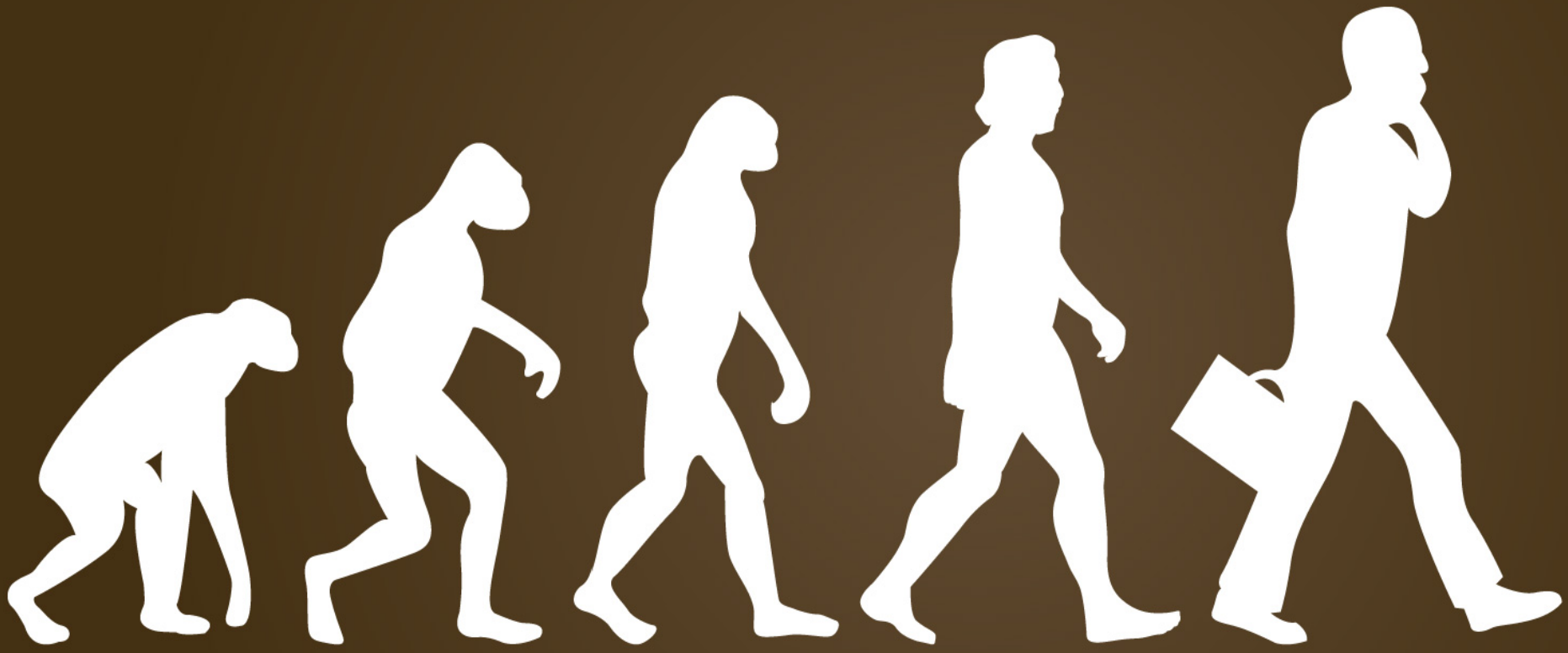
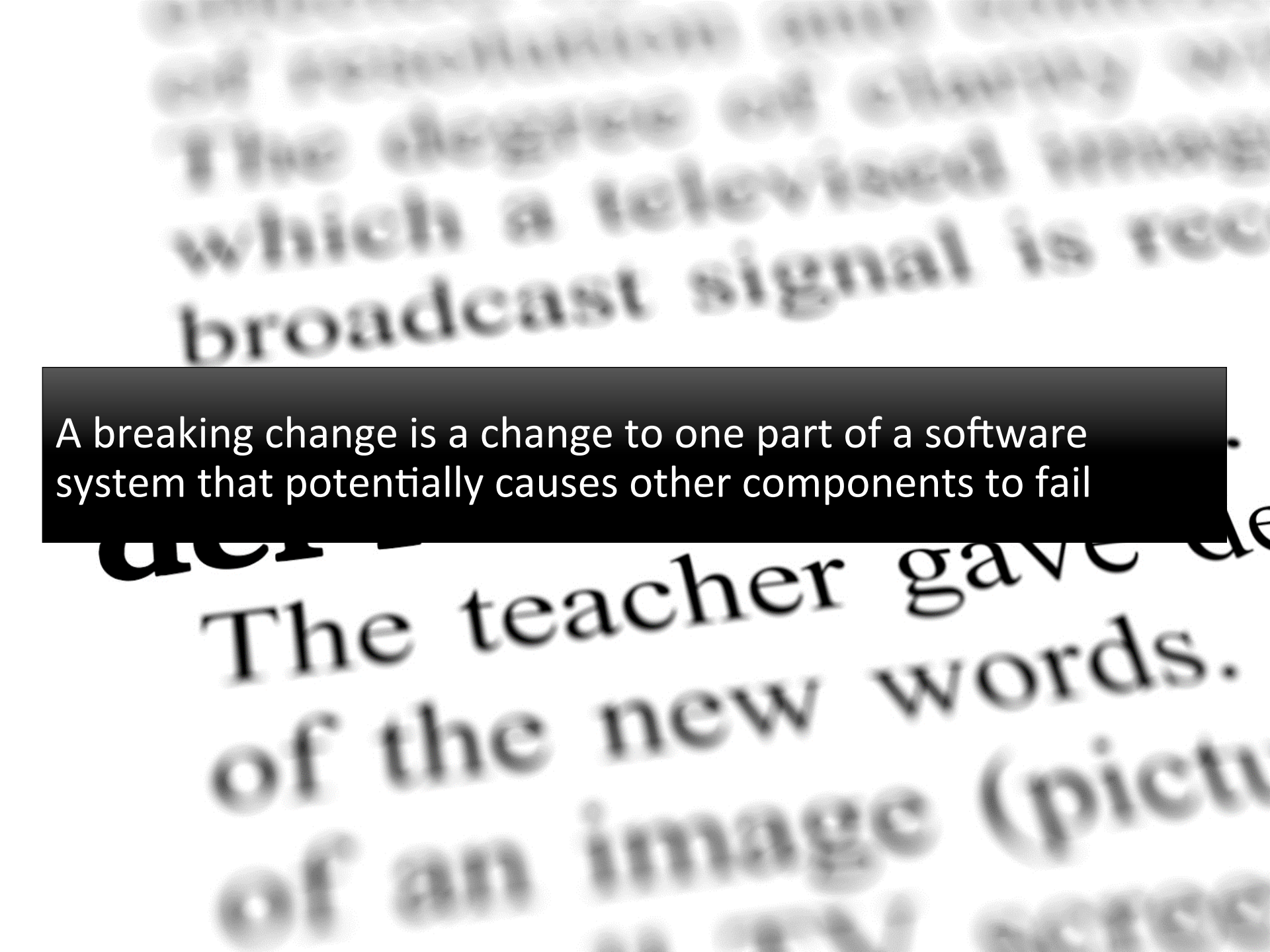
Mining motivated trends of usage of Haskell libraries

