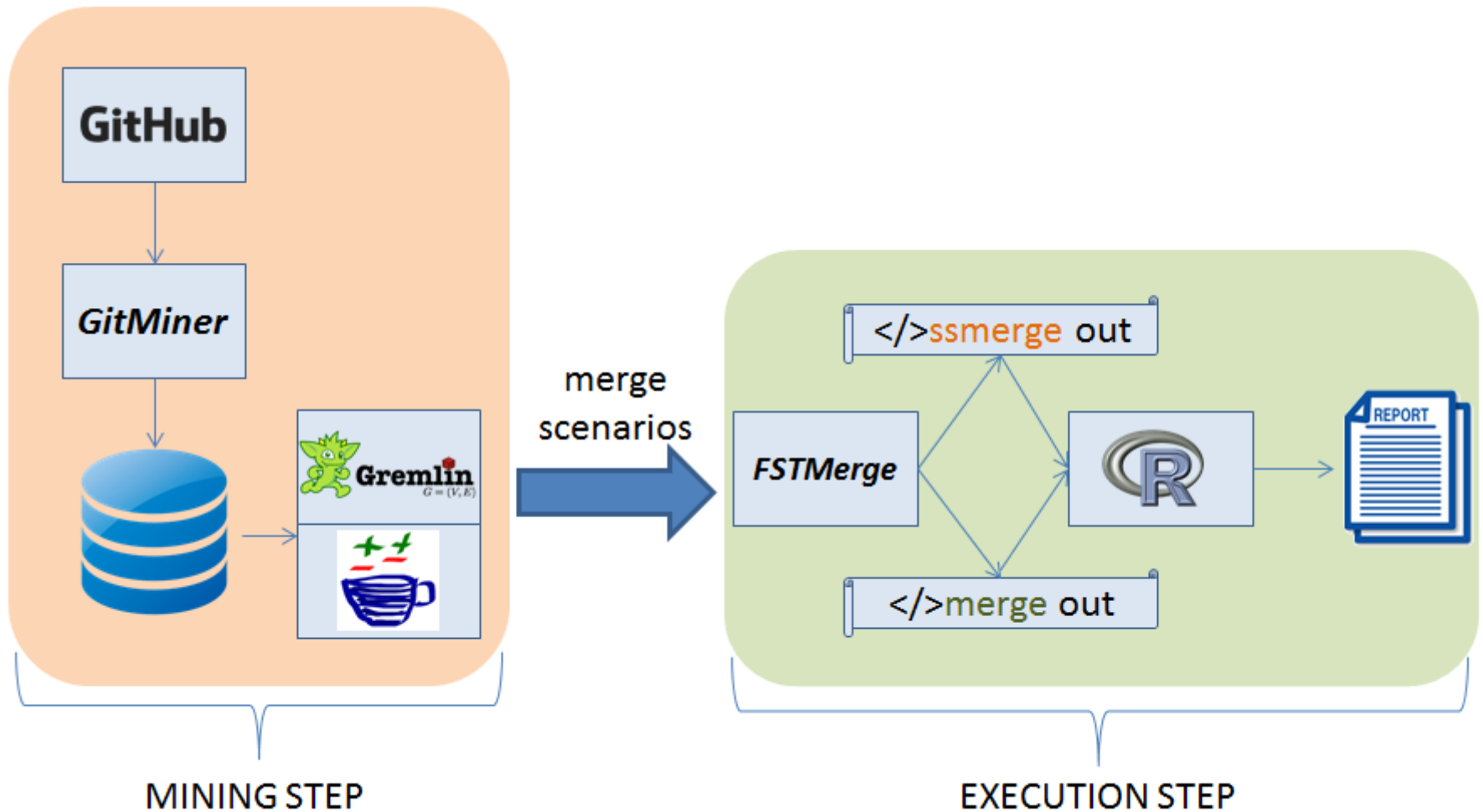
16% of merges resulted in conflicts.  
33% of clean merges resulted in build and test conflicts.

Brun et al 11



*semistructured merge*

# Replication Design



# Mining Step

## Project selection criteria

1. Frequency and Recency of the collaborators activities
2. Number of commits
3. Number of collaborators

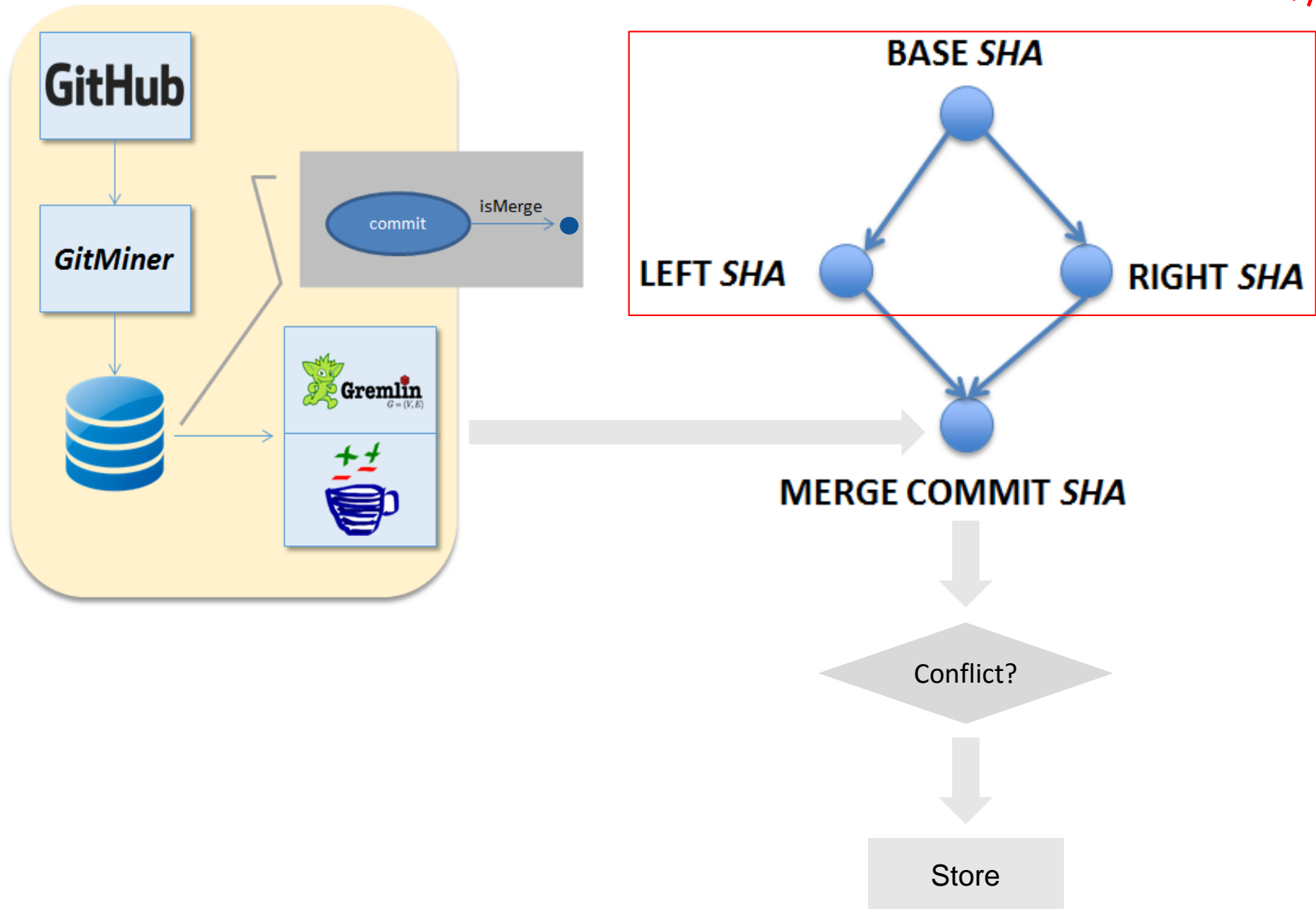
The screenshot shows the GitHub repository page for Django. The repository name is 'django / django'. The page includes buttons for 'Watch' (876), 'Star' (16,110), and 'Fork' (6,489). The description is 'The Web framework for perfectionists with deadlines. <https://www.djangoproject.com/>'. The repository statistics are: 21,430 commits, 41 branches, 115 releases, and 988 contributors. The page also shows a 'Code' button and a 'Pull requests' button. The branch is set to 'master'.

1. Star 16,110

2. 21,430 commits

3. 988 contributors

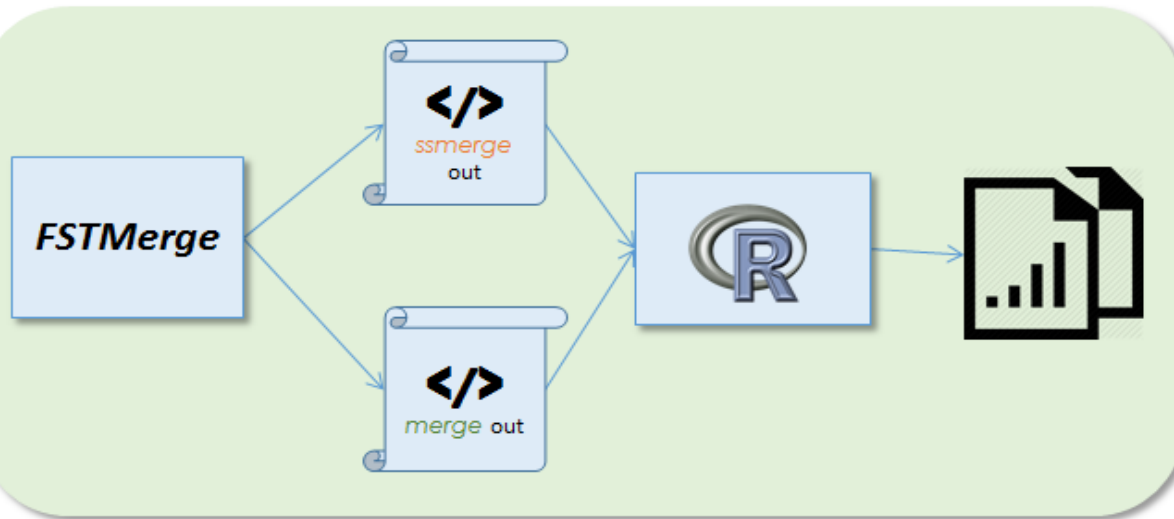
merge scenario



[illegible]



# Execution Step



METRICS

```
public class Member {
```

```
    String email;  
    String username;  
    String name;
```

```
<<<<<<<
```

```
    public String ToString() {  
        return this.name + email;  
    }
```

```
=====
```

```
    public String ToString() {  
        return this.name + username;  
    }
```

```
>>>>>>>
```

```
}
```

- Textual conflict
- Conflicting LOC
- Conflicting file



Conflict Handlers

# Evaluation Results

---

Overall results:

Hypothesis ✓

Total <i>unstructured merge</i> conflicts	18021
Total <i>semistructured merge</i> conflicts	14320

- At least **3.7K** conflicts are ordering conflicts
- Number of *semantic* conflicts found: **1.5K**

# In Merge Scenarios where Semistructured Merge Reduced the Numbers (average)

- Original study (180 merge scenarios, 24 projects):

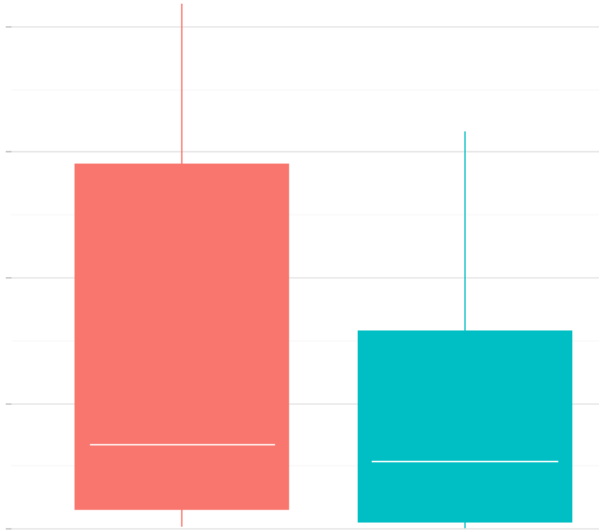
	Reduction by	In Merge Scenarios	SD
Textual conf.	34%	60%	21%
Conf. LOC	61%	82%	22%
Conf. files	28%	72%	12%

- Our replication (3266 merge scenarios, 60 projects):

	Reduction by	In Merge Scenarios	SD
Textual conf.	62%	55.2%	24%
Conf. LOC	81%	71.1%	14%
Conf. files	66%	47.9%	25%

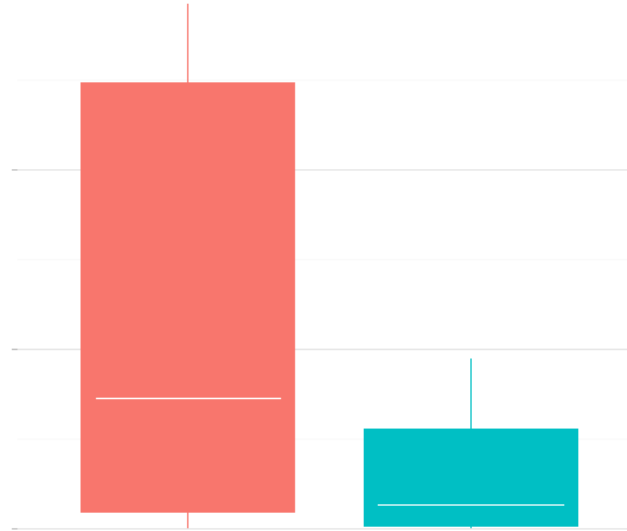
## Conflicts

Unstructured Semistructured



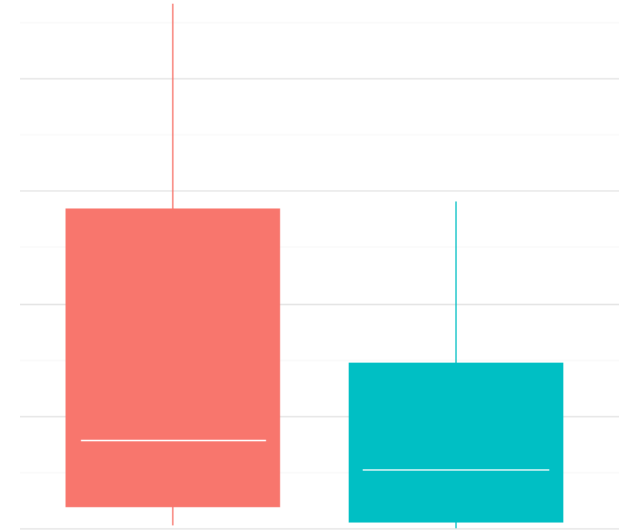
## Conflicting LOC

Unstructured Semistructured



## Conflicting Files

Unstructured Semistructured



**p-value < 0.05**

Wilcoxon-Signed Rank Test  
There is significant reduction!

## Comparison with previous studies

Author	Unstructured Merge Scenario Conflict Rate	Semistructured Merge Scenario Conflict Rate
Brun et al.'11	16%	-
Kasi and Sarma'13	7% - 16%	-
<b>Our replication</b>	<b>7%</b>	<b>3%</b>



**New evidence!**

# Semistructured merge reduces the metrics

```
...
protected void validateFields(List<Throwable> errors) {
<<<<<<<
    for (FrameworkField each : ruleFields())
        validateInterceptorField(each.getField(), errors);
=====
    for (FrameworkField each : ruleFields())
        validateRuleField(each.getField(), errors);
>>>>>>>
}
...
```

*semistructured merge* conflict  
(from project JUnit)

**Conflicting code legibility**

```

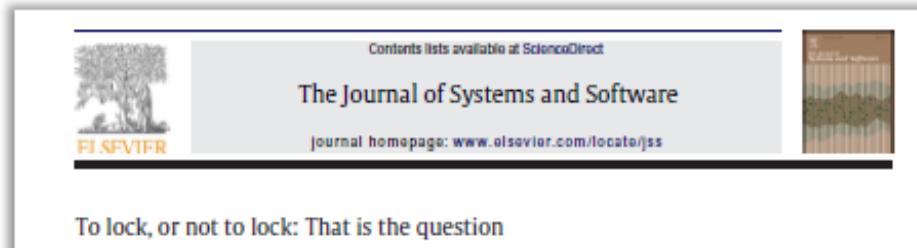
...
<<<<<<<
public static void validateMem...
=====
public ColumnDefinition getColu...

public ColumnDefinition getCol...
>>>>>>>
<<<<<<<
if (cf_def.memtable_flush_after_mins != null)
...
if (cf_def.memtable_throughput_in_mb != null)
...
if (cf_def.memtable_operations_in_millions != null)
...
public ColumnDefinition getColu...
...
public ColumnDefinition getColumnDe...{
    for (ColumnDefinition def : column_metadata.values())
=====
    for (ColumnDefinition def : column_metadata.values())
>>>>>>>
...

```

*unstructured merge*

conflict  
(from project  
Infinispan)



*Merge effort* is the number of extra actions to conciliate the changes made in different revisions.

**Prudêncio et al. 12**

Evaluating the Branch Merging Effort in  
Version Control Systems

Number of Conflicts reached up to 99% correlation when compared to the actual *merge effort*.

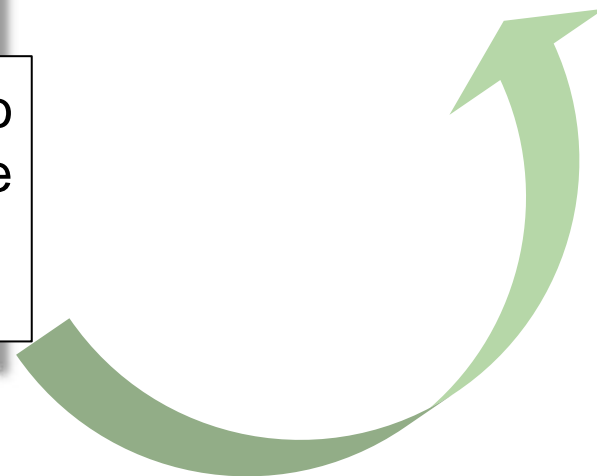
**Santos et al. 12**

*semistructured merge*

reducing **integration effort**

when compared to

*unstructured merge* ?





