Patched Clones and Missed Patches among Variants of a Software Family

John Businge
Assistant Professor – UNLV
Never Work in Theory (NWiT) – Spring 2023
The Equifax data breach occurred between May and July 2017 at the American credit bureau Equifax. Private records of 147.9 million Americans along with 15.2 million British citizens and about 19,000 Canadian citizens were compromised in the breach, making it one of the largest cybercrimes related to identity theft.

Wired Magazine, “Equifax has no excuse”, September 2017
Patched Clones and Missed Patches among Variants of a Software Family

1. Social Forks
   - Social Fork
   - Original project
   - Social Fork

2. Variant Forks
   - Social Fork
   - Original project
   - Variant 1
   - Variant 2
   - Social Fork

Software family — >= 2 variants
Reuse and maintenance practices among divergent forks in three software ecosystems

John Businge, Moses Openia, Sarah Nadi & Thorsten Berger

Abstract

With the rise of social coding platforms that rely on distributed version control systems, software reuse is also on the rise. Many software developers leverage this reuse by creating variants through forking, to account for different customer needs, markets, or environments. Forked variants then form a so-called software family; they share a common code base and are maintained in parallel by same or different developers. As such, software families can easily arise within software ecosystems, which are large collections of interdependent software components maintained by communities of collaborating contributors. However, little is known about the existence and characteristics of such families within ecosystems, especially about their maintenance practices. Improving our empirical understanding of such families will help build better tools for maintaining and evolving such families. We empirically explore maintenance practices in such fork-based software families within ecosystems of open-source software. Our focus is on three of the largest software ecosystems existence today: Android,
Problem

variant1 (source)

variant2 (target)

Inherited commits

Synchronized commits

divergence_date

fork_date

current_date

Uninteresting

Split case

Missed opportunity

Effort duplication

PR merge_commit

target git_head

foo

foo

foo

foo

foo

Social fork

Pull request

Synchronized commits

Unique commits

Uninteresting

Split case

Missed opportunity

Effort duplication
Concrete Example

This is the version of Kafka running at LinkedIn.

---

**415**

LinkedIn unique commits

**1,787**

Apache unique commits

---

extraction_date 2023-03-27
Concrete Example: Missed Opportunity

1 file - Pull request

```
Concrete Example:
Missed Opportunity
```

```
Buggy code from upstream

```c
1 return;
2 }
3 } while (p < (uint16_t *)SYMVAL(__eeprom_workarea_end__));
4 flashend = (uint32_t)((uint16_t *)SYMVAL(__eeprom_workarea_end__) - 1);
5 }
```

```
Patched code from upstream (Pull request)

```c
1 return;
2 }
3 } while (p < (uint16_t *)SYMVAL(__eeprom_workarea_end__));
4 flashend = (uint32_t)(p - 1);
5 }
```

```
Diff for patch in upstream

```
1 @@ -363,7 +363,7 @@
2 }
3 } while (p < (uint16_t *)SYMVAL(__eeprom_workarea_end__));
4 - flashend = (uint32_t)((uint16_t *)SYMVAL(__eeprom_workarea_end__) - 1);
5 + flashend = (uint32_t)(p - 1);
```

```
File from divergent fork at git_head

```c
1 return;
2 }
3 } while (p < (uint16_t *)SYMVAL(__eeprom_workarea_end__));
4 flashend = (uint32_t)((uint16_t *)SYMVAL(__eeprom_workarea_end__) - 1);
5 }
```

```
```

```
gcc10 [...] build warning #12587
```

```
```

```
extraction_date 2021-07-20
```

```
Buggy line
```

```
Patched line
```

```
Hunk
```

```
Buggy line
```

```
sekigon-gonnoc/qmk_firmware
```

```
qmk/qmk_firmware
```

```
```
Research Questions

1. **RQ1**: How many cases of effort duplication and missed opportunities exist between divergent variants?

2. **RQ2**: How much patch technical lag exists between the source and target variants in divergent variants?
keywords \{fix, fixes, resolves, \ldots\}

Method

clone detection tool (PaReco)

ED

MO

keywords \{fix, fixes, resolves, \ldots\}

Search PR Title

variant1 (source)

variant2 (target)

Inherited commits

Synchronized commits

Unique commits

PR merge commit

target git_head

fork_date
divergence_date
current_date

Uninteresting

Split case

Missed opportunity

Effort duplication

Pull request

Source variant

Git head

Target variant

Extract files

Pull request

Extract files

Patched lines

Buggy lines
Results

**RQ1:** How many cases of effort duplication and missed opportunities exist between divergent variants?

![Diagram](image-url)
**RQ1:** How many cases of effort duplication and missed opportunities exist between divergent variants?

8,323 patches from 364 source variants

2,225 interesting patches

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>Accuracy</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>91.0%</td>
<td>80.2%</td>
<td>88.0%</td>
<td>85.3%</td>
</tr>
</tbody>
</table>
Results

**RQ2**: How much patch technical lag exists between the source and target variants in divergent variants?

![Graph showing technical lag between target variants and (MO and SP patches). The graph indicates a significant lag period of 52 weeks later in the target variants.](image-url)
Variants on social coding platforms exhibit suboptimal maintenance.

PaReco: Proof-of-Concept patch recommender tool
Current Work on PaReco

PaReco: Proof-of-Concept patch recommender tool