Understanding Collaboration Conflicts

Paola Accioly (prga@cin.ufpe.br)
Advisor: Paulo Borba (phmb@cin.ufpe.br)
Collaborative development environment

Task Assignment Board

Developer: Bob
Task: Authentication Feature

Developer: Alice
Task: Research Group Feature
public class Member{
    String name;
    String username;
    public String toString(){
        return this.username;
    }
}

public class Member{
    String name;
    public String toString(){
        return this.name;
    }
}
While merging, conflicts might occur...
Merge conflict

<<<<<<<<<<< Authentication
    String username;
    public String toString(){
        return this.username;
    }
} 

=======
    public String toString(){
        return this.name;
    }
} 

>>>>>>>>> Research Group
Build conflict

```java
public String toString(){
    return this.name;
}
...

public String toString(){
    return this.username;
}
```
Test conflict

```java
public String toString(){
    return this.name;
}
...
```

One test from the authentication feature failed after code integration
Previous studies show that conflicts occur frequently and resolving them is a time consuming and error prone task.

Which might impact both development’s productivity and the resulting product’s quality.
Such problem guides the development of tools and strategies that try to mitigate or minimize conflict resolution effort
FSTMerge - semistructured merge

Represent code artifacts as partial trees containing information (through an annotated grammar) about how nodes of certain types (methods, classes, fields, etc.), and its subtrees can be merged
Using FSTMerge

```java
public class Member{
    String name;
    String username;
}

public class Member{
    String name;
    public String toString(){
        return this.name;
    }
}
```
Using diff3 - unstructured merge

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

public class Member{
    String name;
    public String toString(){
        return this.name;
    }
}

<<<<<<< mine
String username;
=======
public String toString(){
    return this.name;
}
>>>>>>> yours
```
Using diff3 - unstructured merge

```java
public class Member{
    String name;
    String username;
}
```

```java
public class Member{
    String name;
    public String toString(){
        return this.name;
    }
}
```

```java
<<<<<<< mine
    String username;
=======
    public String toString(){
        return this.name;
    }
>>>>>>> yours

Ordering conflict

Merging
However, despite the existing evidence about conflicts frequency and their impact, the structure of the changes that lead to conflicts has not been studied yet
What are the most frequent language elements involved in conflicts?

Are frequent conflict patterns good conflict predictors?
Deriving and collecting the frequency of semistructured merge conflict patterns
Goal

Understand characteristics—such as structural patterns, causes, and frequency—of merge conflicts reported by semistructured merge tools when reproducing real merge scenarios from the development history of different software projects.
RQ1: What are the structural conflict patterns that can be found by a semistructured merge tool?

FSTMerge Java grammar

We checked all node annotations leading to conflicts

FSTMerge Java grammar

9 conflict patterns
Developers edit the same or consecutive lines of the same method or constructor - EditSameMC

```java
private List<WalkStep> getWalkSteps(List<State> states) {
    ...
    boolean optionsBefore = pathService.multipleOptionsBefore(edge, backState);
    boolean optionsBefore = currState.multipleOptionsBefore();
    ...
}
```

https://github.com/opentripplanner/OpenTripPlanner
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EditSameFD</td>
<td>Edit the same field declaration</td>
</tr>
<tr>
<td>SameSignatureMC</td>
<td>Add methods with the same signature</td>
</tr>
<tr>
<td>AddSameFD</td>
<td>Add field declarations with the same identifier</td>
</tr>
<tr>
<td>ModifiersList</td>
<td>Edit the same modifiers’ list</td>
</tr>
<tr>
<td>ImplementsList</td>
<td>Edit the same implements’ list</td>
</tr>
<tr>
<td>ExtendsList</td>
<td>Edit the same extends’ declaration</td>
</tr>
<tr>
<td>EditSameEnumConst</td>
<td>Edit the same Enum constant declaration</td>
</tr>
<tr>
<td>DefaultValueA</td>
<td>Edit the same annotation method default value declaration</td>
</tr>
</tbody>
</table>
RQ2: How frequently does each merge conflict pattern occur?

