

I stand on the cliffs
with my son next to me
This island
Our prison
Our home



I stand on the cliffs
with my **son** next to me
This **island**
Our **prison**
Our **home**

Icarus

Overconfidence

Recklessness

"Don't fly too
close to the sun"





Icarus

Overconfidence

Recklessness

"Don't fly too close to the sun"



Daedalus

Master craftsman

Inventions often have
unintended consequences

I stand on the cliffs
with my son next to me

This island
Our prison
Our home

The unintended consequences of analyzing continuous software data

Overconfidence

Shane
McIntosh
Asst. Prof.
Lead Rebel



Recklessness



I'm an academic now, but I wasn't always so respectable



Rockstar
'03-'08



MSc
'09-'10



SW Eng.
'10-'12



PhD
'12-'15



Asst. Prof.
'15-now





Repository Excavation



@SoftwareREBELs



rebels.ece.mcgill.ca

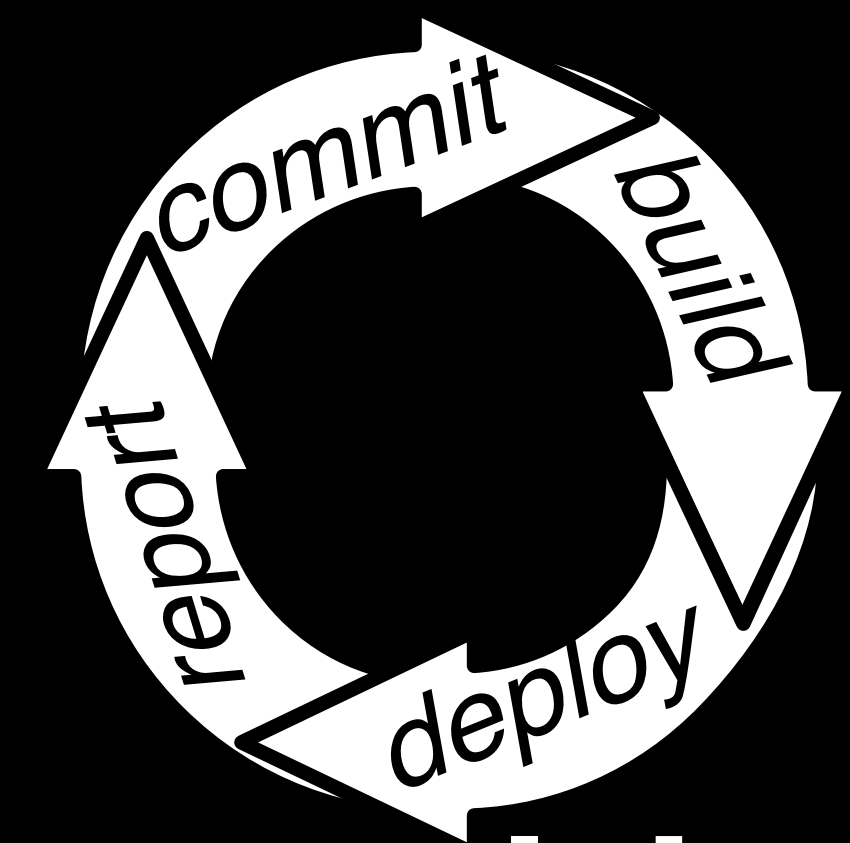
SOFTWARE REBELS



Repository Excavation

What would
your **software**
repositories
say if they
could talk?

How can we
support and
leverage the
pipeline to
improve dev
and delivery?



Build Engineering



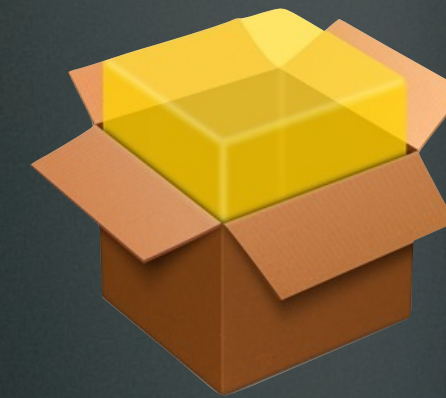
Release pipelines



1. Integrate



2. Build



3. Deploy



4. Monitor

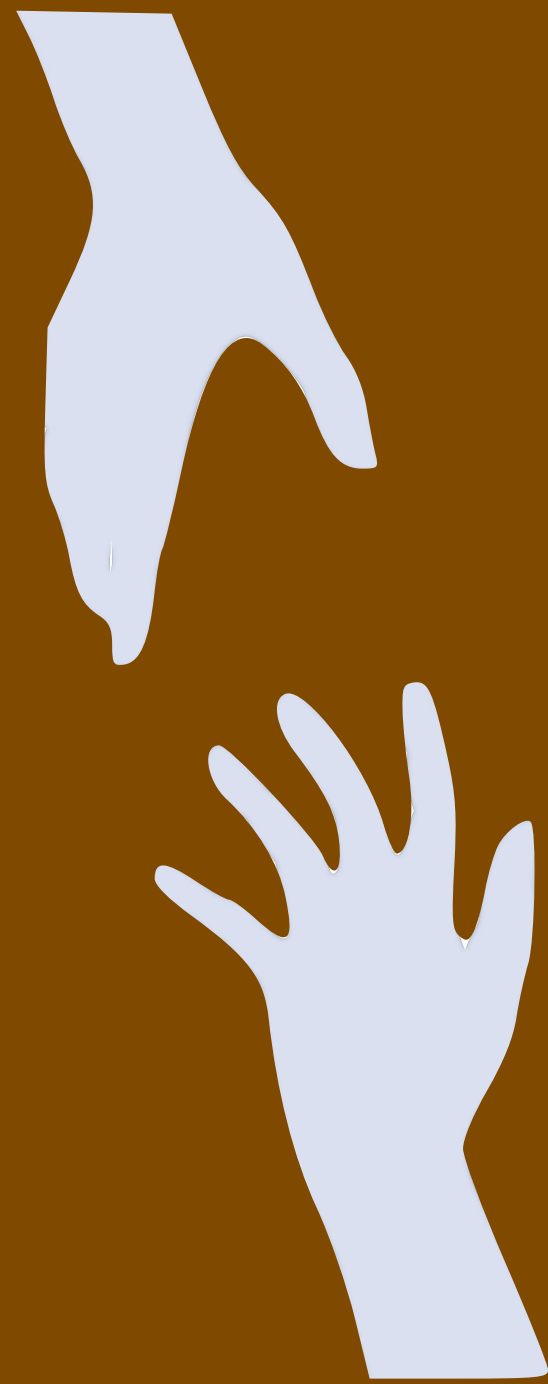


I've got a **plan**
And some **wax** and some **string**
Some **feathers** I stole from the **birds**

