



Build Conflicts in The Wild



Léuson da Silva
lmps2@cin.ufpe.br



Paulo Borba
phmb@cin.ufpe.br



Arthur Pires
ahlp@cin.ufpe.br

Federal University of Pernambuco, Recife, Brazil



SCAN ME

Accepted in Journal of Software:
Evolution and Process



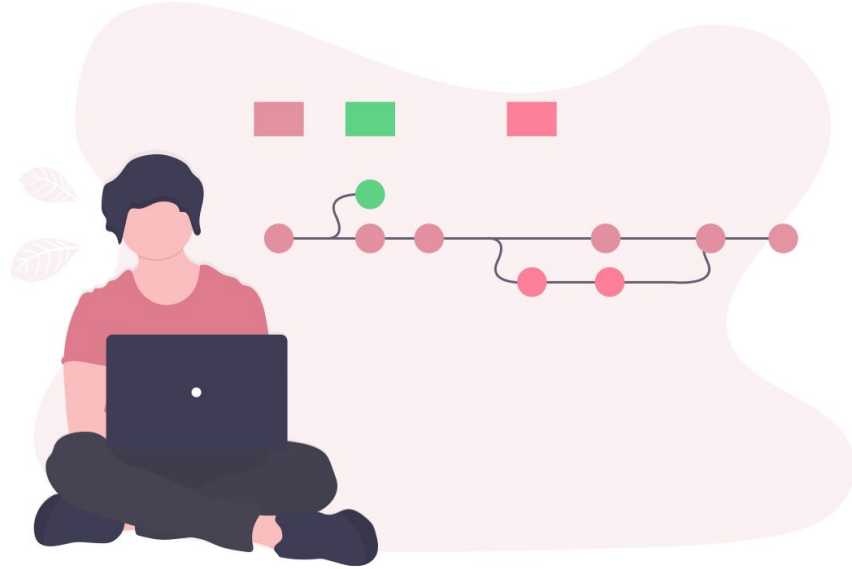
Collaborative Software Development



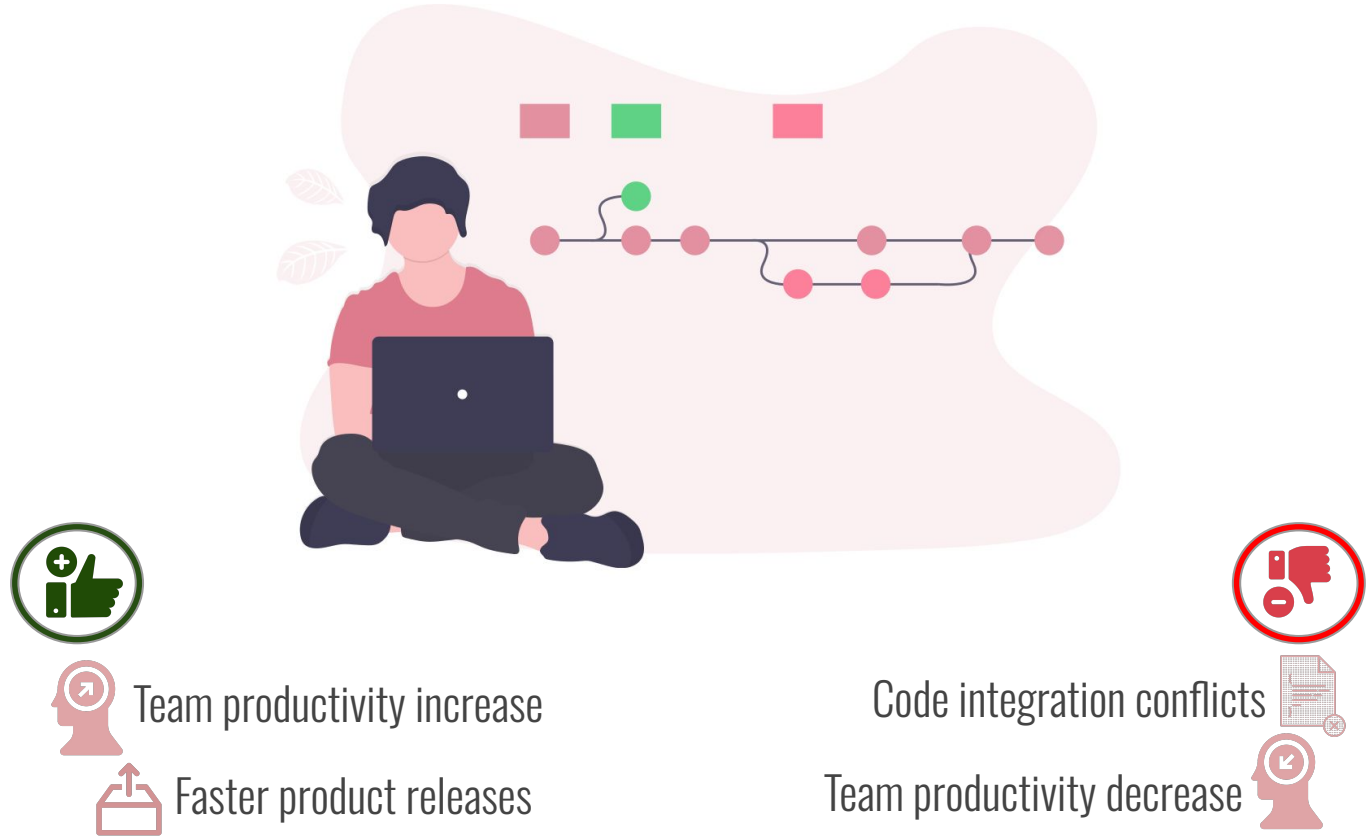
Team productivity increase



Faster product releases



Collaborative Software Development



Conflicting Contributions performed in parallel

```
public class TreeBuilder {  
    public static TreeBuilder ignoreAttributeAtNodeProbability(){  
        {...}  
    }  
}
```