Marc Juchli, Lars Krombeen,
Shashank Rao, Chak Shun Yu
Anand Ashok Sawant and Alberto Bacchelli

TU Delft

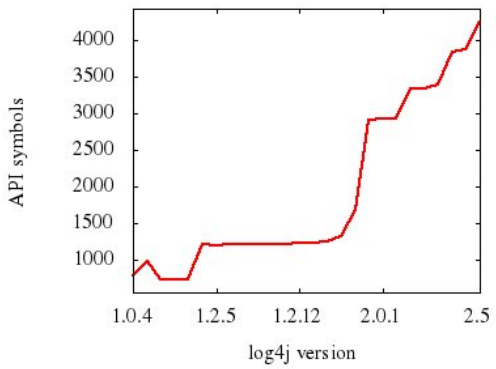




A breaking change is a change to one part of a software system that potentially causes other components to fail

Breaking changes in APIs

Version	Date	Backward Compatibility		Added Methods	Removed Methods
		BC	SC		
2.5	2015-12-07	94.71%	93.50%	402 new	25 removed
2.4.1	2015-10-09	99.27%		41 new	6 removed
2.4	2015-09-20	94.32%	83.85%	503 new	52 removed
2.3	2015-05-11	99.82%	99.64%	55 new	6 removed
2.2	2015-02-23	99.64%	99.52%	17 new	12 removed
2.1	2014-10-20	93.66% added 2 archives	91.04%	291 new	140 removed
2.0.2	2014-08-17	99.81%		7 new	6 removed
2.0.1	2014-07-30	99.87%		24 new	4 removed
2.0	2014-07-13	0% added 9 archives removed 1 archive	0%	0	0



Breaking changes are common!