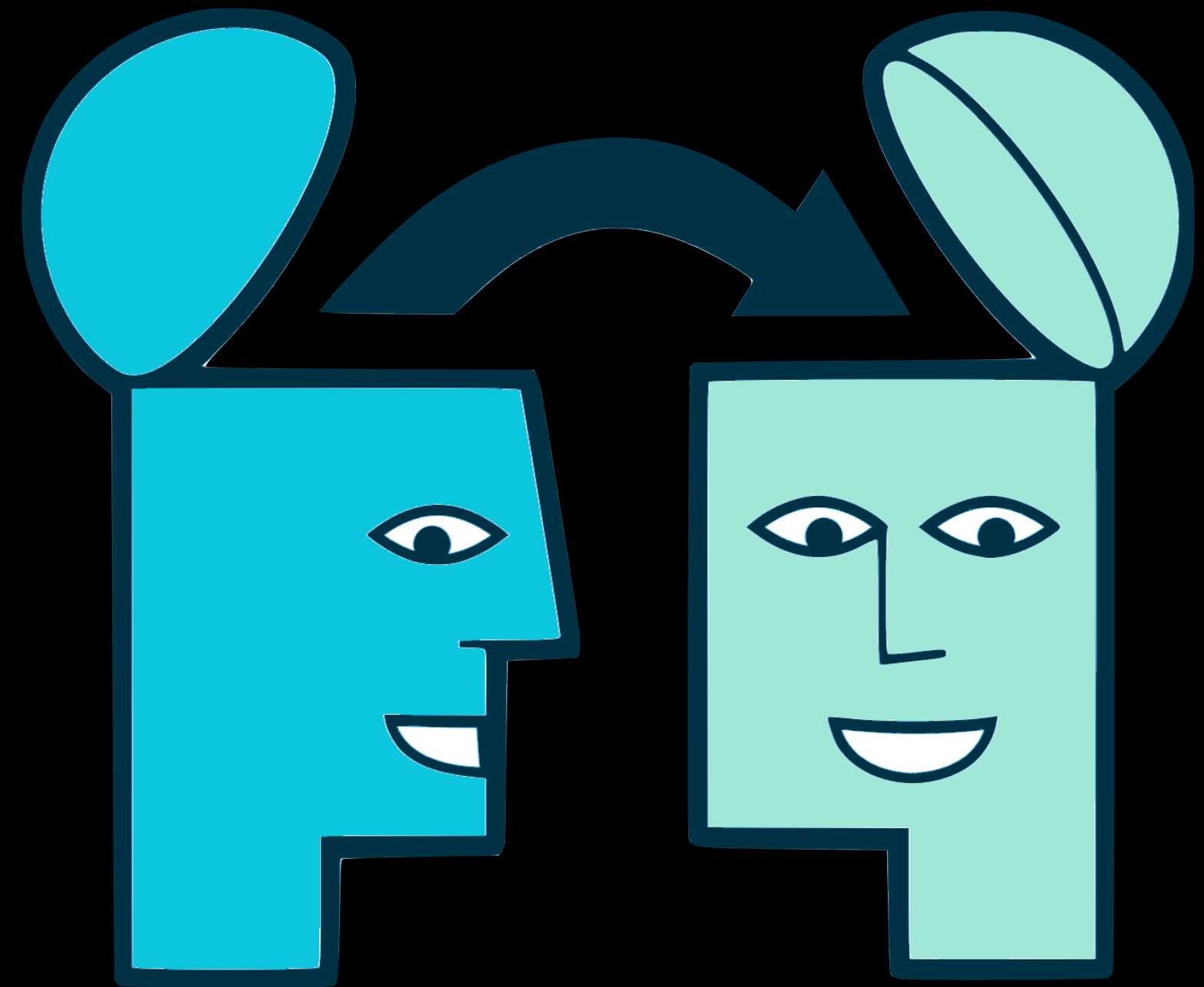


Artist: Markus Stadlober

Supporting



Leveraging




```
package rebels;
```

```
import org.apache.logging.log4j.LogManager;
```

```
import org.apache.logging.log4j.Logger;
```

```
public class App {
```

```
    public static void main (String[] args) {
```

```
        final Logger logger = LogManager.getLogger("HelloWorld");
```

```
        logger.info("Hello World!");
```

```
    }
```

```
}
```

```
<project ...>
```

```
    <groupId>rebels</groupId>
```

```
    <artifactId>helloworld</artifactId>
```

```
    <version>1.0-SNAPSHOT</version>
```

```
    <packaging>jar</packaging>
```

```
</project>
```



```
$ mvn compile
```

```
[INFO] -----  
[INFO] BUILD FAILURE  
[INFO] -----
```




```
$ mvn compile
```

```
[INFO] -----
```

```
[INFO] BUILD FAILURE
```

```
[INFO] -----
```

```
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:3.1:compile  
(default-compile) on project helloworld: Compilation failure: Compilation failure:
```