Conflicting Contributions performed in parallel

```
public class TreeBuilder {  
  public static TreeBuilder ignoreAttributeAtNodeProbability(){  
    {...}  
  }  
}
```


```
public class TreeBuilder {  
- public static TreeBuilder ignoreAttributeAtNodeProbability(){  
+ public static TreeBuilder attributeIgnoringStrategy(){  
    {...}  
  }  
}
```

Conflicting Contributions performed in parallel

```
public class TreeBuilder {  
  public static TreeBuilder ignoreAttributeAtNodeProbability(){  
    {...}  
  }  
}
```

```
public class TreeBuilder {  
- public static TreeBuilder ignoreAttributeAtNodeProbability(){  
+ public static TreeBuilder attributeIgnoringStrategy(){  
    {...}  
  }  
}
```

```
public class StaticBuilder {  
  public void getOptimizer() {  
    {...} = TreeBuilder().ignoreAttributeAtNodeProbability();  
  }  
}
```



Compilation problem during integration of conflicting contributions

```
public class TreeBuilder {  
    public static TreeBuilder attributeIgnoringStrategy(){  
        {...}  
    }  
}
```

```
public class StaticBuilder {  
    public void getOptimizer() {  
        {...} = TreeBuilder().ignoreAttributeAtNodeProbability();  
    }  
}
```

Compilation problem during integration of conflicting contributions

```
public class TreeBuilder {  
    public static TreeBuilder attributeIgnoringStrategy(){  
        {...}  
    }  
}
```

```
public class StaticBuilder {  
    public void getOptimizer() {  
        {...} = TreeBuilder().ignoreAttributeAtNodeProbability();  
    }  
}
```



[ERROR]

StaticBuilder.java: cannot find method `ignoreAttributeAtNodeProbability()` in `TreeBuilder` class

Compilation problem during integration of conflicting contributions

```
public class TreeBuilder {  
    public static TreeBuilder attributeIgnoringStrategy(){  
        {...}  
    }  
}  
  
public class StaticBuilder {  
    public void getOptimizer() {  
        {...} = TreeBuilder().ignoreAttributeAtNodeProbability();  
    }  
}
```



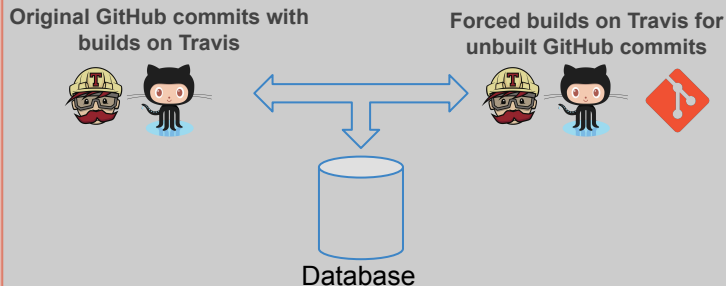
[ERROR]

StaticBuilder.java: cannot find method `ignoreAttributeAtNodeProbability()` in `TreeBuilder` class



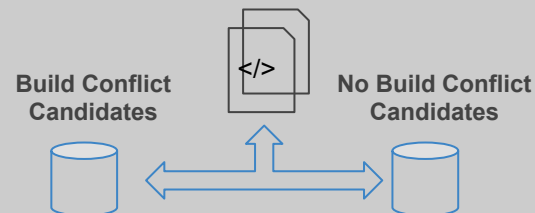
GOAL: Investigation of **Build
Conflicts** *frequency, causes, and
resolution patterns*

1 Selecting Candidate Merge Scenarios



2 Removing Merge Scenarios broken by Build System Issues

Log Analysis of Build Process



3 Classifying Conflicting Contributions and Resolution Patterns

Catalogue of Build Conflict Causes

Syntactic Analysis of Parent Commits

Catalogue of Resolution Patterns

Syntactic Analysis of Merge and Fix Commits

Left Commit

```
public class MethodEditor{...
public boolean isEmpty(){
    HttpClient h1 =
        new HttpClient();
    ...
    + h1.updateList();
}
```

Right Commit

```
public class MethodEditor{...
public boolean isEmpty(){
    - HttpClient h1 =
    - new HttpClient();
    ...
}
```

