ROSEAU (REED)

Fast, Accurate, Source-based API Breaking Change Analysis in Java

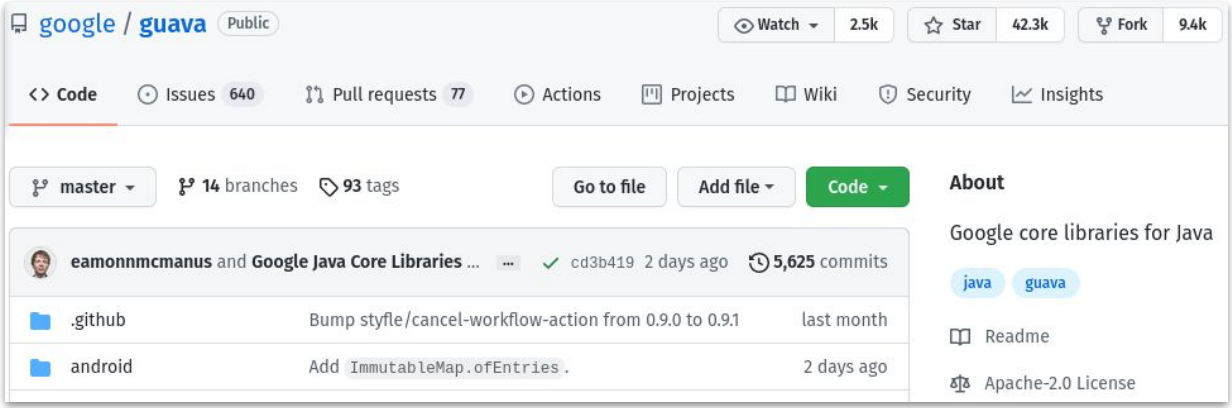
LaBRI TU/e

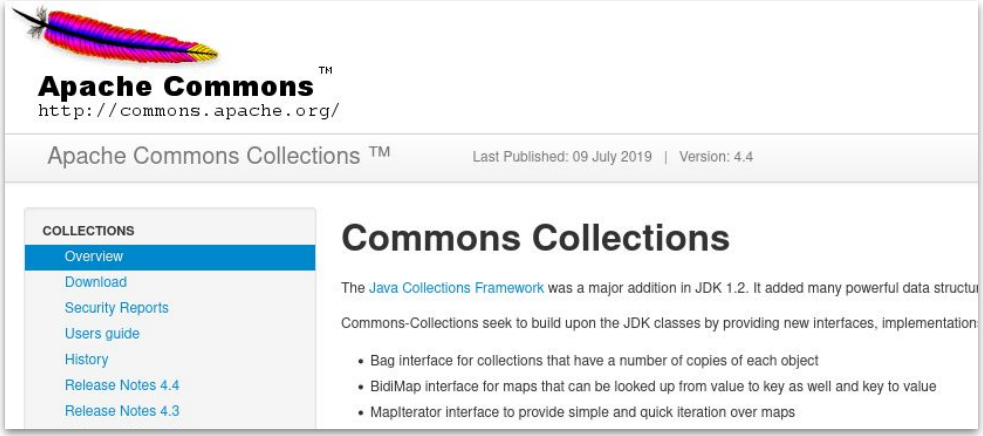




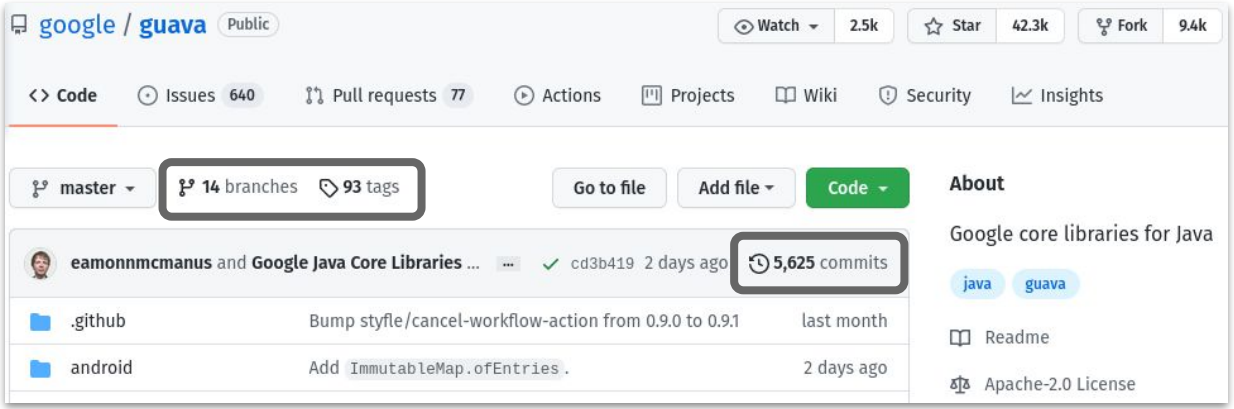
Corentin Latappy Thomas Degueule Jean-Rémy Falleri Romain Robbes Lina Ochoa Venegas

