Should We Replace Our Merge Tools?

Guilherme Cavalcanti

Advisor: Paulo Borba
Collaborative development is a common characteristic of today’s software projects...

Task B
-M Member.java

Task A
-M Member.java
... but it might lead to merge conflicts

```java
public class Member{
    String username;
    String email;
    String name;
    ...
}
```
public class CryptoUtils {

    private static final String TRANSFORMATION = "AES/CBC/PKCS7"

    private static final SecretKey SECRETKEY = CryptoKey.getFIPS186_3();

    public static void encrypt(File inputFile, File outputFile)
        doCrypto(Cipher.ENCRYPT_MODE, inputFile, outputFile);
    }

    public static void decrypt(File inputFile, File outputFile)
        doCrypto(Cipher.DECRYPT_MODE, inputFile, outputFile);
    }
}
Structured merge

Full Parse Trees
Semistructured merge

Simplified Parse Trees

Leaves as Plain Text

```
23 final String TRANSFORMATION
27 public static void encrypt(File input
28 doCrypto(Cipher.ENCRYPT_MODE, input)
29 )
```
### Semistructured merge vs. Unstructured merge

<table>
<thead>
<tr>
<th>Author</th>
<th>Average Reduction of Merge Conflicts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apel et al’11</td>
<td>34%</td>
</tr>
<tr>
<td>Cavalcanti et al’15</td>
<td>62%</td>
</tr>
</tbody>
</table>

### Structured merge vs. Unstructured merge

<table>
<thead>
<tr>
<th>Author</th>
<th>Average Reduction of Merge Conflicts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apel et al’12</td>
<td>61%</td>
</tr>
</tbody>
</table>
Unstructured Merge False Positive

```java
public class Calculator {
    public int sum(int a, int b) {
        return a + b;
    }

    public int sub(int a, int b) {
        return a - b;
    }
}
```

Ordering Conflict
To understand impact on productivity and quality...

**false positives**: unnecessary integration effort

**false negatives**: build or behavioral errors
Semistructured and Structured Merge False Negative

```java
import java.util.*;
import java.awt./*;

public class TestTypeAmbiguity {
    public static void main(String[] args) {
        List list;
    }
}
```

Type Ambiguity Error
Semistructured and Structured Merge
False Positive

```java
public class Calculator{
    public int calculate(int a, int b)
    {
        return (a + b)*2;
    }
}

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

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

Renaming Conflict
Thesis Outline

Unstructured merge **vs.**

Semistructured merge **vs.**

Structured merge
Evaluating and Improving Semistructured Merge

Unstructured merge

Semistructured merge

RQ1. When compared to unstructured merge, does semistructured merge reduce unnecessary integration effort by reporting fewer false positives?
RQ2. When compared to unstructured merge, does semistructured merge compromise integration correctness by having more false negatives?
Comparing additional false positives and false negatives from one strategy in relation to the other

```
<<<<<<<
  public String username;
=======
  public String email;
>>>>>>>
```

<table>
<thead>
<tr>
<th>Unstructured</th>
<th>Semistructured</th>
</tr>
</thead>
<tbody>
<tr>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

Unstructured merge *additional* false positive
(Ordering conflict)
Experimental Design

MINING STEP

GitHub

GitMiner

merge scenarios

FSTMerge

Kdiff3

execution step

list of false positives and false negatives candidates

ANALYSIS STEP

Eclipse

Java

R

REPORT
| Scenario | 34,030 merge scenarios | 50 Java projects |
- **aFP(SS)** - Overestimated number of additional false positives of semistructured merge

- **aFN(SS)** - Overestimated number of additional false negatives of semistructured merge

- **aFP(UN)** - Underestimated number of additional false positives of unstructured merge

- **aFN(UN)** - Underestimated number of additional false negatives of unstructured merge
upper bound  real value  lower bound  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 integration effort by reporting fewer false positives?

<table>
<thead>
<tr>
<th></th>
<th>False Positives</th>
<th>Merge Scenarios with False Positives</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unstructured Merge</strong></td>
<td>43.5% ± 19% <em>(AVG ± SD)</em></td>
<td>6.6% ± 6.1%</td>
</tr>
<tr>
<td><strong>Semistructured Merge</strong></td>
<td>30.2% ± 20.7%</td>
<td>3.1% ± 3.5%</td>
</tr>
</tbody>
</table>
Different conflicts demand different effort
Semistructured merge additional false positives easier to resolve