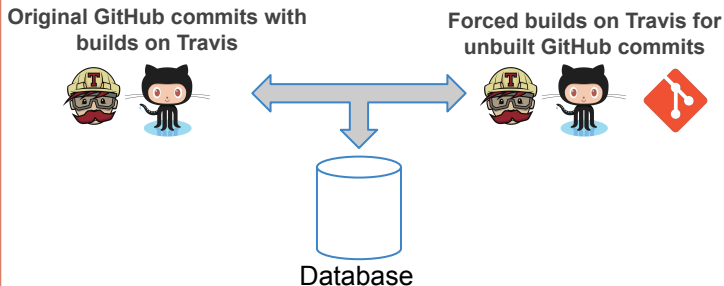
Merge Commit

```
public class MethodEditor{...
public boolean isEmpty(){
    ...
    h1.updateList();
}
```

Fix Commit

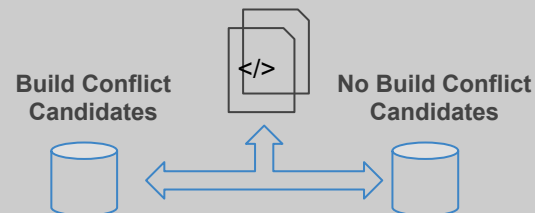
```
public class MethodEditor{...
public boolean isEmpty(){
    + HttpClient h1 =
    + new HttpClient();
    ...
    h1.updateList();
}
```

1 Selecting Candidate Merge Scenarios



2 Removing Merge Scenarios broken by Build System Issues

Log Analysis of Build Process



3 Classifying Conflicting Contributions and Resolution Patterns

Catalogue of Build Conflict Causes

Syntactic Analysis of Parent Commits

Catalogue of Resolution Patterns

Syntactic Analysis of Merge and Fix Commits

Left Commit

```
public class MethodEditor{...
public boolean isEmpty(){
    HttpClient h1 =
        new HttpClient();
    ...
    + h1.updateList();
}
```

Right Commit

```
public class MethodEditor{...
public boolean isEmpty(){
    - HttpClient h1 =
    - new HttpClient();
    ...
}
```

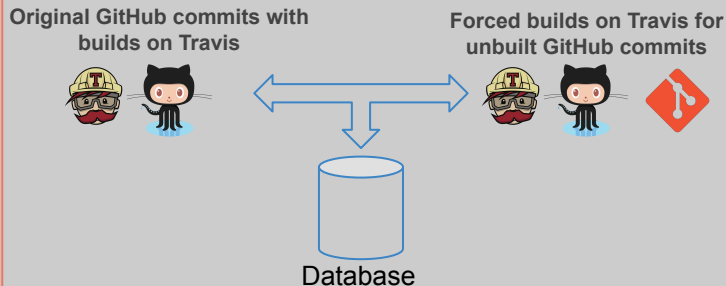
Merge Commit

```
public class MethodEditor{...
public boolean isEmpty(){
    ...
    h1.updateList();
}
```

Fix Commit

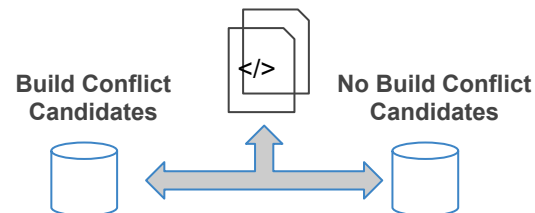
```
public class MethodEditor{...
public boolean isEmpty(){
    + HttpClient h1 =
    + new HttpClient();
    ...
    h1.updateList();
}
```

1 Selecting Candidate Merge Scenarios



2 Removing Merge Scenarios broken by Build System Issues

Log Analysis of Build Process



3 Classifying Conflicting Contributions and Resolution Patterns

Catalogue of Build Conflict Causes

Syntactic Analysis of Parent Commits

Catalogue of Resolution Patterns

Syntactic Analysis of Merge and Fix Commits

Left Commit

```
public class MethodEditor{...
  public boolean isEmpty(){
    HttpClient h1 =
      new HttpClient();
    ...
    + h1.updateList();
  }
}
```

Right Commit

```
public class MethodEditor{...
  public boolean isEmpty(){
    - HttpClient h1 =
    - new HttpClient();
    ...
  }
}
```

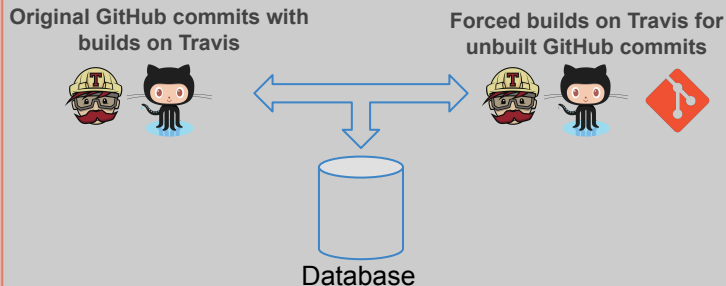
Merge Commit

```
public class MethodEditor{...
  public boolean isEmpty(){
    ...
    h1.updateList();
  }
}
```

