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.
public class Member {
    String name;
    ...
}

Authentication

public class Member {
    String username;
    String name;
    ...
}

Research Group

public class Member {
    String email;
    String name;
    ...
}

Code integration (three-way merge)

public class Member {
    String email;
    String username;
    String name;
    ...
}
Conflicts are frequent and time-consuming!
Merge Approaches

unstructured merge

structured merge

JAVA

```
stmt → identifier = exp
stmt → return exp
stmt → if exp compoundstmt
stmt → if exp compoundstmt else compoundstmt
compoundstmt → { stmts }
stmts → stmt ; stmts
stmts → ε
```
special structure merge

JAVA

GRAMMAR

\[
\begin{align*}
\text{stmt} & \rightarrow \text{id} : \text{exp} \\
\text{stmt} & \rightarrow \text{return} \ \text{exp} \\
\text{stmt} & \rightarrow \text{if} \ \text{exp} \ \text{compoundstmt} \\
\text{stmt} & \rightarrow \text{if} \ \text{exp} \ \text{compoundstmt} \ \text{else} \ \text{compoundstmt} \\
\text{compoundstmt} & \rightarrow \{ \ \text{stmt} S \} \\
\text{stmt} S & \rightarrow \text{stmt} \ , \ \text{stmt} S \\
\text{stmt} S & \rightarrow \epsilon
\end{align*}
\]
public class Member {
    String name;
    String username;
    String email;
    ...
}

UNSTRUCTURED MERGE

public class Member {
    String username;
    String email;
    String name;
    ...  
}
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 \to identifier = exp  
stmt \to return exp  
stmt \to if exp compoundstmt  
stmt \to if exp compoundstmt else compoundstmt  
compoundstmt \to \{ stmt S \}  
stmt S \to stmt ; stmt S  
stmt S \to \epsilon
```

**DVCS**

**GitHub**

**Conflict Handlers**
unstructured merge

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

GitHub

GitMiner

merge scenarios

Gremlin

EXECUTION STEP

FSTMerge

<>ssmerge out

<>merge out

REPORT
Mining Step

Project selection criteria

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


- 21,430 commits
- 41 branches
- 115 releases
- 988 contributors
merge scenario
<table>
<thead>
<tr>
<th>3266 merge scenarios</th>
</tr>
</thead>
<tbody>
<tr>
<td>60 projects</td>
</tr>
</tbody>
</table>
public class Member {
    String email;
    String username;
    String name;

    // Textual conflict
    public String toString() {
        return this.name + email;
    }

    // Conflicting LOC
    public String toString() {
        return this.name + username;
    }

    // Conflicting file
}

Execution Step

Textual conflict
Conflicting LOC
Conflicting file
Evaluation Results

Overall results:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total <em>unstructured merge</em> conflicts</td>
<td>18021</td>
</tr>
<tr>
<td>Total <em>semistructured merge</em> conflicts</td>
<td>14320</td>
</tr>
</tbody>
</table>

- 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):

<table>
<thead>
<tr>
<th></th>
<th>Reduction by</th>
<th>In Merge Scenarios</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Textual conf.</td>
<td>34%</td>
<td>60%</td>
<td>21%</td>
</tr>
<tr>
<td>Conf. LOC</td>
<td>61%</td>
<td>82%</td>
<td>22%</td>
</tr>
<tr>
<td>Conf. files</td>
<td>28%</td>
<td>72%</td>
<td>12%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>Reduction by</th>
<th>In Merge Scenarios</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Textual conf.</td>
<td>62%</td>
<td>55.2%</td>
<td>24%</td>
</tr>
<tr>
<td>Conf. LOC</td>
<td>81%</td>
<td>71.1%</td>
<td>14%</td>
</tr>
<tr>
<td>Conf. files</td>
<td>66%</td>
<td>47.9%</td>
<td>25%</td>
</tr>
</tbody>
</table>
p-value < 0.05
Wilcoxon-Signed Rank Test
There is significant reduction!
Comparison with previous studies

<table>
<thead>
<tr>
<th>Author</th>
<th>Unstructured Merge Scenario Conflict Rate</th>
<th>Semistructured Merge Scenario Conflict Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brun et al.’11</td>
<td>16%</td>
<td>-</td>
</tr>
<tr>
<td>Kasi and Sarma’13</td>
<td>7% - 16%</td>
<td>-</td>
</tr>
<tr>
<td>Our replication</td>
<td>7%</td>
<td>3%</td>
</tr>
</tbody>
</table>

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

Semistructured merge reduces the metrics

**semistructured merge** conflict
(from project JUnit)

Conflicting code legibility
... <<<<<<<<
public static void validateMem...

=======
public ColumnDefinition getColu...

public ColumnDefinition getColumnDef...

>>>>>>>
<<<<<<<<<<<
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 getColumnDef...
...
public ColumnDefinition getColumnDe...

for (ColumnDefinition def : column_metadata.values())

=======
for (ColumnDefinition def : column_metadata.values())

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

Semistructured merge reducing integration effort when compared to unstructured merge?

Number of Conflicts reached up to 99% correlation when compared to the actual merge effort.
unstructured merge conflict = semistructured merge conflict
(from project kotlin)
increased numbers due to renamings or deletions
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.

Goguen and Meseguer 82
increased numbers due to renamings or deletions

The attributes are unlikely to change

I hope no one edit this signature

BASE

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

LEFT

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

RIGHT

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

Code Integration

UNSTRUCTURED MERGE

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

SEMISTRUCTURED MERGE