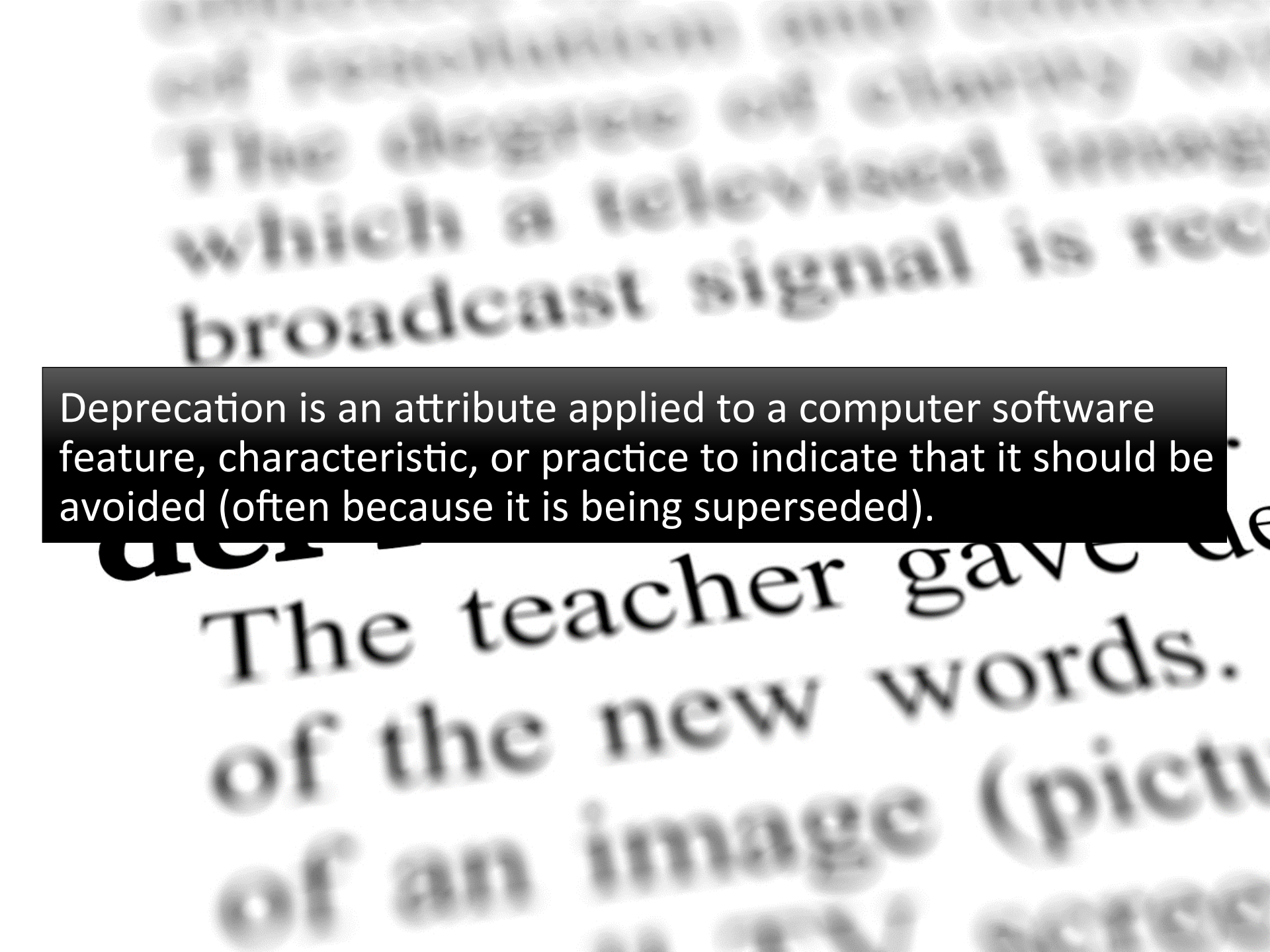
ISSUE

Not enough migration support

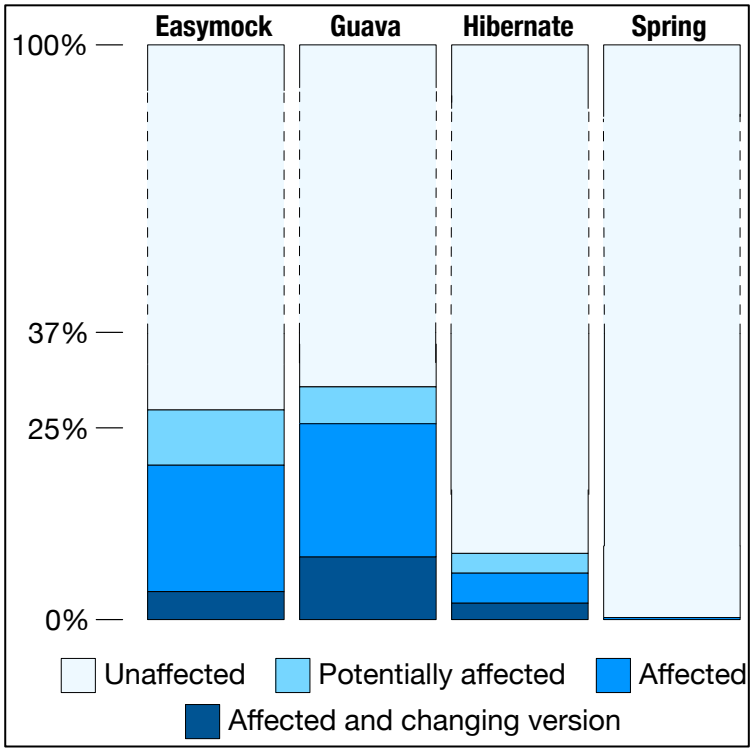
API client's actual behavior

```
if (useVersion2) {
  com.mycompany.library.v2.Converter c = new com.mycompany.library.v2.Converter();
  // configure and run
  c.setOption(...);
  c.convert(in, out);
} else {
  com.mycompany.library.Converter c = new com.mycompany.library.Converter();
  // configure and run
  c.setOption(...);
  c.convert(in, out);
}
```

Feature toggling


Deprecation is an attribute applied to a computer software feature, characteristic, or practice to indicate that it should be avoided (often because it is being superseded).

API client's affected



In some cases many clients are affected

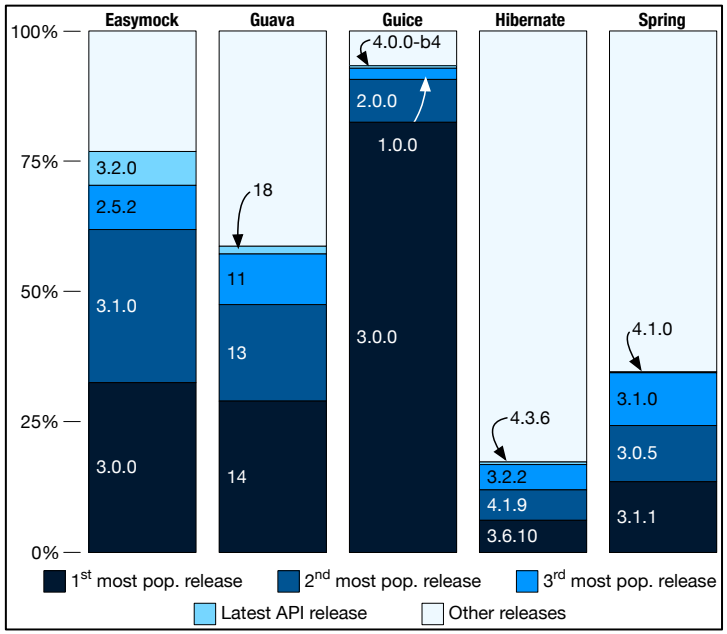
API client's that react

API	Affected clients that react	Median	Max
Easymock	17	11	109
Guava	161	3	283
Hibernate	40	5	59
Spring	10	31	131