Fix Commit

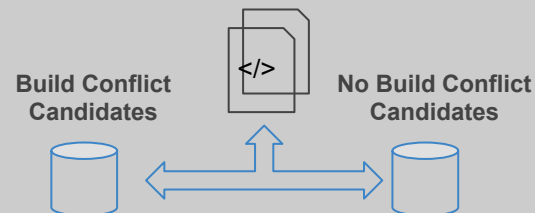
```
public class MethodEditor{...
  public boolean isEmpty(){
    + HttpClient h1 =
    + new HttpClient();
    ...
    h1.updateList();
  }
}
```

1 Selecting Candidate Merge Scenarios



2 Removing Merge Scenarios broken by Build System Issues

Log Analysis of Build Process



3 Classifying Conflicting Contributions and Resolution Patterns

Catalogue of Build Conflict Causes

Syntactic Analysis of Parent Commits

Left Commit

```
public class MethodEditor{...
  public boolean isEmpty(){
    HttpClient h1 =
      new HttpClient();
    ...
    + h1.updateList();
  }
}
```

Right Commit

```
public class MethodEditor{...
  public boolean isEmpty(){
    - HttpClient h1 =
    - new HttpClient();
    ...
  }
}
```

Catalogue of Resolution Patterns

Syntactic Analysis of Merge and Fix Commits

Merge Commit

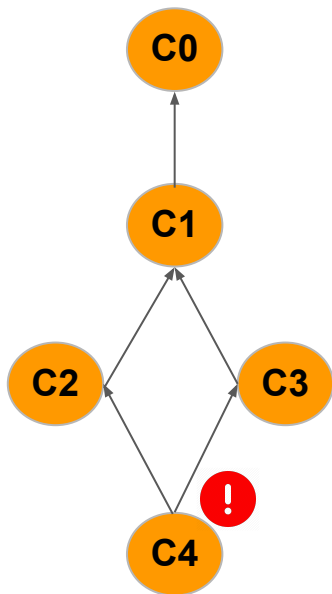
```
public class MethodEditor{...
  public boolean isEmpty(){
    ...
    h1.updateList();
  }
}
```

Fix Commit

```
public class MethodEditor{...
  public boolean isEmpty(){
    + HttpClient h1 =
    + new HttpClient();
    ...
    h1.updateList();
  }
}
```

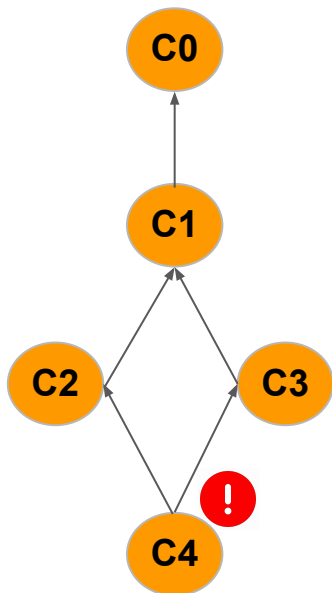
Detecting Conflict Occurrence by Parent Changes Analysis

Project History



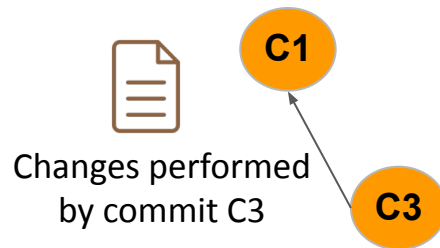
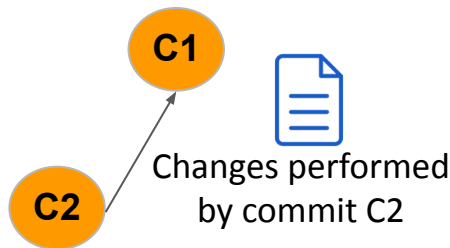
Detecting Conflict Occurrence by Parent Changes Analysis

Project History



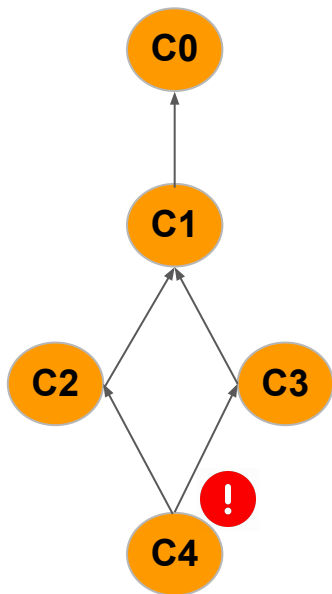
Gumtree - Syntactic Diff

Accessing parent contributions from project history



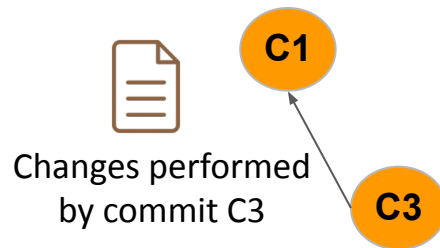
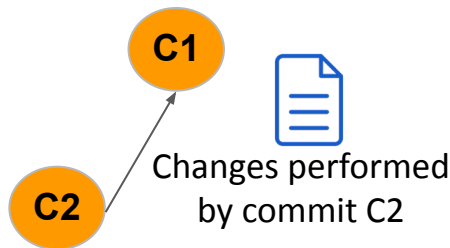
Detecting Conflict Occurrence by Parent Changes Analysis

Project History



Gumtree - Syntactic Diff

Accessing parent contributions from project history



TreeBuilder: **Update** SimpleName:
ignoreAttributeAtNodeProbability
to **attributeIgnoringStrategy**

StaticBuilder: **Insert** SimpleName:
ignoreAttributeAtNodeProbability



How frequently do build conflicts occur?