```java
public void removeAllAttachments() {
    runOnUiThread(new Runnable() {
        public void run() {
            for (int i = 0, count = mAttachments.getChildren(); i < count; i++) {
                mAttachments.removeView(mAttachments.getChildAt(i));
            }
        }
    });
}

public void removeAllAttachments() {
    runOnUiThread(new Runnable() {
        public void run() {
            for (int i = 0, count = mAttachments.getChildren(); i < count; i++) {
                mAttachments.removeView(mAttachments.getChildAt(i));
            }
        }
    });
}
```

(from project k-9)
Unstructured merge additional false positives harder to resolve

(from project cassandra)
Semistructured merge reduced the overall number of reported conflicts, has fewer additional false positives, and they are easier to understand and resolve.
RQ2. When compared to unstructured merge, does semistructured merge compromise integration correctness by having more false negatives?

<table>
<thead>
<tr>
<th></th>
<th>False Negatives</th>
<th>Merge Scenarios with False Negatives</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unstructured Merge</strong></td>
<td>9.6% ± 16.3% (AVG ± SD)</td>
<td>0.9% ± 1.1%</td>
</tr>
<tr>
<td><strong>Semistructured Merge</strong></td>
<td>20.6% ± 21.3%</td>
<td>4.4% ± 5.5%</td>
</tr>
</tbody>
</table>
Different false negatives have different impact on correctness
Unstructured merge additional false negatives easier to track

duplicated declaration error
(from project lucene-solr)
Semistructured merge additional false negatives harder to detect

```java
public class PlaybackService{
    ...

    private boid setSatus(PlayerStatus newStatus){
        ...
        bluetoothNotifyChange();
    }

    private void bluetoothNotifyChange(){
        ...
        if(queue!= null){
            i.putExtra("ListSize", queue.size());
        }
        ...
    }
    ...
}
```

new element referencing edited one  
(from project AntennaPod)
Semistructured merge has more additional false negatives, and they are harder to detect and resolve.
An Improved Semistructured Tool

https://github.com/guilhermejccavalcanti/jFSTMerge
Features

- Modules for each known kind of false positive and false negative

- Ensure that, whenever possible, it is not worse than an unstructured merge tool

- Universally applicable to files it cannot process

- Able to resolve conflicts due to code indentation
Features

- Full and transparent integration with *git*

```plaintext
crowdoctor@Plague:~
git merge
```

```plaintext
1 file changed, 2 insertions(+), 2 deletions(-)
crowdoctor@Plague:~/fusquinha$ git checkout master
Switched to branch 'master'
crowdoctor@Plague:~/fusquinha$ git merge left
changed source code builtin/merge.c:1134 Updating b803596..e1d2fd8
Fast-forward
diff called here? merge.c:411 base.java | 3 +++
1 file changed, 1 insertion(+), 2 deletions(-)
crowdoctor@Plague:~/fusquinha$ git merge right
changed source code builtin/merge.c:1134 reached merge.c:641 MERGE OUTPUT:
public class Test {
    public String name = "Daniel";
    long fibo(int n) {
        if (n < 2) {
            return n;
        } else {
            return fibo(n - 1) + fibo(n - 2);
        }
    }
    public int idade = 10;
}
```

Merge files finished.
Auto-merging base.java
Merge made by the 'recursive' strategy.
diff called here? merge.c:411 base.java | 24 ++
1 file changed, 14 insertions(+), 10 deletions(-)
crowdoctor@Plague:~/fusquinha$ cd
```
Type Ambiguity Error Handler