ISSUE

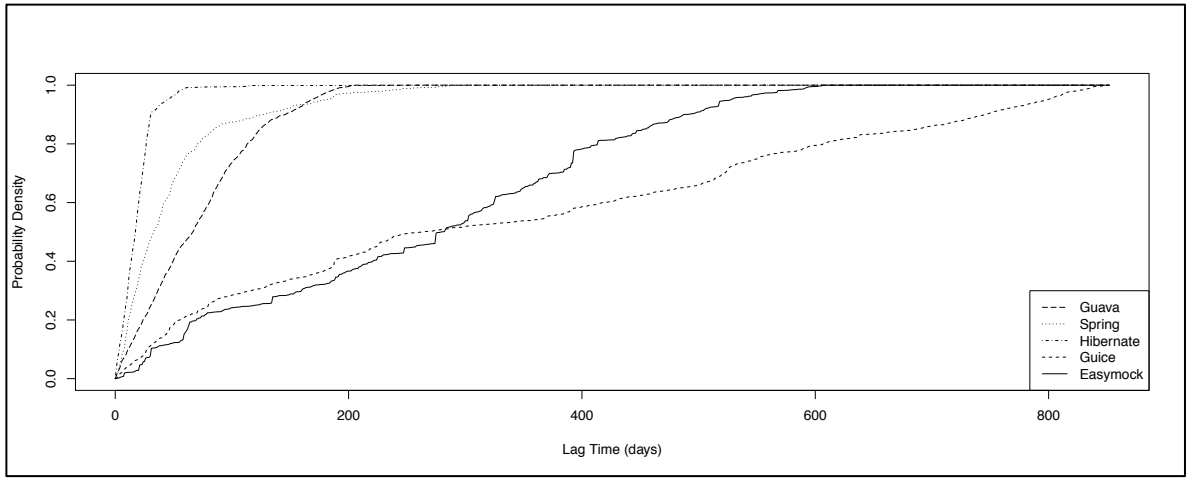
Nobody changes versions of a library!

Usage of API versions in Java



Very few clients upgrade to the latest version

Update time of clients



Median time to upgrade is high

API version trends

Issues in a version of an API lead to its abandonment

Popular versions remain popular

Some versions are untouched

ISSUE

Reasons behind change unknown!

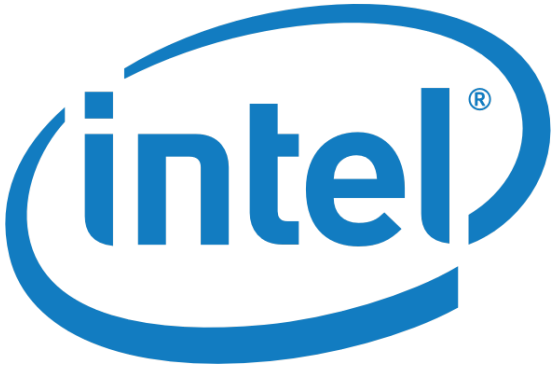

Haskell

Purely functional language

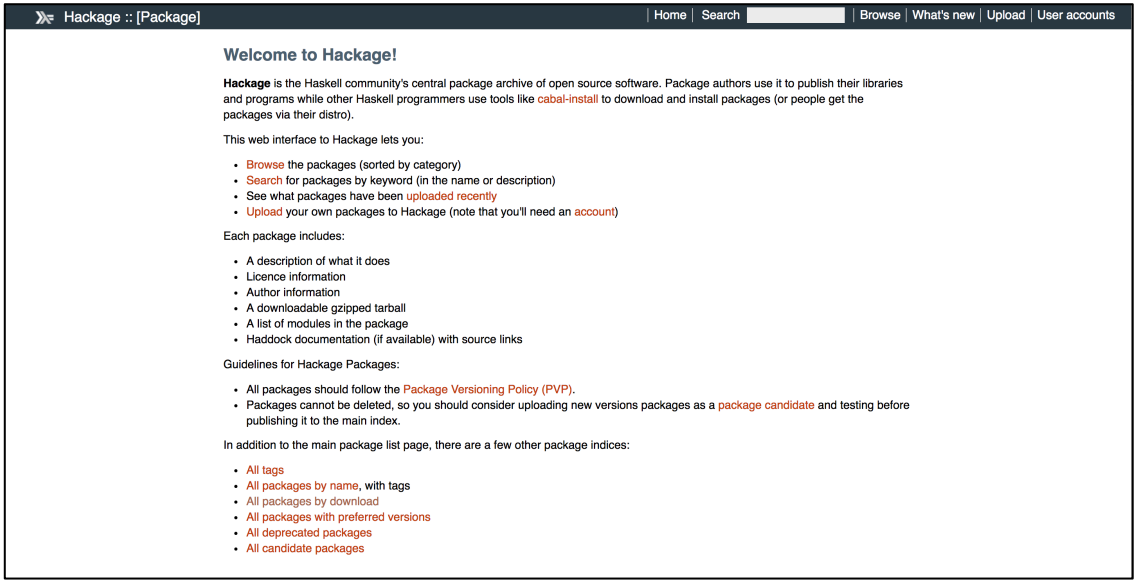
Statically typed

Type inference system

Usage in industry



Hackage



Hackage is a host for Haskell libraries

Popular Haskell packages

Downloaded packages

Package name	Downloads
hashable	18087
transformers-compat	17654
mtl	17541
semigroups	17155
text	16969
tagged	16014
primitive	15940
unordered-containers	15353
random	15300
async	14932
aeson	14557
stm	13457
scientific	13413
parsec	12952
vector	12452
dlist	12344
exceptions	12185
QuickCheck	12176
network	11978
cryptonite	11908
conduit	11624
attoparsec	11237
ansi-terminal	11197
lifted-base	11006
resourcet	10828
tf-random	10360
cereal	10301
base-compat	10210
integer-logarithms	10172
data-default	10152

Hackage indexes the most popular Haskell libaries

Sample CABAL file

```
name:                Poker
version:             0.1.0.0
synopsis:            Initial project template from stack
description:         Please see README.md
build-type:          Simple
extra-source-files:  README.md
cabal-version:       >=1.10

library
  hs-source-dirs:      src
  exposed-modules:     Poker.Data, Poker.Analysis
  build-depends:       base >= 4.7 && < 5,
                       containers
  default-language:    Haskell2010

executable poker-exe
  hs-source-dirs:      app
  main-is:             Main.hs
  ghc-options:         -threaded -rtsopts -with-rtsopts=-N
  build-depends:       base, poker,containers
  default-language:    Haskell2010

source-repository head
  type:                Git
  location:            https://github.com/githubuser/poker
```