```java
<<<<<<<<<
=======
public void Init(Address address,
TransactionSettings transactionSettings,
Func<bool> commitTransation)
{
    this.settings = transactionSettings;
    this.address = address;
} >>>>>>>>
...
Integration Effort and Correcteness
Semistructured merge vs. Unstructured merge

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

```java
package util;
class Stack {
    Object top() { return data.getFirst(); }
}

package util;
class Stack {
    LinkedList data = new LinkedList();
    void push(Object obj) { data.addFirst(obj); }
    Object pop() { return data.removeFirst(); }
    Object top() { return data.getFirst(); }
}
```
Semistructured ≠ Unstructured
Outside methods declarations.
public class Calc {
...

public int sum(int a, int b) {
    return a + b;
}

public int doMath(int a, int b) {
    return (a + b)^2;
}

public int doMath(int a, int b) {
    return a + b;
}

public int sum(int a, int b) {
    return a + b;

ordering conflict (unstructured merge)

renaming/deletion conflict (semistructured merge)
Duplicated Declaration Error
(Unstructured Merge)

```java
public class Calc {
    int doMath(int a, int b) {
        return a*b;
    }

    int fib(int n) {
        if (n <= 1) return n;
        else return fib(n-1) + fib(n-2);
    }

    int doMath(int a, int b) {
        return a+b;
    }
}
```

Duplicate method doMath(int, int) in type Calc
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
Execution and Analysis Steps

- **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

**upper bound**

**lower bound**

real value

real value
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?

There is no significant difference between unstructured and semistructured merge in terms of added false positives by conflicts. The results show that the semistructured merge has a slightly lower percentage of added false positives compared to the unstructured merge, but the difference is not statistically significant (p-value > 0.05).

For added false positives by merge scenarios, the results indicate a significant difference (p-value < 0.05). The semistructured merge has fewer added false positives compared to the unstructured merge.
simple ordering conflicts

(from project gradle)

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

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

(from project jedis)

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();
}
crosscutting ordering conflicts

...
false positive renaming conflict

```java
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);
    ...
}
```

```java
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);
    ...
}
```

(from project lucene-solr)
suggestions for **improving** FSTMerge tool

```plaintext
void m(){
    return;
}

void m(){
    return 10;
}

void new_m(){
    return;
}

void m(){
    return 10;
}

...;

void new_m(){
    return;
}
```

(keeping the two versions)
indentation renaming/deletion conflict

(from project k-9)
suggestions for **improving** FSTMerge tool

```c
void m()
{
  return;
}

void new_m()
{
  return;
}
```

(ignoring the spacings)
true positive renaming conflict

```java
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) ...
}
```

(from project cassandra)
RQ2 - When compared to unstructured merge, does semistructured merge compromise integration correctness by missing more task interferences?