# Semistructured merge keeps or increases the number

**similar** numbers due to **conflicts inside method bodies**

```
...
private void analyzeAndReportSemanticErrors() {
...
<<<<<<<
    environment.getProject(), sourceFiles, filesToAnalyzeCompletely,
    JetControlFlowDataTraceFactory.EMPTY, compilerSpecialMode);
=====
    environment.getProject(), sourceFiles, filesToAnalyzeCompletely,
    JetControlFlowDataTraceFactory.EMPTY);
>>>>>>>
...
}
...
```

*unstructured merge* conflict = *semistructured merge* conflict  
(from project kotlin)

# increased numbers due to renamings or deletions

BASE

```
...  
public void Init(Address address,  
TransactionSettings transactionSettings,  
Func<bool> commitTransation)  
{  
    this.address = address;  
}  
...
```

LEFT

```
...  
public void Init(Address address,  
TransactionSettings transactionSettings)  
{  
    this.address = address;  
}  
...
```

RIGHT

```
...  
public void Init(Address address,  
TransactionSettings transactionSettings,  
Func<bool> commitTransation)  
{  
    this.settings = transactionSettings;  
    this.address = address;  
}  
...
```

Code  
Integration

UNSTRUCTURED MERGE

```
...  
public void Init(Address address,  
TransactionSettings transactionSettings)  
{  
    this.settings = transactionSettings;  
    this.address = address;  
}  
...
```

SEMISTRUCTURED MERGE

```
...  
<<<<<<<  
=====  
public void Init(Address address,  
TransactionSettings transactionSettings,  
Func<bool> commitTransation)  
{  
    this.settings = transactionSettings;  
    this.address = address;  
}  
>>>>>>>  
...
```

One group of users using a certain set of commands is *noninterfering* with another group of users if what the first group does with those commands has no effect on what the second group expects.

This paper motivates and outlines a new approach to secure systems with the following novel properties:

- It introduces a simple and general

taxonomy of access-control policies, and by letting them appear at their proper level of abstraction.

However, our approach does not address the problems of user authentication, of security breaches

# increased numbers due to renamings or deletions

The attributes are unlikely to change



BASE

```
...
public void Init(Address address,
TransactionSettings transactionSettings,
Func<bool> commitTransation)
{
    this.address = address;
}
...
```

I hope no one edit this signature



LEFT

```
...
public void Init(Address address,
TransactionSettings transactionSettings)
{
    this.address = address;
}
...
```

RIGHT

```
...
public void Init(Address address,
TransactionSettings transactionSettings,
Func<bool> commitTransation)
{
    this.settings = transactionSettings;
    this.address = address;
}
...
```

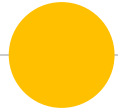
Code  
Integration

UNSTRUCTURED MERGE

```
...
public void Init(Address address,
TransactionSettings transactionSettings)
{
    this.settings = transactionSettings;
    this.address = address;
}
...
```

SEMISTRUCTURED MERGE

```
...
<<<<<<<
=====
public void Init(Address address,
TransactionSettings transactionSettings,
Func<bool> commitTransation)
{
    this.settings = transactionSettings;
    this.address = address;
}
>>>>>>>
...
```

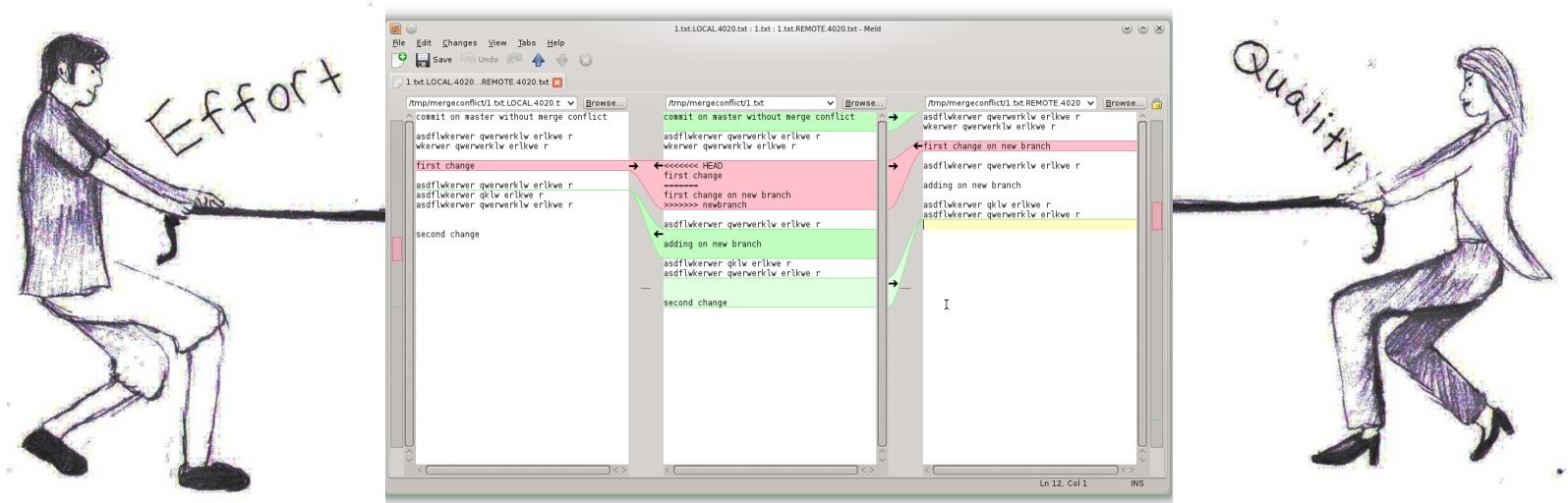


# Integration Effort and Correctness

# Semistructured merge vs. Unstructured merge

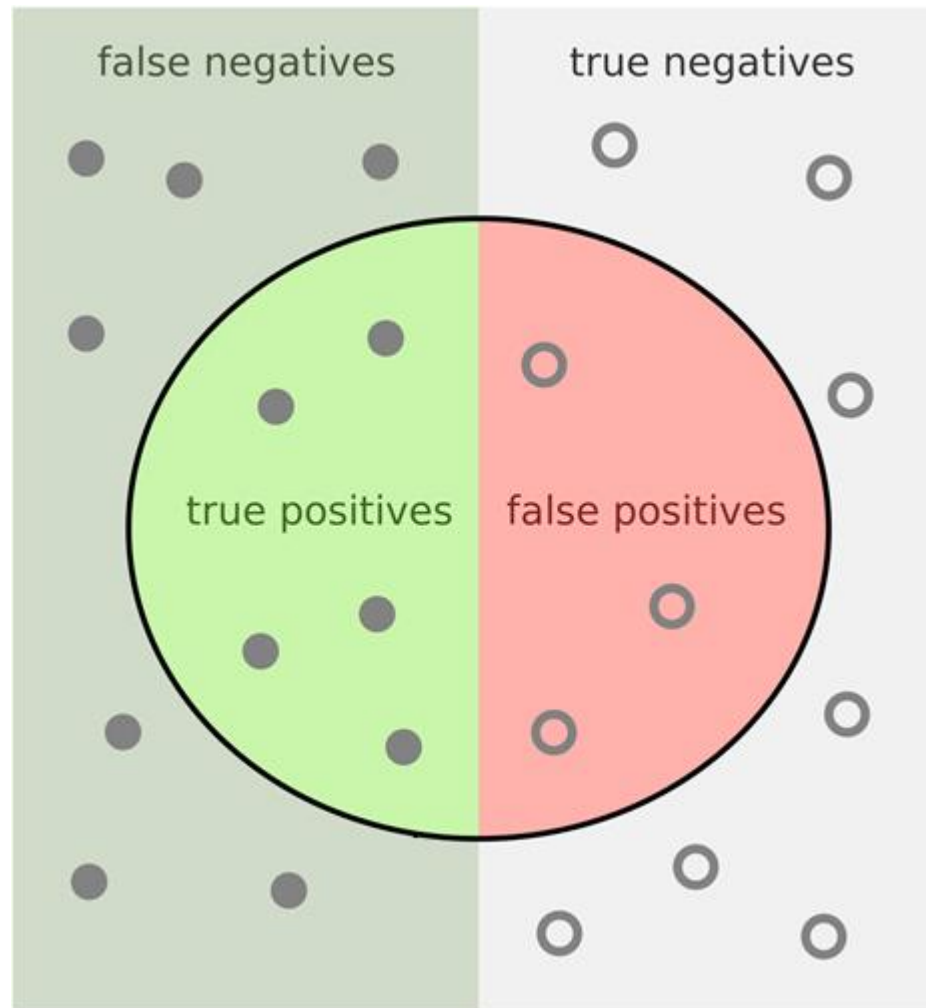
Author	VCS	Projects	Merge Scenarios	Conflicts	Conflicting LOC	Conflicting Files
Apel et al. 11	Subversion	24	180	34%	61%	28%
<b>Our replication</b>	<b>Git</b>	<b>60</b>	<b>3266</b>	<b>62%</b>	<b>81%</b>	<b>65%</b>

false positives: unnecessary integration effort



false negatives: build or behavioral errors

Comparing added false positives and false negatives from one approach in relation to the other





# Research Questions

---

- **RQ1** – When compared to unstructured merge, does semistructured merge reduce unnecessary developer's integration effort?
- **RQ2** – When compared to unstructured merge, does semistructured merge compromise integration correctness by missing more task interferences?

# semistructured merge's superimposition

```
1 package util;  
2 class Stack {  
3   Object top() { return data.getFirst(); }  
4 }
```

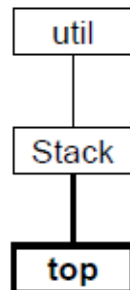
•

```
1 package util;  
2 class Stack {  
3   LinkedList data = new LinkedList();  
4   void push(Object obj) { data.addFirst(obj); }  
5   Object pop() { return data.removeFirst(); }  
6 }
```

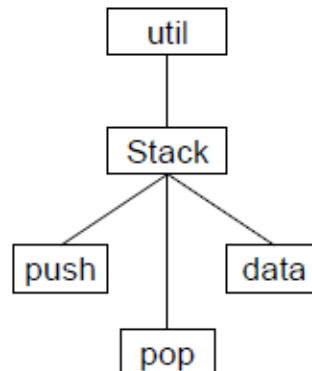
=

```
1 package util;  
2 class Stack {  
3   LinkedList data = new LinkedList();  
4   void push(Object obj) { data.addFirst(obj); }  
5   Object pop() { return data.removeFirst(); }  
6   Object top() { return data.getFirst(); }  
7 }
```

**TopOfStack**

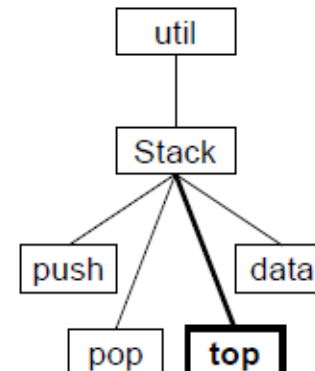


**BasicStack**



=

**CompStack**



Semistructured  $\neq$   
Unstructured  
Outside methods  
declarations.

```
public class Calc {
```

```
...
```

```
<<<<<<<
```

```
10. public int sum(int a, int b) {  
11.     return a + b;  
12. }
```

```
=====
```

```
10. public int sub(int a, int b) {  
11.     return a-b;  
12. }
```

```
>>>>>>>
```

```
...
```

```
}
```

Developer A

Developer B

Ordering Conflict  
(Unstructured Merge)

```
public class Calc {
```

```
...
```

```
<<<<<<<
```

```
10. public int doMath(int a, int b) {  
11.     return (a + b)2;  
12. }
```

```
|||||
```

```
10. public int doMath(int a, int b) {  
11.     return a + b;  
12. }
```

```
=====
```

```
>>>>>>>
```

```
...
```

```
20. public int sum(int a, int b) {  
21.     return a + b;  
22. }
```

```
...
```

```
}
```

Developer A

Developer B


Base

Renaming/Deletion Conflict  
(Semistructured Merge)

```

3 public class Calc {
4
5     int doMath(int a, int b){
6         return a+b;
7     }
8
9     long fib(int n) {
10         if (n <= 1) return n;
11         else return fib(n-1) + fib(n-2);
12     }
13
14     int doMath(int a, int b){
15         return a*b;
16     }
17 }

```

 Duplicate method doMath(int, int) in type Calc

Developer A

Developer B

Duplicated Declaration Error  
(Unstructured Merge)

```
2
3 import java.util.*;
4 import java.awt.*;
5
6 public class Test {
7
8     public static void main(String[] args) {
9         List list;
10    }
11 }
12
```

The type List is ambiguous

- ← Explicitly import 'java.awt.List'
- ← Explicitly import 'java.util.List'
- 🔧 Rename in file (Ctrl+2, R)

## **member x member**

import java.util.List and import java.awt.List

## **package x package**

import java.util.\* and import java.awt.\*

## **package x member**

import java.util.List and import java.awt.\*

Type Ambiguity Error  
(Semistructured Merge)

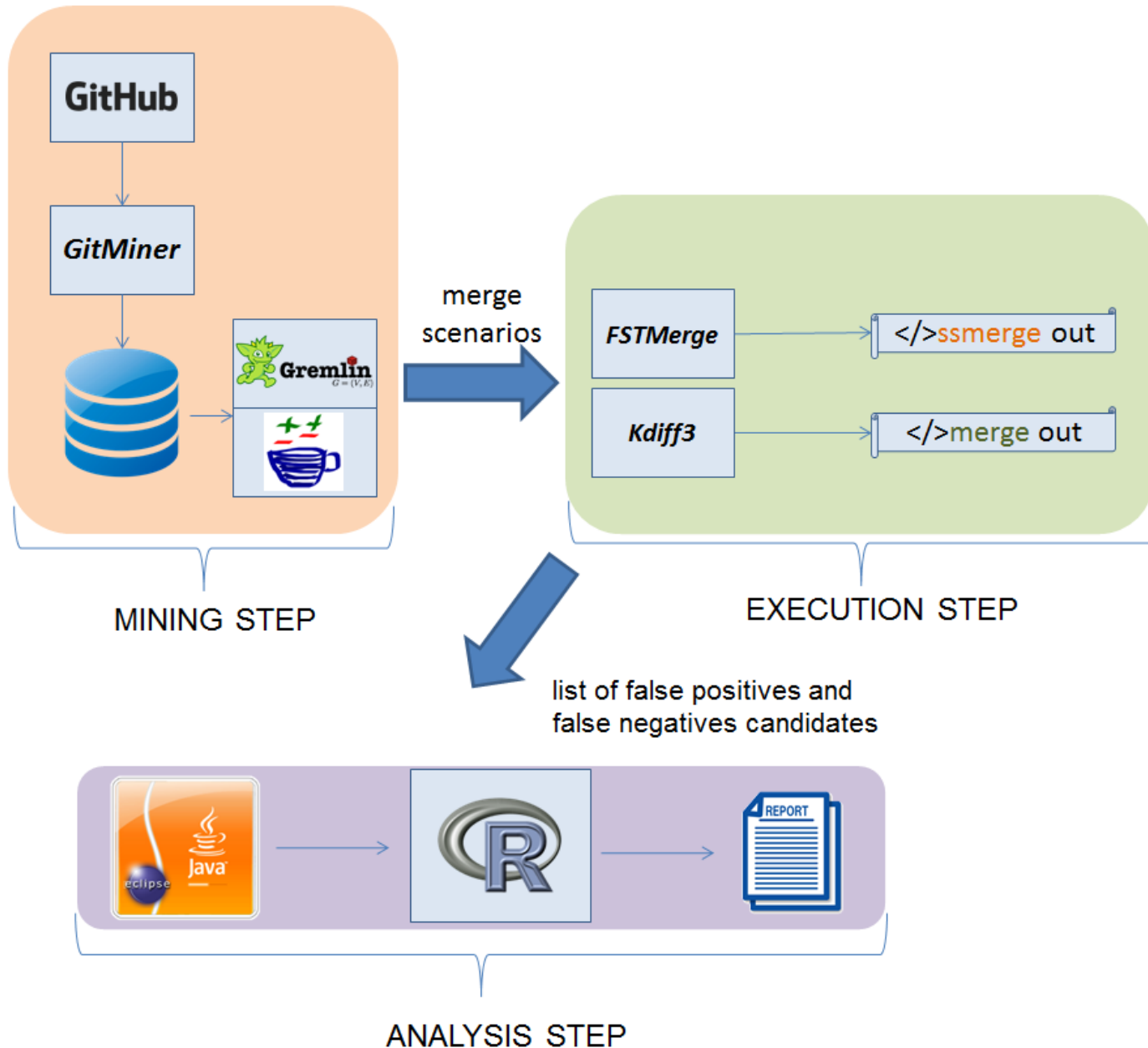
```
public class Calc {  
...  
10. public int doMath(int a, int b) {  
11.   return (a + b)2;  
12. }  
13.  
14. public int composed(int a, int b){  
15.   return doMath(a + b)3;  
16. }  
...
```

Developer A

Developer B

New Artefact Referencing  
Edited One  
(Semistructured Merge)