Build conflicts might not reach the
related remote project repositories

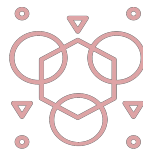


239 build conflicts in 65
merge scenarios from 37
projects

Build conflicts might not reach the related remote project repositories

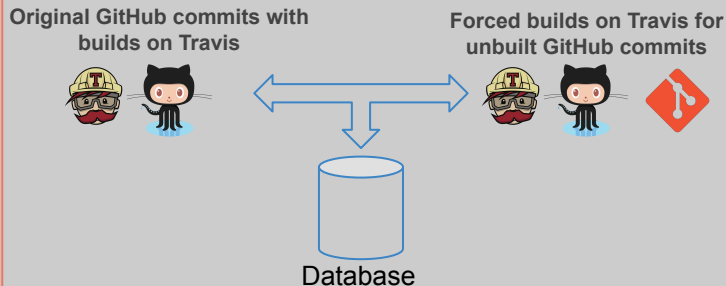


239 build conflicts in 65
merge scenarios from 37
projects



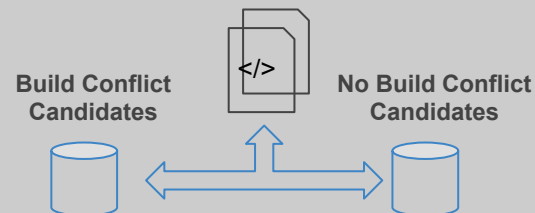
Different build conflict
types may happen in one
single merge scenario

1 Selecting Candidate Merge Scenarios



2 Removing Merge Scenarios broken by Build System Issues

Log Analysis of Build Process



3 Classifying Conflicting Contributions and Resolution Patterns

Catalogue of Build Conflict Causes

Syntactic Analysis of Parent Commits

Left Commit

```
public class MethodEditor{...
    public boolean isEmpty(){
        HttpClient h1 =
            new HttpClient();
        ...
        + h1.updateList();
    }
}
```

Right Commit

```
public class MethodEditor{...
    public boolean isEmpty(){
        - HttpClient h1 =
        - new HttpClient();
        ...
    }
}
```

Catalogue of Resolution Patterns

Syntactic Analysis of Merge and Fix Commits

Merge Commit

```
public class MethodEditor{...
    public boolean isEmpty(){
        ...
        h1.updateList();
    }
}
```

Fix Commit

```
public class MethodEditor{...
    public boolean isEmpty(){
        + HttpClient h1 =
        + new HttpClient();
        ...
        h1.updateList();
    }
}
```

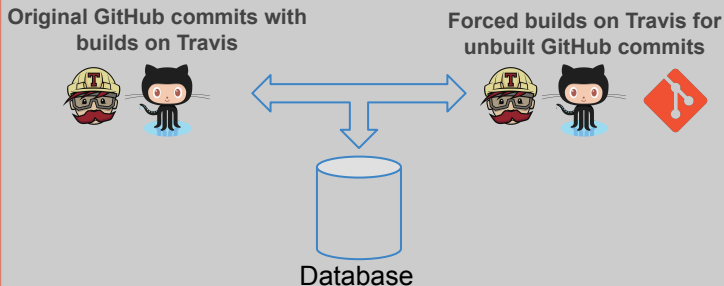


What are the build conflict causes?

Most Build Conflicts are caused by Unavailable Symbols

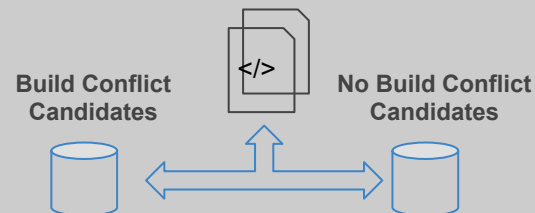
Cause	Description	#
Unimplemented Method	Unimplemented method from supertype/interface	12 (5%)
Duplicated Declaration	Elements with the same identifier	5 (2.1%)
Unavailable Symbol	Reference for a missing symbol	157 (65.7%)
Incompatible Method Signature	Unmatched method reference	26 (10.9%)
Incompatible Types	Mismatch between expected and received types	17 (7.1%)
Project Rules	Unfollowed project guidelines	22 (9.2%)
Total		239

1 Selecting Candidate Merge Scenarios



2 Removing Merge Scenarios broken by Build System Issues

Log Analysis of Build Process



3 Classifying Conflicting Contributions and Resolution Patterns

Catalogue of Build Conflict Causes

Syntactic Analysis of Parent Commits

Left Commit

```
public class MethodEditor{...
public boolean isEmpty(){
    HttpClient h1 =
        new HttpClient();
    ...
    + h1.updateList();
}
```

Right Commit

```
public class MethodEditor{...
public boolean isEmpty(){
    - HttpClient h1 =
    - new HttpClient();
    ...
}
```