Number of conflicts for each conflict pattern reported while reproducing merge scenarios using FSTMerge
Different spacing conflicts

Base

1. void m() {
2. int x;
3. int y;
4. ...
5. }

Left

1. void m() {
2. {
3. int x;
4. int y;
5. ...
6. }

Right

1. void m(int a) {
2. int x;
3. int y;
4. ...
5. }
Consecutive lines conflicts

Base

1. void m() {
2.     int x;
3.     int y;
4.     ...
5. }

1. void m() {
2.     int z;
3.     int y;
4.     ...
5. }

1. void m(int a) {
2.     int x;
3.     int y;
4.     ...
5. }
RQ3: What kinds of code changes most likely lead to conflicts?

\[
\text{CHANGES} = \text{CHANGES}_{BA} + \text{CHANGES}_{CA} + \text{CHANGES}_{DA}
\]

\[
\text{CONFLICTS} = \text{CONFLICTS}_E + \text{CONFLICTS}_G
\]

\[
\text{NORMALIZED\_CONFLICTS} = \text{CONFLICTS} / \text{CHANGES}
\]
RQ4: How frequently do merge conflicts occur?

\[
\text{conflicting scenarios} = \frac{\text{merge scenarios with conflicts}}{\text{merge scenarios}}
\]
Study setup

Merge Scenarios

Conflict Analyzer

Metrics Report

Base

Left

Merge

Right

Adapted FSTMerge

Merge Conflicts

<<<<<<<<< mine
int x = 0;

=========
int x = 1;

>>>>>>>
>>>

yours
A pilot study revealed that around 13% of the conflicts were from the SameSignatureMC pattern

How come?
Why developers add methods with the same signature?

- Small methods
- Copied files
- Copied methods
- Renamed methods
RQ5: How frequent are the underlying causes of the SameSignatureMC pattern?

- Small methods
- Copied files
- Copied methods
- Renamed methods
- None of the above
Sample

- 123 Java projects from GitHub chosen by their popularity
  - > 500 stars
- Projects from different domains
  - game engines, databases, development frameworks and APIs, GPS app, e-commerce platforms
- Different sizes (4 - 1000 KLOCs)
- Different number of collaborators (20 - 112)
- Projects like JUnit, Jenkins, Cassandra and Gradle
Results and Discussion
RQ2: How frequently does each merge conflict pattern occur?

28,883 conflicts collected
RQ3: What kinds of code changes most likely lead to conflicts?

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Normalized numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>EditSameMC Nodes</td>
<td>0.30%</td>
</tr>
<tr>
<td>EditSameMC Chunks</td>
<td>0.26%</td>
</tr>
<tr>
<td>EditSameMC Lines</td>
<td>0.03%</td>
</tr>
<tr>
<td>SameSignatureMC</td>
<td>0.03%</td>
</tr>
<tr>
<td>EditSameFd</td>
<td>0.06%</td>
</tr>
<tr>
<td>AddSameFd</td>
<td>0.01%</td>
</tr>
<tr>
<td>EditSameEnumConst</td>
<td>0.07%</td>
</tr>
<tr>
<td>ExtendsList</td>
<td>0.04%</td>
</tr>
<tr>
<td>ModifiersList</td>
<td>0.06%</td>
</tr>
<tr>
<td>ImplementsList</td>
<td>Approx. 0%</td>
</tr>
</tbody>
</table>
**RQ3**: What kinds of code changes most likely lead to conflicts?

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Normalized numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>EditSameMC Nodes</td>
<td>0.30%</td>
</tr>
<tr>
<td>EditSameMC Chunks</td>
<td>0.26%</td>
</tr>
<tr>
<td>EditSameMC Lines</td>
<td>0.03%</td>
</tr>
<tr>
<td>SameSignatureMC</td>
<td>0.03%</td>
</tr>
<tr>
<td>EditSameFd</td>
<td>0.06%</td>
</tr>
<tr>
<td>AddSameFd</td>
<td>0.01%</td>
</tr>
<tr>
<td>EditSameEnumConst</td>
<td>0.07%</td>
</tr>
<tr>
<td>ExtendsList</td>
<td>0.04%</td>
</tr>
<tr>
<td>ModifiersList</td>
<td>0.06%</td>
</tr>
<tr>
<td>ImplementsList</td>
<td>Approx. 0%</td>
</tr>
</tbody>
</table>

There is no statistically significant difference between EditSameMC lines, EditSameFd and ModifiersList.
Most merge conflicts happen when developers edit the same or consecutive lines of the same method…