# Experimental Design

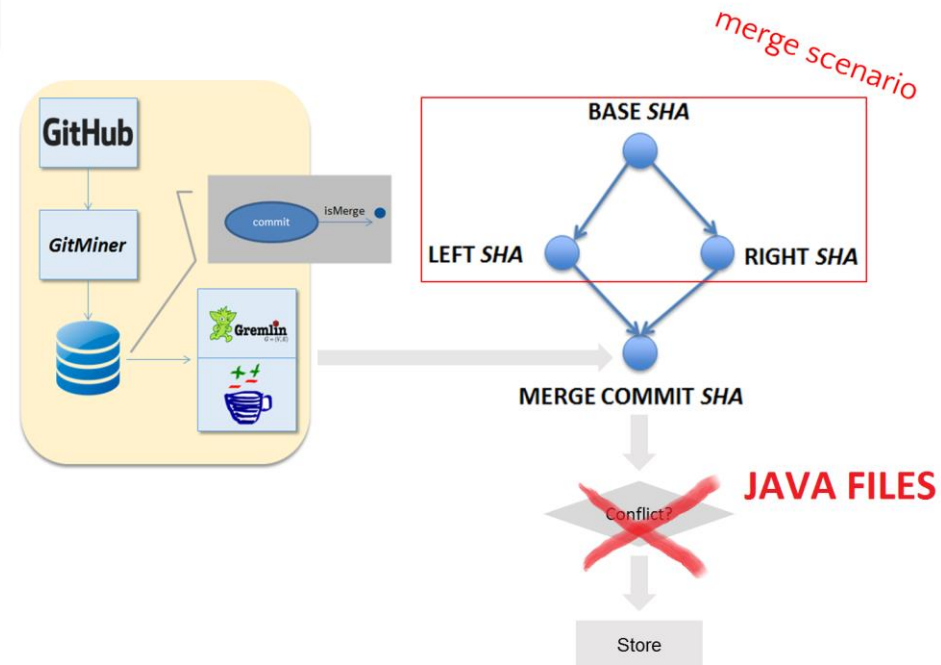
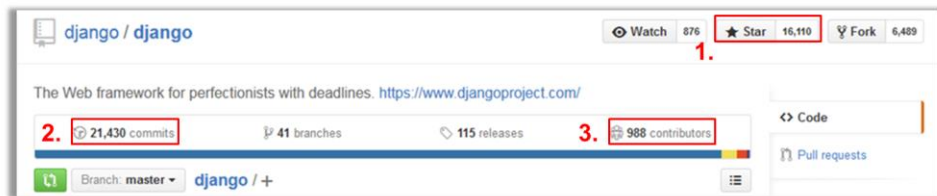




# Mining Step

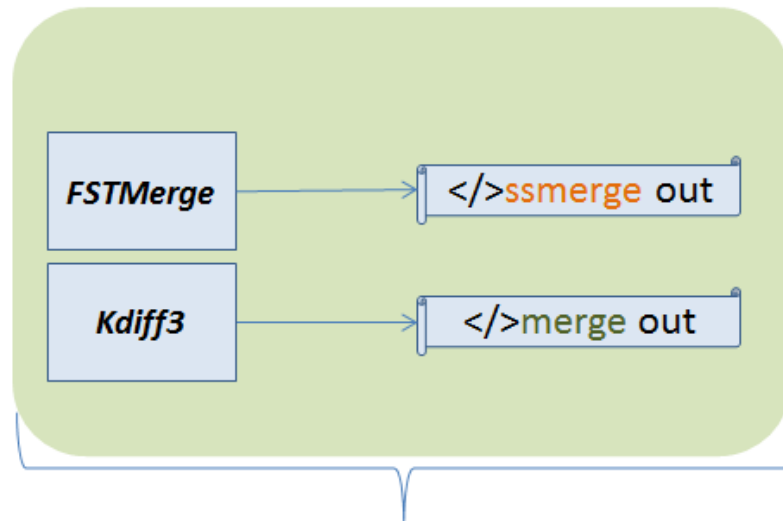
Project selection criteria

1. Frequency and Recency of the collaborators activities
2. Number of commits
3. Number of collaborators



Project	Project Name	Project Size	Project Type	Project Status	Project Location	Project Date
Project 1	Project 1 Name	100	Project 1 Type	Project 1 Status	Project 1 Location	Project 1 Date
Project 2	Project 2 Name	200	Project 2 Type	Project 2 Status	Project 2 Location	Project 2 Date
Project 3	Project 3 Name	300	Project 3 Type	Project 3 Status	Project 3 Location	Project 3 Date
Project 4	Project 4 Name	400	Project 4 Type	Project 4 Status	Project 4 Location	Project 4 Date
Project 5	Project 5 Name	500	Project 5 Type	Project 5 Status	Project 5 Location	Project 5 Date
Project 6	Project 6 Name	600	Project 6 Type	Project 6 Status	Project 6 Location	Project 6 Date
Project 7	Project 7 Name	700	Project 7 Type	Project 7 Status	Project 7 Location	Project 7 Date
Project 8	Project 8 Name	800	Project 8 Type	Project 8 Status	Project 8 Location	Project 8 Date
Project 9	Project 9 Name	900	Project 9 Type	Project 9 Status	Project 9 Location	Project 9 Date
Project 10	Project 10 Name	1000	Project 10 Type	Project 10 Status	Project 10 Location	Project 10 Date
Project 11	Project 11 Name	1100	Project 11 Type	Project 11 Status	Project 11 Location	Project 11 Date
Project 12	Project 12 Name	1200	Project 12 Type	Project 12 Status	Project 12 Location	Project 12 Date
Project 13	Project 13 Name	1300	Project 13 Type	Project 13 Status	Project 13 Location	Project 13 Date
Project 14	Project 14 Name	1400	Project 14 Type	Project 14 Status	Project 14 Location	Project 14 Date
Project 15	Project 15 Name	1500	Project 15 Type	Project 15 Status	Project 15 Location	Project 15 Date
Project 16	Project 16 Name	1600	Project 16 Type	Project 16 Status	Project 16 Location	Project 16 Date
Project 17	Project 17 Name	1700	Project 17 Type	Project 17 Status	Project 17 Location	Project 17 Date
Project 18	Project 18 Name	1800	Project 18 Type	Project 18 Status	Project 18 Location	Project 18 Date
Project 19	Project 19 Name	1900	Project 19 Type	Project 19 Status	Project 19 Location	Project 19 Date
Project 20	Project 20 Name	2000	Project 20 Type	Project 20 Status	Project 20 Location	Project 20 Date
Project 21	Project 21 Name	2100	Project 21 Type	Project 21 Status	Project 21 Location	Project 21 Date
Project 22	Project 22 Name	2200	Project 22 Type	Project 22 Status	Project 22 Location	Project 22 Date
Project 23	Project 23 Name	2300	Project 23 Type	Project 23 Status	Project 23 Location	Project 23 Date
Project 24	Project 24 Name	2400	Project 24 Type	Project 24 Status	Project 24 Location	Project 24 Date
Project 25	Project 25 Name	2500	Project 25 Type	Project 25 Status	Project 25 Location	Project 25 Date
Project 26	Project 26 Name	2600	Project 26 Type	Project 26 Status	Project 26 Location	Project 26 Date
Project 27	Project 27 Name	2700	Project 27 Type	Project 27 Status	Project 27 Location	Project 27 Date
Project 28	Project 28 Name	2800	Project 28 Type	Project 28 Status	Project 28 Location	Project 28 Date
Project 29	Project 29 Name	2900	Project 29 Type	Project 29 Status	Project 29 Location	Project 29 Date
Project 30	Project 30 Name	3000	Project 30 Type	Project 30 Status	Project 30 Location	Project 30 Date
Project 31	Project 31 Name	3100	Project 31 Type	Project 31 Status	Project 31 Location	Project 31 Date
Project 32	Project 32 Name	3200	Project 32 Type	Project 32 Status	Project 32 Location	Project 32 Date
Project 33	Project 33 Name	3300	Project 33 Type	Project 33 Status	Project 33 Location	Project 33 Date
Project 34	Project 34 Name	3400	Project 34 Type	Project 34 Status	Project 34 Location	Project 34 Date
Project 35	Project 35 Name	3500	Project 35 Type	Project 35 Status	Project 35 Location	Project 35 Date
Project 36	Project 36 Name	3600	Project 36 Type	Project 36 Status	Project 36 Location	Project 36 Date
Project 37	Project 37 Name	3700	Project 37 Type	Project 37 Status	Project 37 Location	Project 37 Date
Project 38	Project 38 Name	3800	Project 38 Type	Project 38 Status	Project 38 Location	Project 38 Date
Project 39	Project 39 Name	3900	Project 39 Type	Project 39 Status	Project 39 Location	Project 39 Date
Project 40	Project 40 Name	4000	Project 40 Type	Project 40 Status	Project 40 Location	Project 40 Date
Project 41	Project 41 Name	4100	Project 41 Type	Project 41 Status	Project 41 Location	Project 41 Date
Project 42	Project 42 Name	4200	Project 42 Type	Project 42 Status	Project 42 Location	Project 42 Date
Project 43	Project 43 Name	4300	Project 43 Type	Project 43 Status	Project 43 Location	Project 43 Date
Project 44	Project 44 Name	4400	Project 44 Type	Project 44 Status	Project 44 Location	Project 44 Date
Project 45	Project 45 Name	4500	Project 45 Type	Project 45 Status	Project 45 Location	Project 45 Date
Project 46	Project 46 Name	4600	Project 46 Type	Project 46 Status	Project 46 Location	Project 46 Date
Project 47	Project 47 Name	4700	Project 47 Type	Project 47 Status	Project 47 Location	Project 47 Date
Project 48	Project 48 Name	4800	Project 48 Type	Project 48 Status	Project 48 Location	Project 48 Date
Project 49	Project 49 Name	4900	Project 49 Type	Project 49 Status	Project 49 Location	Project 49 Date
Project 50	Project 50 Name	5000	Project 50 Type	Project 50 Status	Project 50 Location	Project 50 Date

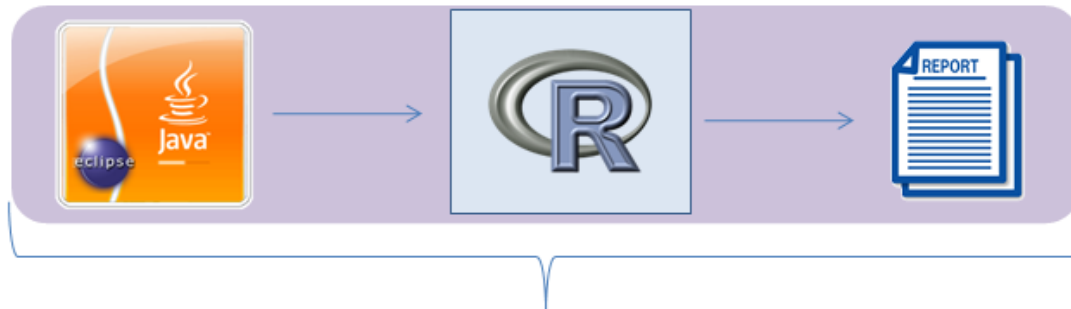
# Execution and Analysis Steps



EXECUTION STEP



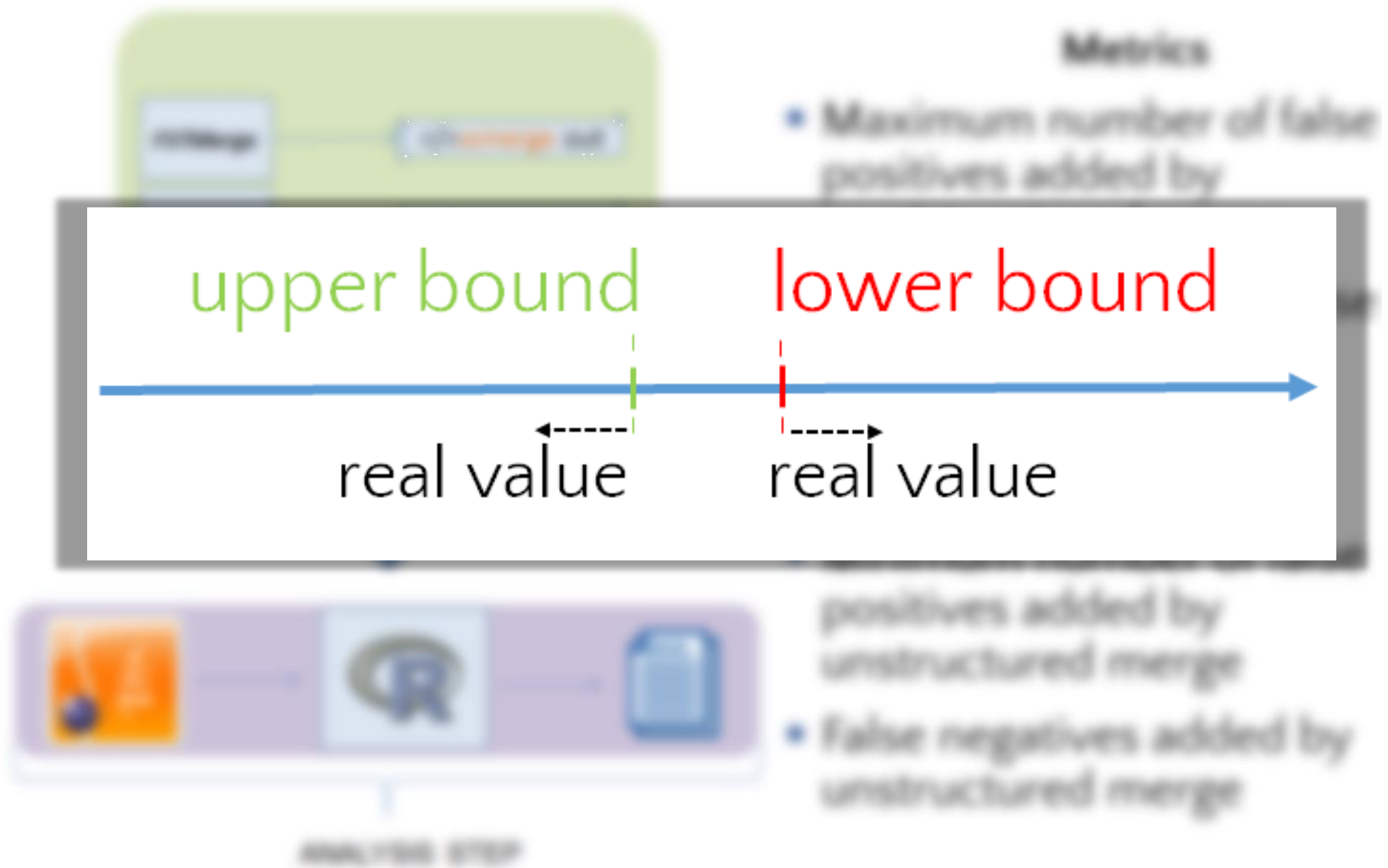
list of false positives and  
false negatives candidates



ANALYSIS STEP

- **FPa(SS)** – Maximum number of false positives added by semistructured merge
- **FNa(SS)** – Maximum number of false negatives added by semistructured merge
- **FPa(UN)** – Minimum number of false positives added by unstructured merge
- **FNa(UN)** – False negatives added by unstructured merge

# Execution and Analysis Steps



# Evaluation Results

---

**19,238**  
unstructured merge conflicts

**14,544**  
semistructured merge conflicts

**24%**

Reduction!

# RQ1 - When compared to unstructured merge, does semistructured merge reduce unnecessary developer's integration effort?

Added false positives by conflicts(%)

Unstructured Semistructured



38.11% ± 23.49%

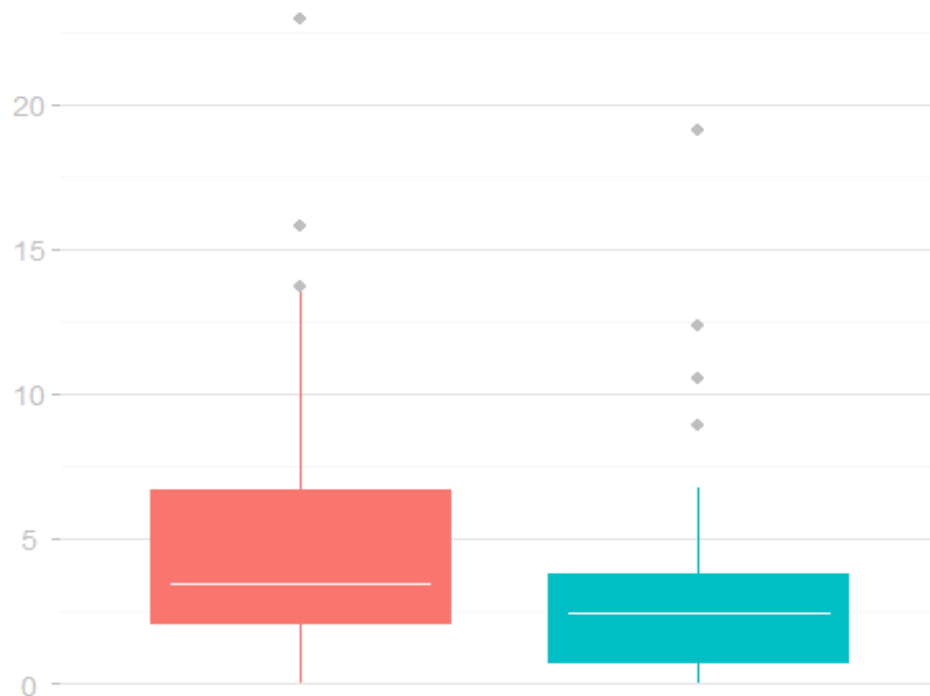
30.21% ± 20.68%

**p-value > 0.05**

There is no significant difference

Added false positives by merge scenarios(%)

Unstructured Semistructured



5.35% ± 4.85%

3.12% ± 3.55%

**p-value < 0.05**

There is significant difference

Median test statistics by confidence (%)

100% 90% 80% 70% 60% 50% 40% 30% 20% 10% 0%

Median test statistics by range confidence (%)

100% 90% 80% 70% 60% 50% 40% 30% 20% 10% 0%

FPa(SS)

FPa(UN)

real value

real value

28.37% & 23.49%

30.27% & 20.48%

5.20% & 4.80%

2.07% & 2.00%

**p-value > 0.05**

Wilcoxon Signed Rank Test  
There is no significant difference

**p-value < 0.05**

Wilcoxon Signed Rank Test  
There is significant difference

# simple ordering conflicts

<<<<<<

```
private final Multimap<ModelPath, ImmutableList<ModelPath>> usedMutators =  
ArrayListMultimap.create();
```

=====

```
private final Multimap<ModelPath, ModelMutation<?>> finalizers =  
ArrayListMultimap.create();
```

>>>>>>

Developer A

Developer B

(from project gradle)

<<<<<<

```
public Map<String, String> pubsubNumSub(String... channels) {  
    checkIsInMulti();  
    client.pubsubNumSub(channels);  
    return BuilderFactory.STRING_MAP.build(client.getBinaryMultiBulkReply());  
}
```

=====

```
public String asking() {  
    checkIsInMulti();  
    client.asking();  
    return client.getStatusCodeReply();  
}
```

>>>>>>

Developer A

Developer B

(from project jedis)



# crosscutting ordering conflicts

```
...
<<<<<<
    public static void validateMemtableSettings(org.apache.cassandra.db.migration.avro.CfDef cf_def) throws
    ConfigurationException
=====
    public ColumnDefinition getColumnDefinition(ByteBuffer name)
    {
        return column_metadata.get(name);
    }
    public ColumnDefinition getColumnDefinitionForIndex(String indexName)
>>>>>>
<<<<<<
    if (cf_def.memtable_flush_after_mins != null)
        DatabaseDescriptor.validateMemtableFlushPeriod(cf_def.memtable_flush_after_mins);
    if (cf_def.memtable_throughput_in_mb != null)
        DatabaseDescriptor.validateMemtableThroughput(cf_def.memtable_throughput_in_mb);
    if (cf_def.memtable_operations_in_millions != null)
        DatabaseDescriptor.validateMemtableOperations(cf_def.memtable_operations_in_millions);
    }
    public ColumnDefinition getColumnDefinition(ByteBuffer name)
    {
        return column_metadata.get(name);
    }
    public ColumnDefinition getColumnDefinitionForIndex(String indexName)
    {
        for (ColumnDefinition def : column_metadata.values())
=====
        for (ColumnDefinition def : column_metadata.values())
>>>>>>
...