English m%@\$#!
Do you *speak* it?




```
$ mvn compile
```

```
[INFO] -----  
[INFO] BUILD FAILURE  
[INFO] -----
```

```
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:3.1:compile  
(default-compile) on project helloworld: Compilation failure: Compilation failure:
```

```
[ERROR] /Users/shanemcintosh/src/foo/helloworld/src/main/java/rebels/App.java:[3,32] package  
org.apache.logging.log4j does not exist
```

```
[ERROR] /Users/shanemcintosh/src/foo/helloworld/src/main/java/rebels/App.java:[4,32] package  
org.apache.logging.log4j does not exist
```

```
[ERROR] /Users/shanemcintosh/src/foo/helloworld/src/main/java/rebels/App.java:[9,11] cannot find  
symbol
```

```
[ERROR] symbol:    class Logger
```

```
[ERROR] location: class rebels.App
```

```
[ERROR] /Users/shanemcintosh/src/foo/helloworld/src/main/java/rebels/App.java:[9,27] cannot find  
symbol
```

```
[ERROR] symbol:    variable LogManager
```

```
[ERROR] location: class rebels.App
```

```
[ERROR] -> [Help 1]
```



```
$ mvn compile
```

```
[INFO] -----  
[INFO] BUILD FAILURE  
[INFO] -----
```

```
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:3.1:compile  
(default-compile) on project helloworld: Compilation failure: Compilation failure:
```

```
[ERROR] /Users/shanemcintosh/src/foo/helloworld/src/main/java/rebels/App.java:[3,32] package  
org.apache.logging.log4j does not exist
```

```
[ERROR] /Users/shanemcintosh/src/foo/helloworld/src/main/java/rebels/App.java:[4,32] package  
org.apache.logging.log4j does not exist
```

```
[ERROR] /Users/shanemcintosh/src/foo/helloworld/src/main/java/rebels/App.java:[9,11] cannot find  
symbol
```

```
[ERROR] symbol: class Logger
```

```
[ERROR] location: class rebels.App
```

```
[ERROR] /Users/shanemcintosh/src/foo/helloworld/src/main/java/rebels/App.java:[9,27] cannot find  
symbol
```

```
[ERROR] symbol: variable LogManager
```

```
[ERROR] location: class rebels.App
```

```
[ERROR] -> [Help 1]
```



```
package rebels;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class App {
    public static void main (String[] args) {
        final Logger logger = LogManager.getLogger("HelloWorld");
        logger.info("Hello World!");
    }
}

<project ...>
    <groupId>rebels</groupId>
    <artifactId>helloworld</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
</project>
```



```
package rebels;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class App {
    public static void main (String[] args) {
        final Logger logger = LogManager.getLogger("HelloWorld");
        logger.info("Hello World!");
    }
}

<project ...>
  <groupId>rebels</groupId>
  <artifactId>helloworld</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <dependencies>
    <dependency>
      <groupId>org.apache.logging.log4j</groupId>
      <artifactId>log4j-api</artifactId>
      <version>2.10.0</version>
    </dependency>
  </dependencies>
</project>
```



```
$ mvn compile
```

```
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----
```

Why can't
Maven fix this
automatically?



We **leap** from the cliff
And we **hear** the **wind sing**
A **song** that's **too perfect** for words