Software Libraries

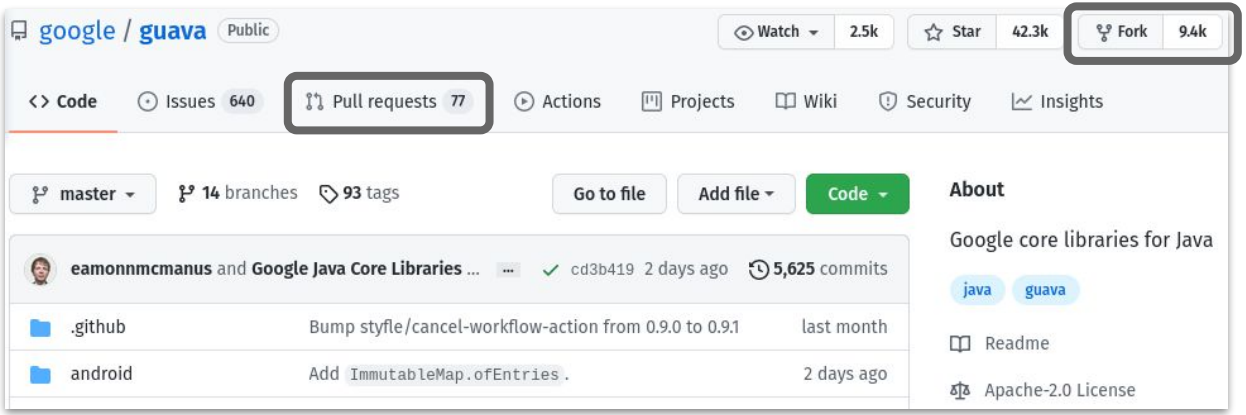


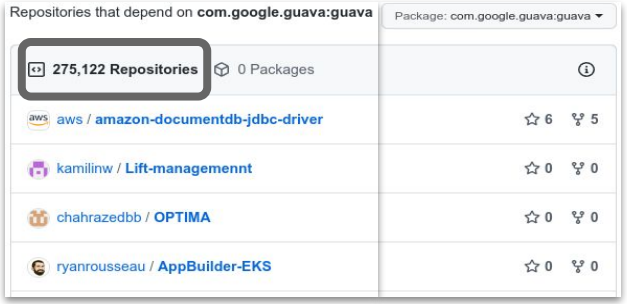


Software Libraries Evolution

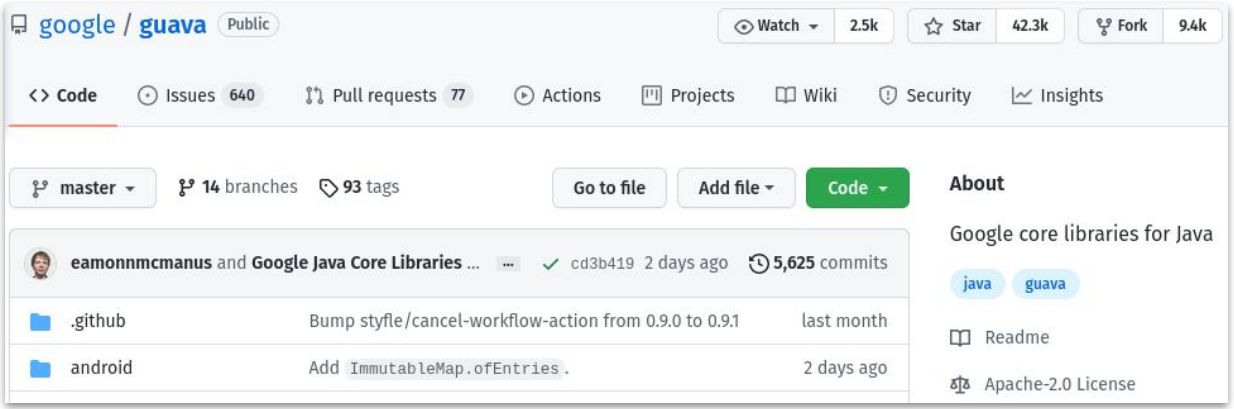


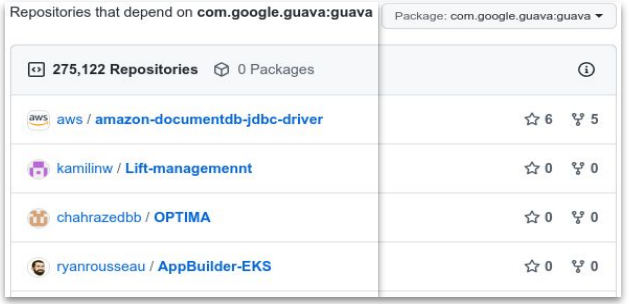
Software Libraries Evolution

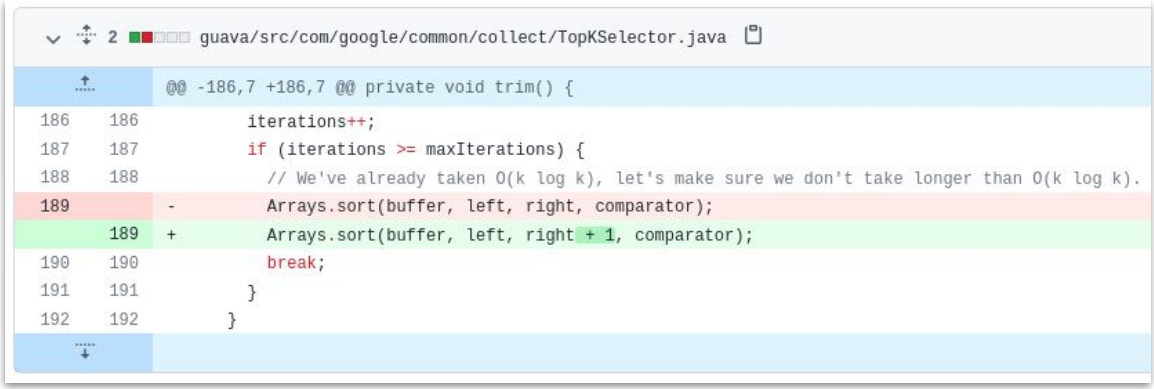




Software Libraries Evolution





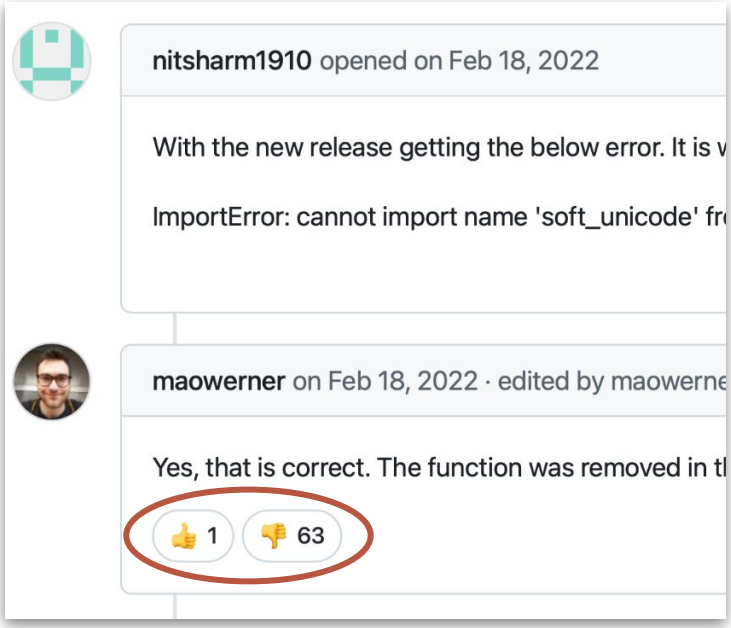


Breaking Changes in Practice

- Why?

Breaking Changes in Practice

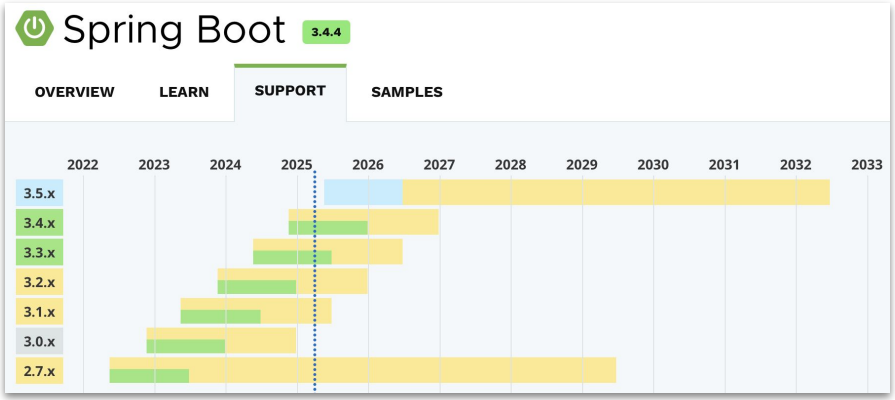
- Why?
  - Keep users confidence



https://github.com/pallets/markupsafe/issues/284

Breaking Changes in Practice

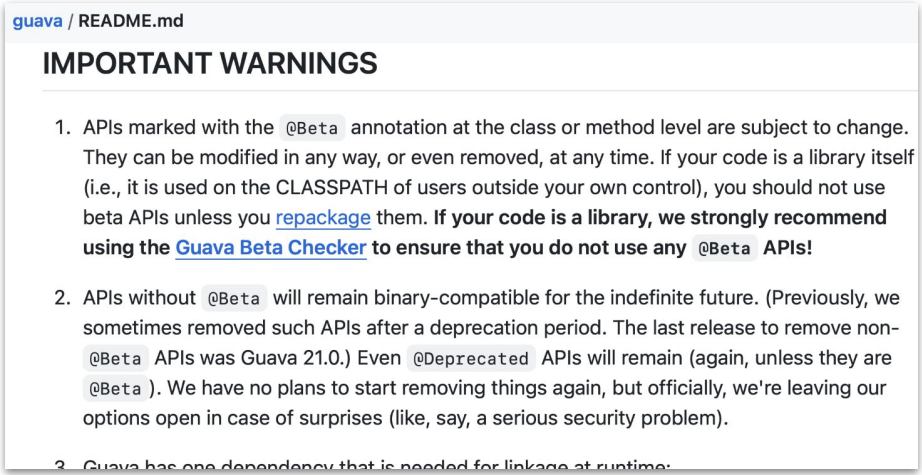
- Why?
  - Keep users confidence
  - Avoid multiple versions support



https://spring.io/projects/spring-boot#support

Breaking Changes in Practice

- Why?
  - Keep users confidence
  - Avoid multiple versions support
- How?
  - Policies



guava / README.md

IMPORTANT WARNINGS

1. APIs marked with the `@Beta` annotation at the class or method level are subject to change. They can be modified in any way, or even removed, at any time. If your code is a library itself (i.e., it is used on the CLASSPATH of users outside your own control), you should not use beta APIs unless you repackage them. **If your code is a library, we strongly recommend using the Guava Beta Checker to ensure that you do not use any `@Beta` APIs!**

2. APIs without `@Beta` will remain binary-compatible for the indefinite future. (Previously, we sometimes removed such APIs after a deprecation period. The last release to remove non-`@Beta` APIs was Guava 21.0.) Even `@Deprecated` APIs will remain (again, unless they are `@Beta`). We have no plans to start removing things again, but officially, we're leaving our options open in case of surprises (like, say, a serious security problem).