```

Developer A

Developer B

(from project cassandra)

# false positive renaming conflict

LEFT

```
public void (Collection<DocConsumerPerThread> threads, SegmentWriteState state)
throws IOException {
    ...
    for ( DocConsumerPerThread thread : threads) {
        DocFieldProcessorPerThread perThread = (DocFieldProcessorPerThread) thread;
        childThreadsAndFields.put(perThread.consumer, perThread.fields());
        perThread.trimFields(state);
    }
    trimFields(state);

    fieldsWriter.flush(state);
    consumer.flush(childFields, state);
    ...
}
```

```
public void flush(SegmentWriteState state) throws IOException {
    ...
    Collection<DocFieldConsumerPerField> fields = fields();
    for (DocFieldConsumerPerField f : fields) {
        childFields.put(f.getFieldInfo(), f);
    }
    trimFields(state);

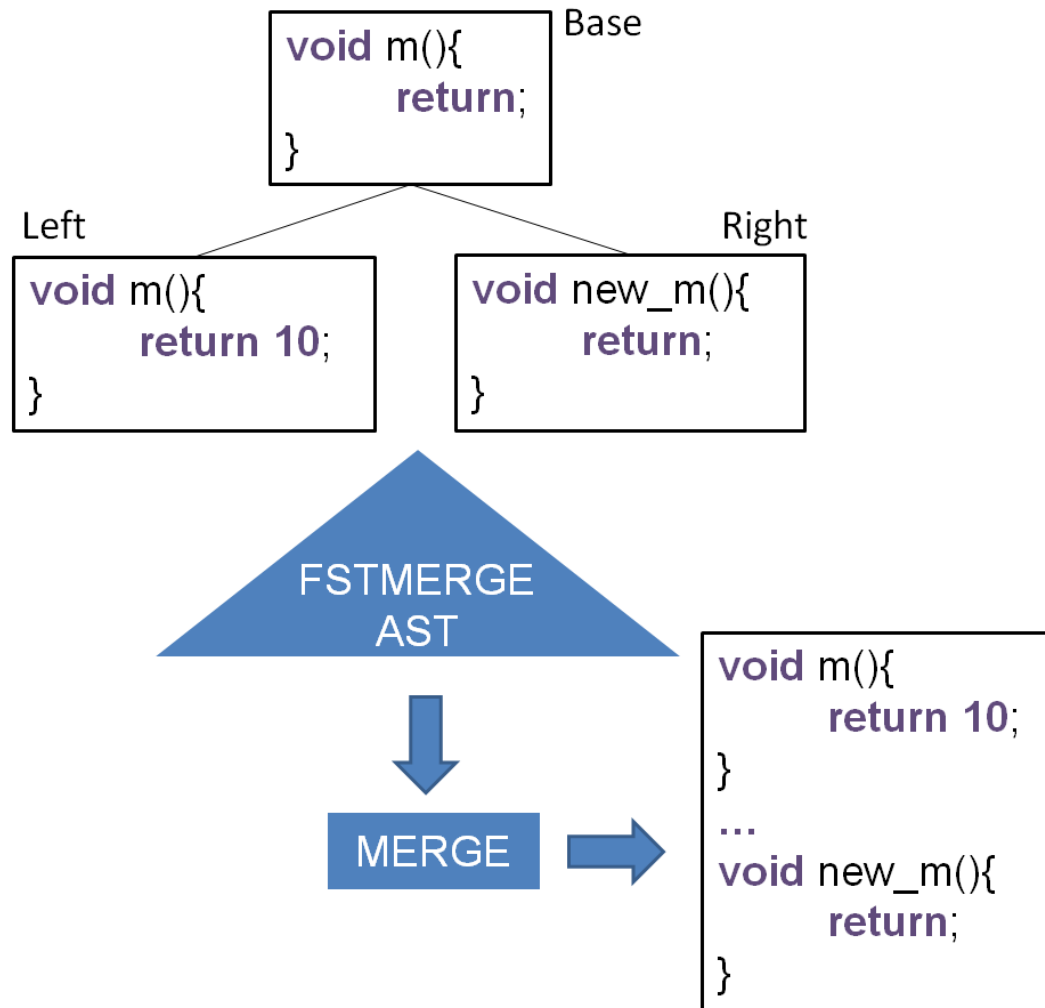
    fieldsWriter.flush(state);
    consumer.flush(childFields, state);
    ...
}
```

 Developer A  
 Developer B

RIGHT/INTEGRATOR DECISION/CURRENT VERSION

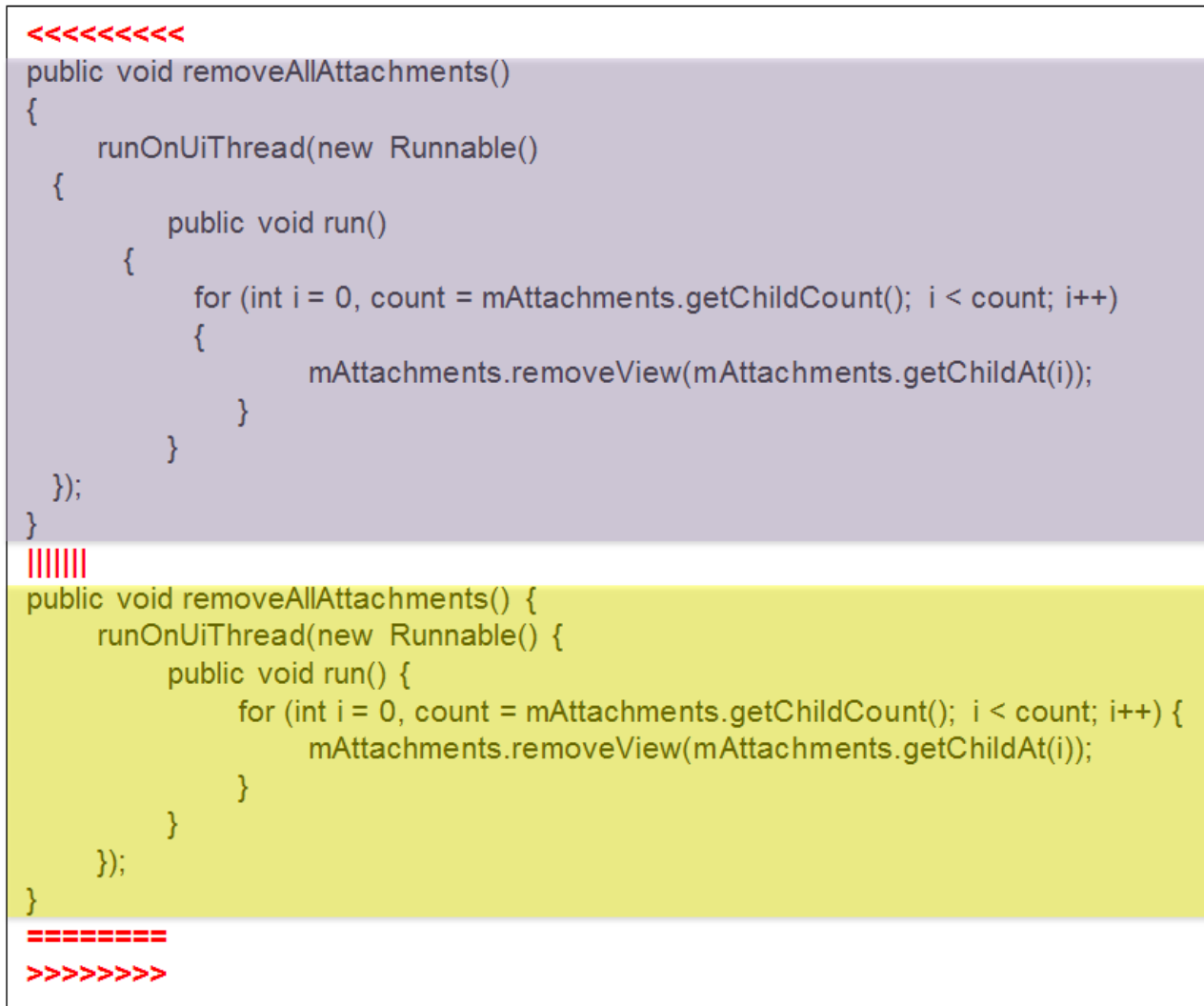
(from project lucene-solr)



# suggestions for **improving** FSTMerge tool



(keeping the two versions)

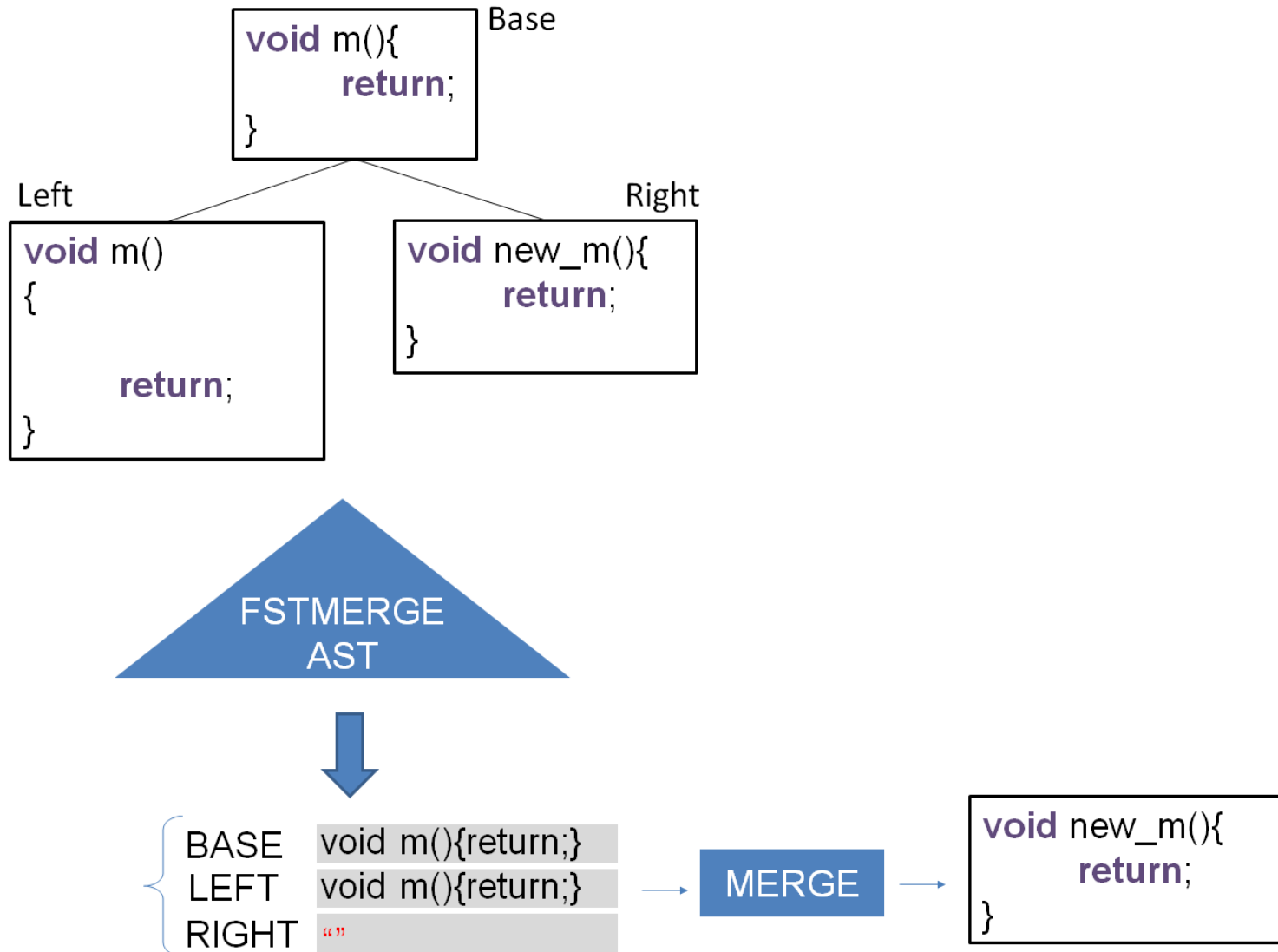
# indentation renaming/deletion conflict



 Developer A  
 Base

(from project k-9)

# suggestions for **improving** FSTMerge tool



(ignoring the spacings)

# true positive renaming conflict

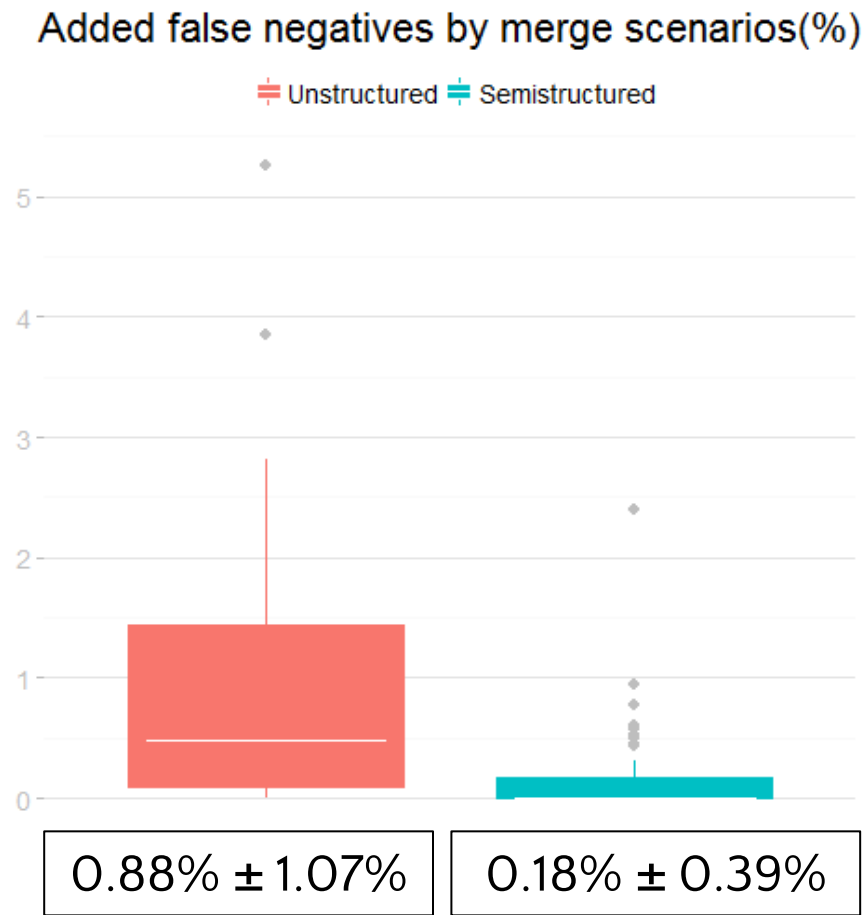
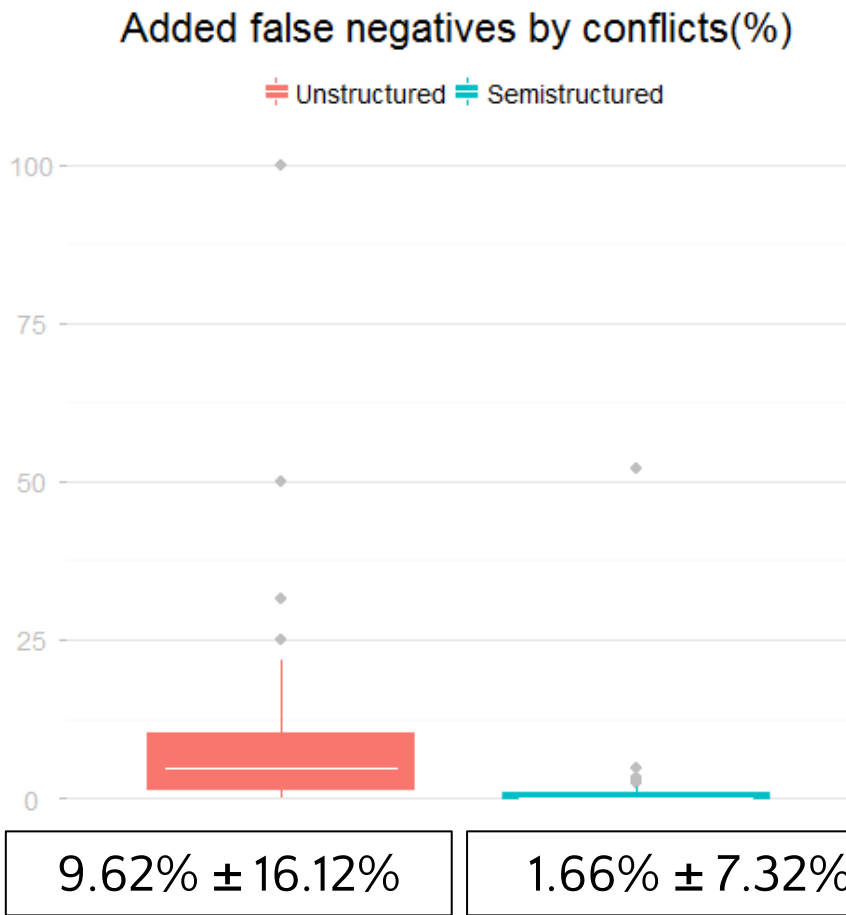
```
public void apply(org.apache.cassandra.avro.CfDef cf_def) throws  
ConfigurationException  
{  
...  
    if (!cf_def.keyspace.toString().equals(tableName))  
...  
        validateMemtableSettings(cf_def);  
...  
        for (ByteBuffer indexName : column_metadata.keySet())  
...  
            for (org.apache.cassandra.avro.ColumnDef def : cf_def.column_metadata  
...  
            }  
}
```

Developer A

Developer B

(from project cassandra)

# RQ2 - When compared to unstructured merge, does semistructured merge compromise integration correctness by missing more task interferences?