(using compiler features)
<table>
<thead>
<tr>
<th></th>
<th>Unstructured tool</th>
<th>Improved Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reported Conflicts</td>
<td>19,238 (100%)</td>
<td>9,343 (49%)</td>
</tr>
<tr>
<td>Additional False Positives</td>
<td>7,423 (100%)</td>
<td>0</td>
</tr>
<tr>
<td>Additional False Negatives</td>
<td>2,714 (100%)</td>
<td>2,489 (92%)</td>
</tr>
</tbody>
</table>
Performance Evaluation

- On 1,731 merge scenarios, 25 java projects

<table>
<thead>
<tr>
<th></th>
<th>Entire sample</th>
<th>Average per Scenario</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unstructured tool</td>
<td>45 sec</td>
<td>0.03 sec</td>
<td>0.09 sec</td>
</tr>
<tr>
<td>Improved tool</td>
<td>24 min</td>
<td>0.83 sec</td>
<td>2.47 sec</td>
</tr>
</tbody>
</table>

- Improved tool performance *non prohibitive*
  - Less than 1 second in 80% of the merge scenarios
  - More than 5 seconds in 2% of the merge scenarios
The Impact of Structure on Software Merging: Semistructured Versus Structured Merge

Structured merge (as in JDime)
Structured merge conflict (as in JDime)
Structured merge (as in JDime)

Unordered Nodes

Ordered Nodes

Semistructured merge

TYPE_N
class TempBuilder {...
    void createParameters(){...
        map.put("h", new C(5));
    }
    ...
}
```java
class TempBuilder {
    void createParameters() {
        key = "h";
        map.put(key, new C(5, 50));
    }
}
```
class TempBuilder {
void createParameters() {
    // update map
    map.put("h", new C(5, 50));
    ...
}
...
}

Semistructured

Structured

Merge

class TempBuilder {...
void createParameters() {...
<<<<<<<
    // update map
    map.put("h", new C(5));
    ========
    map.put("h", new C(5, 50));
    >>>>>>>>
    ...
}
...

Semistructured

Semistructured False Positive
(Consecutive Lines)
```java
class TempBuilder {
    void createParameters() {
        key = "j";
        map.put(key, new C(5, 50));
    }
}
```

### Structured

```java
class TempBuilder {
    void createParameters() {
        key = "j";
        map.put(key, new C(5, 50));
    }
}
```

### Semistructured

```java
class TempBuilder {
    void createParameters() {
        key = "j";
        map.put(key, new C(5));
        map.put("h", new C(5, 50));
    }
}
```

**Structured False Negative**
RQ1. How many conflicts arise when using semistructured and structured merge?
RQ2. How often do semistructured and structured merge differ with respect to the occurrence of conflicts?
RQ3. Why do semistructured and structured merge differ?
RQ4. Which of the two strategies reports fewer false positives?
RQ5.

Which of the two strategies has fewer false negatives?
RQ6. Does ignoring conflicts caused by changes to consecutive lines make the two merge strategies more similar?
Setup

**MERGE COMMIT**

- **SEMI**: ✓
- **STRUCT**: ✗

**Travis CI**

- **Building**
  - Build breaks
  - Build succeeds

- **Testing**
  - Fail
  - Pass

**Interference**

- False Negative
- False Positive

**Inspection**

- Undetermined Scenario
- No interference

**RQ1, RQ2, RQ3 and RQ6**

**RQ4 and RQ5**
S3M: to merge and match class level declarations (*unordered nodes*)

JDime: to merge expressions and statements (declarations’s body, *ordered nodes*)

```java
public class Bike {
    public int speed;
    public int gear;

    public Bike(int startSpeed, int startGear) {
        this.gear = startGear;
        this.speed = startSpeed;
    }

    public void speedUp(int increment) {
        this.speed += increment;
    }
}
```
Setup


Sample Selection
On 510 Java projects and 43,509 merge scenarios...
RQ1. How many conflicts arise when using semistructured and structured merge?

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total semistructured confs</td>
<td>4,732</td>
</tr>
<tr>
<td>Total structured confs</td>
<td>4,793</td>
</tr>
</tbody>
</table>

1.29% Difference!
Structured merge conflicts are granular

(from Project neo4j-framework)
RQ1. How many conflicts arise when using semistructured and structured merge?