Catalogue of Resolution Patterns

Syntactic Analysis of Merge and Fix Commits

Merge Commit

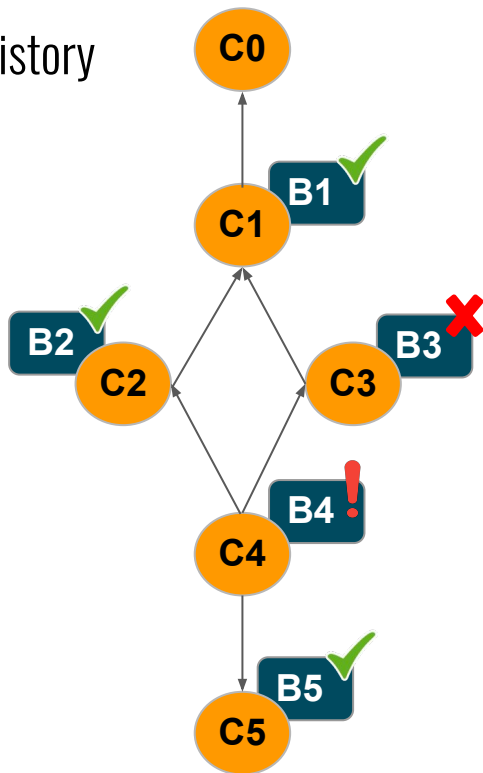
```
public class MethodEditor{...
public boolean isEmpty(){
    ...
    h1.updateList();
}
```

Fix Commit

```
public class MethodEditor{...
public boolean isEmpty(){
    + HttpClient h1 =
    + new HttpClient();
    ...
    h1.updateList();
}
```

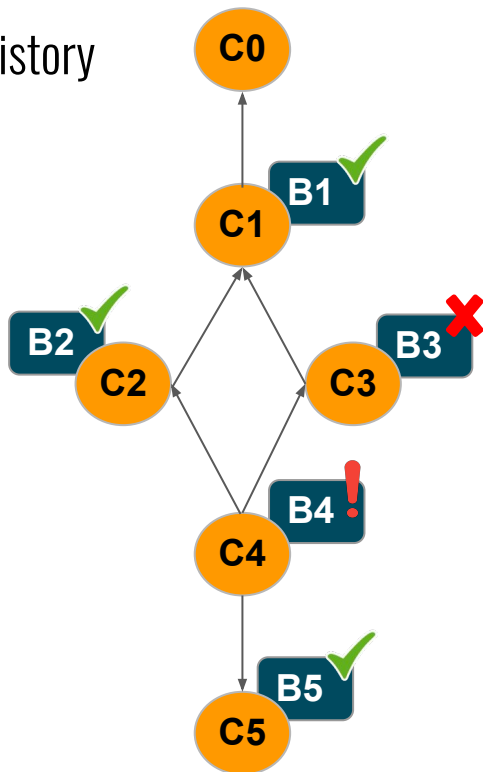

Detecting Resolution Patterns based on the closest no broken build commit

Project History



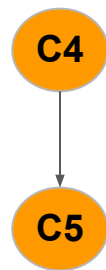
Detecting Resolution Patterns based on the closest no broken build commit

Project History



Gumtree - Syntactic Diff

Accessing fix information from project history



Changes performed
by commit C5

StaticBuilder: **Remove** SimpleName:
ignoreAttributeAtNodeProbability



What are the resolution patterns
adopted for build conflicts?

Build Conflict Fixes adapt the project to its new state

Cause	Resolution Patterns	#
Unimplemented Method	Supertype class addition	5 (3.4%)
	Method implementation	5 (3.4%)
Duplicated Declaration	Duplicated element removal	2 (1.3%)
Unavailable Symbol	Missing symbol import update	66 (44.6%)
	Missing symbol reference update	37 (25.0%)
	Missing symbol reference removal	9 (6.0%)
Incompatible Method Signature	Required method reference update	3 (2.1%)
	Required method reference removal	2 (1.3%)
	JDK Setup	7 (4.8%)
Incompatible Types	Type update	9 (6.0%)
Project Rules	License header addition/update	3 (2.1%)
Total		100

Most Build Conflict Fixes are done by
parent commit authors

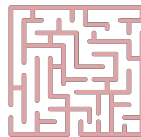


137 out of 148 fixes
follow this pattern

Most Build Conflict Fixes are done by parent commit authors



137 out of 148 fixes
follow this pattern

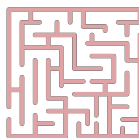


Build Conflicts are harder
than merge conflict fixes

Most Build Conflict Fixes are done by parent commit authors



137 out of 148 fixes
follow this pattern



Build Conflicts are harder
than merge conflict fixes



Parent commit authors may
fix these conflicts faster than
not involved authors

We propose a build conflict automatic repair tool

```
public class TreeBuilder {  
- public static TreeBuilder ignoreAttributeAtNodeProbability() {  
+ public static TreeBuilder attributeIgnoringStrategy() {  
    {...}  
}  
}
```

```
public class StaticBuilder {  
    public void getOptimizer() {  
+ {...} = TreeBuilder().ignoreAttributeAtNodeProbability();  
    }  
}
```