3. Guava has one dependency that is needed for linkage at runtime:

https://github.com/google/guava/blob/master/README.md#important-warnings

Breaking Changes in Practice

- Why?
  - Keep users confidence
  - Avoid multiple versions support
- How?
  - Policies
  - Tools

cargo-semver-checks

gocompat
gorelease

AexPy

JApiCmp
Revapi

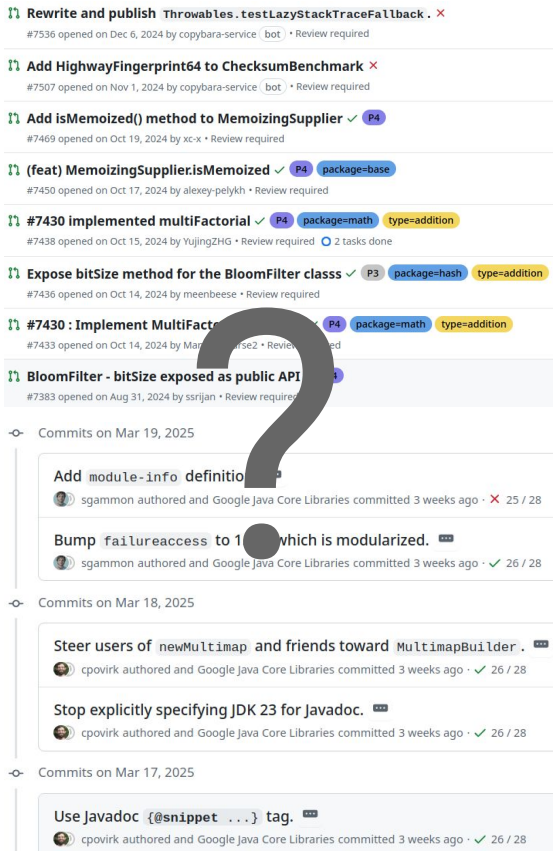
Breaking Changes in SE Research

- Studies: *why*, *how*, *when* do libraries break, how does it affect clients?
  - Xavier *et al.*: 317 libraries, 9k releases[1]
  - Ochoa *et al.*: 14k libraries, 120k releases[2]
  - Jayasuriya *et al.*: 7k libraries, 18k releases[3]

[1] "*Historical and impact analysis of API breaking changes: A large-scale study*". L. Xavier; A. Brito; A. Hora, M. Valente. SANER'17
[2] "*Breaking Bad? Semantic Versioning and Impact of Breaking Changes in Maven Central*". L. Ochoa, T. Degueule, JR. Falleri, J. Vinju. EMSE'22
[3] "*An extended study of syntactic breaking changes in the wild*". D. Jayasuriya, S. Ou, S. Hegde, V. Terragni, J. Dietrich, K. Blincoe. EMSE'25

Breaking Changes in SE Research

- Studies: *why*, *how*, *when* do libraries break, how does it affect clients?
  - Xavier *et al.*: 317 libraries, 9k releases[1]
  - Ochoa *et al.*: 14k libraries, 120k releases[2]
  - Jayasuriya *et al.*: 7k libraries, 18k releases[3]
- Researchers are limited in their ability to study library evolution
  - Japicmp is the *de facto* standard, but requires compiled JAR files
    - Only about 50% of arbitrary versions on GitHub can be built automatically[4]
    - Even when possible, compilation costs are impractical for large-scale studies
  - Empirical research is limited to analyzing *pre-built releases*
- Analyzing commits, pull requests, etc. would be valuable but is impractical[5]



[1] "*Historical and impact analysis of API breaking changes: A large-scale study*". L. Xavier; A. Brito; A. Hora, M. Valente. SANER'17
[2] "*Breaking Bad? Semantic Versioning and Impact of Breaking Changes in Maven Central*". L. Ochoa, T. Degueule, JR. Falleri, J. Vinju. EMSE'22
[3] "*An extended study of syntactic breaking changes in the wild*". D. Jayasuriya, S. Ou, S. Hegde, V. Terragni, J. Dietrich, K. Blincoe. EMSE'25
[4] "*Automatic building of java projects in software repositories: A study on feasibility and challenges*". F. Hassan, M. Shaikh, E. Lam, X. Wang. ESEM'17
[5] "*BreakBot: Analyzing the Impact of Breaking Changes to Assist Library Evolution*". L. Ochoa, T. Degueule, JR. Falleri. ICSE'22 (NIER)

V1

```
public final class List<E> {
  private int size;
  public E add(int pos, E elem) {
    ...
  }
}
```

V2

```
public final class List<E T> {
  private int double size listSize;
  public E void add(int pos, E T elem) {
    ...
  }
}
```

V1

```
public final class List<E> {
  private int size;
  public E add(int pos, E elem) {
    ...
  }
}
```

Breaking Changes:
- METHOD_RETURN_TYPE_CHANGED

V2

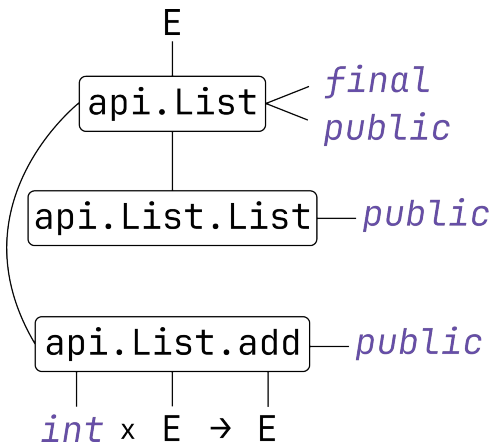
```
public ~~final~~ class List<~~E~~ T> {
  private ~~int~~ double ~~size~~ listSize;
  public ~~E~~ void add(int pos, ~~E~~ T elem) {
    ...
  }
}
```

Roseau – API Model

```
public final class List<E> {
  private int size;
  public E add(int pos, E elem) {
    ...
  }
}
```

Roseau – API Model

```
public final class List<E> {
  private int size;
  public E add(int pos, E elem) {
    ...
  }
}
```

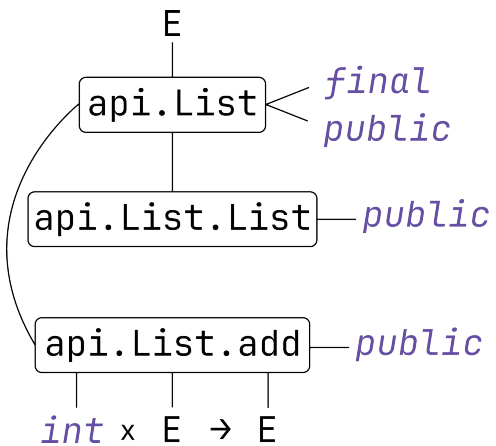


Roseau – API Model

```
public final class List<E> {
  private int size;
  public E add(int pos, E elem) {
    ...
  }
}
```

```
public final class
api.List<E> {
  public api.List();
    0: aload_0
    1: invokespecial #1
    4: return

  public E add(int, E);
    0: aconst_null
    1: areturn
}
```



