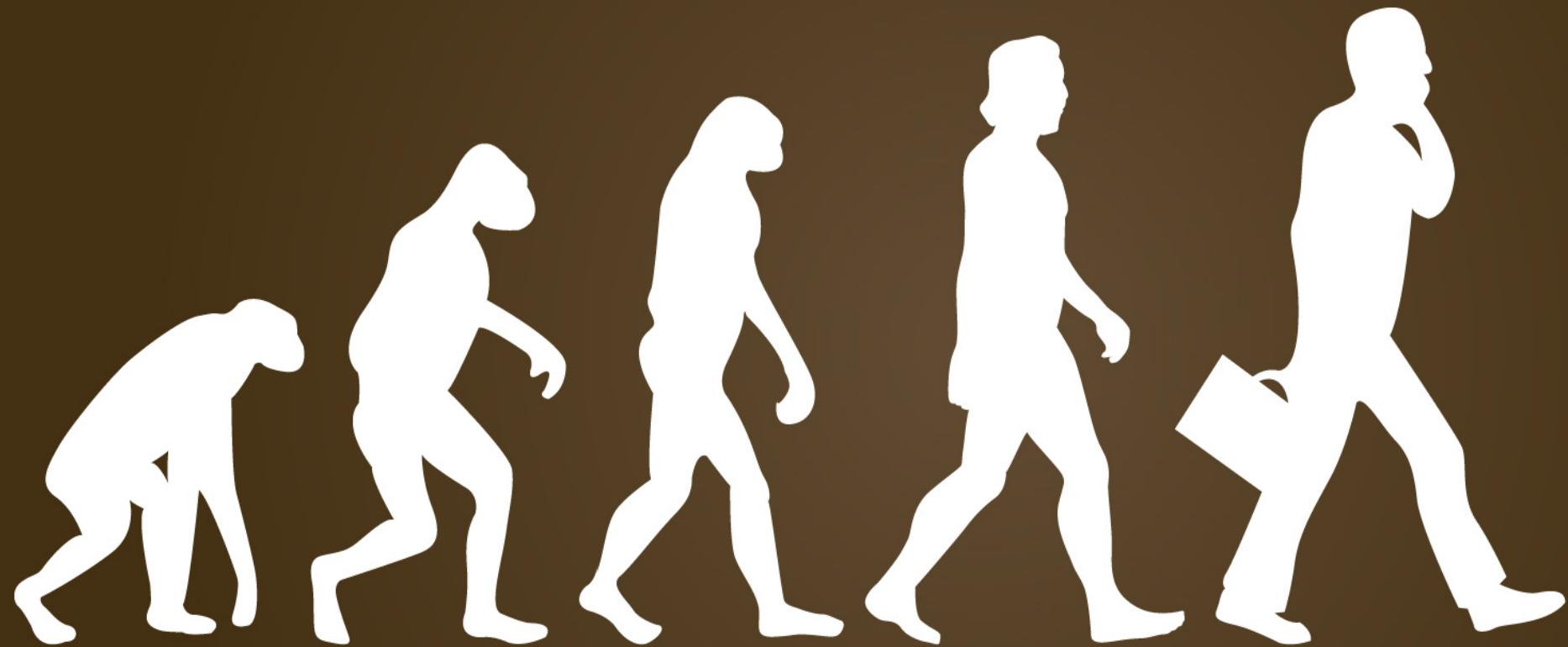


Mining motivated trends of usage of Haskell libraries

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

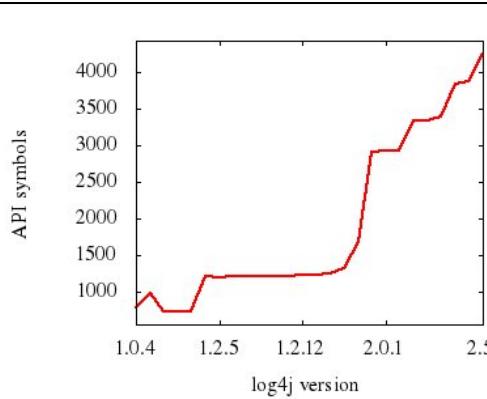




A breaking change is a change to one part of a software system that potentially causes other components to fail

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

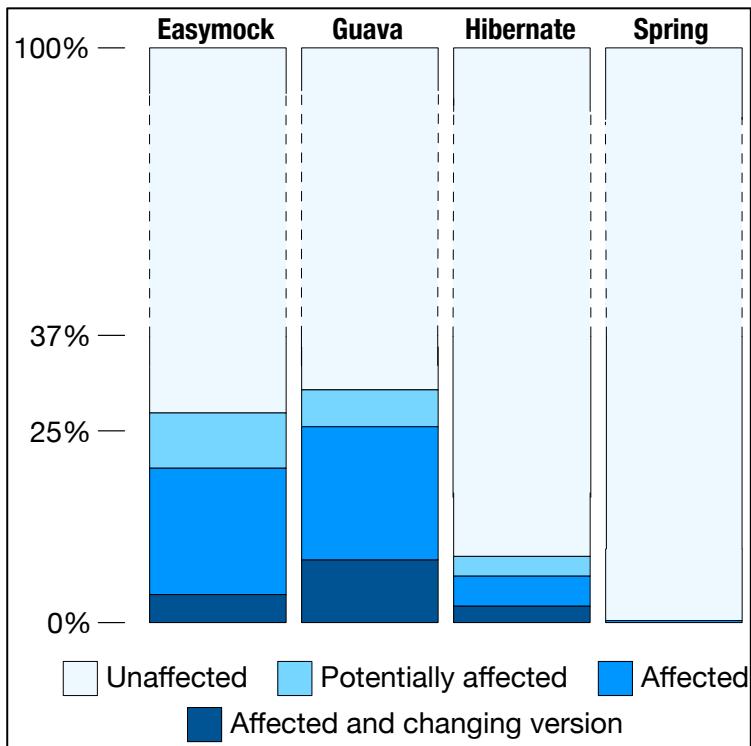
API client's actual behavior

```