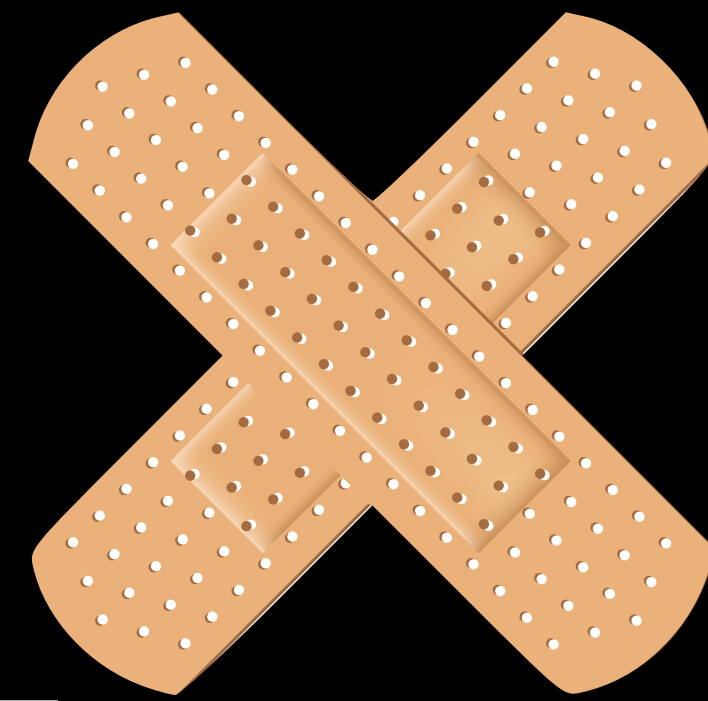


Artist: Charles Paul Landon

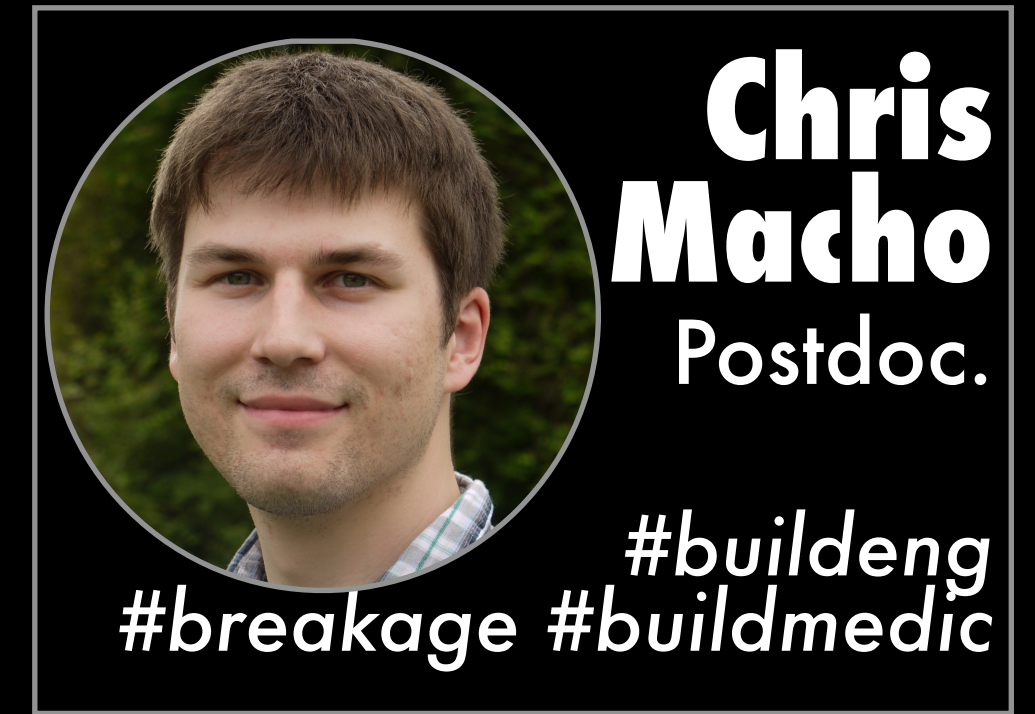


@SoftwareREBELs





Build Medic



```
[INFO] -----  
[INFO] BUILD FAILURE  
[INFO] -----  
...
```

Detect Build Failure Type

Apply Repair Strategy

Evaluate Revision

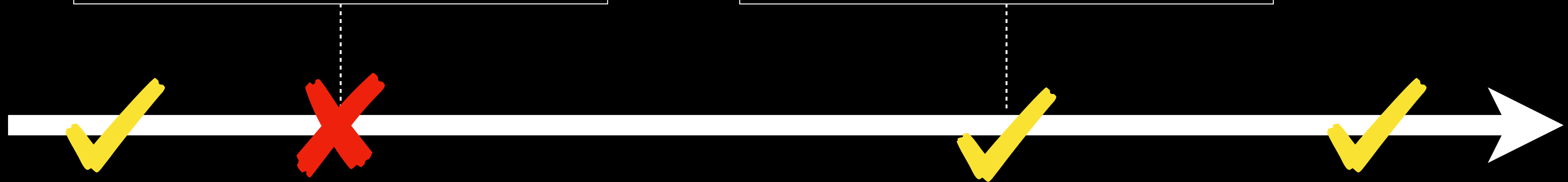
```
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
...
```

**Automatically Repairing
Dependency-Related Build
Breakage**
C. Macho, S. McIntosh, M. Pinzger
[SANER 2018]

Studying how developers repair build breakage

```
[INFO] -----  
[INFO] BUILD FAILURE  
[INFO] -----  
...
```

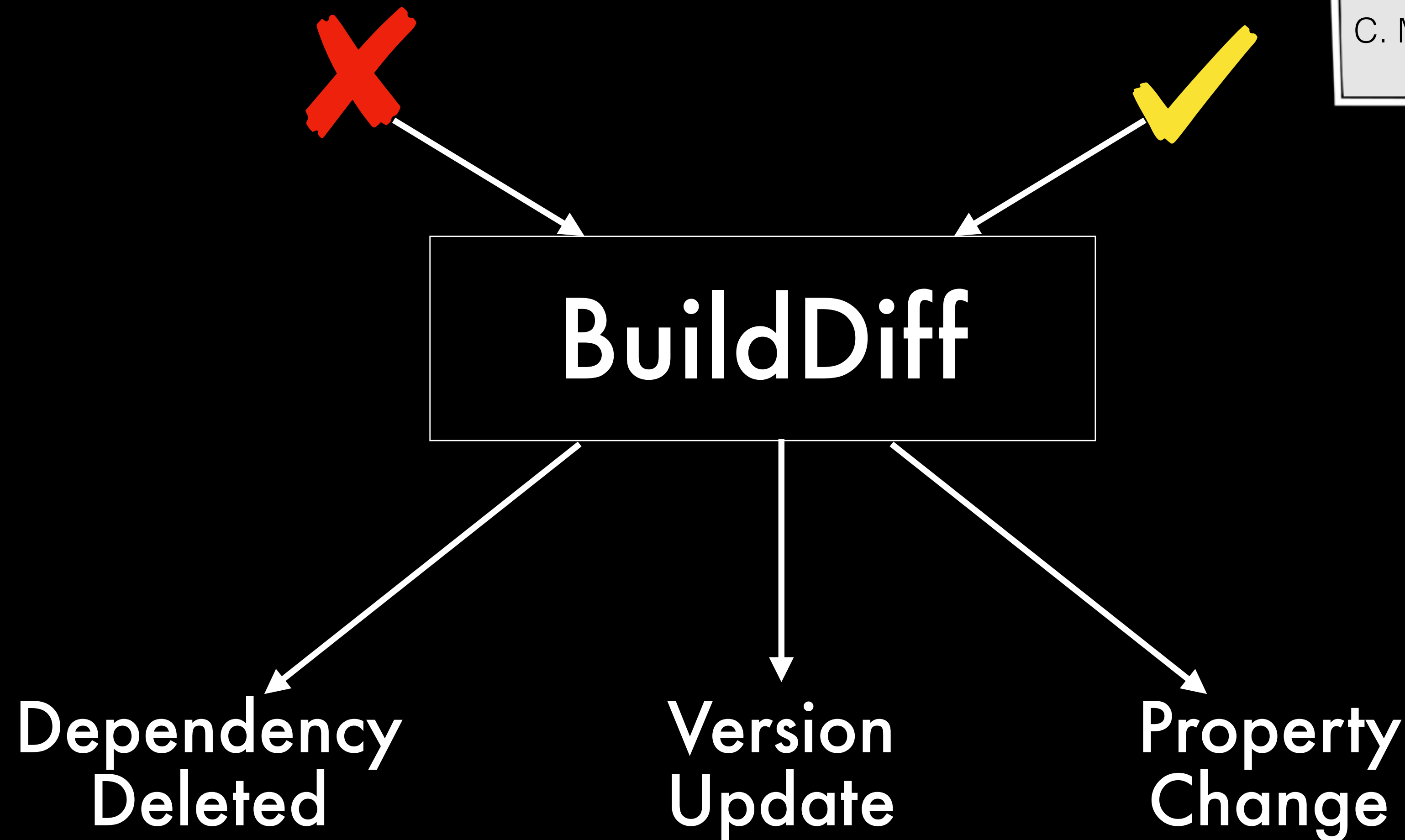
```
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
...
```