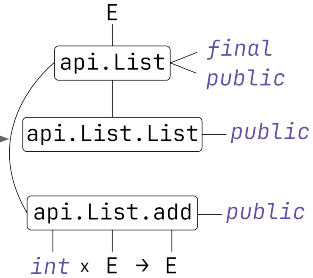
Roseau

V1

```
public final class List<E> {
  private int size;
  public E add(int pos, E elem) {
    ...
  }
}
```

```
public final class api.List<E> {
  public api.List();
                0: aload_0
                1: invokespecial #1
                4: return

  public E add(int, E);
                0: aconst_null
                1: areturn
}
```

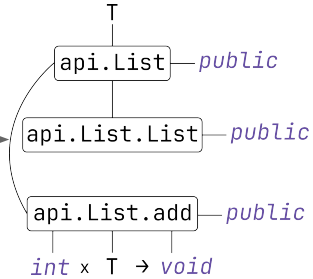


V2

```
public final class List<E> T> {
  private int double size listSize;
  public E void add(int pos, E T elem) {
    ...
  }
}
```

```
public class api.List<T> {
  public api.List();
                0: aload_0
                1: invokespecial #1
                4: return

  public void add(int, T);
                0: aconst_null
}
```



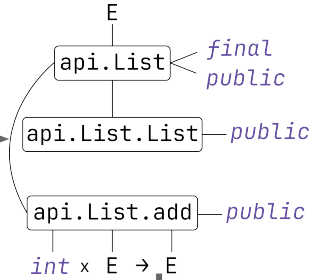
Roseau

V1

```
public final class List<E> {
  private int size;
  public E add(int pos, E elem) {
    ...
  }
}
```

```
public final class api.List<E> {
  public api.List();
                0: aload_0
                1: invokespecial #1
                4: return

  public E add(int, E);
                0: aconst_null
                1: areturn
}
```



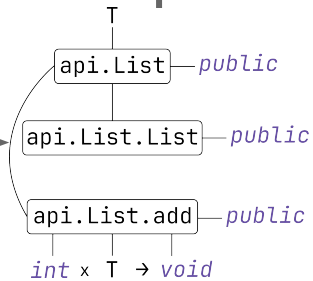
Diff

V2

```
public final class List<E T> {
  private int double size listSize;
  public E void add(int pos, E T elem) {
    ...
  }
}
```

```
public class api.List<T> {
  public api.List();
                0: aload_0
                1: invokespecial #1
                4: return

  public void add(int, T);
                0: aconst_null
}
```



Roseau – BCs as Diffs of API Models

- Translating JLS rules as detection rules on API models
- Expressing these rules at the API level is much easier than at the AST/classfile level

Kind	Detection Rule	Binary	Source
Type removed	$T \in A \wedge \nexists\, T' \in A' \mid T \sim T'$	✗	✗
Type now protected	$T \sim T' \wedge \text{public}(T) \wedge \text{protected}(T')$	✗	✗
Type kind changed	$T \sim T' \wedge \text{typeKind}(T) \neq \text{typeKind}(T')$	✗	✗
Supertype removed	$T \sim T' \wedge \exists\, S \in \text{superTypes}(T) \mid \text{exported}(S) \wedge T' \not\sqsubseteq S$	✗	✗
Type variable removed	$T \sim T' \wedge \lvert\text{typeVars}(T)\rvert > \lvert\text{typeVars}(T')\rvert$	✓	✗
Type variable added	$T \sim T' \wedge \lvert\text{typeVars}(T')\rvert > \lvert\text{typeVars}(T)\rvert > 0$	✓	✗
Type variable changed	$T \sim T' \wedge \exists i \in [1, \lvert\text{typeVars}(T)\rvert], \exists b' \in \text{bounds}(\text{typeVar}_i(T')), \forall b \in \text{bounds}(\text{typeVar}_i(T)) \mid b \not\sqsubseteq b'$	✓	✗
Type now final	$T \sim T' \wedge \neg\text{effectivelyFinal}(T) \wedge \text{effectivelyFinal}(T')$	✗	✗
Type now abstract	$T \sim T' \wedge \neg\text{effectivelyFinal}(T) \wedge \text{effectivelyFinal}(T')$	✗	✗
Nested type now static	$T \sim T' \wedge \neg\text{static}(T) \wedge \text{static}(T')$	✗	✗
Nested type no longer static	$T \sim T' \wedge \text{static}(T) \wedge \neg\text{static}(T')$	✗	✗
Class now checked exception	$T \sim T' \wedge \text{uncheckedException}(T) \wedge \text{checkedException}(T')$	✓	✗
Abstract method added to type	$T \sim T' \wedge \exists m' \in \text{allMethods}(T') \mid \text{abstract}(m') \wedge \nexists m \in \text{allMethods}(T) \mid m \sim m'$	✓	✗
Executable removed	$e \in \text{executables}(T) \wedge \nexists\, e' \in \text{executables}(T') \mid e \sim e'$	✗	✗
Executable now protected	$e \sim e' \wedge \text{public}(e) \wedge \text{protected}(e')$	✗	✗
Executable checked exception removed	$e \sim e' \wedge \exists\, t \in \text{thrown}(e) \mid \nexists t' \in \text{thrown}(e'),\ t' \sqsubseteq t$	✓	✗
Executable checked exception added	$e \sim e' \wedge \exists\, t' \in \text{thrown}(e') \mid \nexists\, t \in \text{thrown}(e),\ t' \not\sqsubseteq t$	✓	✗
Executable parameter generics changed	$e \sim e' \wedge \exists\, i \in [1, \lvert\text{params}(e)\rvert] \mid \lvert\text{typeArgs}(\text{param}_i(e))\rvert \neq \lvert\text{typeArgs}(\text{param}_i(e'))\rvert$	✓	✗
Executable type variable removed	$e \sim e' \wedge \lvert\text{typeVars}(e)\rvert > \lvert\text{typeVars}(e')\rvert \wedge (\text{isMethod}(e) \vee \lvert\text{typeVars}(e)\rvert > 1)$	✓	✗
Executable type variable added	$e \sim e' \wedge \lvert\text{typeVars}(e')\rvert > \lvert\text{typeVars}(e)\rvert > 0$	✓	✗
Executable type variable changed	$e \sim e' \wedge \exists\, i \in [1, \lvert\text{typeVars}(e)\rvert], \exists\, b' \in \text{bounds}(\text{typeVar}_i(e')), \forall\, b \in \text{bounds}(\text{typeVar}_i(e)) \mid b \not\sqsubseteq b'$	✓	✗
Method now final	$m \sim m' \wedge \neg\text{effectivelyFinal}(m) \wedge \text{effectivelyFinal}(m')$	✗	✗
Method now static	$m \sim m' \wedge \neg\text{static}(m) \wedge \text{static}(m')$	✗	✓
Method no longer static	$m \sim m' \wedge \text{static}(m) \wedge \neg\text{static}(m')$	✗	✗
Method now abstract	$m \sim m' \wedge \neg\text{abstract}(m) \wedge \text{abstract}(m')$	✗	✗
Method return type changed	$m \sim m' \wedge \text{returnType}(m) \neq \text{returnType}(m')$	✗	✗
Field removed	$f \in \text{fields}(T) \wedge \nexists\, f' \in \text{fields}(T') \mid f \sim f'$	✗	✗
Field now protected	$f \sim f' \wedge \text{public}(f) \wedge \text{protected}(f')$	✗	✗
Field now final	$f \sim f' \wedge \neg\text{final}(f) \wedge \text{final}(f')$	✗	✗
Field now static	$f \sim f' \wedge \neg\text{static}(f) \wedge \text{static}(f')$	✗	✓
Field no longer static	$f \sim f' \wedge \text{static}(f) \wedge \neg\text{static}(f')$	✗	✗
Field type changed	$f \sim f' \wedge \text{type}(f) \neq \text{type}(f')$	✗	✗

CHAPTER 13

Binary Compatibility

13.4.6 Class Body and Member Declarations

No incompatibility with pre-existing binaries is caused by adding an instance (respectively `static`) member that has the same name and accessibility (for fields), or same name and accessibility and signature and return type (for methods), as an instance (respectively `static`) member of a superclass or subclass. No error occurs even if the set of classes being linked would encounter a compile-time error.