We propose a build conflict automatic repair tool

```
public class TreeBuilder {  
- public static TreeBuilder ignoreAttributeAtNodeProbability(){  
+ public static TreeBuilder attributeIgnoringStrategy(){  
    {...}  
}  
}
```

```
public class StaticBuilder {  
    public void getOptimizer() {  
+ {...} = TreeBuilder().ignoreAttributeAtNodeProbability();  
    }  
}
```



Recommendation for replacing the missing symbol

```
public class TreeBuilder {  
    public static TreeBuilder attributeIgnoringStrategy(){  
        {...}  
    }  
}
```

```
public class StaticBuilder {  
    public void getOptimizer() {  
        {...} = TreeBuilder().ignoreAttributeAtNodeProbability();  
    }  
}
```

We propose a build conflict automatic repair tool

```
public class TreeBuilder {  
- public static TreeBuilder ignoreAttributeAtNodeProbability(){  
+ public static TreeBuilder attributeIgnoringStrategy(){  
    {...}  
}  
}
```

```
public class StaticBuilder {  
    public void getOptimizer() {  
+ {...} = TreeBuilder().ignoreAttributeAtNodeProbability();  
    }  
}
```

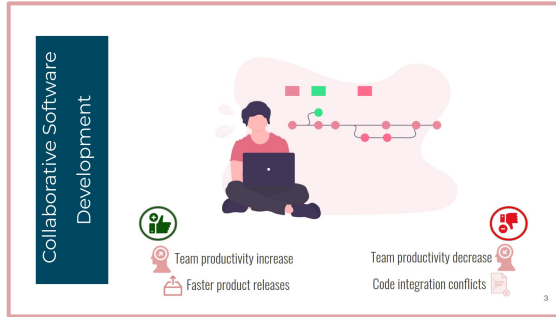


Recommendation for replacing the missing symbol

```
public class TreeBuilder {  
    public static TreeBuilder attributeIgnoringStrategy(){  
        {...}  
    }  
}
```

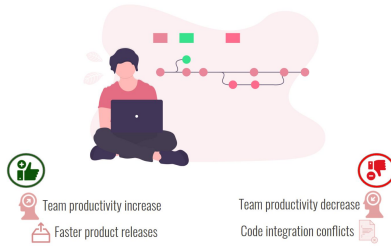
```
public class StaticBuilder {  
    public void getOptimizer() {  
        {...} = TreeBuilder().attributeIgnoringStrategy();  
    }  
}
```

In Summary



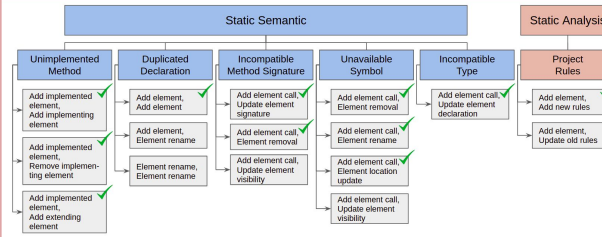
In Summary

Collaborative Software Development



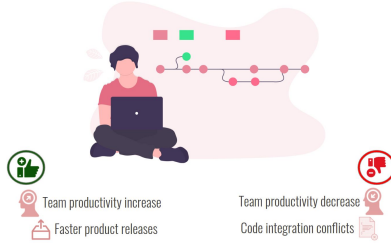
3

Build Conflict Catalogue



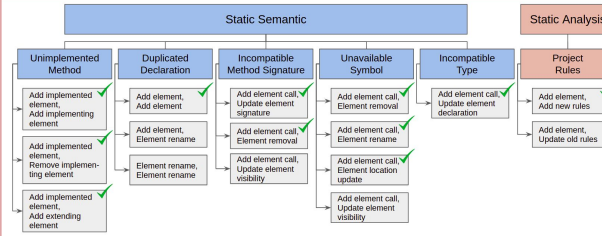
In Summary

Collaborative Software Development



3

Build Conflict Catalogue

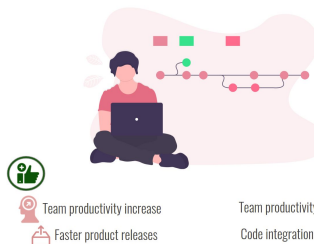


Build Conflict Fixes adapt the project to its new state

Cause	Resolution Patterns	#
Unimplemented Method	Supertype class addition	5 (3.4%)
	Method implementation	5 (3.4%)
Duplicated Declaration	Duplicated element removal	2 (1.3%)
	Missing symbol import update	66 (44.6%)
Unavailable Symbol	Missing symbol reference update	37 (25.0%)
	Missing symbol reference removal	9 (6.0%)
	Required method reference update	3 (2.1%)
Incompatible Method Signature	Required method reference removal	2 (1.3%)
	JDK Setup	7 (4.8%)
Incompatible Types	Type update	9 (6.0%)
Project Rules	License header addition/update	3 (2.1%)
Total		100