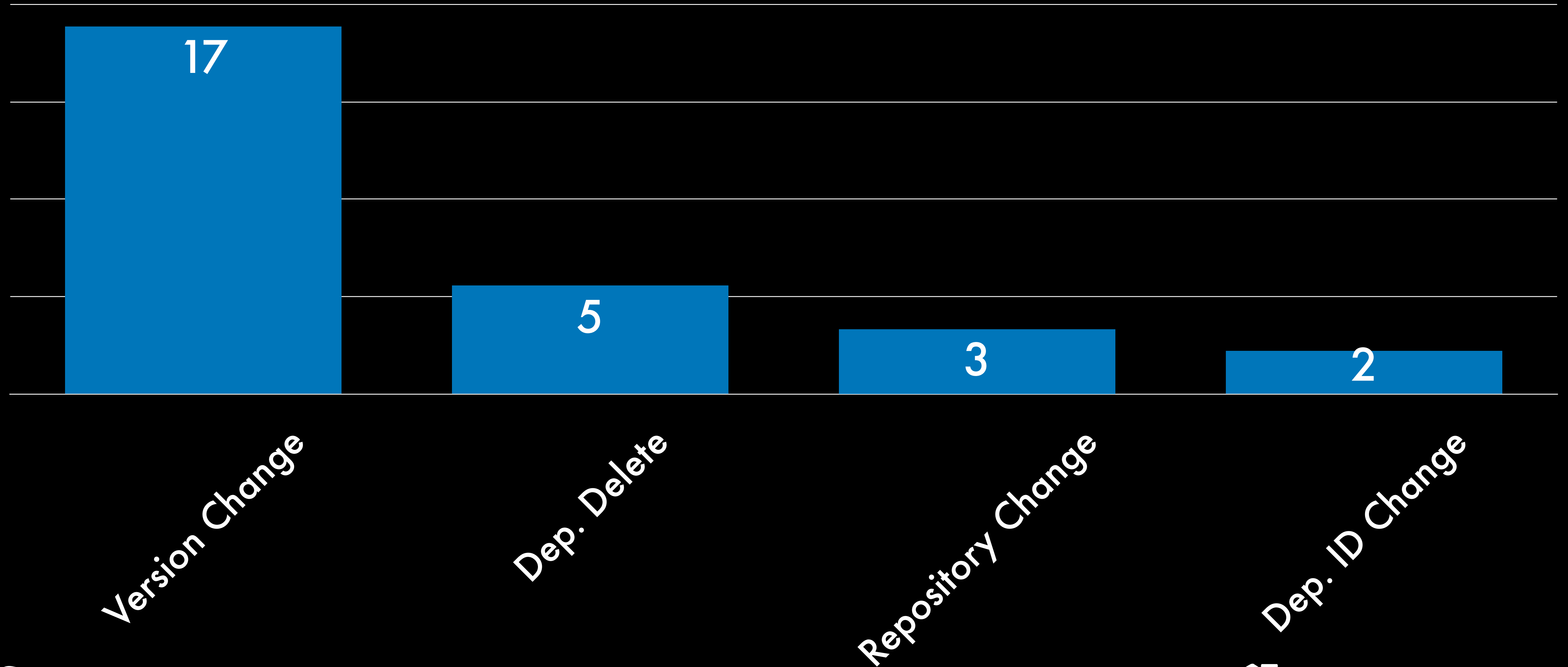
Studying how developers repair build breakage

**Extracting Build Changes
with BuildDiff**

C. Macho, S. McIntosh, M. Pinzger
[SANER 2018]



Repeat and investigate 37 breakage-fix pairs, 27 of which are fixed by one change



We formulated repair strategies for the three most frequent breakage repairs

Version Change

```
+ <dependency>
+   <groupId>org.robvm</groupId>
+   <artifactId>robvm-cocoatouch</artifactId>
+   <version>1.0.0</version>
-   <version>1.0.0-SNAPSHOT</version>
- </dependency>
```

Dependency Delete

```
- <dependency>
-   <groupId>org.robvm</groupId>
-   <artifactId>robvm-cocoatouch</artifactId>
-   <version>1.0.0-SNAPSHOT</version>
- </dependency>
```

Add Repository

```
+ <repository>
+   <id>spring</id>
+   <name>springRepo</name>
+   <url>http://repo.spring.io/libs-milestone/</url>
+ </repository>
```

BuildMedic achieves promising
results on 84 unseen breakage pairs

54%

of repair attempts are
eventually successful

76%

of successful repairs are
identified in one iteration

The 45 BuildMedic repairs are usually identical or very similar to developer repairs

36%

of successful repairs are identical to the corresponding developer repairs

Another

44%

of successful repairs edit the same elements as the corresponding developer repairs

Steer clear of the sun
Or you'll find yourself
In the sea

Key Assumption:
Historical build breakages are
meaningful!

Artist: Amy Adkins



Experimenting with new platforms... Is it breakage?

The screenshot shows a GitHub Actions workflow for the repository 'lesstif / php-jira-rest-client'. The workflow is in a 'passing' state. The build #131 is highlighted with a red box and the text '#131 passed'. Below this, a summary of 'Allowed Failures' is shown, also highlighted with a red box, indicating a failure in step #131.1. The workflow steps are listed on the right, showing a sequence of steps that all passed, each taking 53 seconds or less. The bottom of the screenshot shows the workflow status as 'passed' and the PHP version as 'hhvm'.

lesstif / php-jira-rest-client build passing

Current Branches Build History Pull Requests > Build #131 More options

✓ master changed allow_failure

Commit 968c6f0

Compare 4d4ff42..968c6f0

Branch master

KwangSeob Jeong authored and committed

2 years ago

Allowed Failures ?

✗ # 131.1

✓ # 131.5 PHP: hhvm

no environment variables set	53 sec
no environment variables set	53 sec
no environment variables set	53 sec
no environment variables set	50 sec
no environment variables set	1 min 13 sec

https://github.com/zdavatz/spreadsheet/blame/master/.travis.yml#L24


```
15 - 1.9.2
16 - 1.8.7
17 - rbx-19mode
18 - rbx-18mode
19 - jruby-head
20 - jruby-19mode
21 - jruby-18mode
22 - ree
23 matrix:
24   allow_failures:
25     - rvm: rbx-19mode
26     - rvm: rbx-18mode
27     - rvm: jruby-head
28     - rvm: jruby-19mode
29     - rvm: jruby-18mode
30     - rvm: ree
31
```

5 years ago


6 years ago

This project has had
allow_failures
enabled for its
entire lifetime!

Can we rely
on build data?



Keheliya Gallaba
Ph.D. Student
#buildeng
#robust #efficient #CI #CD



Chris Macho
Postdoc.
#buildeng
#breakage #buildmedic

**Noise and Heterogeneity in Historical Build Data:
An Empirical Study of Travis CI**
K. Gallaba, C. Macho, M. Pinzger, S. McIntosh
[ASE 2018]

To what extent is build data....