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computer software feature, characteristic, or practice to indicate that it should be avoided (often because it is being superseded).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

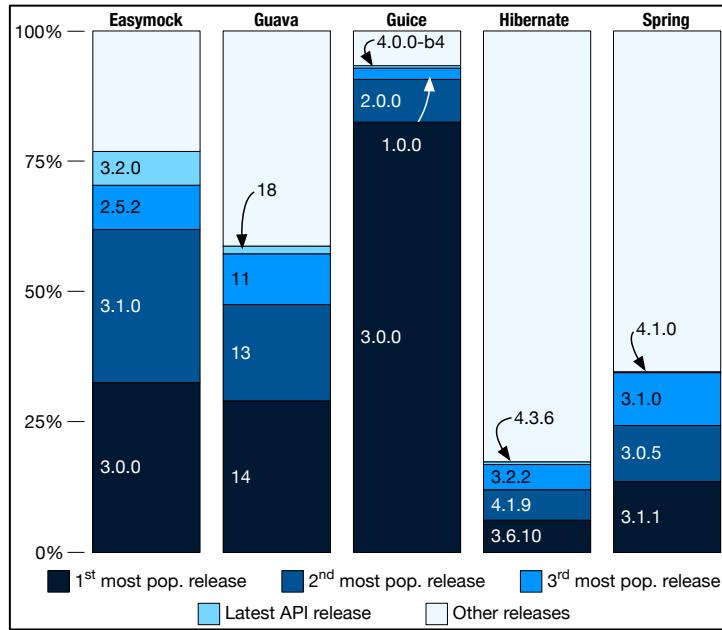
API client's that react

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

ISSUE

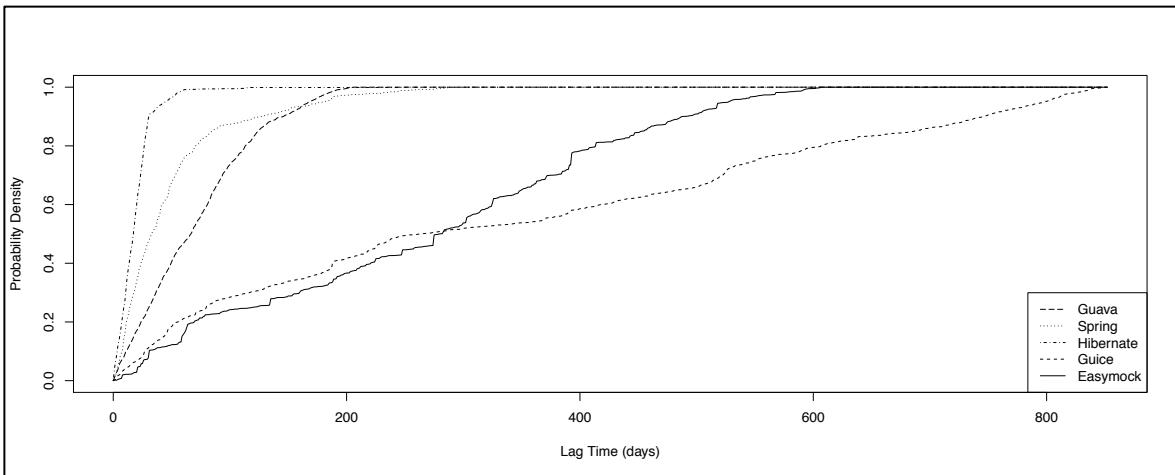
Nobody changes versions of a library!