Deleting a class member or constructor that is not declared `private` may cause a linkage error if the member or constructor is used by a pre-existing binary.

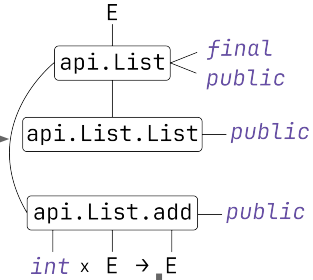
Roseau

V1

```
public final class List<E> {
  private int size;
  public E add(int pos, E elem) {
    ...
  }
}
```

```
public final class api.List<E> {
  public api.List();
                0: aload_0
                1: invokespecial #1
                4: return

  public E add(int, E);
                0: aconst_null
                1: areturn
}
```



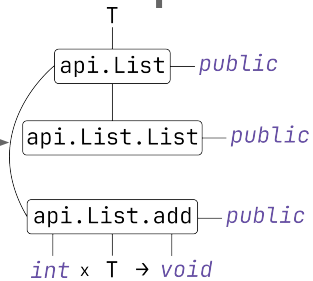
Diff

V2

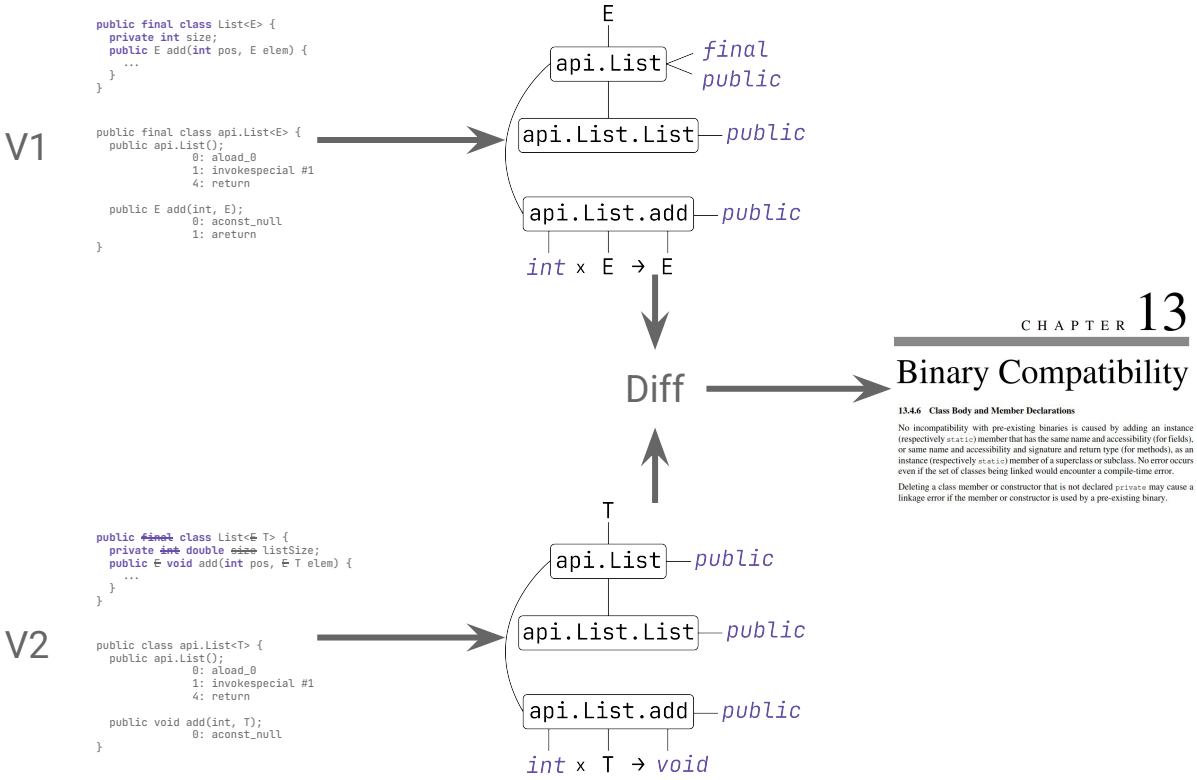
```
public final class List<E T> {
  private int double size listSize;
  public E void add(int pos, E T elem) {
    ...
  }
}
```

```
public class api.List<T> {
  public api.List();
                0: aload_0
                1: invokespecial #1
                4: return

  public void add(int, T);
                0: aconst_null
}
```



Roseau


V1
public final class List<E> {
private int size;
public E add(int pos, E elem) {
...
}
}
public final class api.List<E> {
public api.List();
0: aload_0
1: invokespecial #1
4: return
public E add(int, E);
0: aconst_null
1: areturn
}
E
api.List
final
public
api.List.List
public
api.List.add
public
int x E → E
Diff
CHAPTER 13
Binary Compatibility
13.4.6 Class Body and Member Declarations
No incompatibility with pre-existing binaries is caused by adding an instance (respectively static) member that has the same name and accessibility (for fields), or same name and accessibility and signature and return type (for methods), as an instance (respectively static) member of a superclass or subclass. No error occurs even if the set of classes being linked would encounter a compile-time error.
Deleting a class member or constructor that is not declared private may cause a linkage error if the member or constructor is used by a pre-existing binary.
V2
public final class List<E> T> {
private int double size listSize;
public E void add(int pos, E T elem) {
...
}
}
public class api.List<T> {
public api.List();
0: aload_0
1: invokespecial #1
4: return
public void add(int, T);
0: aconst_null
}
T
api.List
public
api.List.List
public
api.List.add
public
int x T → void

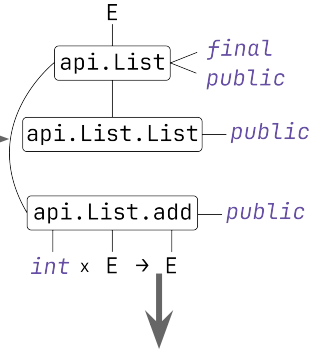
Roseau

V1

```
public final class List<E> {
  private int size;
  public E add(int pos, E elem) {
    ...
  }
}
```

```
public final class api.List<E> {
  public api.List();
              0: aload_0
              1: invokespecial #1
              4: return

  public E add(int, E);
              0: aconst_null
              1: areturn
}
```

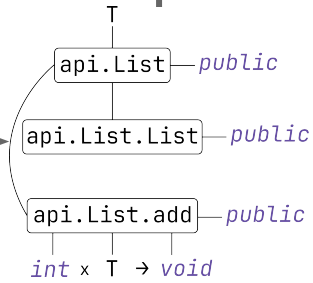


V2

```
public final class List<E> T> {
  private int double size listSize;
  public E void add(int pos, E T elem) {
    ...
  }
}
```

```
public class api.List<T> {
  public api.List();
              0: aload_0
              1: invokespecial #1
              4: return

  public void add(int, T);
              0: aconst_null
}
```