However, perhaps awareness tools should be more careful with class field, and modifier list edits as well.
Mitigate Syde’s information overload

```java
@Entity has been concurrently added/changed, but not committed. Author: anja

* Gets a new ChangeAlert for a given user
  *
  * @return ChangeAlert object
  */

public ChangeAlert getChangeAlert(String author, long timestamp)
  throws RemoteException {
  return new NotifierFacade().getChangeAlert(author, timestamp);
}
**RQ4: How frequently do merge conflicts occur?**

We analyzed a total of 70,047 merge scenarios from 123 Java projects.

<table>
<thead>
<tr>
<th></th>
<th>Total</th>
<th>Median</th>
<th>IQR</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR</td>
<td>9.38%</td>
<td>6.64%</td>
<td>8.81%</td>
</tr>
<tr>
<td>CR without DS</td>
<td>9.04%</td>
<td>6.50%</td>
<td>8.72%</td>
</tr>
<tr>
<td>CR without CL</td>
<td>8.64%</td>
<td>6.39%</td>
<td>7.76%</td>
</tr>
<tr>
<td>CR without DS and CL</td>
<td>8.39%</td>
<td>6.00%</td>
<td>7.21%</td>
</tr>
</tbody>
</table>
Previous studies using the unstructured merge showed higher conflicting scenario rates (approximately 13% and 16%)
Sophisticated merge tools reduce the number of conflicts and might improve productivity and quality
RQ5: How frequent are the underlying causes of the SameSignatureMC pattern?

1,505 conflicts collected
**RQ5:** How frequent are the underlying causes of the SameSignatureMC pattern?

1,505 conflicts collected
Developers do not take full advantage of proper code version and end up creating conflicts.
A tool that enables partial merges, such as Commit Bubbles

1. Break commits into smaller changes
2. Merge only selected commits
RQ5.1: How many developers are involved in merges, conflicting merges, and merge conflicts?

For the majority of cases (>50%), merges, conflicting merges and merge conflicts involved more than two developers
Although the number of developers involved in merge conflicts does not measure directly the effort to resolve them, we believe that solving conflicts involving a single developer is probably easier than solving conflicts involving more developers.
Threats to validity

- FSTMerge false negatives - Import List
- FSTMerge false positives - Renaming
- We might be missing merge commits in Git history due to commands that rewrite Git history such as rebase
Most semistructured merge conflicts happen when developers edit the same or consecutive lines of the same method.
However, even when there are no merge conflicts, editing the same method can cause other types of conflicts such as build and test conflicts.
In addition, editing directly dependent methods are also a frequent cause for build and test conflicts.
Are those good conflict predictors?
Analyzing conflict predictors precision and recall
Goal

Analyze editions to the same method—EditSameMC—and editions to directly dependent methods—EditDepMC—effectiveness as conflict predictors during the development history of different Java projects
Strategy

Reproduce merge scenarios while collecting conflict predictor instances together with merge, build, and test conflict instances to compute how often the predictors are associated to conflicts
To compute merge conflicts we continue to use FSTMerge tool and our conflict patterns
For establishing build and test conflicts ground truth, we rely on the status of building and testing processes executed by the Travis CI service.
How frequently predictors are associated to conflicts

Analyzed Merge Scenarios

Predictors: False positives, True positives, False negatives
Conflicts: True negatives
RQ1: How precise are EditSameMC and EditDepMC predictors?

$$Precision = \frac{true \ positives}{true \ positives + false \ positives}$$
RQ2: How many conflicts can we avoid by detecting EditSameMC and EditDepMC predictors?

\[
\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}
\]
RQ3: Why EditSameMC and EditDepMC instances are not associated with merge, build, or test conflicts?

Considering a sub-sample of the false positives we conduct a manual analysis to check if there are contributions’ interferences missed by our oracle (Travis CI testing process)
**RQ4:** What other change patterns are associated with conflicts?

We identify the false negatives’ conflict cause

![Diagram showing the relationship between predictors, conflicts, true positives, false positives, false negatives, and true negatives.](image-url)
Study Setup

1- Filter projects from GitHub, Travis CI, and Maven to identify build and test conflicts

2- Reproduce merge scenarios while collecting merge conflicts and conflict predictors

3- Manual analysis of false positives and false negatives
Results and discussion
Sample

- 45 analyzed projects