Noisy?

Heterogeneous?

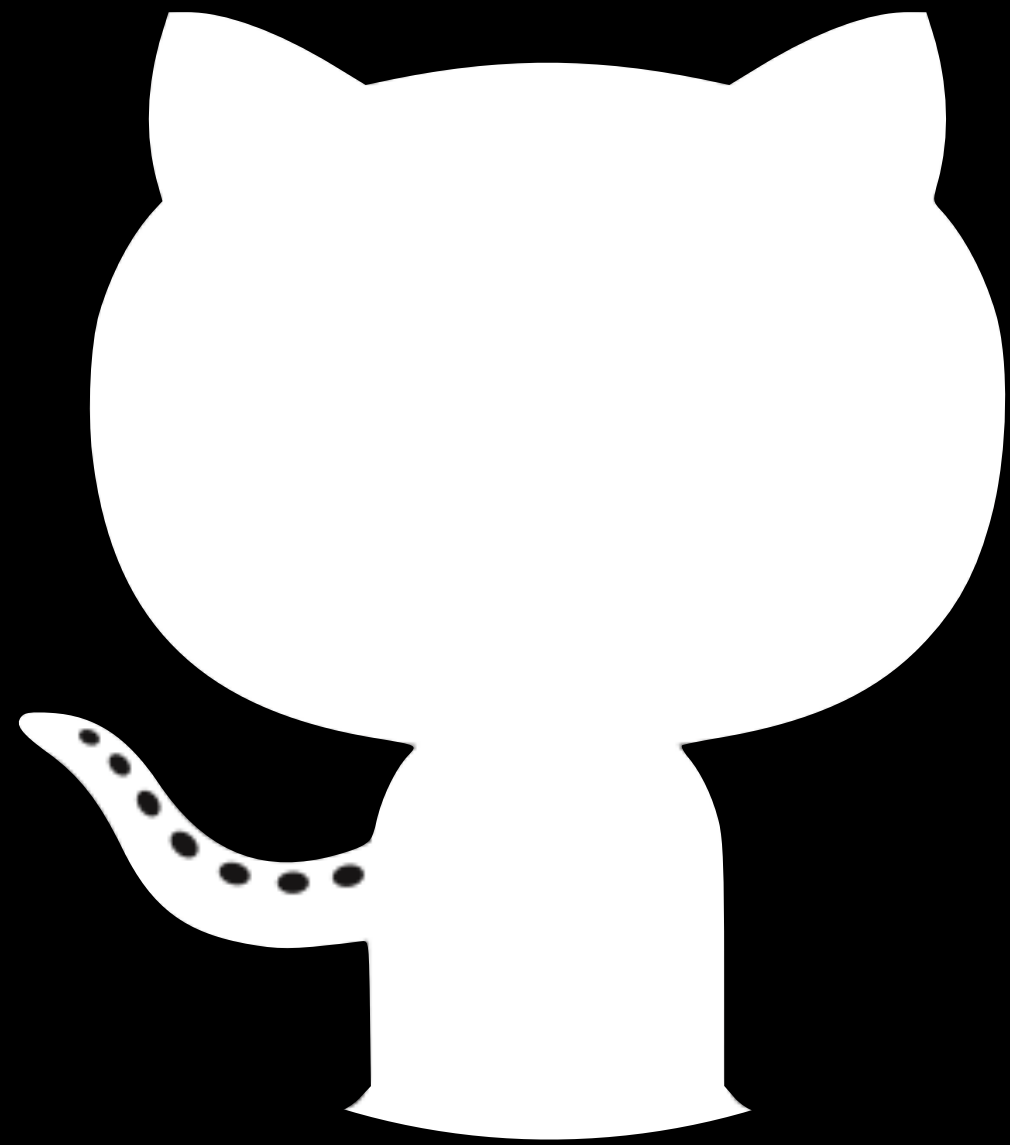


@SoftwareREBELs

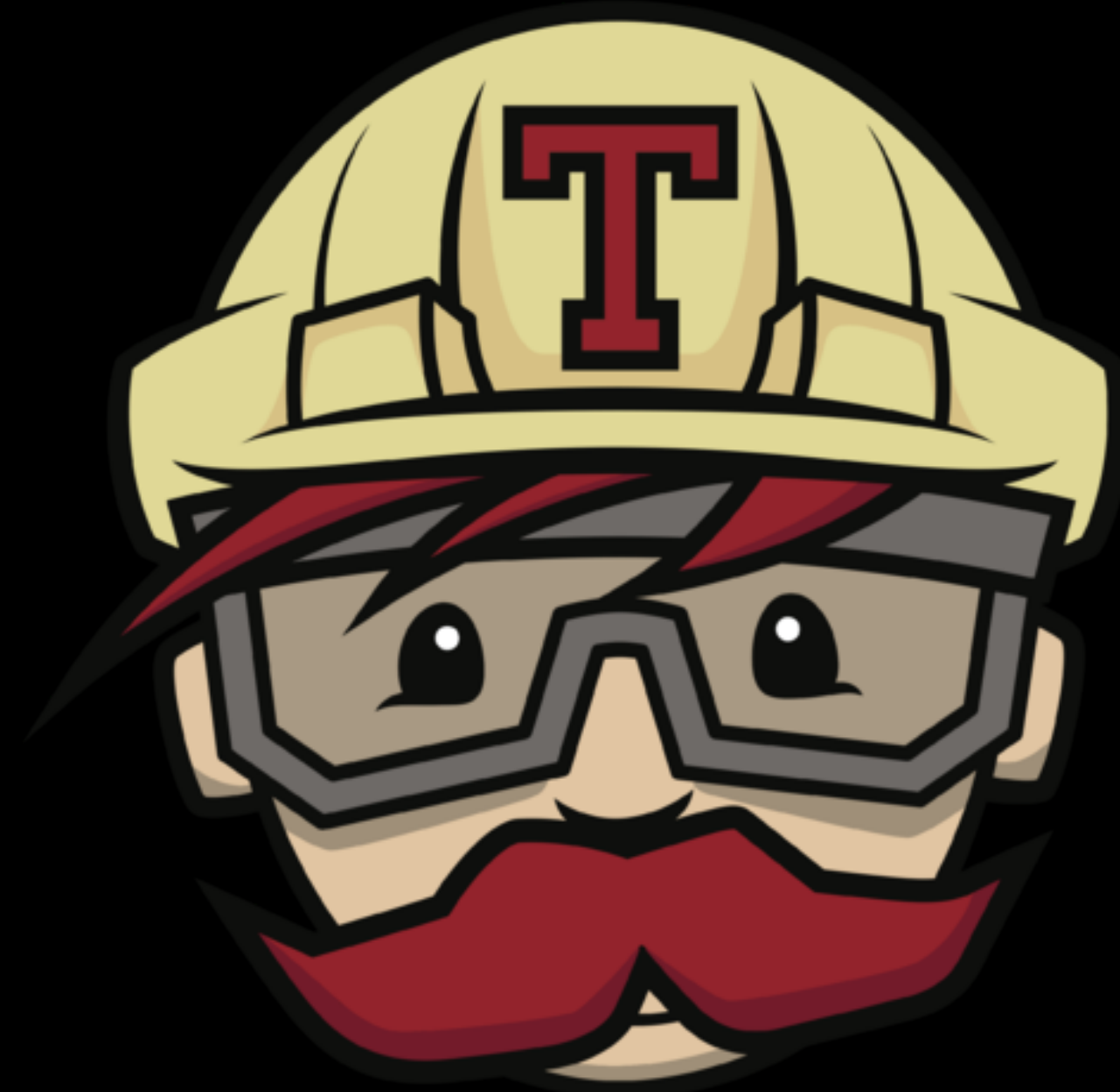


rebels.ece.mcgill.ca

We study open source projects that use the Travis CI service



1,276
projects



680,209
Travis CI builds

To what extent is build data...

Noisy?