Diff

CHAPTER 13

Binary Compatibility

13.4.6 Class Body and Member Declarations

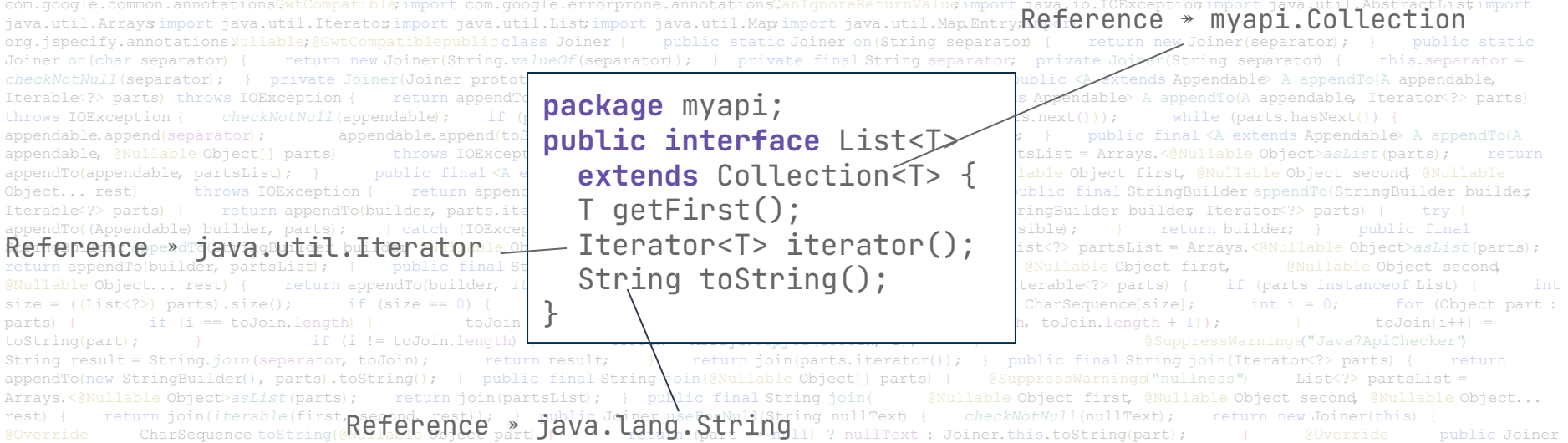
No incompatibility with pre-existing binaries is caused by adding an instance (respectively `static`) member that has the same name and accessibility (for fields), or same name and accessibility and signature and return type (for methods), as an instance (respectively `static`) member of a superclass or subclass. No error occurs even if the set of classes being linked would encounter a compile-time error.

Deleting a class member or constructor that is not declared `private` may cause a linkage error if the member or constructor is used by a pre-existing binary.

Breaking Changes:
- METHOD_RETURN_TYPE_CHANGED

Features / Optimizations

- Incremental Parsing



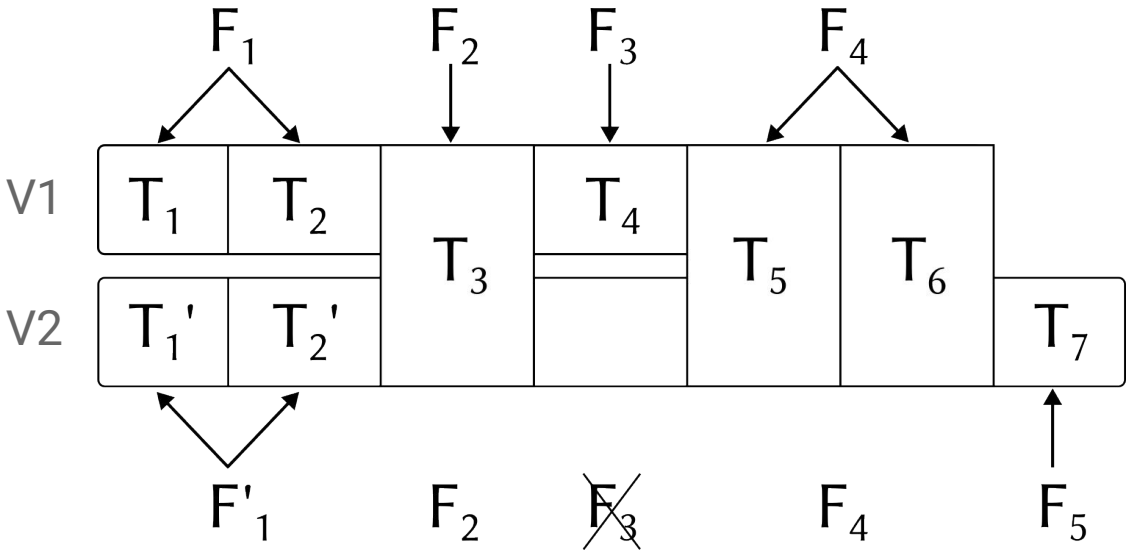
Features / Optimizations

- Incremental Parsing
- *“Diet”* Parsing

```
public class ArrayList {
  public int indexOf(Object o) {
    E[] a = this.a;
    if (o == null) {
      for (int i = 0; i < a.length; i++)
        if (a[i] == null)
          return i;
    } else {
      for (int i = 0; i < a.length; i++)
        if (o.equals(a[i]))
          return i;
    }
    return -1;
  }
}
```

Features / Optimizations

- Incremental Parsing
- *“Diet”* Parsing
- Partial Updates



Evaluation

Accuracy

API Evolution Data Corpus (*Jezek et al.*[1] and *Ochoa et al.*[2])

[1] K. Jezek and J. Dietrich, “Api evolution and compatibility: A data corpus and tool evaluation.” *J. Object Technol.*, vol. 16, no. 4, pp. 2–1, 2017.
[2] L. Ochoa, T. Degueule, J.-R. Falleri, and J. Vinju, “Breaking bad? semantic versioning and impact of breaking changes in maven central: An external and differentiated replication study,” *Empirical Software Engineering*, vol. 27, no. 3, p. 61, 2022.

Accuracy

API Evolution Data Corpus (*Jezek et al.*[1] and *Ochoa et al.*[2])

	All			Source			Binary		
	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1
JAPICMP	0.90	0.82	0.86	0.78	0.80	0.79	0.90	**0.98**	**0.94**
REVAPI	0.85	0.97	0.91	0.77	0.96	0.85	0.89	0.95	0.92
ROSEAU	**0.98**	**0.99**	**0.99**	**0.88**	**1.00**	**0.94**	**0.91**	**0.98**	**0.94**

[1] K. Jezek and J. Dietrich, “Api evolution and compatibility: A data corpus and tool evaluation.” *J. Object Technol.*, vol. 16, no. 4, pp. 2–1, 2017.
[2] L. Ochoa, T. Degueule, J.-R. Falleri, and J. Vinju, “Breaking bad? semantic versioning and impact of breaking changes in maven central: An external and differentiated replication study,” *Empirical Software Engineering*, vol. 27, no. 3, p. 61, 2022.

Performance

Dataset w/ Top 100 Artifacts from Maven

⏱️ Artifact last stable version on itself

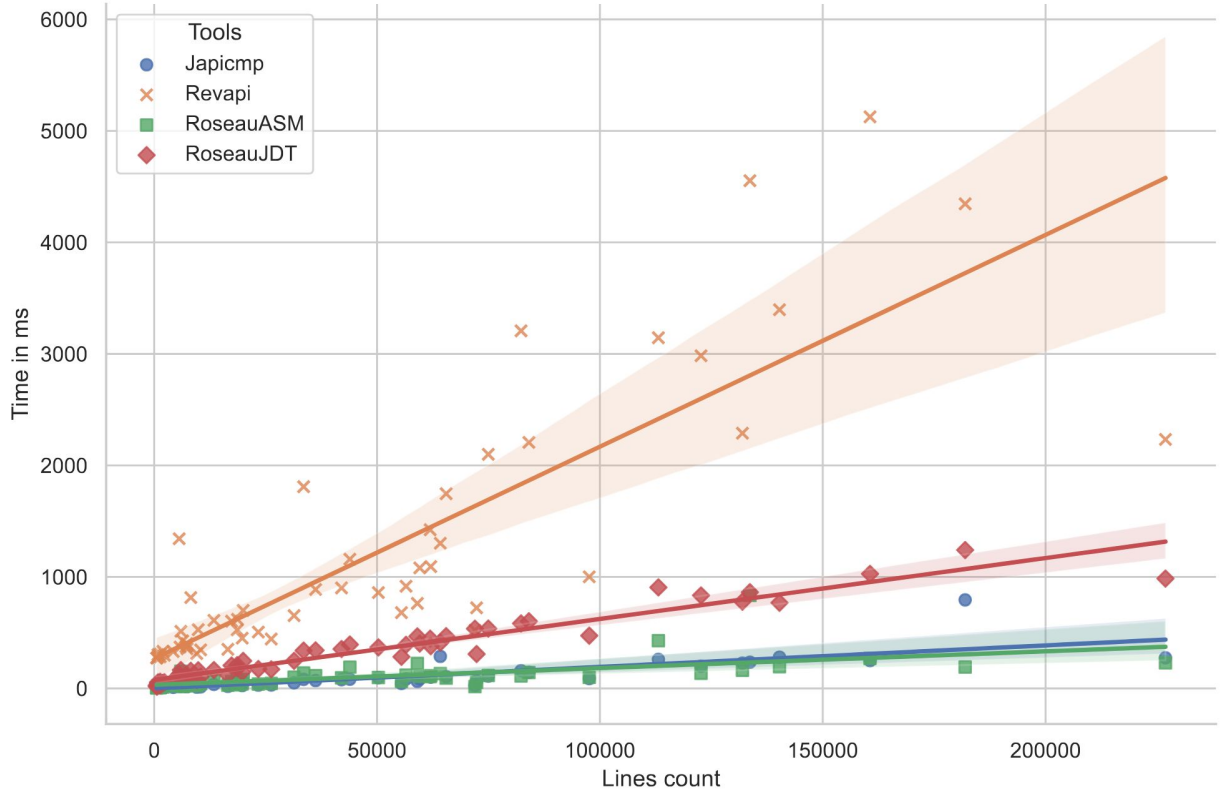
Java Microbenchmark Harness (JMH)
→ **50** measures by artifact and tool