**p-value < 0.05**  
There is significant difference!

Notes: Notes negative by confidence (%)

Notes: Notes negative by range confidence (%)

Notes: Notes negative by range confidence (%)

Notes: Notes negative by range confidence (%)

$FN_{Na}(SS)$

$FN_{Na}(UN)$

real value

5x

real value

0.00% & 0.00%

1.00% & 1.00%

0.00% & 1.00%

0.00% & 0.00%

**p-value < 0.05**

Notes: Notes negative by range confidence (%)  
Notes: Notes negative by range confidence (%)



# tracking false negatives

```
public class SegmentInfo{  
    ...  
    public void setDocStoreSegment(String segment) {  
        docStoreSegment = segment;  
    }  
    ...  
    public void setDocStoreSegment(String docStoreSegment) {  
        this.docStoreSegment = docStoreSegment;  
        clearFiles();  
    }  
    ...  
}
```

Developer A  
Developer B

duplicated declaration error  
(from project lucene-solr)

```

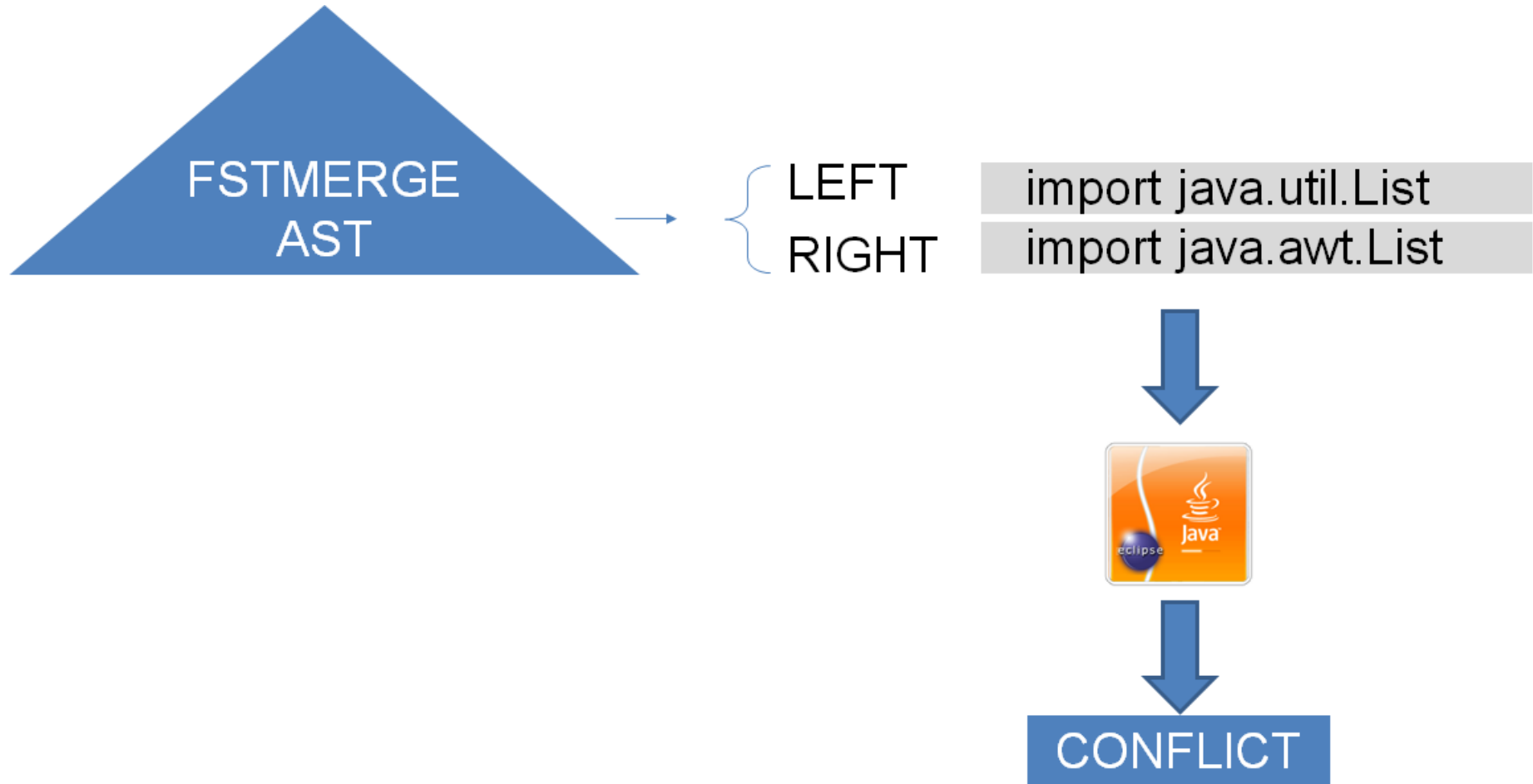
public class PlaybackService {
    ...
    private void setStatus(PlayerStatus newStatus) {
        ...
        bluetoothNotifyChange();
    }
    ...
    private void bluetoothNotifyChange() {
        ...
        if (queue != null) {
            i.putExtra("ListSize", queue.size());
        }
        ...
    }
    ...
}

```

 Developer A  
 Developer B

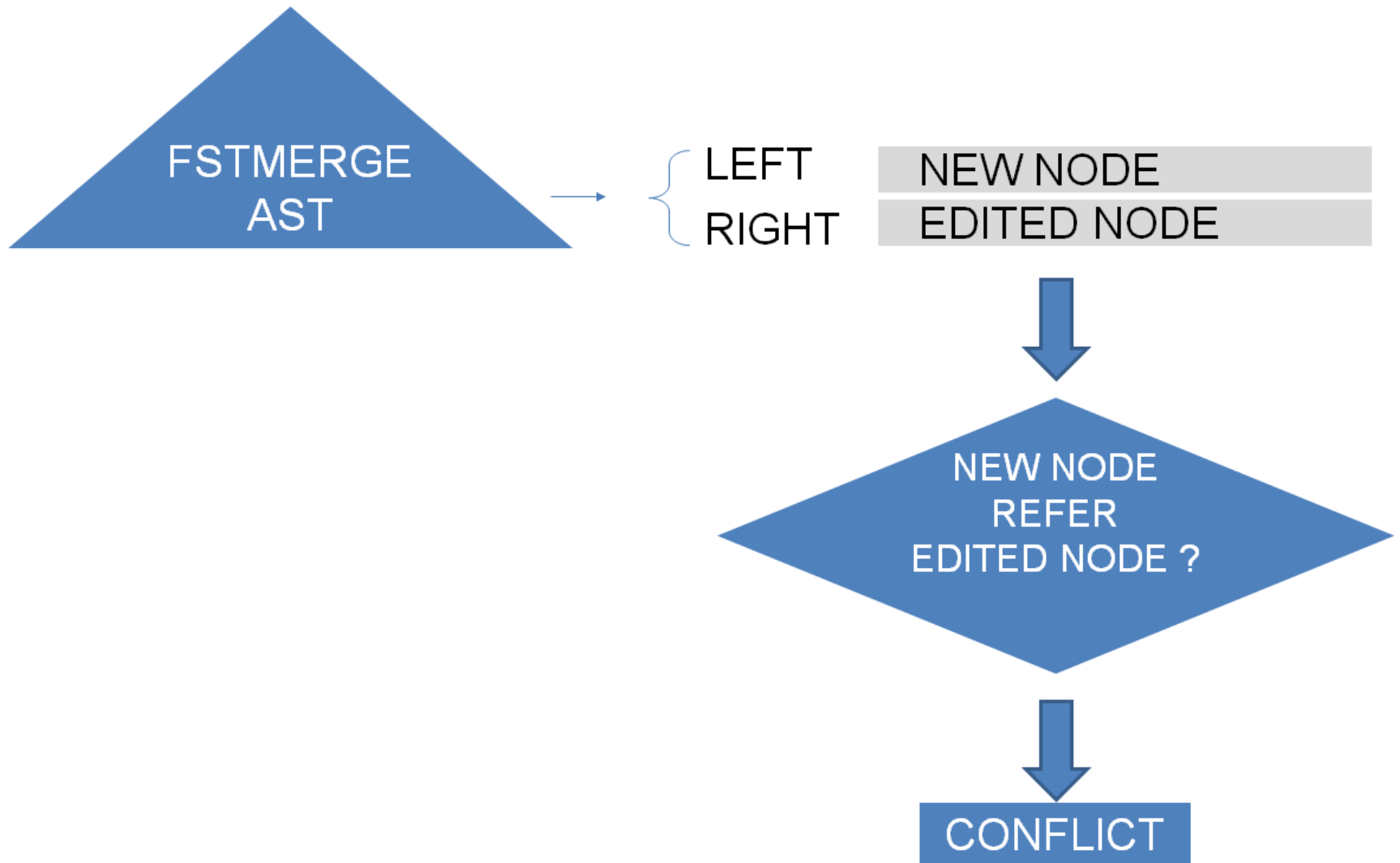
new element referencing edited one  
(from project AntennaPod)

# suggestions for **improving** FSTMerge tool



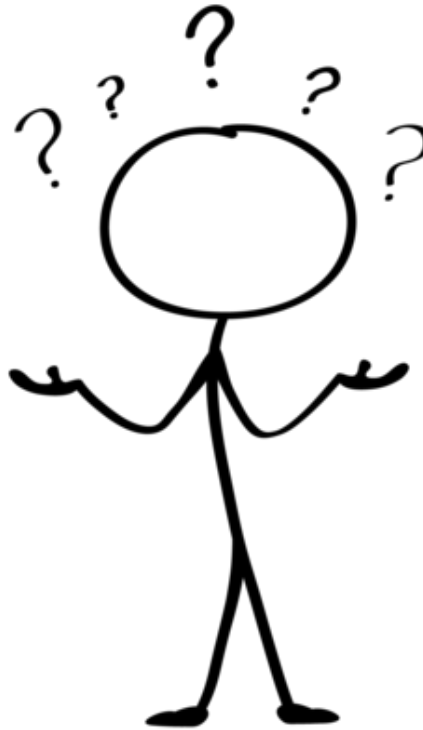
(using compilation features)

# suggestions for **improving** FSTMerge tool



(infering interference)

**unstructured  
or  
semistructured merge?**



# Future Work

---

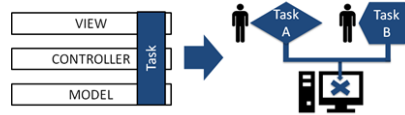
Unstructured vs.  
Semistructured vs. Structured

False Positives and False  
Negatives in general

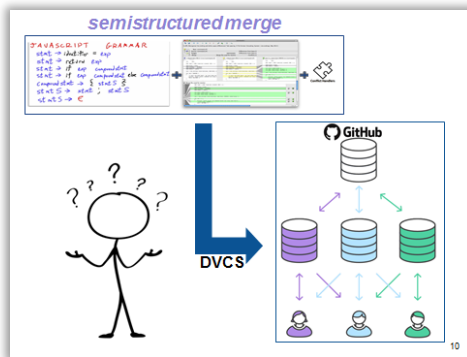
Required effort to resolve  
conflicts

# Thanks!

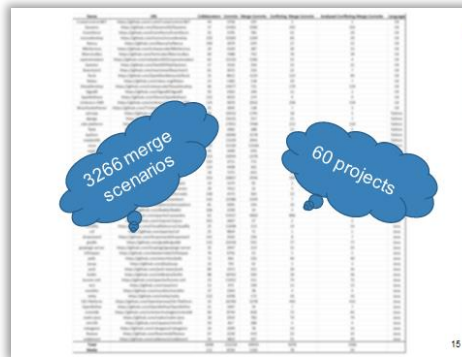
Collaborative development is an essential characteristic of large software projects



2



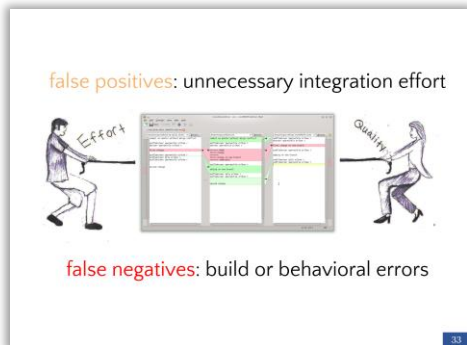
10



15



20



33



42

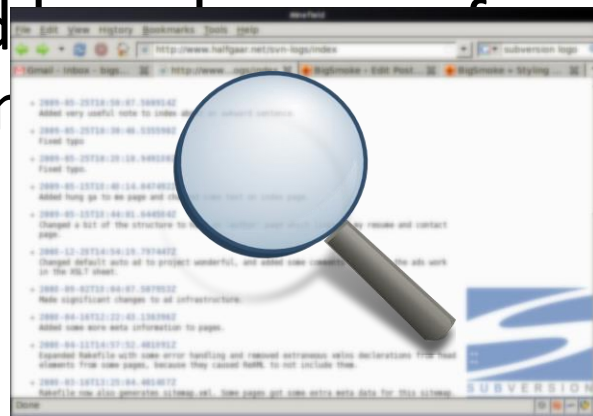


Comparing **integration effort** and **correctness** of different merge approaches in Version Control Systems





- Projects of varying sizes, and with at least one conflict by either semistructured and unstructured merge.
- Merges that developers actually performed and the revisions involved
- Merges that could not be performed or that are unrealistic considering



Apel's et al *mining step*

# Merge Conflicts and Their Types

**false-negative:** a conflict not detected

**false-positive:** a conflict that *does not* represent a  
interference between developers' tasks

**true-positive:** a conflict that represents a *real*  
interference between developers' tasks

	Semistructured ↓	Semistructured = Unstructured	Unstructured ↓
Textual Conf.	1804 (55.2%)	1179 (36.1%)	283 (08.7%)
Conf. LOC	2323 (71.1%)	581 (17.8%)	362 (11.1%)
Conf. Files	1566 (47.9%)	1691 (51.8%)	9 (0.3%)
<b>Total Merge Scenarios</b>			<b>3266</b>

# Threats to Validity

---

- Construct: the output of semistructured merge in the presence of renaming.
- Internal: our approach of selecting conflict scenarios. Discard of scenarios.
- External: the size of our sample.

```
public class Calc {
```

```
...
```

```
<<<<<<<
```

```
10. public int sum(int a, int b) {  
11.     return a + b;  
12. }
```

```
=====
```

```
10. public int sub(int a, int b) {  
11.     return a-b;  
12. }
```

```
>>>>>>>
```

```
...
```

```
}
```

 Developer A

 Developer B

## Ordering Conflict (Unstructured Merge)

```

public class Calc {
...
<<<<<<<
10. public int doMath(int a, int b) {
11.     return (a + b)2;
12. }
|||||
10. public int doMath(int a, int b) {
11.     return a + b;
12. }
=====
>>>>>>>
...
20. public int sum(int a, int b) {
21.     return a + b;
22. }
...
}

```

 Developer A  
 Developer B  
 Base

Renaming/Deletion Conflict  
(Semistructured Merge)