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!



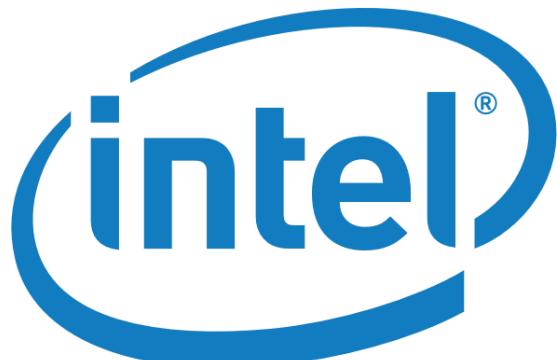
Purely functional language

Statically typed



Type inference system

Usage in industry



Hackage

Hackage :: [Package]

| Home | Search | Browse | What's new | Upload | User accounts

Welcome to Hackage!

Hackage is the Haskell community's central package archive of open source software. Package authors use it to publish their libraries and programs while other Haskell programmers use tools like `cabal-install` to download and install packages (or people get the packages via their distro).

This web interface to Hackage lets you:

- [Browse](#) the packages (sorted by category)
- [Search](#) for packages by keyword (in the name or description)
- See what packages have been [uploaded recently](#)
- [Upload](#) your own packages to Hackage (note that you'll need an [account](#))

Each package includes:

- A description of what it does
- Licence information
- Author information
- A downloadable gzipped tarball
- A list of modules in the package
- Haddock documentation (if available) with source links

Guidelines for Hackage Packages:

- All packages should follow the [Package Versioning Policy \(PVP\)](#).
- Packages cannot be deleted, so you should consider uploading new versions packages as a [package candidate](#) and testing before publishing it to the main index.

In addition to the main package list page, there are a few other package indices:

- [All tags](#)
- [All packages by name, with tags](#)
- [All packages by download](#)
- [All packages with preferred versions](#)
- [All deprecated packages](#)
- [All candidate packages](#)

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
resource	10828
tf-random	10360
cereal	10301
base-compat	10210
integer-logarithms	10172
data-default	10152

Hackage
indexes the
most popular
Haskell libraries

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:
exposed-modules:
build-depends:
default-language: Haskell2010

src
Poker.Data, Poker.Analysis
base >= 4.7 && < 5,
containers
Haskell2010

executable poker-exe
hs-source-dirs:
main-is: Main.hs
ghc-options: -threaded -rtsopts -with-rtsopts=-N
build-depends:
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:
exposed-modules:          Poker.Data, Poker.Analysis
build-depends:
src
base >= 4.7 && < 5,
containers
default-language:          Haskell2010

executable poker-exe
hs-source-dirs:
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

ISSUE

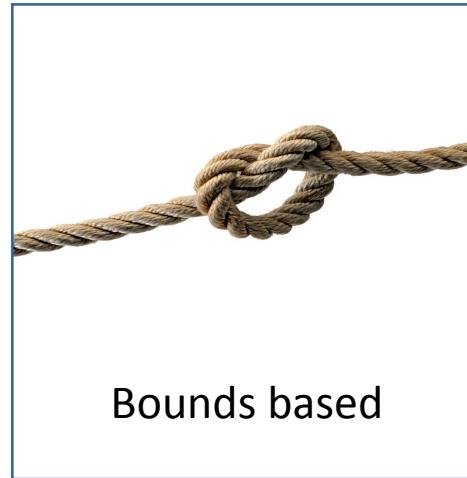
Accurate version resolution is a challenge!