<table>
<thead>
<tr>
<th>All scenarios</th>
<th>43,509</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conflicting <em>Semistructured</em> Merge Scenarios</td>
<td>1,007 (2.31%)</td>
</tr>
<tr>
<td>Conflicting <em>Structured</em> Merge Scenarios</td>
<td>814 (1.87%)</td>
</tr>
</tbody>
</table>

19.17% Difference!
**RQ2.** How often do semistructured and structured merge differ with respect to the occurrence of conflicts?

<table>
<thead>
<tr>
<th>Scenario Description</th>
<th>Number of Scenarios</th>
</tr>
</thead>
<tbody>
<tr>
<td>All scenarios</td>
<td>43,509</td>
</tr>
<tr>
<td>Only Conflicting <strong>Semistructured</strong> Merge Scenarios</td>
<td>223 (0.51%)</td>
</tr>
<tr>
<td>Only Conflicting <strong>Structured</strong> Merge Scenarios</td>
<td>30 (0.07%)</td>
</tr>
</tbody>
</table>

0.58%
RQ2. How often do semistructured and structured merge differ with respect to the occurrence of conflicts?

<table>
<thead>
<tr>
<th>Conflicting Scenarios</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Only Conflicting <strong>Semistructured</strong> Merge Scenarios</td>
<td>223 (20.86%)</td>
</tr>
<tr>
<td>Only Conflicting <strong>Structured</strong> Merge Scenarios</td>
<td>30 (2.81%)</td>
</tr>
</tbody>
</table>

23.67%
RQ3. Why do semistructured and structured merge differ?

When changes occur in overlapping text areas that correspond to different AST nodes
public static void main(String[] args) {
    ...
    Instance instance = ...
    VolumeManager fs = ...
    Accumulo.init(fs, instance, conf, app);
    final TabletServer server = ...
    MetricsSystemHelper.configure(...);
    Accumulo.init(fs, instance, conf, app);
    final TabletServer server = ...
    ...
}

public static void main(String[] args) {
    ...
    Instance instance = ...
    VolumeManager fs = ...
    Accumulo.init(fs, instance, conf, app);
    MetricsSystemHelper.configure(...);
    Accumulo.init(fs, instance, conf, app);
    final TabletServer server = ...
    ...
}

Semistructured

Structured

(from project accumulo)
**RQ3.** Why do semistructured and structured merge differ?

When changes occur in non-overlapping text areas that correspond to (a) different but incorrectly matched nodes and to (b) the same node
... ViewTuple<...> load = Loader.fxmlView(...)
    .viewModel(scopedViewModelD)
    .context(context)
    .provided(new Example1Scope())
    .load();
...

Semistructured

(Structured)

(from project mvvmFX)
(from project mvvmFX)
**RQ4.** Which of the two strategies reports fewer false positives?

<table>
<thead>
<tr>
<th>Merge Scenarios with False Positives</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Semistructured</strong></td>
</tr>
<tr>
<td>Merge</td>
</tr>
<tr>
<td>66</td>
</tr>
<tr>
<td><strong>Structured</strong></td>
</tr>
<tr>
<td>Merge</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>
RQ5. Which of the two strategies has fewer false negatives?

<table>
<thead>
<tr>
<th>Merge Scenarios with False Negatives</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Semistructured</strong></td>
</tr>
<tr>
<td>Merge</td>
</tr>
<tr>
<td><strong>Structured</strong></td>
</tr>
<tr>
<td>Merge</td>
</tr>
</tbody>
</table>
RQ6. Does ignoring conflicts caused by changes to consecutive lines make the two merge strategies more similar?

<table>
<thead>
<tr>
<th>All scenarios</th>
<th>43,509</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conflicting Semistructured Merge Scenarios</td>
<td>1,007 973</td>
</tr>
</tbody>
</table>

3.38% Reduction!

| Merge Scenarios in which Semistructured and Structured Merge Differ | 253 225 |

11.07% Reduction!
Should We Replace Our Merge Tools?

Accuracy

Syntactic Structure

Literature
Should We Replace Our Merge Tools?

Our Observations

Syntactic Structure

Accuracy

Unstructured

Ideals?

Structured

Unstructured
Thanks!