# Maximum False Negatives Added by Semistructured Merge – $FN_{\alpha}(SS)$

## *Type Ambiguity Errors*

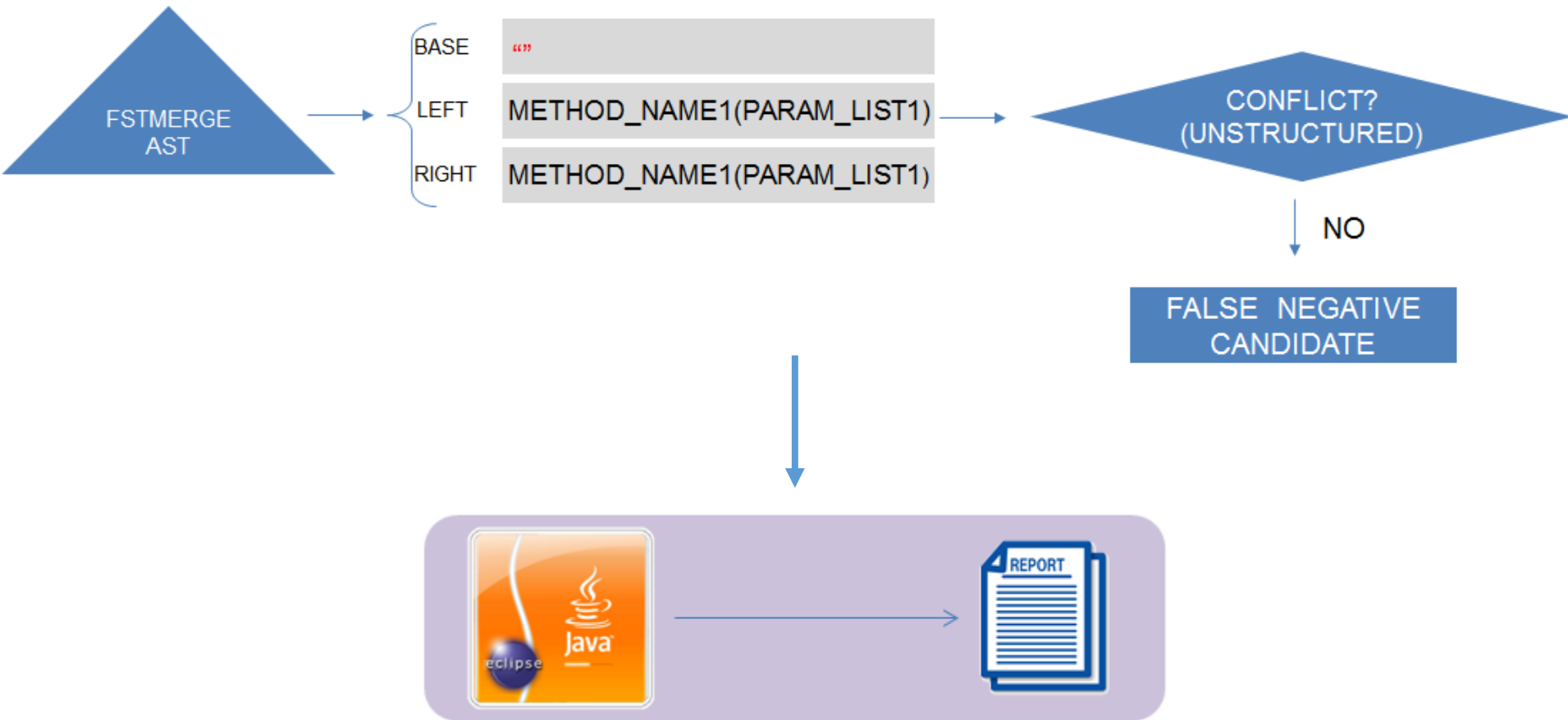
import java.util.List and import java.awt.List

import java.util.\* and import java.awt.\*

import java.util.List and import java.awt\*

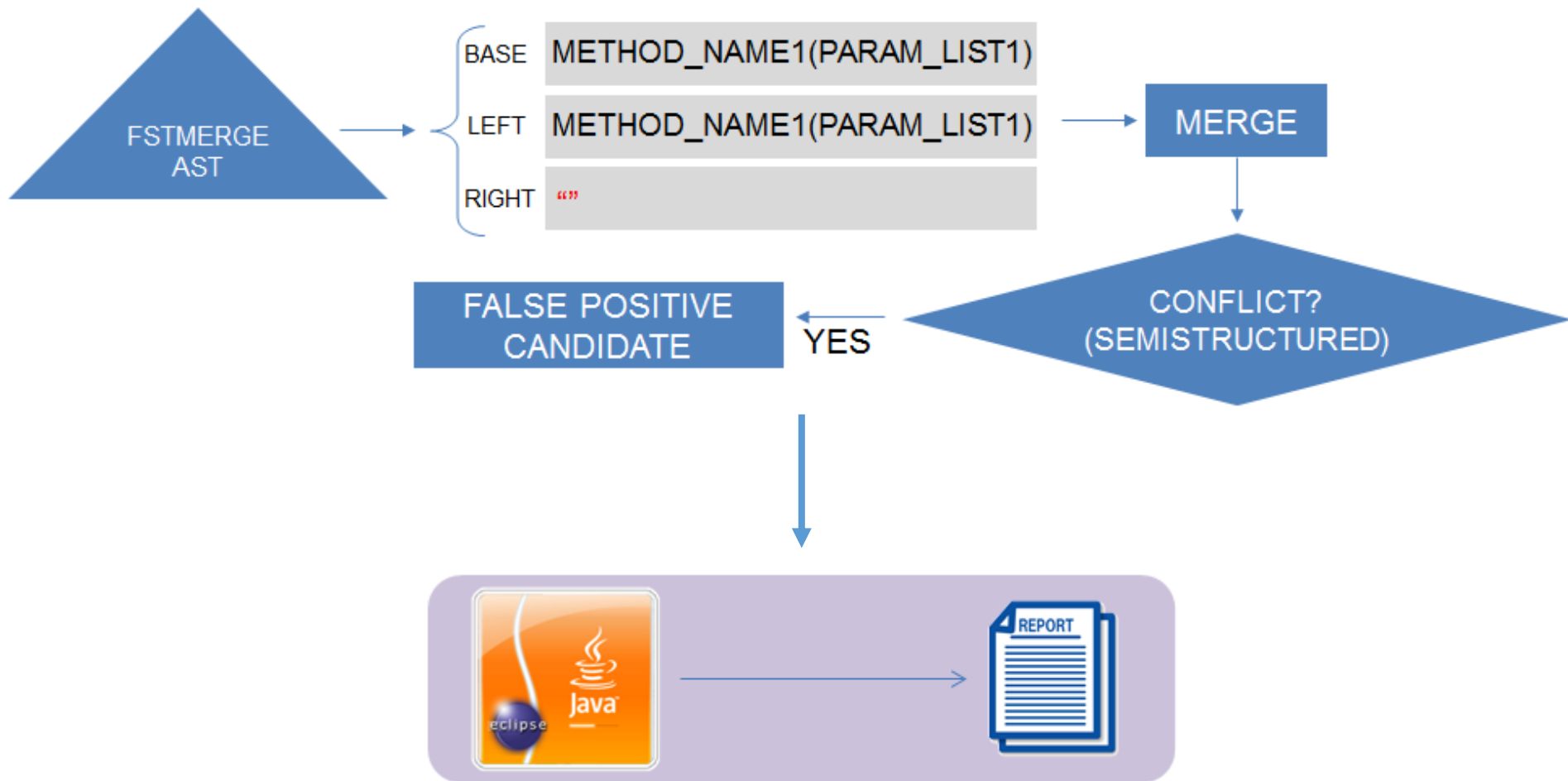
# False Negatives Added by Unstructured Merge – $FN_{\alpha}(SS)$

## *Duplicated Declaration Errors*

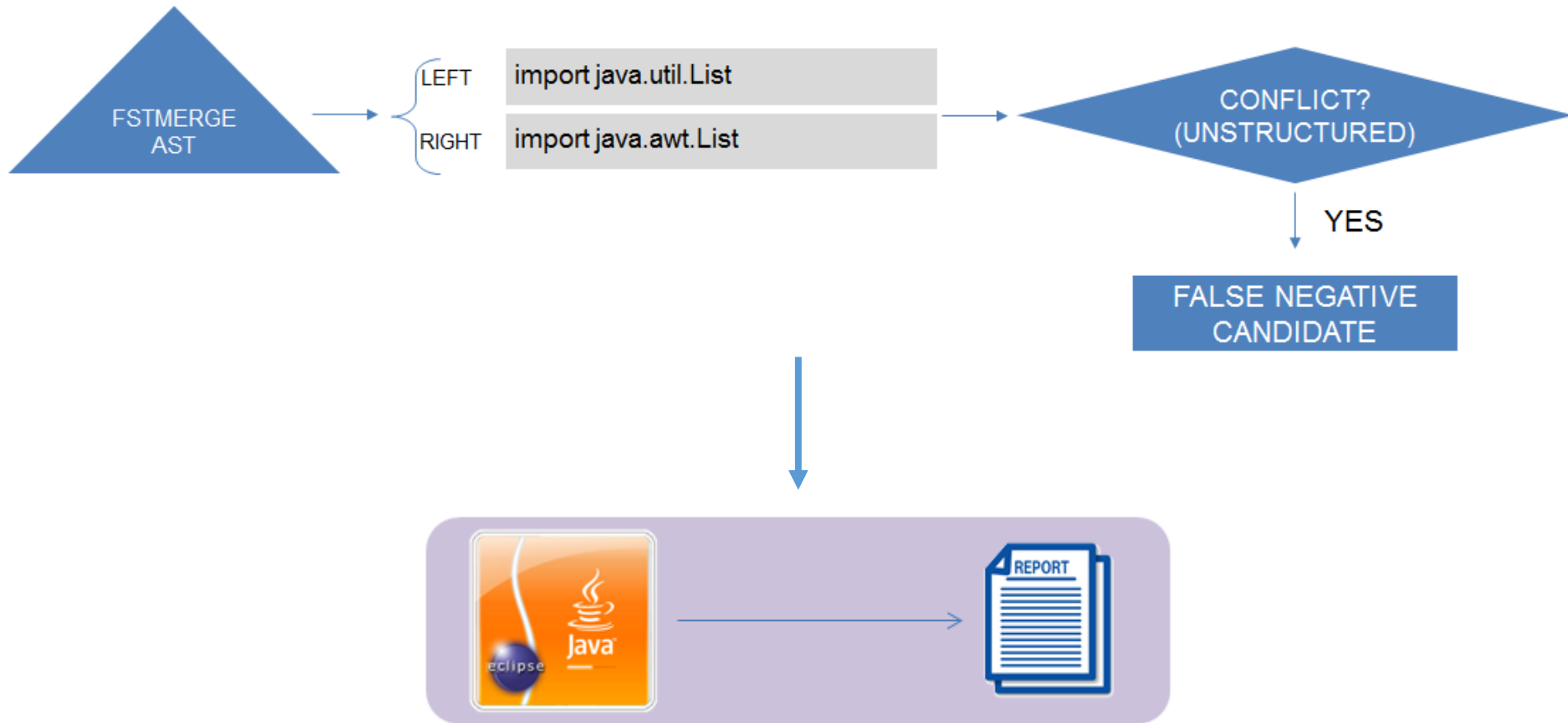




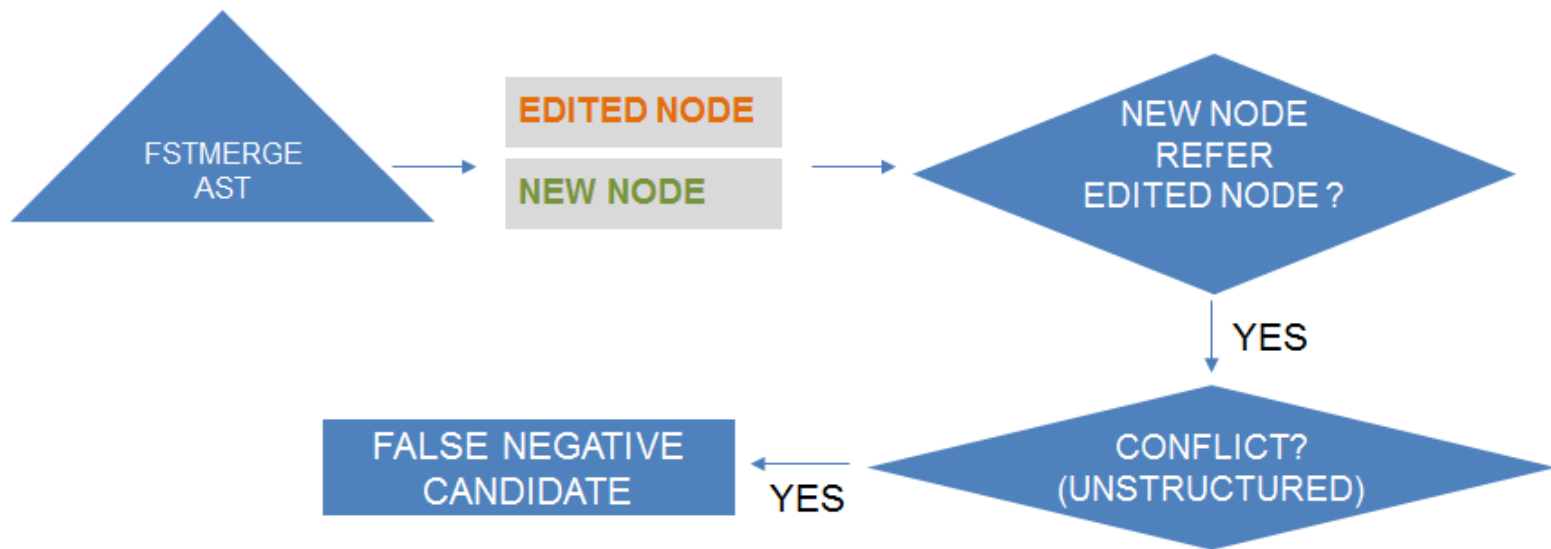
# Maximum Number of False Positives Added by Semistructured Merge – $FPa(SS)$ *Renaming or Deletion Conflicts*



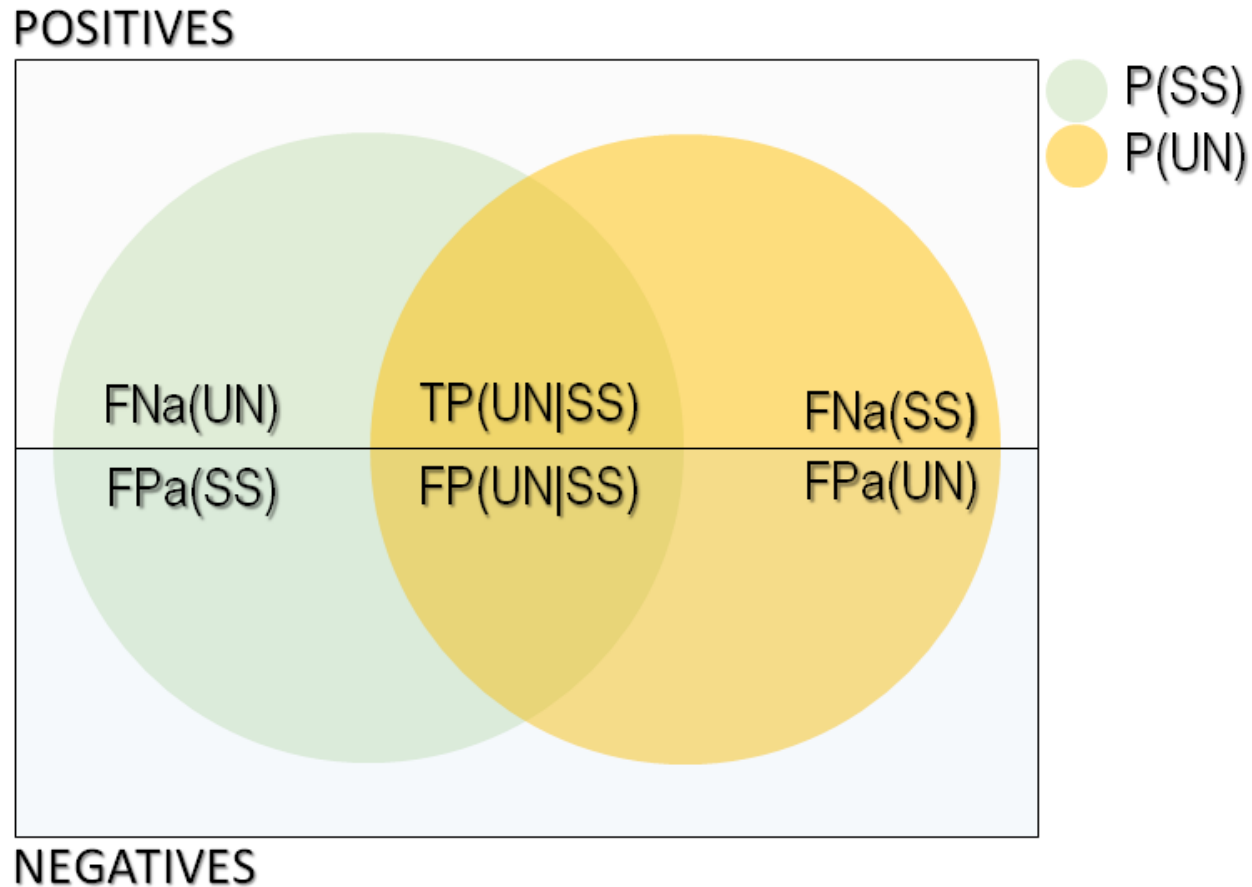
# Maximum Number of False Negatives Added by Semistructured Merge – $FNa(SS)$ *Type Ambiguity Errors*