Added false negatives by conflicts(%)  

<table>
<thead>
<tr>
<th>Unstructured</th>
<th>Semistructured</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.62% ± 16.12%</td>
<td>1.66% ± 7.32%</td>
</tr>
</tbody>
</table>

Added false negatives by merge scenarios(%)  

<table>
<thead>
<tr>
<th>Unstructured</th>
<th>Semistructured</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.88% ± 1.07%</td>
<td>0.18% ± 0.39%</td>
</tr>
</tbody>
</table>

*p-value < 0.05*  
There is significant difference!
**FNa(SS)**

real value

[5x]

**FNa(UN)**

real value
tracking false negatives

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

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());
    }
  }

  
}

new element referencing edited one
(from project AntennaPod)
suggestions for improving FSTMerge tool

FSTMERGE AST

\{ LEFT \\
RIGHT \}

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

CONFLICT

(using compilation features)
suggestions for **improving** FSTMerge tool

(FSTMERGE AST) → \{ LEFT, RIGHT \} NEW NODE

EDITED NODE

NEW NODE REFER EDITED NODE ?

CONFLICT

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

Thanks!
● 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 have been performed or that are realistic considering the revision history
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
<table>
<thead>
<tr>
<th></th>
<th>Semistructured ↓</th>
<th>Semistructured = Unstructured</th>
<th>Unstructured ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Textual Conf.</td>
<td>1804 (55.2%)</td>
<td>1179 (36.1%)</td>
<td>283 (08.7%)</td>
</tr>
<tr>
<td>Conf. LOC</td>
<td>2323 (71.1%)</td>
<td>581 (17.8%)</td>
<td>362 (11.1%)</td>
</tr>
<tr>
<td>Conf. Files</td>
<td>1566 (47.9%)</td>
<td>1691 (51.8%)</td>
<td>9 (0.3%)</td>
</tr>
<tr>
<td><strong>Total Merge Scenarios</strong></td>
<td></td>
<td></td>
<td><strong>3266</strong></td>
</tr>
</tbody>
</table>
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. }

>>>>>>>
...
}

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. }

...
}

Renaming/Deletion Conflict
(Semistructured Merge)
Maximum False Negatives Added by Semistructured Merge – $FNa(SS)$

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 – \( FNa(SS) \)

**Duplicated Declaration Errors**

1. FSTMERGE AST
2. BASE
   - METHOD_NAME1(PARAM_LIST1)
3. LEFT
   - METHOD_NAME1(PARAM_LIST1)
4. RIGHT
5. CONFLICT? (UNSTRUCTURED)
6. NO
7. FALSE NEGATIVE CANDIDATE

Diagram showing the process of detecting false negatives in a merge scenario using an abstract syntax tree (AST). The steps involve comparing base, left, and right elements to identify conflicts that are not detected (false negatives).
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

FSTMERGE

AST

import java.util.List

import java.awt.List

CONFLICT? (UNSTRUCTURED)

YES

FALSE NEGATIVE CANDIDATE

REPORT
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**

\[
FPa(UN) = P(UN) - (FP(UN|SS) + TP(UN|SS)) - FNa(SS)
\]

\[
FP(UN|SS) + TP(UN|SS) \ ????
\]
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 -> identifier = exp
stmt -> return exp
stmt -> if exp compoundstmt
stmt -> if exp compoundstmt else compoundstmt
compoundstmt -> { stmt S }
stmt S -> stmt ; stmt S
stmt S -> ε
```

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

Sven Apel, Olaf Leßennich, 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.
- Renaming/deletion conflicts
- Type ambiguity errors
- New artefact referencing edited one

- Ordering Conflicts
- Duplicated declaration errors
- For ordered nodes, if their position overlap, the nodes are flagged as conflicting

![Ordered Nodes Diagram]

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

![Unordered Nodes Diagram]
False positives added by Unstructured/Semistructured Merge

Consecutive Lines Conflict

Spacing Conflict
False negative added by Structured Merge

Edits to Same Statement

\[
\text{... for(int } i = 1; i < 10; i++) \{
   \text{... }
\}
\]

\[
\text{... for(int } i = 0; i < 10; i++) \{
   \text{... }
\}
\]

Code integration

\[
\text{... for(int } i = 1; i < 9; i++) \{
   \text{... }
\}
\]
Pilot Experiment

MINING STEP

GitHub

GitMiner

merge scenarios

EXECUTION STEP

\$</\$/ssmerge out

\$</\$/merge out

34,030 merge scenarios

50 Java projects
Preliminary Results

**Added false positives by conflicts (%)**

- Spacing: 10.87% ± 10.44%
- Consecutive Lines: 15.56% ± 19.25%

**Added false positives by merge scenarios (%)**

- Spacing: 1.46% ± 1.63%
- Consecutive Lines: 2.53% ± 2.84%