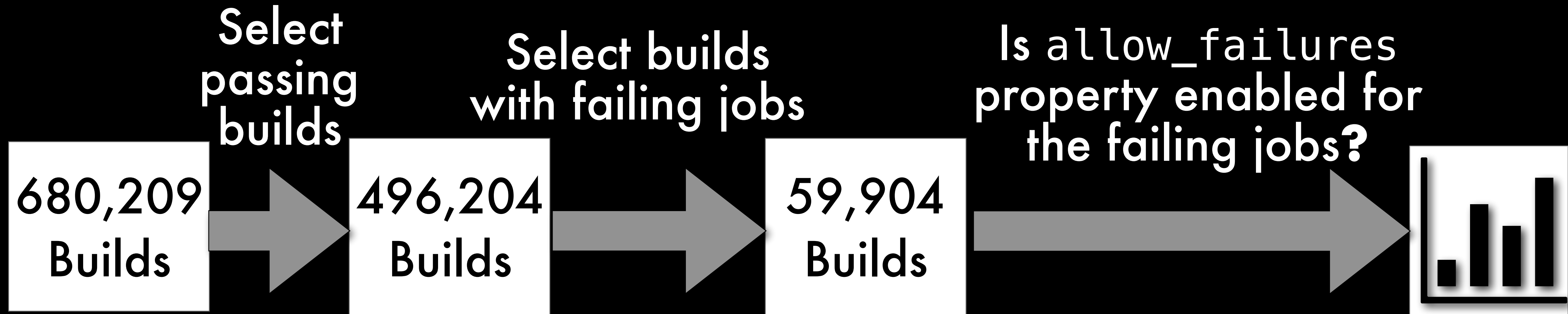


@SoftwareREBELs



rebels.ece.mcgill.ca

We look for passing builds with actively ignored failures



12%

of passing builds have at least one actively ignored failure

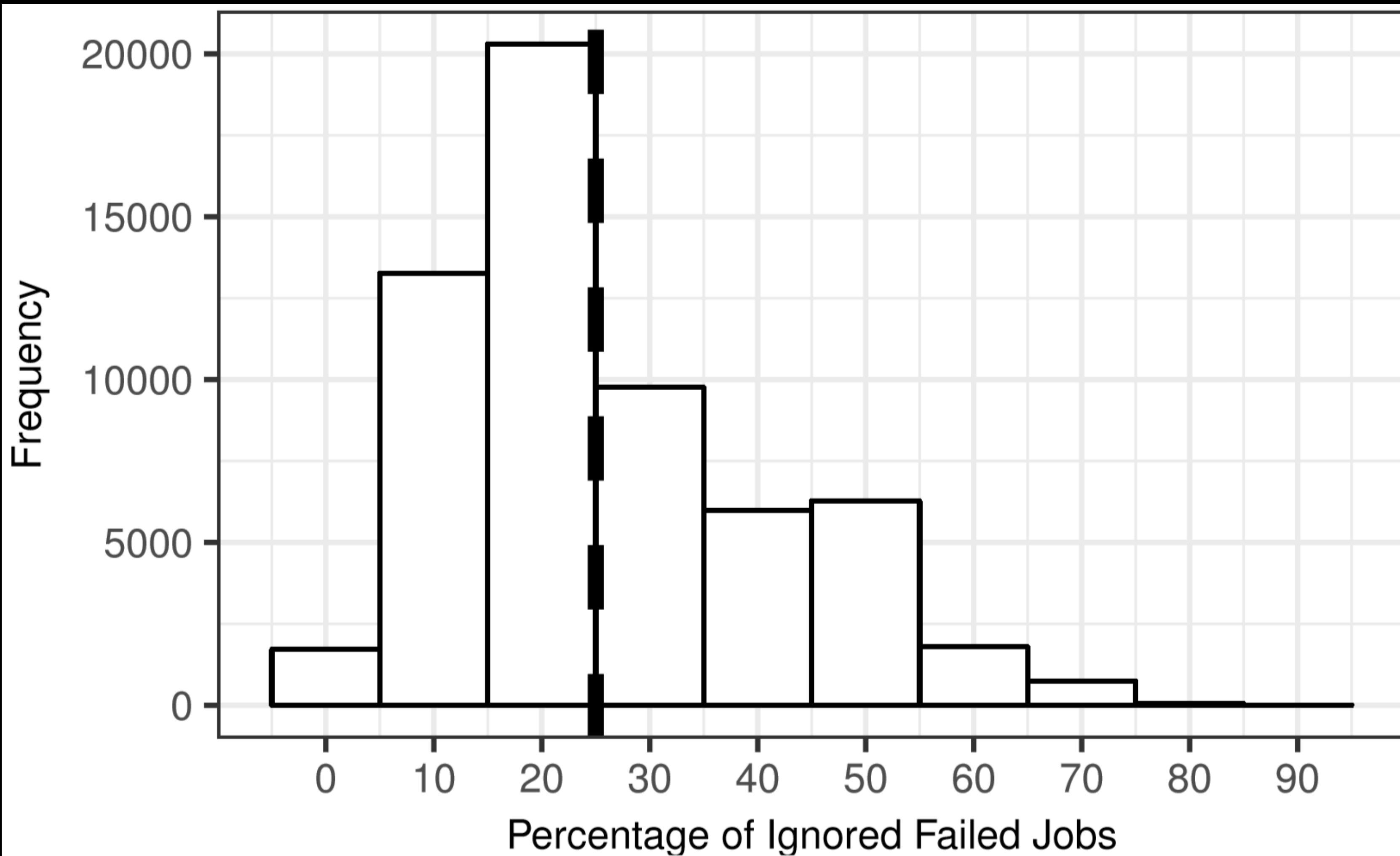
In those passing builds that have actively ignored failures...