# Maximum Number of False Negatives Added by Semistructured Merge – $FNa(SS)$ *New Element Referencing Edited One*



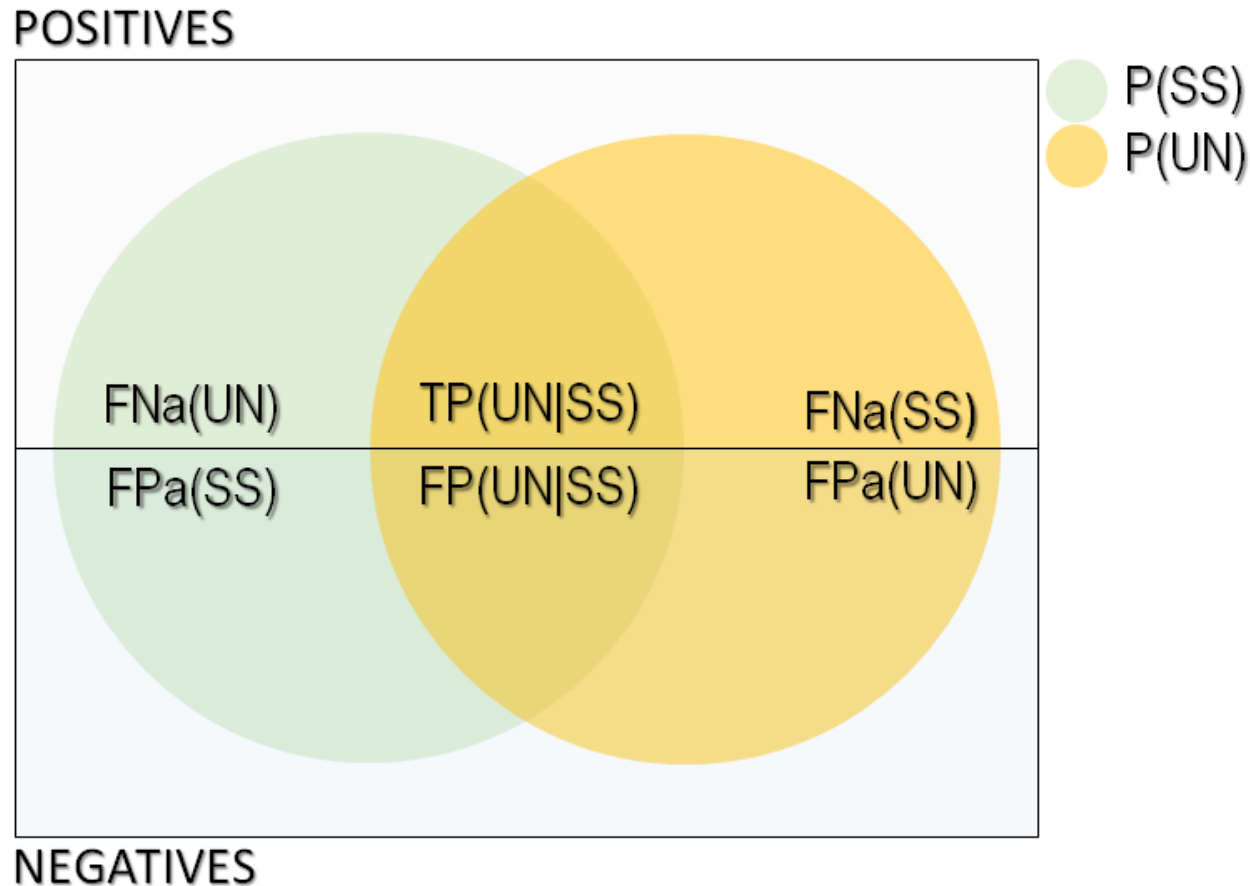
# Minimum Number of False Positives Added by Unstructured Merge – $FPa(UN)$ *Ordering Conflicts*



$$FPa(UN) = P(UN) - (FP(UN|SS) + TP(UN|SS)) - FNa(SS)$$

$$FP(UN|SS) + TP(UN|SS) \text{ ???}$$

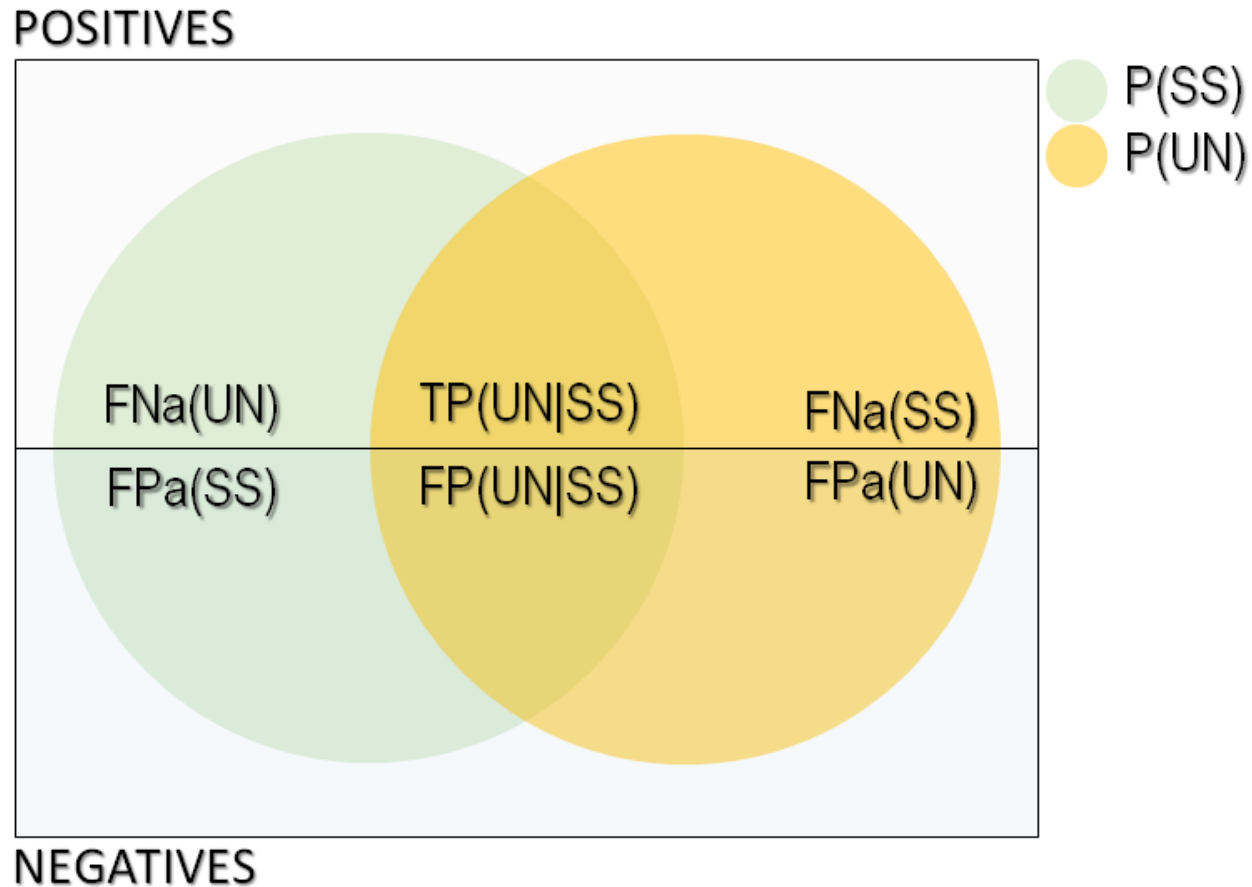
# Minimum Number of False Positives Added by Unstructured Merge – $FPa(UN)$ *Ordering Conflicts*



$$P(SS) = FP(UN|SS) + TP(UN|SS) + FPa(SS) + FNa(UN)$$

$$FP(UN|SS) + TP(UN|SS) + FPa(SS) = P(SS) - FNa(UN)$$

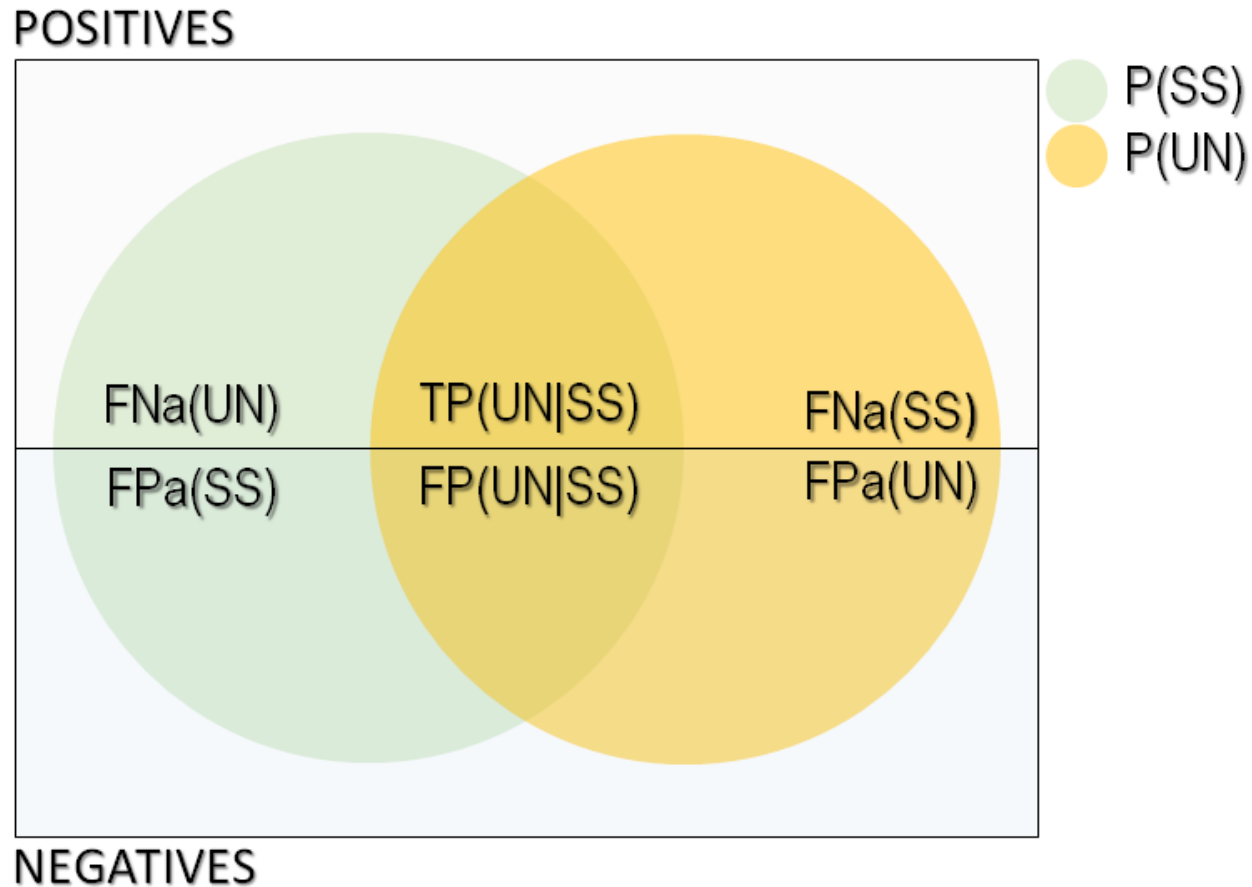
# Minimum Number of False Positives Added by Unstructured Merge – $FPa(UN)$ *Ordering Conflicts*



$$FP(UN|SS) + TP(UN|SS) + FPa(SS) = P(SS) - FNa(UN)$$

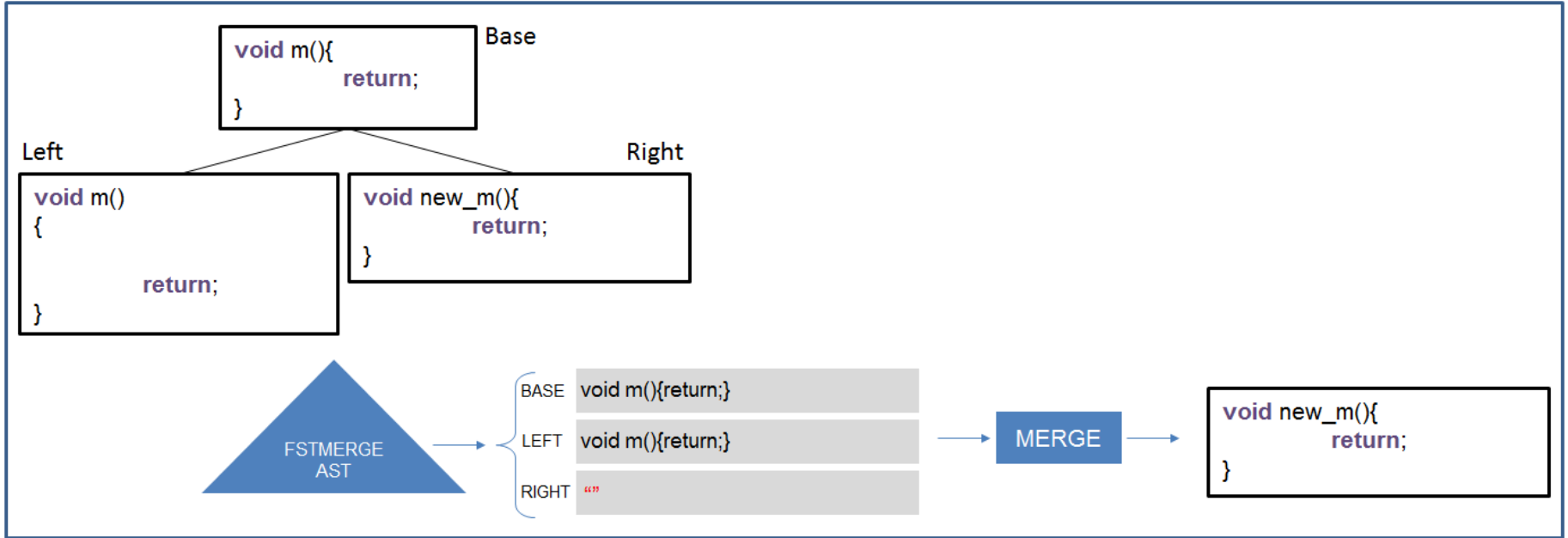
$$FP(UN|SS) + TP(UN|SS) \leq P(SS) - FNa(UN)$$

# Minimum Number of False Positives Added by Unstructured Merge – $FPa(UN)$ *Ordering Conflicts*

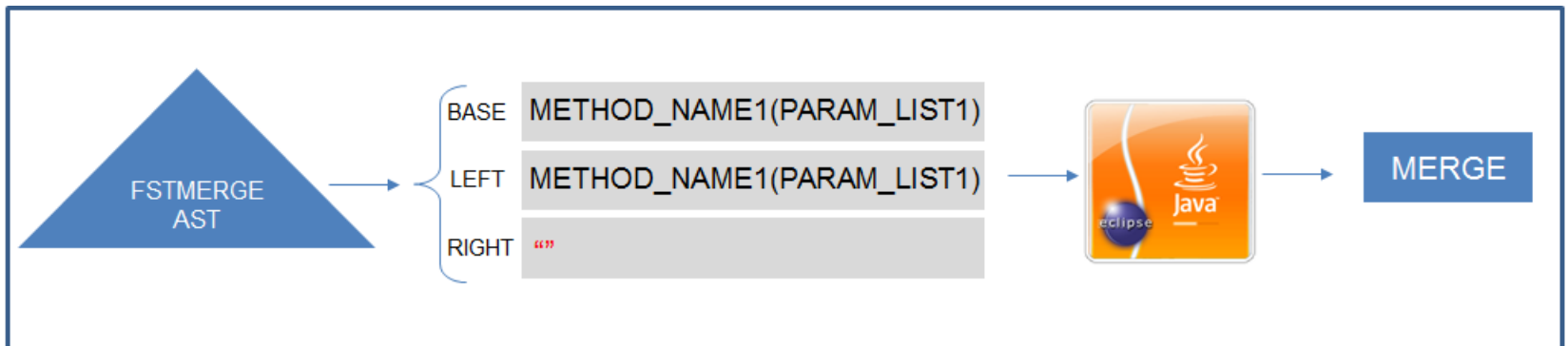


$$FPa(UN) \geq P(UN) - P(SS) + FNa(UN) - FNa(SS)$$

# improving FSTMerge tool



(ignoring spacings)



(using compilation features)



# Threats to Validity

---

- Construct: integration effort mainly based on the number of false positives; metrics are approximations
- Internal: selection of merge scenarios; discarded files
- External: only open-source Java projects

# The Structured Merge Approach

JAVA

GRAMMAR

stmt  $\rightarrow$  identifier = exp

stmt  $\rightarrow$  return exp

stmt  $\rightarrow$  if exp compoundstmt

stmt  $\rightarrow$  if exp compoundstmt else compoundstmt

compoundstmt  $\rightarrow$  { stmt S }

stmt S  $\rightarrow$  stmt ; stmt S

stmt S  $\rightarrow$   $\epsilon$

## Structured Merge with Auto-Tuning: Balancing Precision and Performance

Sven Apel, Olaf Leßenich, and Christian Lengauer  
University of Passau, Germany  
{apel, lessenic, lengauer}@fim.uni-passau.de

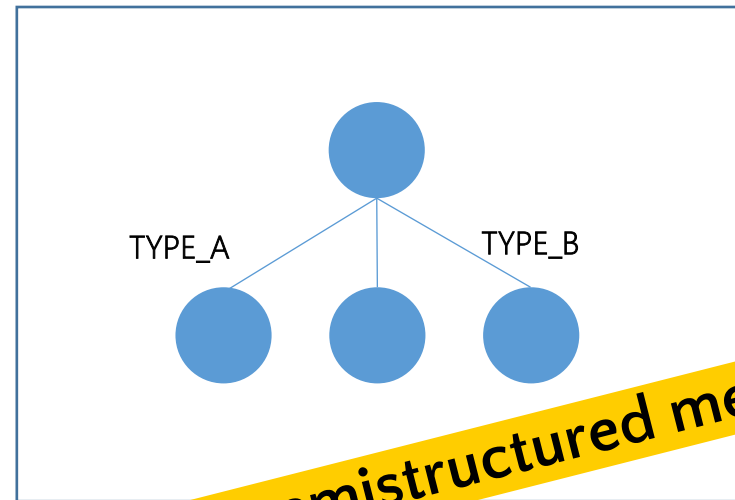
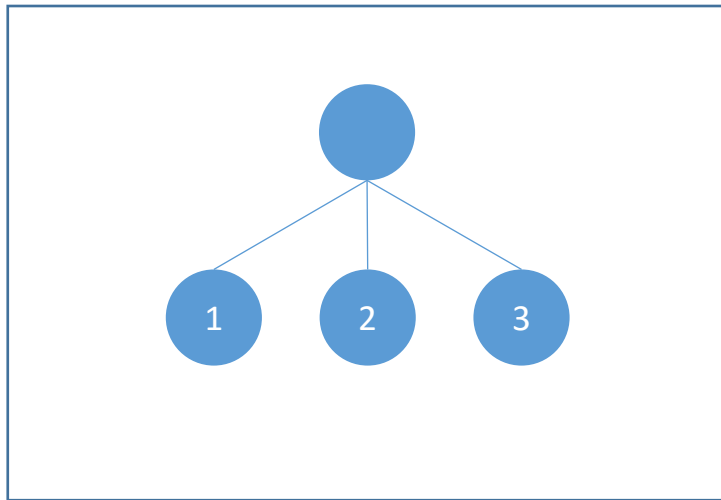
### ABSTRACT

Software-merging techniques face the challenge of finding a balance between precision and performance. In practice, developers use unstructured-merge (i.e., line-based) tools, which are fast but imprecise. In academia, many approaches incorporate information on the structure of the artifacts being merged. While this increases precision in conflict detection and resolution, it can induce severe performance penalties. Striving for a proper balance between precision and

important tools for programmers and software engineers not only in version control systems but also in product-line and model-driven engineering.