- 5,647 merge scenarios
- 290 merge conflicts
- 84 build conflicts
- 5 test conflicts
**RQ1 and RQ2 - conflict predictors precision and recall**

<table>
<thead>
<tr>
<th></th>
<th>Both predictors</th>
<th>EditSameMC</th>
<th>EditDepMC</th>
<th>Both predictors WDS</th>
<th>EditSameMC WDS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Precision</strong></td>
<td>56.29%</td>
<td>55.51%</td>
<td>8.85%</td>
<td>57.99%</td>
<td>56.71%</td>
</tr>
<tr>
<td><strong>Recall</strong></td>
<td>83.62%</td>
<td>82.45%</td>
<td>13.15%</td>
<td>82.67%</td>
<td>80.85%</td>
</tr>
</tbody>
</table>
RQ3: False positives analysis

20 manually analyzed cases

10 EditSameMC
- 5 have interference
- 5 do not have interference

10 EditDepMC
- 3 have interference
- 8 do not have interference
RQ4: False negatives analysis

- 56 false negatives
- 20 merge conflicts
- 33 build conflicts
- 3 test conflicts
3 test conflicts

1: updated test case executing an edited method

2: Methods with more than one level of dependency were concurrently edited
How effective are EditSameMC and EditDepMC?

These results can be used to guide different conflict awareness strategies depending on each team preferences:

- Conservative vs. Precise

<table>
<thead>
<tr>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>57.99%</td>
<td>82.67%</td>
</tr>
</tbody>
</table>
Strategies to improve precision and recall

1. Implement strategies to identify and ignore cases where clearly there is no interference
   a. Detect different spacing changes
   b. Detect refactoring changes

2. Implement strategies to identify possible interferences
   a. Identify information flow between contributions [Filho, 2017]
   b. Generate test cases to explore contributions interaction [Böhme, 2013]
Threats to validity

- Same threats from first study
- Using better tests would likely increase the precision and decrease the recall
Understanding Collaboration Conflicts

Characteristics

While merging, conflicts might occur...

Study setup

RQ2 - How frequently does each merge conflict pattern occur?

28,883 conflicts collected

How frequently predictors are associated to conflicts

Study Setup

How effective are EditSameMC and EditDepMC?
These results can be used to guide different conflict awareness strategies depending on each team preferences:

- Conservative vs. Precise

Thank you!

Paola Accioly (prga@cin.ufpe.br)
<table>
<thead>
<tr>
<th></th>
<th>Projects</th>
<th>Merge Scenarios</th>
<th>Conflicting Merge Scenarios</th>
<th>Conflict Rate</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brun et al 11</td>
<td>9</td>
<td>3635</td>
<td>572</td>
<td>16.7%</td>
<td>+- 11</td>
</tr>
<tr>
<td>Kasi and Sarma 13</td>
<td>4</td>
<td>1158</td>
<td>154</td>
<td>13.7%</td>
<td>+- 4.2</td>
</tr>
</tbody>
</table>
Add extra alarms to Palantír
Reduce Crystal’s number of build and test routines
We consider methods to be the same if the similarity value is greater than or equal to 70%. To choose this value, we executed our analyses considering 68 randomly selected projects from our original sample, using 3 different thresholds (>=60%, >=70%, and >=80%). We found out that, for the Renamed Method category, a total of 78.6% of the renamed methods in the sub sample fall in the >=80% category, and an additional >=11% is considered if we use the >=70% category. For the Copied Method category, 84.4% fall in the >=80% category, and an additional 8% is considered using the >=70% category. Hence, we considered 70% to be an acceptable threshold value, since we get most part of the renamed and copied methods (more than 80% similar), and we are still able to get some of the renamed, and copied methods having similarities between 70% and 80%.
Broadcasting API refactoring changes
Strategy

1. `void m (int a) {`
2. `    int x;`
3. `    int y;`
4. `    if(a > 0){`
5. `        x++;`
6. `    }`
7. `}

1- Annotate contributions

2- Run Joana
How effective is this approximation?
Research Questions [Filho, 2017]

- Research Question 1 (RQ1) - Configuration: Which SDG option is the most appropriate to identify information flow between merge scenario same-method contributions?
- Research Question 2 (RQ2) - Severity: Is there direct information flow between merge scenario same-method contributions? How often?
- Research Question 3 (RQ3) - Limitations: In which situations is there information flow and no interference?

Research Question 2 (RQ2) - Severity: Is there direct information flow between merge scenario same-method contributions? How often?

There is control information flow for 64.56% (51 out of 79) of the scenarios.
Research Question 3 (RQ3) - Limitations: In which situations is there information flow and no interference?