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

Accurate version resolution



Date based

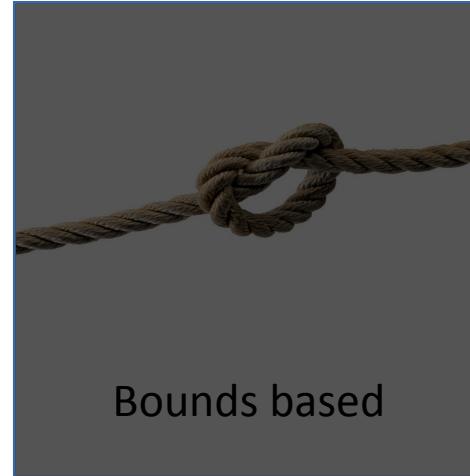


Bounds based

Accurate version resolution



Date based



Bounds based

Date based resolution

Version 2
Jan 2006

Version 3
Mar 2006

Version 4
Jun 2006

Version 5
Jan 2007

foo > 2
Apr 2006

Date based resolution

Version 2
Jan 2006

Version 3
Mar 2006

Version 4
Jun 2006

Version 5
Jan 2007

foo > 2
Apr 2006

Date based resolution

Version 2
Jan 2006

Version 3
Mar 2006

Version 4
Jun 2006

Version 5
Jan 2007

foo > 2 && <=5
Apr 2008

Date based resolution

Version 2
Jan 2006

Version 3
Mar 2006

Version 4
Jun 2006

Version 5
Jan 2007

foo > 2 && <=5
Apr 2008

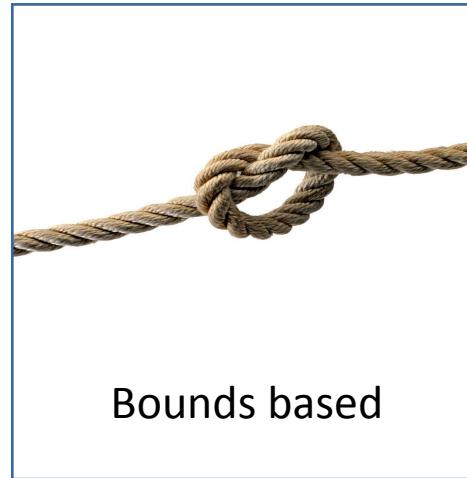


Results in more version upgrades
Upgrade decision unknown

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



Corrective



Adaptive



Deletion of lines



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

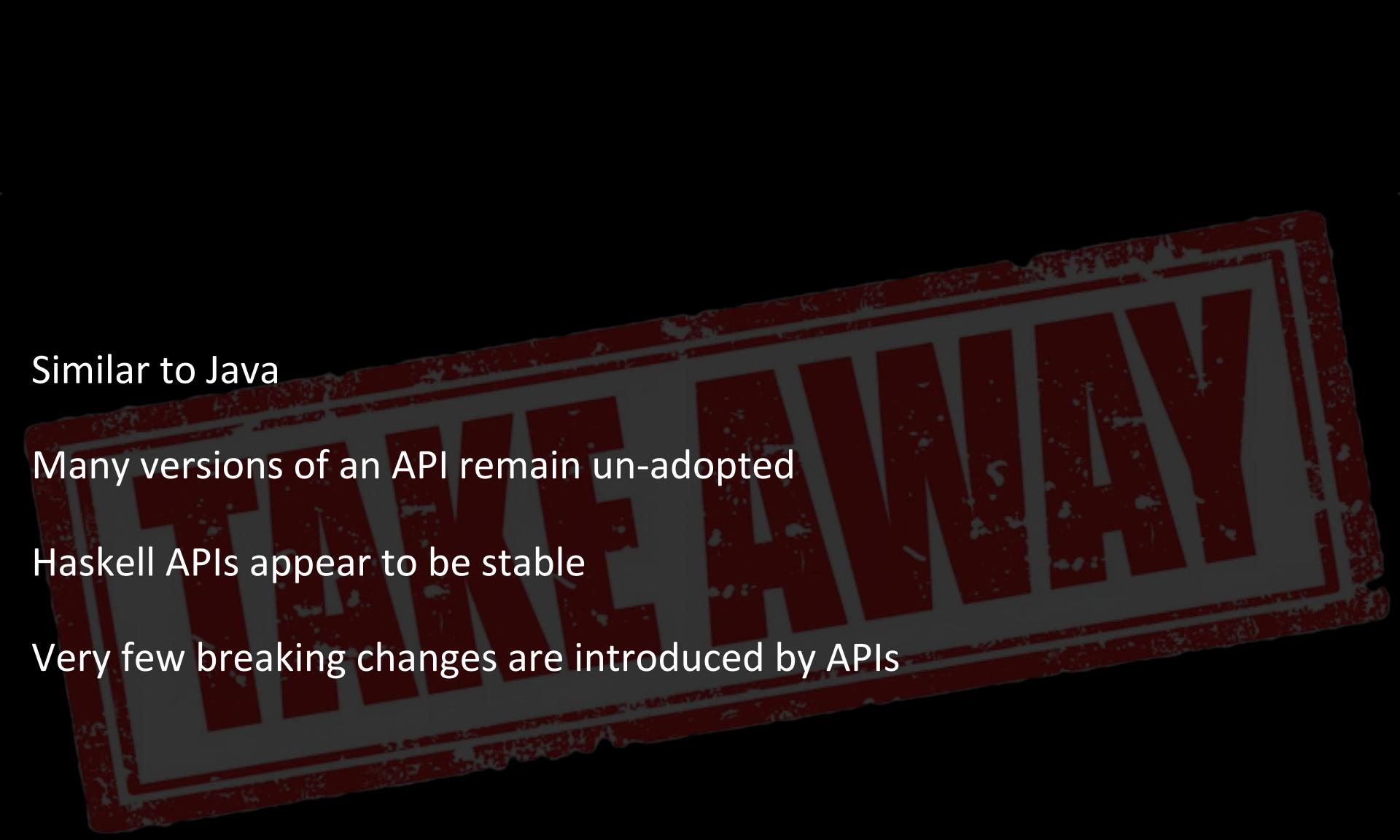
What does it this mean?

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



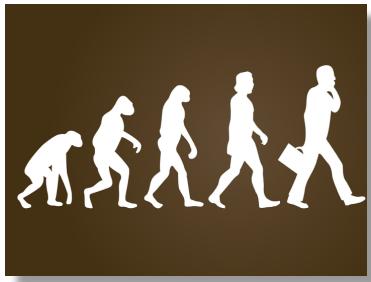
Similar to Java

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