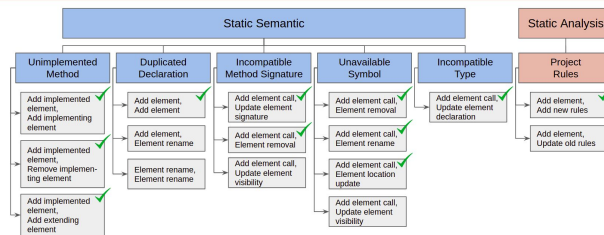
In Summary

Collaborative Software Development



3

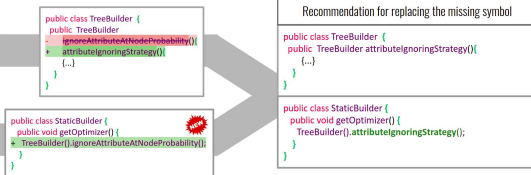
Build Conflict Catalogue



Build Conflict Fixes adapt the project to its new state

Cause	Resolution Patterns	#
Unimplemented Method	Supertype class addition	5 (3.4%)
	Method implementation	5 (3.4%)
Duplicated Declaration	Duplicated element removal	2 (1.3%)
	Missing symbol import update	66 (44.6%)
Unavailable Symbol	Missing symbol reference update	37 (25.0%)
	Missing symbol reference removal	9 (6.0%)
	Required method reference update	3 (2.1%)
Incompatible Method Signature	Required method reference removal	2 (1.3%)
	JDK Setup	7 (4.8%)
	Type update	9 (6.0%)
Project Rules	License header addition/update	3 (2.1%)
Total		100

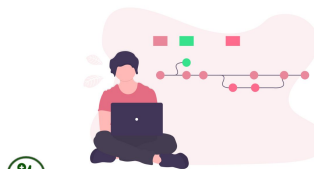
We propose a build conflict automatic repair tool



31

In Summary

Collaborative Software Development



Team productivity increase

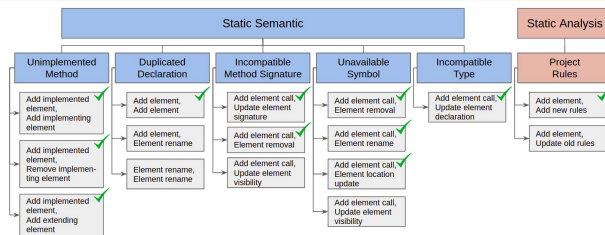
Faster product releases

Team productivity decrease

Code integration conflicts

3

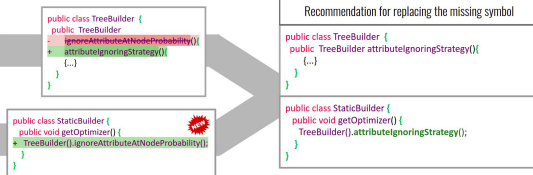
Build Conflict Catalogue



Build Conflict Fixes adapt the project to its new state

Cause	Resolution Patterns	#
Unimplemented Method	Supertype class addition	5 (3.4%)
	Method implementation	5 (3.4%)
Duplicated Declaration	Duplicated element removal	2 (1.3%)
	Missing symbol import update	66 (44.6%)
Unavailable Symbol	Missing symbol reference update	37 (25.0%)
	Missing symbol reference removal	9 (6.0%)
Incompatible Method Signature	Required method reference update	3 (2.1%)
	Required method reference removal	2 (1.3%)
Incompatible Types	JDK Setup	7 (4.8%)
	Type update	9 (6.0%)
Project Rules	License header addition/update	3 (2.1%)
Total		100

We propose a build conflict automatic repair tool



31

Build Conflicts in The Wild

Abstract

When collaborating, developers often create and change software artifacts without being fully aware of other team member's work. While such independence is essential for increasing development productivity, it might also result in conflicts when integrating developers' code contributions. To better understand some of these conflicts—the ones revealed by failures when building integrated code—we investigate their frequency, structure, and adopted resolution patterns in 431 open-source Java projects. To detect such build conflicts, we select merge scenarios from git repositories, parse the Travis logs generated when building the commits, and check whether the logged build error messages are related to the merged changes. We find and classify 239 build conflicts and their resolution patterns. Most build conflicts are caused by missing declarations removed or renamed by one developer but referenced by another developer. Conflicts caused by renaming are often resolved by updating the missing reference, whereas removed declarations are often reintroduced. Most fix commits are authored by one of the contributors involved in the merge scenario. We also detect and analyze build failures caused by immediate post integration changes, which are often performed with the aim of fixing merge conflicts but end up leading to build issues. Based on our catalogue of build conflict causes, awareness tools could alert developers about the risk of conflict situations. Program repair tools could benefit from our catalogue of build conflict resolution patterns to automatically fix conflicts. We illustrate that with a proof of concept implementation of a tool that recommends fixes for conflicts.

Keywords

- Lifson De Silva (lifson@cs.cup.edu)
- Paulo Borba (pb@cs.cup.edu)
- Arthur Pires (pires@cs.cup.edu)

How frequently do build conflicts occur?



Build Conflicts in The Wild



Léuson da Silva
lmps2@cin.ufpe.br



Paulo Borba
phmb@cin.ufpe.br



Arthur Pires
ahlp@cin.ufpe.br

Federal University of Pernambuco, Recife, Brazil



SCAN ME

Accepted in Journal of Software:
Evolution and Process