Sample CABAL file

```
name:                Poker
version:             0.1.0.0
synopsis:            Initial project template from stack
description:         Please see README.md
build-type:          Simple
extra-source-files:  README.md
cabal-version:       >=1.10

library
  hs-source-dirs:      src
  exposed-modules:     Poker.Data, Poker.Analysis
  build-depends:       base >= 4.7 && < 5,
                       containers
  default-language:    Haskell2010

executable poker-exe
  hs-source-dirs:      app
  main-is:             Main.hs
  ghc-options:         -threaded -rtsopts -with-rtsopts=-N
  build-depends:       base, poker,containers
  default-language:    Haskell2010

source-repository head
  type:                Git
  location:            https://github.com/githubuser/poker
```

Sample CABAL file

```
name:                Poker
version:             0.1.0.0
synopsis:            Initial project template from stack
description:         Please see README.md
build-type:          Simple
extra-source-files:  README.md
cabal-version:       >=1.10

library
  hs-source-dirs:      src
  exposed-modules:     Poker.Data, Poker.Analysis
  build-depends:       base >= 4.7 && < 5,
                       containers
  default-language:    Haskell2010

executable poker-exe
  hs-source-dirs:      app
  main-is:             Main.hs
  ghc-options:         -threaded -rtsopts -with-rtsopts=-N
  build-depends:       base, poker,containers
  default-language:    Haskell2010

source-repository head
  type:                Git
  location:            https://github.com/githubuser/poker
```

Sample CABAL file

```
name:                Poker
version:             0.1.0.0
synopsis:            Initial project template from stack
description:         Please see README.md
build-type:          Simple
extra-source-files:  README.md
cabal-version:       >=1.10

library
  hs-source-dirs:      src
  exposed-modules:     Poker.Data, Poker.Analysis
  build-depends:       base >= 4.7 && < 5,
                       containers
  default-language:    Haskell2010

executable poker-exe
  hs-source-dirs:      app
  main-is:             Main.hs
  ghc-options:         -threaded -rtsopts -with-rtsopts=-N
  build-depends:       base, poker,containers
  default-language:    Haskell2010

source-repository head
  type:                Git
  location:            https://github.com/githubuser/poker
```

Sample CABAL file

```
name:                Poker
version:             0.1.0.0
synopsis:            Initial project template from stack
description:         Please see README.md
build-type:          Simple
extra-source-files:  README.md
cabal-version:       >=1.10

library
  hs-source-dirs:      src
  exposed-modules:     Poker.Data, Poker.Analysis
  build-depends:       base >= 4.7 && < 5,
                       containers
  default-language:    Haskell2010

executable poker-exe
  hs-source-dirs:      app
  main-is:             Main.hs
  ghc-options:         -threaded -rtsopts -with-rtsopts=-N
  build-depends:       base, poker,containers
  default-language:    Haskell2010

source-repository head
  type:                Git
  location:            https://github.com/githubuser/poker
```

Studies performed

1

2

Studies performed

Version popularity

2

Version popularity

Goal: Understand what trends exist regarding the adoption of a version of a package

Outcome: Some versions are adopted immediately most are left unused

Version definitions in Cabal

```
base >= 2

base == 2

base > 2

base < 2

base <= 2
```

Accurate version resolution is a challenge!