Contemporary software-merging techniques can be classified into (1) syntactic approaches and (2) semantic approaches. The former include (a) unstructured approaches that treat software artifacts as sequences of text lines and (b) structured approaches that are based on the artifacts' syntactic structure. In our attempt to push back the limits

- Matching of nodes depends on their syntactic category  
**similar to semistructured merge**
- Tree matching distinguishes between *ordered nodes* (which must not be permuted) and *unordered nodes* (which can be permuted safely), comparing the input trees level-wise



**similar to semistructured merge**

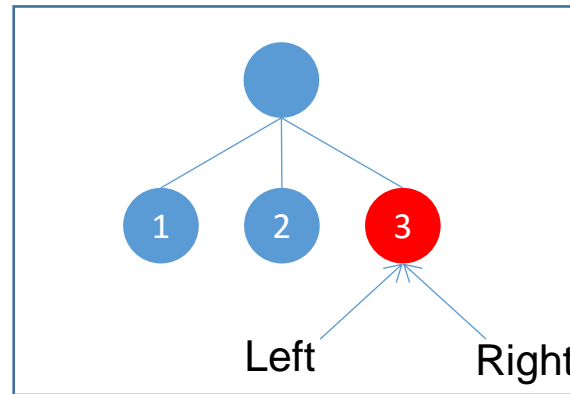
• Matching of nodes depends on **node name**

• Tree merging  
factoring out common nodes  
factoring out common nodes  
tree merging  
comparing the input  
edited one

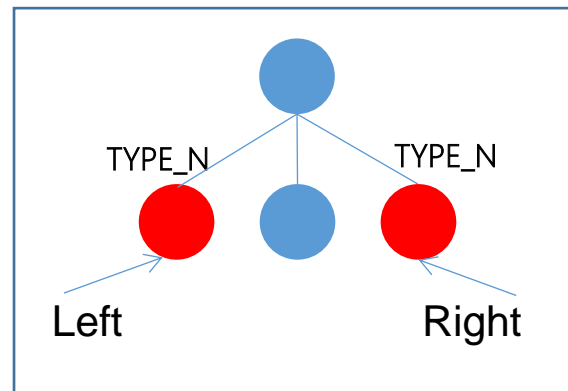
- Renaming/deletion conflicts
- Type ambiguity errors
- New artefact referencing

- Ordering Conflicts
- Duplicated declaration errors

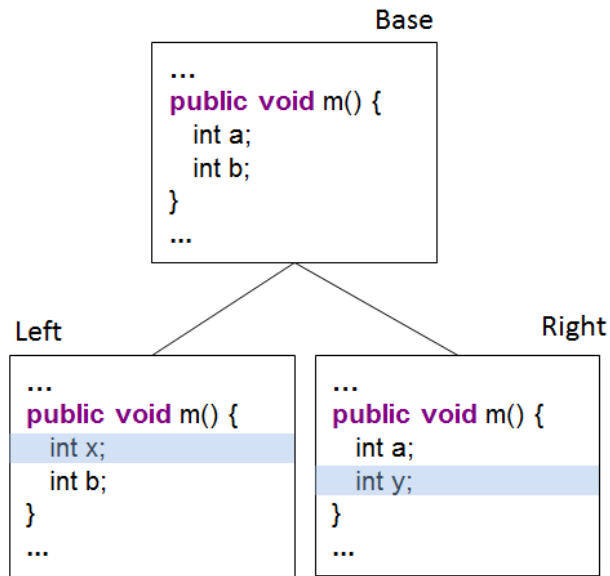
- For ordered nodes, if their position overlap, the nodes are flagged as conflicting



- Whether unordered nodes are in conflict, depends on their type and name



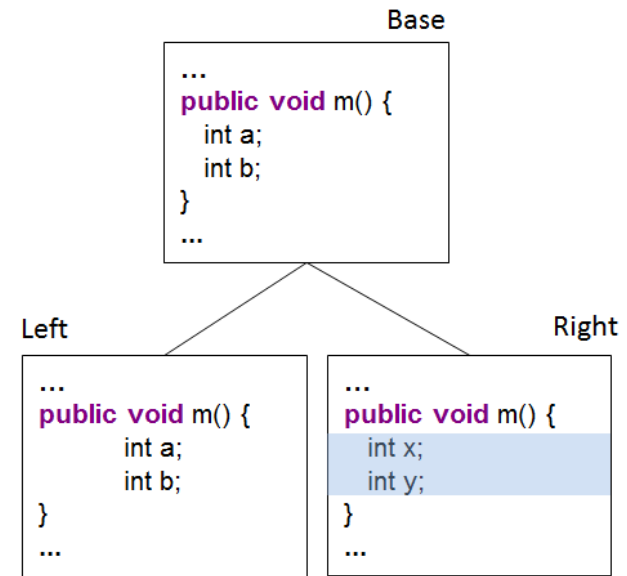
# False positives added by Unstructured/Semistructured Merge



Code  
integration

```
...  
public void m() {  
    <<<<<<<<  
    int x;  
    int b;  
    =====  
    int a;  
    int y;  
    >>>>>>>>  
}  
...
```

Consecutive Lines Conflict



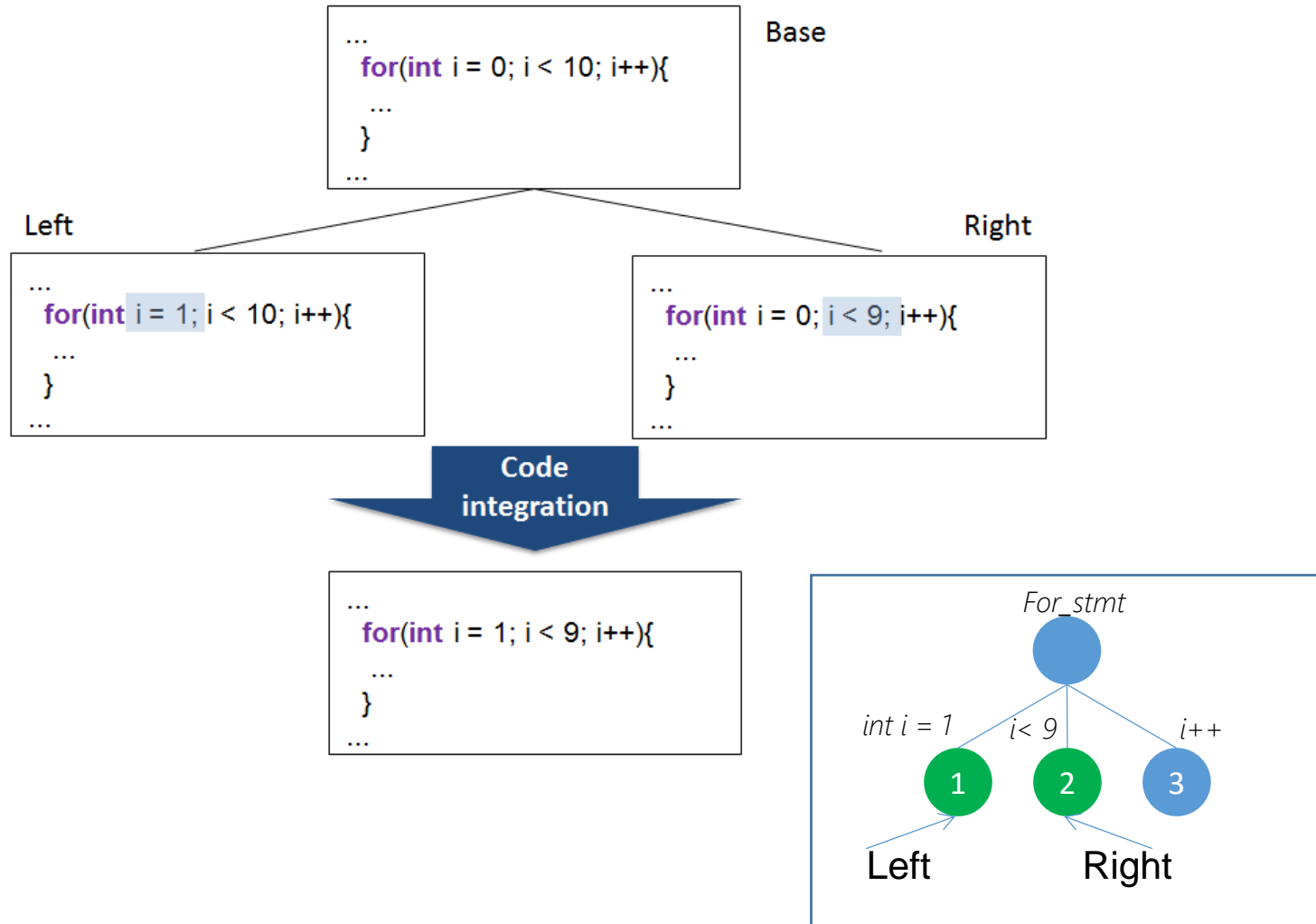
Code  
integration

```
...  
public void m() {  
    <<<<<<<<  
        int a;  
        int b;  
    =====  
    int x;  
    int y;  
    >>>>>>>>  
}  
...
```

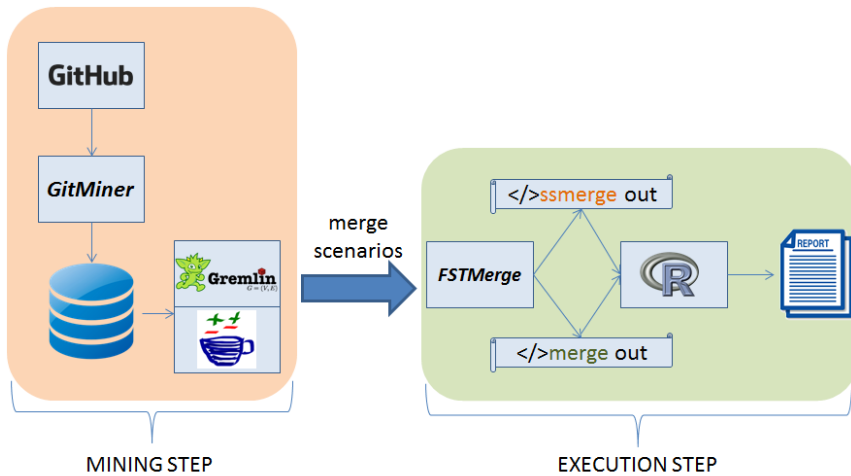
Spacing Conflict

# False negative added by Structured Merge

## *Edits to Same Statement*



# Pilot Experiment



34030 merge scenarios

50 Java projects

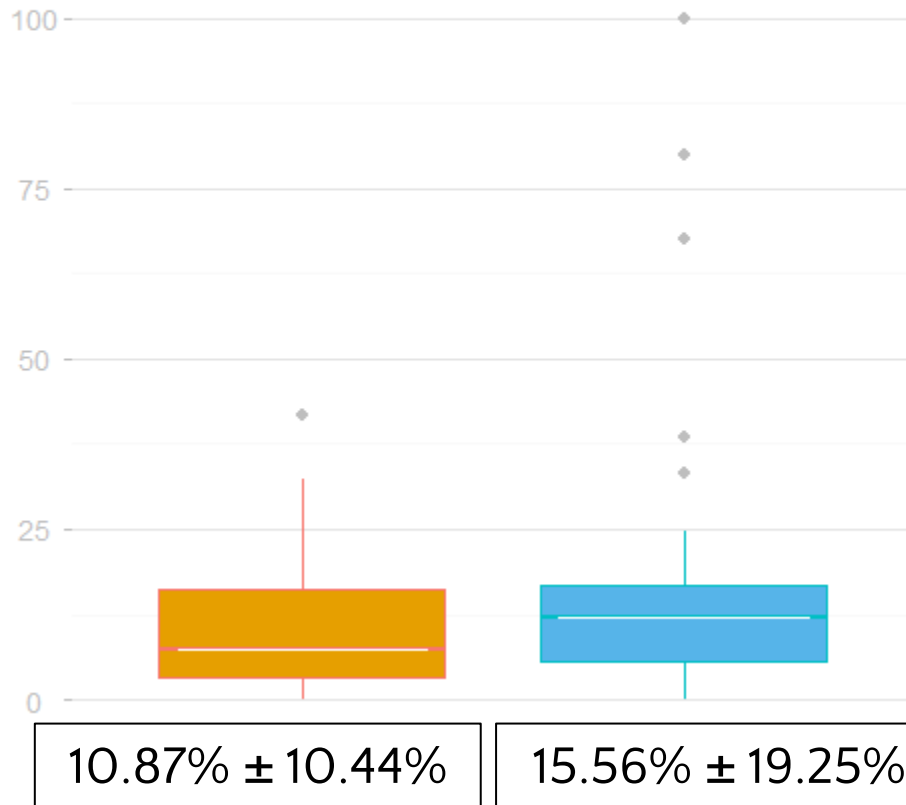
Project	File	Lines	Methods	Classes	Interfaces	Enums	Annotations	Imports	Exports
Project 1	File 1	100	10	1	0	0	0	0	0
Project 1	File 2	200	20	2	0	0	0	0	0
Project 1	File 3	300	30	3	0	0	0	0	0
Project 1	File 4	400	40	4	0	0	0	0	0
Project 1	File 5	500	50	5	0	0	0	0	0
Project 1	File 6	600	60	6	0	0	0	0	0
Project 1	File 7	700	70	7	0	0	0	0	0
Project 1	File 8	800	80	8	0	0	0	0	0
Project 1	File 9	900	90	9	0	0	0	0	0
Project 1	File 10	1000	100	10	0	0	0	0	0
Project 2	File 1	1100	110	11	0	0	0	0	0
Project 2	File 2	1200	120	12	0	0	0	0	0
Project 2	File 3	1300	130	13	0	0	0	0	0
Project 2	File 4	1400	140	14	0	0	0	0	0
Project 2	File 5	1500	150	15	0	0	0	0	0
Project 2	File 6	1600	160	16	0	0	0	0	0
Project 2	File 7	1700	170	17	0	0	0	0	0
Project 2	File 8	1800	180	18	0	0	0	0	0
Project 2	File 9	1900	190	19	0	0	0	0	0
Project 2	File 10	2000	200	20	0	0	0	0	0
Project 3	File 1	2100	210	21	0	0	0	0	0
Project 3	File 2	2200	220	22	0	0	0	0	0
Project 3	File 3	2300	230	23	0	0	0	0	0
Project 3	File 4	2400	240	24	0	0	0	0	0
Project 3	File 5	2500	250	25	0	0	0	0	0
Project 3	File 6	2600	260	26	0	0	0	0	0
Project 3	File 7	2700	270	27	0	0	0	0	0
Project 3	File 8	2800	280	28	0	0	0	0	0
Project 3	File 9	2900	290	29	0	0	0	0	0
Project 3	File 10	3000	300	30	0	0	0	0	0
Project 4	File 1	3100	310	31	0	0	0	0	0
Project 4	File 2	3200	320	32	0	0	0	0	0
Project 4	File 3	3300	330	33	0	0	0	0	0
Project 4	File 4	3400	340	34	0	0	0	0	0
Project 4	File 5	3500	350	35	0	0	0	0	0
Project 4	File 6	3600	360	36	0	0	0	0	0
Project 4	File 7	3700	370	37	0	0	0	0	0
Project 4	File 8	3800	380	38	0	0	0	0	0
Project 4	File 9	3900	390	39	0	0	0	0	0
Project 4	File 10	4000	400	40	0	0	0	0	0
Project 5	File 1	4100	410	41	0	0	0	0	0
Project 5	File 2	4200	420	42	0	0	0	0	0
Project 5	File 3	4300	430	43	0	0	0	0	0
Project 5	File 4	4400	440	44	0	0	0	0	0
Project 5	File 5	4500	450	45	0	0	0	0	0
Project 5	File 6	4600	460	46	0	0	0	0	0
Project 5	File 7	4700	470	47	0	0	0	0	0
Project 5	File 8	4800	480	48	0	0	0	0	0
Project 5	File 9	4900	490	49	0	0	0	0	0
Project 5	File 10	5000	500	50	0	0	0	0	0



# Preliminar Results

Added false positives by conflicts(%)

Spacing Consecutive Lines



Added false positives by merge scenarios(%)

Spacing Consecutive Lines