Version	Date	Backend Changes	Added Methods	Removed Methods
2.3	2013-01-07	31,121	31,100	190
2.4.1	2013-09-09	31,121	4	6 removed
2.4	2013-09-26	31,121	11,125	50 removed
2.3	2013-05-15	31,121	31,100	9 removed
2.2	2013-02-20	31,121	31,121	17 removed
2.1	2013-02-20	31,121	31,121	12 removed
2.0.2	2013-02-20 since 2.0.1	31,121	29,121	180 removed
2.0.1	2013-02-20	31,121	31,121	6 removed
2.0	2013-01-17	31,121	31,121	6 removed
2.0.1	2014-07-13 since 2.0.0	31,121	31,121	6 removed

Breaking changes are common!

A line graph showing the number of breaking changes over time. The x-axis represents time in days, and the y-axis represents the number of changes. The graph shows a sharp increase in the number of changes starting around day 100, with a peak of approximately 300 changes per day around day 150.

Usage of API versions in Java

A stacked bar chart showing the usage of different Java API versions. The x-axis lists API versions: 1.8, 1.7, 1.6, 1.5, 1.4, 1.3, 1.2, 1.1, and 1.0. The y-axis represents the percentage of usage. The chart shows that the vast majority of usage is concentrated in the latest version (1.8), with smaller portions in 1.7 and 1.6.

Very few clients upgrade to the latest version



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