ISSUE

```
foo >= 0.5.2 && < 0.6
bar >= 1.1 && <= 1.2.*
```

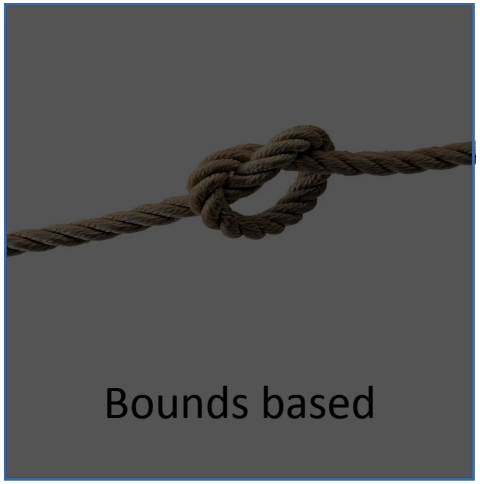
Accurate version resolution

Date based

Bounds based

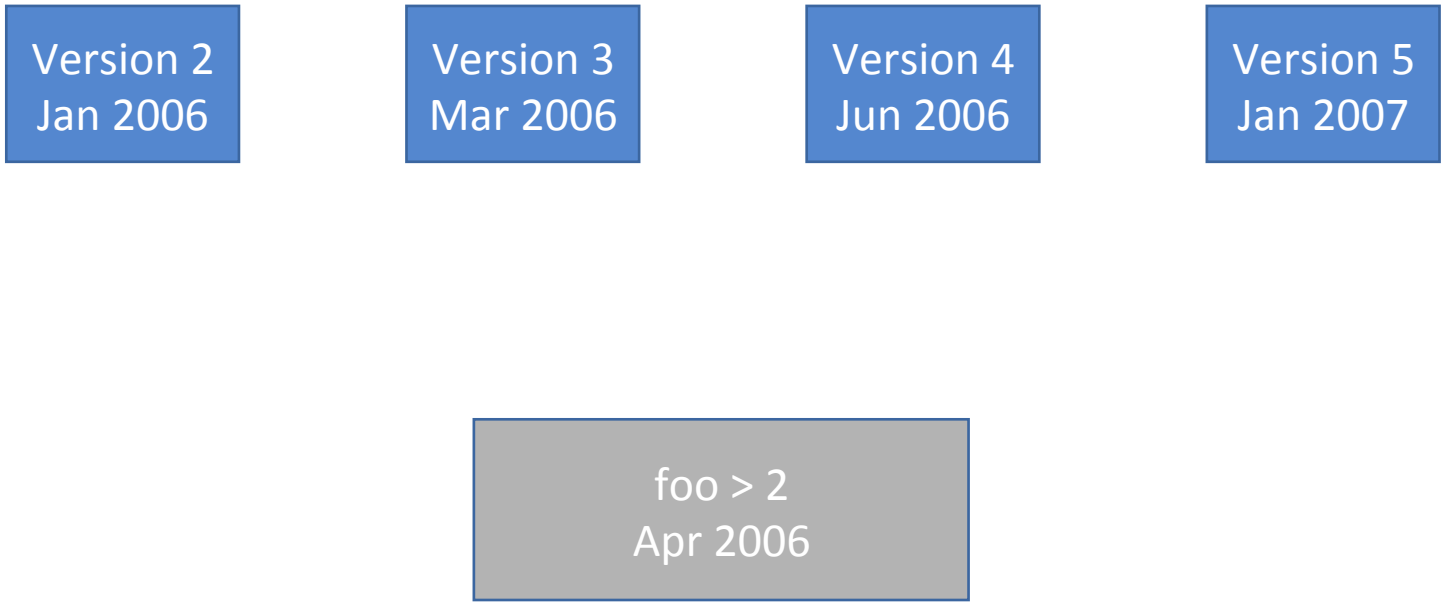
Accurate version resolution

Date based

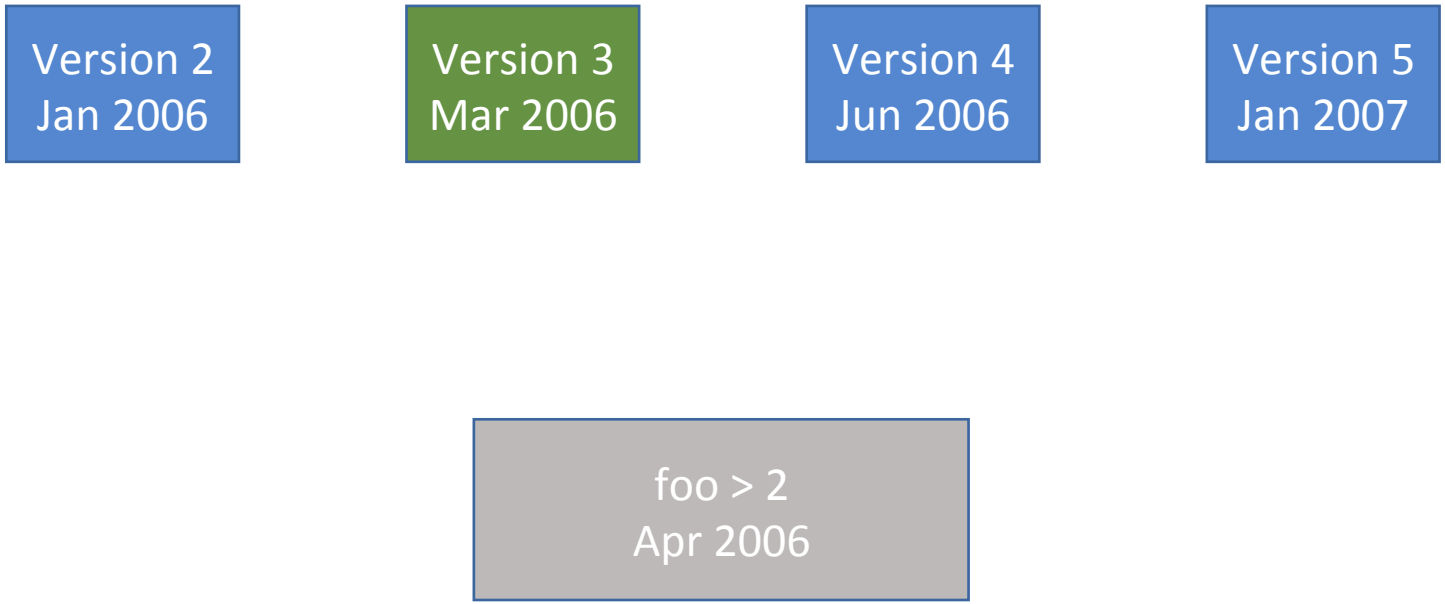


Bounds based

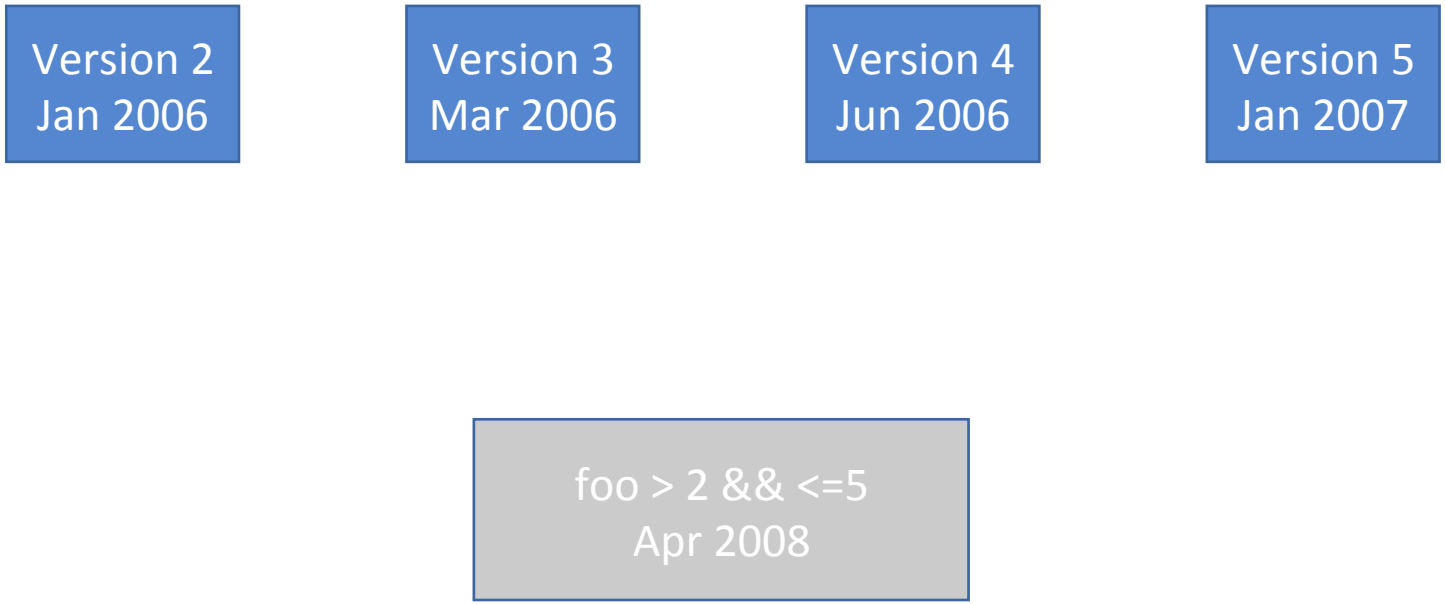
Date based resolution



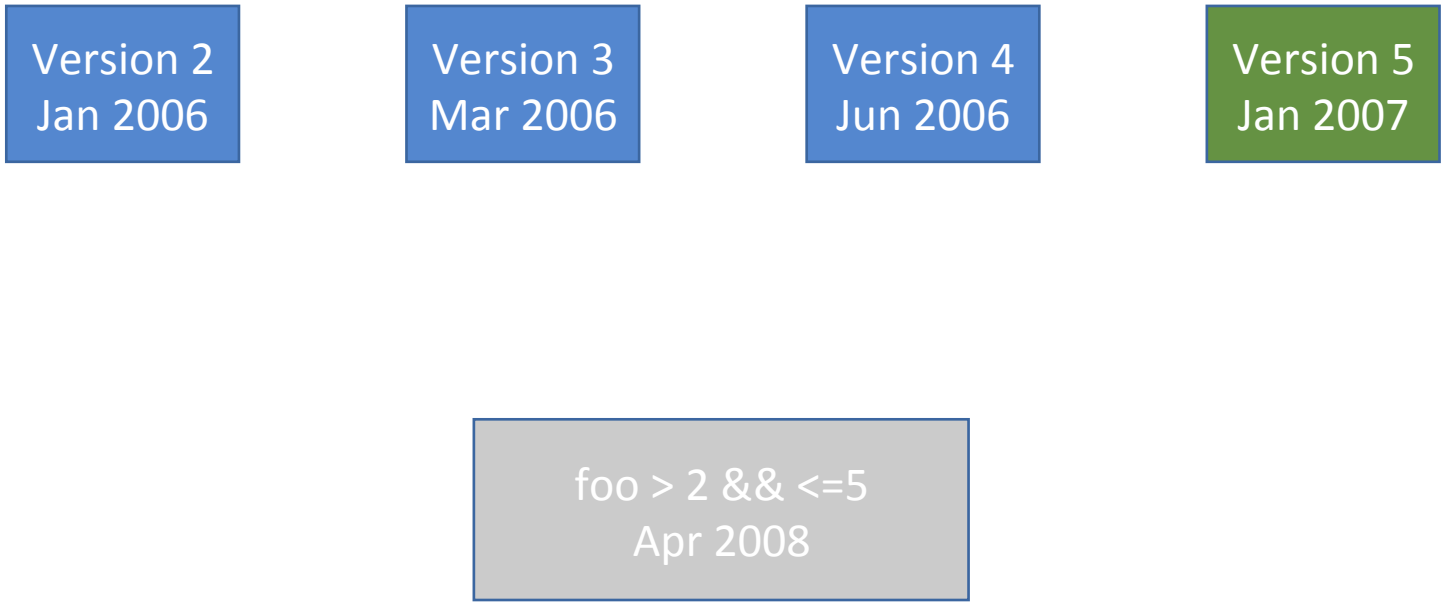
Date based resolution



Date based resolution



Date based resolution



ISSUE

Results in more version upgrades

Upgrade decision unknown

Accurate version resolution

Date based

Bounds based

Bounds based

Select the boundary cases

Exclude wildcards

Accurate version resolution

```
foo >= 0.5.2 && < 0.6

bar >= 1.1 && <= 1.2.*
```

Accurate version resolution

```
foo >= 0.5.2 && < 0.6   -> 0.5

bar >= 1.1 && <= 1.2.* -> 1.1
```

Results

Some versions of an API are never used

Versions that are popular are very popular

Popularity depends on early adopters

Results in comparison with Java

Similar characteristics to the Apache ecosystem

Comparable to our own previous findings

Studies performed

Version popularity

Motivation to change version

Motivation to change version

Goal: Understand why developers change the version of a package being used

Outcome: The most popular reason is to remove a dependency is compatibility issues

Cleaning commit messages

Remove non-alphabetical characters and stop-words

Stem the remaining words

Focus only on commits that change cabal file

Categories of commits

Addition of lines

Deletion of lines

Corrective

Adaptive

Perfective

Reasons behind change

Change	Category	Reason
Addition	Corrective	Compatible with the project
Addition	Adaptive	Dependency needed for new patch or feature
Addition	Perfective	Improvement of project
Deletion	Corrective	Removed for bugfix
Deletion	Adaptive	Compatibility issues
Deletion	Perfective	Not relevant / unused

Size of dataset

1250 projects on Hackage

2,220 unique dependencies

Over 45,000 commits that change dependency

Selected results

Majority of additions were adaptive in nature

Majority of the deletions were adaptive

Very few changes due to issues in package

Usage of a new version is rare

The lens project

Lens had the most changes − 515

30% of them were upgrades

Majority of the changes were to maintain stability

What does it this mean?

