Why and How Java Developers Break APIs

Aline Brito, Laerte Xavier, Andre Hora, Marco Tulio Valente
Motivation

Libraries have a key importance in modern software development

Maven™
200K libraries
Motivation

Libraries have a key importance in modern software development

Maven
200K libraries

Android, JUnit, Mockito, Spring
Motivation

Libraries have a key importance in modern software development
Library services are provided via APIs
In theory, APIs should be stable
But, 28% out of 500K API changes break backward compatibility
Why do Developers Break APIs?
Outline

1. APIDiff Tool
2. Dataset
3. Study Steps
4. Definitions
5. Results
APIDiff

A tool to detect API Changes

APIDiff: Detecting API Breaking Changes  SANER Tool Track, 2018

https://github.com/aser-g-ufmg/apidiff
APIDiff

Breaking changes in types, methods, and fields

- Removal
- Change in access modifiers
- Change in parameter list
APIDiff

Breaking changes in types, methods, and fields

- Removal
- Change in access modifiers
- Change in parameter list
APIDiff

APIDiff warns if a breaking change is performed in internal or deprecated APIs

io.reactivex.internal.util.ExceptionHelper
APIDiff

APIDiff warns if a breaking change is performed in `internal` or `deprecated` APIs

```java
io.reactivex.internal.util.ExceptionHelper
```
Dataset
Top 2,000 projects by stars

We discard projects without the keywords Library(ies), API(s), framework(s), and deprecated projects (449 projects)

We inspect the projects documentation

400 libraries and frameworks
Top 2,000 projects by stars

We discard projects without the keywords Library(ies), API(s), framework(s), and deprecated projects (449 projects)

We inspect the projects documentation

400 libraries and frameworks
Top 2,000 projects by stars

We discard projects without the keywords Library(ies), API(s), framework(s), and deprecated projects (449 projects)

We inspect the projects documentation

400 libraries and frameworks
Top 2,000 projects by stars

We discard projects without the keywords Library(ies), API(s), framework(s), and deprecated projects (449 projects)

We inspect the projects documentation

400 libraries and frameworks
Study Steps
Study Steps

We mined daily commits
Study Steps

We mined **daily commits**

We used **APIDiff** to identify **breaking changes**
Study Steps

We mined **daily commits**

We used **APIDiff** to identify **breaking changes**

We sent **emails** to developers asking the **reasons** behind changes
Study Numbers

116 days, May 8th to August 31th, 2017

102 emails

56 answers (55%)
Definitions
Breaking Changes Candidates (BCC)

Changes detected by APIDiff in public API elements
Breaking Changes (BC)

BCCs confirmed by the surveyed developers
Survey Results
Survey Questions

1. **Why** did you perform these changes?

2. Do you **agree** these changes can break clients? If yes, could you quantify the amount of **work to use** the new implementation?

3. **Why** didn’t you **deprecate** the old implementation?

4. Do you plan to **document** the changes? If yes, how?
Q1: How often do changes impact clients?
How often do changes impact clients?

59 BCCs (39%) detected by APIDiff are BCs
Unconfirmed Breaking Changes

92 changes (61%)
The surveyed developers did not agree they have an impact on clients
Unconfirmed Breaking Changes

Most unconfirmed BCCs are in **internal** or low-level APIs or in **testing branches**.
Most common breaking changes

- Move Method: 19%
- Remove Class: 17%
- Change in Parameter List: 15%
- Rename Method: 14%
- Move Class: 14%
- Add Final Modifier: 10%
- Remove Method: 5%
- Change in Return Type: 3%
- Change in Field Default Value: 2%
- Access Modifier Change: 2%
Most common breaking changes

Most BCs are due to **refactorings (47%)**
Most common breaking changes

- Move Method: 19%
- Remove Class: 17%
- Change in Parameter List: 15%
- Rename Method: 14%
- Move Class: 14%
- Add Final Modifier: 10%
- Remove Method: 5%
- Change in Return Type: 3%
- Change in Field Default Value: 2%
- Access Modifier Change: 2%
Breaking changes per API element

- Method: 59%
- Type: 36%
- Field: 5%
Breaking changes per API element

Most BCs are performed on methods (59%)
Q2: Why do developers break APIs?
## Why do developers break APIs?

<table>
<thead>
<tr>
<th>Motivation</th>
<th>Description</th>
<th>Occur.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEW FEATURE</td>
<td>BCs to implement new features</td>
<td>19</td>
</tr>
<tr>
<td>API SIMPLIFICATION</td>
<td>BCs to simplify and reduce the API complexity and number of elements</td>
<td>17</td>
</tr>
<tr>
<td>MAINTAINABILITY</td>
<td>BCs to improve the maintainability and the structure of the code</td>
<td>14</td>
</tr>
<tr>
<td>BUG FIXING</td>
<td>BCs to fix bugs in the code</td>
<td>3</td>
</tr>
</tbody>
</table>
"The changes in this commit were just a setup before implementing a new feature: chart data retrieval."
“This method should not accept any parameters, because they are **ignored by server.**”
Maintainability (14 occurrences)

“Make support class lighter, by moving methods to Class and Method info.”
Q3: Why don’t developers deprecate broken APIs?
Why don’t developers deprecate broken APIs?

- Increase Maintainability Effort: 44%
- Other Motivations: 33%
- Minor Change/Impact: 22%

(17 answers)
Why don’t developers deprecate broken APIs?

Developers do not deprecate elements affected by BCs mostly due to the *extra effort* to maintain them.
Increase Maintainability Effort (8 answers)

“In such a small library, deprecation will only add complexity and maintenance issues in the long run.”
Q4: What is the effort on clients to migrate?
What is the effort on clients to migrate?

- Small: 86%
- Moderate: 14%
- Large: 0%
What is the effort on clients to migrate?

According to the surveyed developers, the effort to migrate to the new API versions is small.
Q5: How do developers document breaking changes?
Do you plan to document the changes?

14 developers agree
4 developers disagree
How do developers document BCs?

- Release Notes: 22%
- Changelog: 22%
- JavaDoc: 17%
- Website: 11%
- Migration Guide: 11%
- Examples: 11%
- README: 6%
How do developers document breaking changes?

- Release Notes: 22%
- Changelog: 22%
- JavaDoc: 17%
- Website: 11%
- Migration Guide: 11%
- Examples: 11%
- README: 6%
How do developers document breaking changes?

- Release Notes: 22%
- Changelog: 22%
- JavaDoc: 17%
- Website: 11%
- Migration Guide: 11%
- Examples: 11%
- README: 6%
Implications and Conclusions
Implications to Language Designers

Most unconfirmed BCCs affect internal or low-level APIs.

However, **internal APIs can be used** by external clients, since they are public.
Implications to Language Designers

This confirms the relevance of the new module system, being proposed to Java

*http://openjdk.java.net/projects/jigsaw
Implications to Practitioners

Many unconfirmed BCCs do not have internal or experimental in their names.

Practitioners should use internal in low-level or internal API names.
Why do Java developers break APIs?

- New features
- API simplification
- Improve maintainability

How do Java developers break APIs?

- Move Method
- Remove Class
- Change in Parameter List
- Rename Method
- Move Class
Thank you!

Aline Brito, Laerte Xavier, Andre Hora, Marco Tulio Valente