Performance

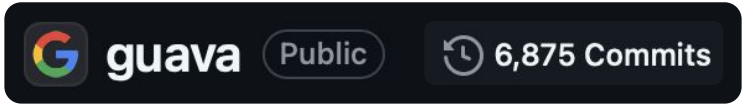
Dataset w/ Top 100 Artifacts from Maven

⏱️ Artifact last stable version on itself

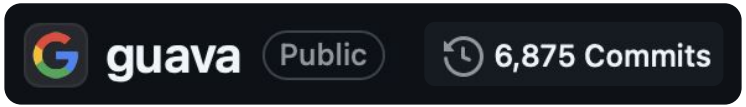
Java Microbenchmark Harness (JMH)
→ **50** measures by artifact and tool



Large-scale Longitudinal Studies



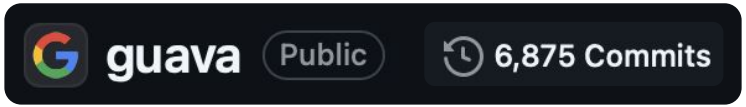
Large-scale Longitudinal Studies



Preliminary study: 100 latest commits of Google's Guava

Method	Mean commit time (ms)	Total time (minutes)
Packaging JAR	68,750	7,836
Roseau w/o incremental	1,065	121
Roseau w/ incremental	81	9

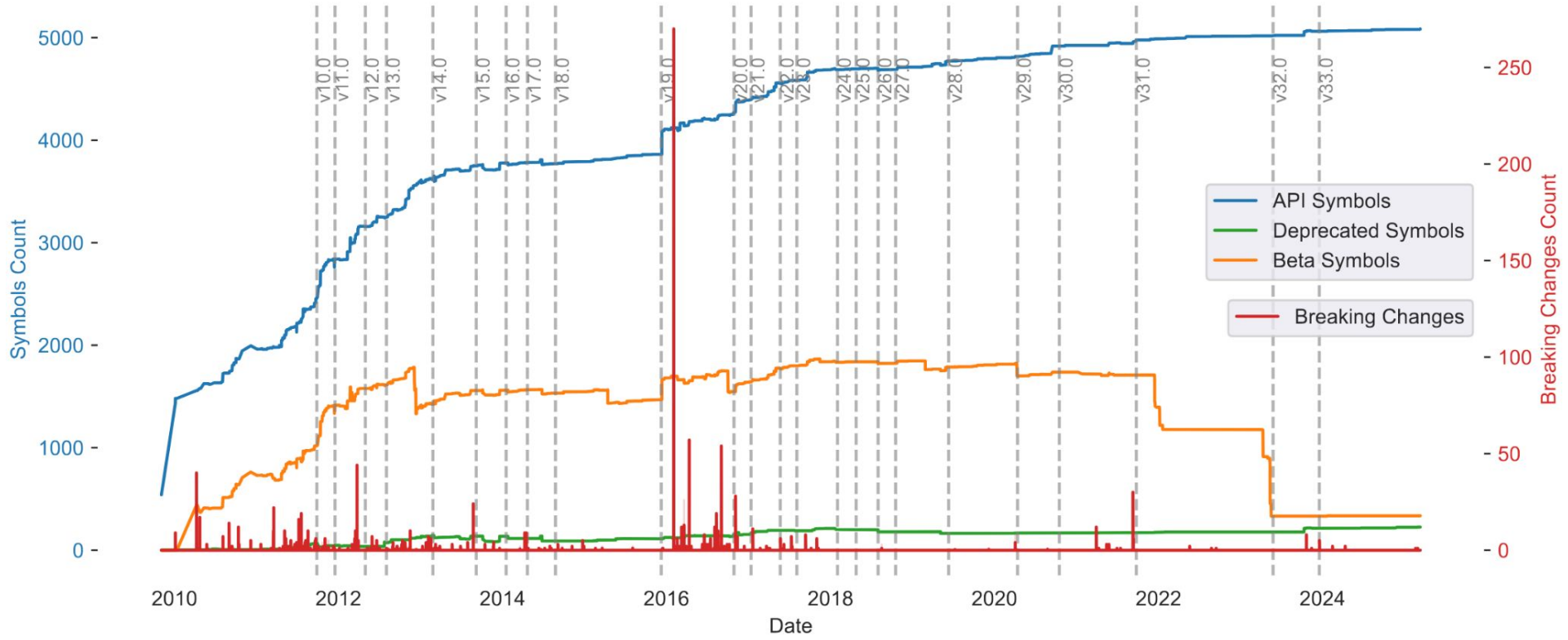
Large-scale Longitudinal Studies



Preliminary study: 100 latest commits of Google's Guava

Method	Mean commit time (ms)	Total time (minutes)
Packaging JAR	68,750	7,836
Roseau w/o incremental	1,065	121
Roseau w/ incremental	81	9