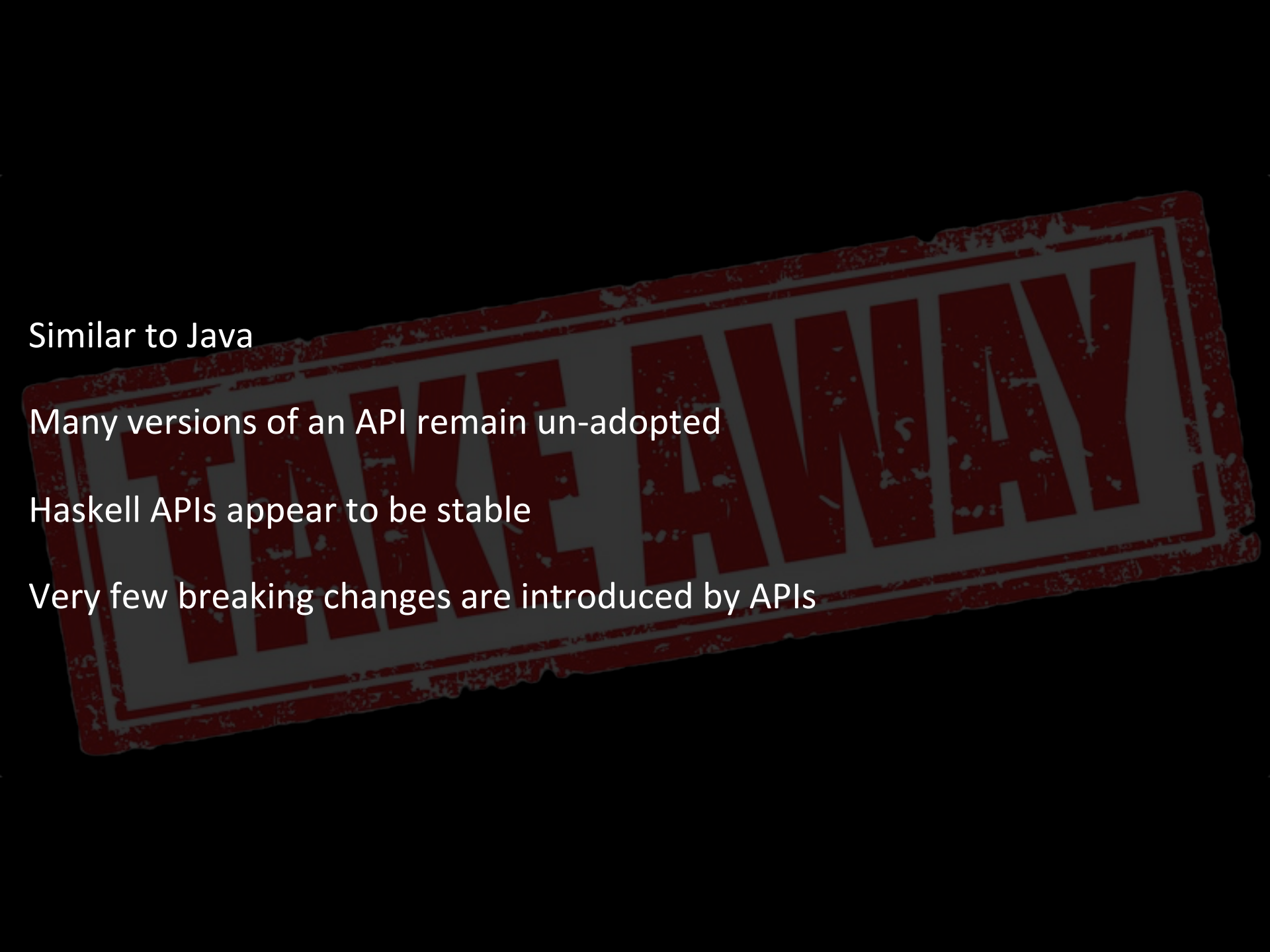
We can distill some version change reasoning

This can be used to understand whether or not an ecosystem is stable

Results can be used to recommend versions of an

API


TAKE AWAY



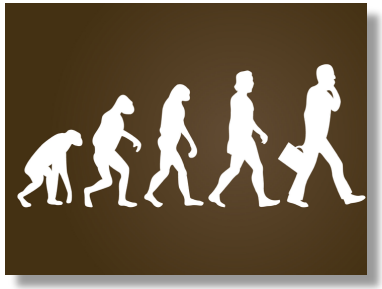
Similar to Java

Many versions of an API remain un-adopted

Haskell APIs appear to be stable

Very few breaking changes are introduced by APIs

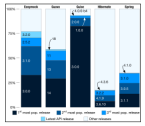
Conclusion



Breaking changes in APIs

Breaking changes are common!

Usage of API versions in Java



Very few clients upgrade to the latest version

Haskell

Hackage

Hackage is a host for Haskell libraries

Accurate version resolution

```
foo >= 0.5.2 && < 0.6  -> 0.5

bar >= 1.1 && <= 1.2.* -> 1.1
```

Reasons behind change

Change	Category	Reason
Addition	Corrective	Compatible with the project
Addition	Adaptive	Dependency needed for new patch or feature
Addition	Perfective	Improvement of project
Deletion	Corrective	Removed for bugfix
Deletion	Adaptive	Compatibility issues
Deletion	Perfective	Not relevant / unused