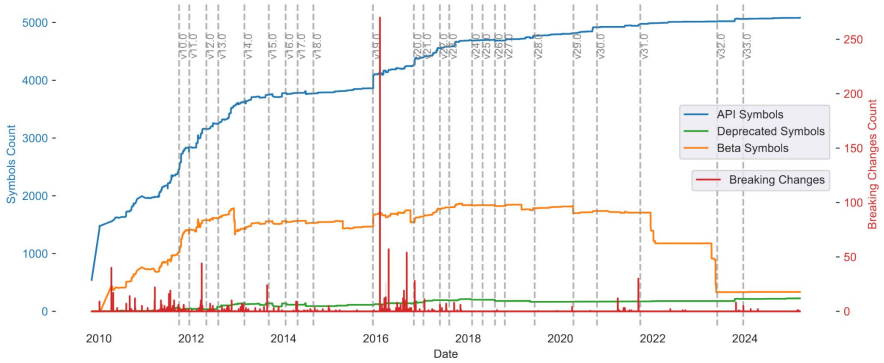
5,5 days vs **9 minutes**



Evolution of Guava API and introduction of Breaking Changes over time

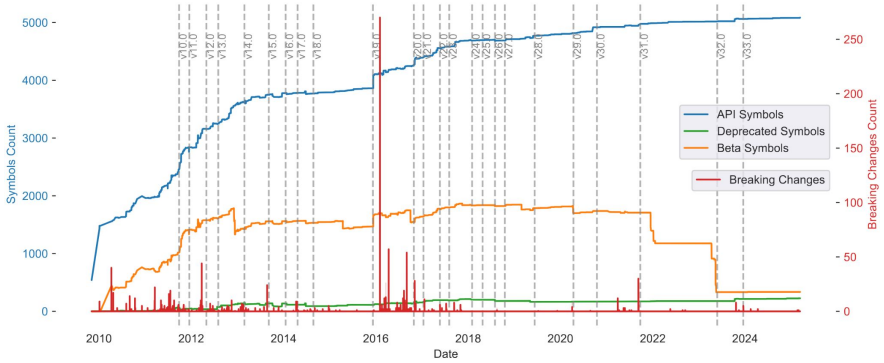
Future Work

Qualitative case studies of some popular libraries



Future Work

Qualitative case studies of some popular libraries



Maintain and improve Roseau

junit-team / **junit-framework**

Commit `465a06f`

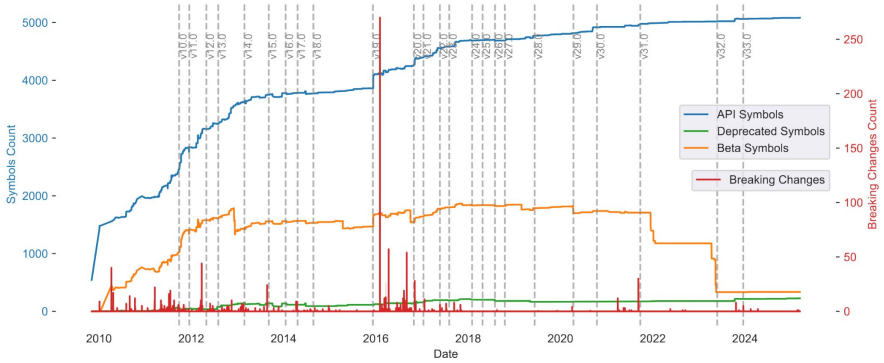
marcphilipp committed on Jul 11 Verified

```
Try out roseau to analyze API changes between versions
```

`marc/roseau` 1 parent `eaf5ae9` commit `465a06f`

Future Work

Qualitative case studies of some popular libraries



Combinatorial benchmark

V1 API :

288 types

47K members → 1,5M V2

~80K clients

Maintain and improve Roseau

junit-team / **junit-framework**

Commit `465a06f`

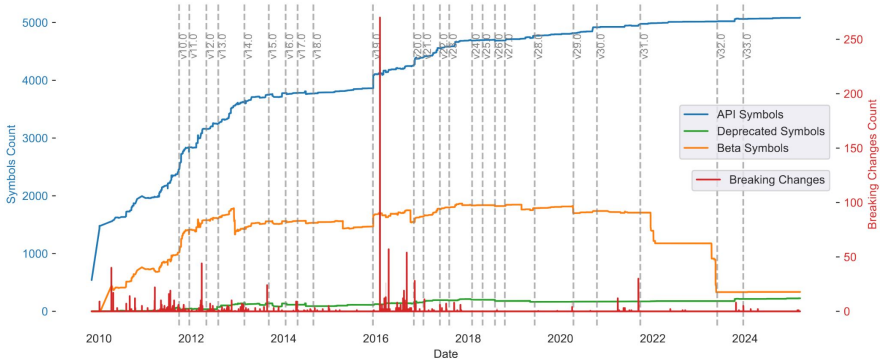
marcphilipp committed on Jul 11 Verified

```
Try out roseau to analyze API changes between versions
```

`marc/roseau` 1 parent `eaf5ae9` commit `465a06f`

Future Work

Qualitative case studies of some popular libraries



Maintain and improve Roseau

junit-team / junit-framework

Commit 465a06f

marcphilipp committed on Jul 11 Verified

Try out roseau to analyze API changes between versions

marc/roseau 1 parent eaf5ae9 commit 465a06f

Combinatorial benchmark

V1 API :

288 types

47K members

→ 1,5M V2

~80K clients

Live feedback in IDEs

```
boolean isEffectivelyFinal() {   6 clients   1 broken   Revert
  return isFinal()
      || isNative()
      || containingType.isEffectivelyFinal();
}
```

ROSEAU (REED)

Accuracy

API Evolution Data Corpus (*Jezek et al.*[1] and *Ochoa et al.*[2])

	All			Source	Binary
	Prec.	Rec.	F_1	Prec.	
JAPICMP	0.90	0.82	0.86	0.78	
REVAPI	0.85	0.97	0.91	0.77	
ROSEAU	**0.98**	**0.99**	**0.99**	**0.88**	

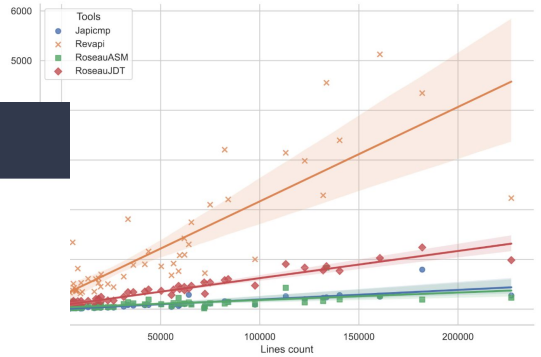
[1] K. Jezek and J. Dietrich, "Api evolution and compatibility: A data corpus and tool evaluation." *J. Object Technol.*, vol. 16, no. 4, pp. 2–1, 20
[2] L. Ochoa, T. Degueule, J.-R. Falleri, and J. Vinju, "Breaking bad? semantic versioning and impact of breaking changes in maven central: A

Performance

Dataset w/ Top 100 Artifacts from Maven

Artifact last stable version on itself

Java Microbenchmark Harness (JMH)



Large-scale Longitudinal Studies

guava Public 6,875 Commits

Preliminary study: 100 latest commits of Google's Guava

Method	Mean commit time (ms)	Total time (minutes)
Packaging JAR	68,750	7,836
ROSEAU w/o incremental	1,065	121
ROSEAU w/ incremental	81	9

5,5 days vs **9 minutes**

ROSEAU (REED)

Accuracy

API Evolution Data Corpus (*Jezek et al.*[1] and *Ochoa et al.*[2])

	All			Source	Binary
	Prec.	Rec.	F_1	Prec.	
JAPICMP	0.90	0.82	0.86	0.78	
REVAPI	0.85	0.97	0.91	0.77	
ROSEAU	**0.98**	**0.99**	**0.99**	**0.88**	

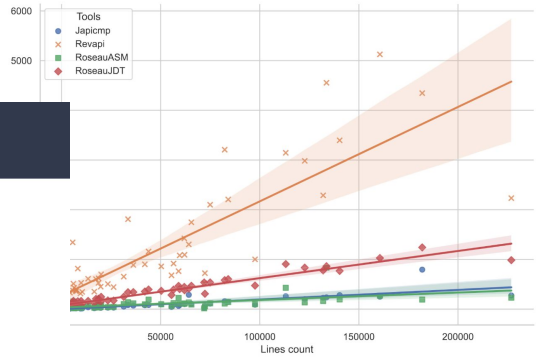
[1] K. Jezek and J. Dietrich, "Api evolution and compatibility: A data corpus and tool evaluation." *J. Object Technol.*, vol. 16, no. 4, pp. 2–1, 20
[2] L. Ochoa, T. Degueule, J.-R. Falleri, and J. Vinju, "Breaking bad? semantic versioning and impact of breaking changes in maven central: A

Performance

Dataset w/ Top 100 Artifacts from Maven

Artifact last stable version on itself

Java Microbenchmark Harness (JMH)



Large-scale Longitudinal Studies

guava Public 6,875 Commits

Preliminary study: 100 latest commits of Google's Guava

Method	Mean commit time (ms)	Total time (minutes)
Packaging JAR	68,750	7,836
ROSEAU w/o incremental	1,065	121
ROSEAU w/ incremental	81	9

5,5 days vs **9 minutes**

"I bend and do not break."

The Oak and the Reed
by Jean de La Fontaine