A median of

25%
of jobs failed

Up to

87%
of jobs are
actively ignored



Passively ignored breakages may introduce noise when all breakages are assumed to be distracting

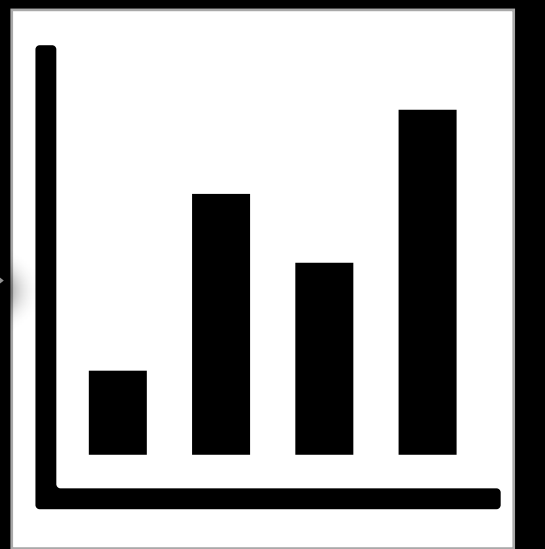
Filter out, e.g.,
scheduled builds

Graph construction
using version
control data

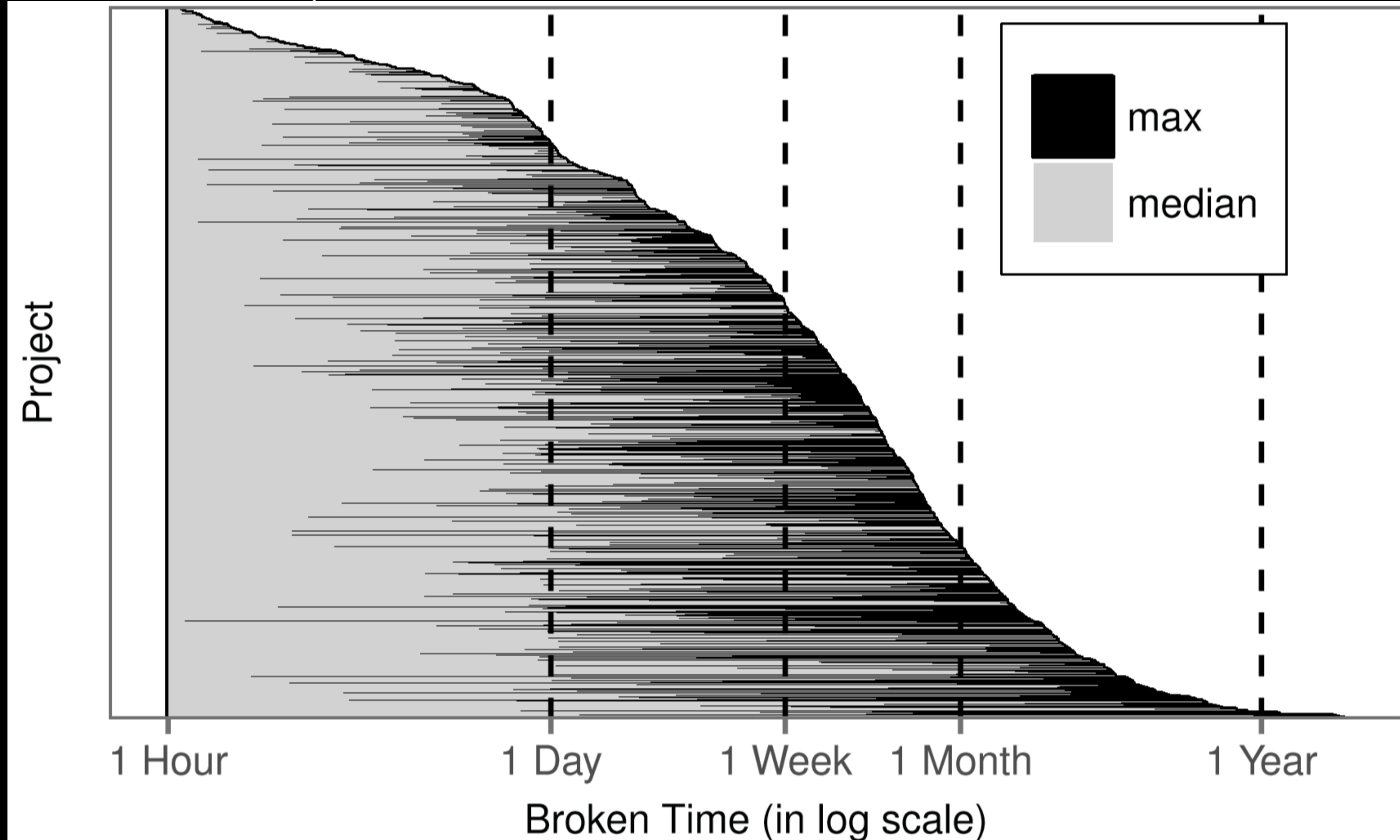
Graph
analysis

680,209
Builds

610,550
Builds



In some cases, builds remain broken for 423 days



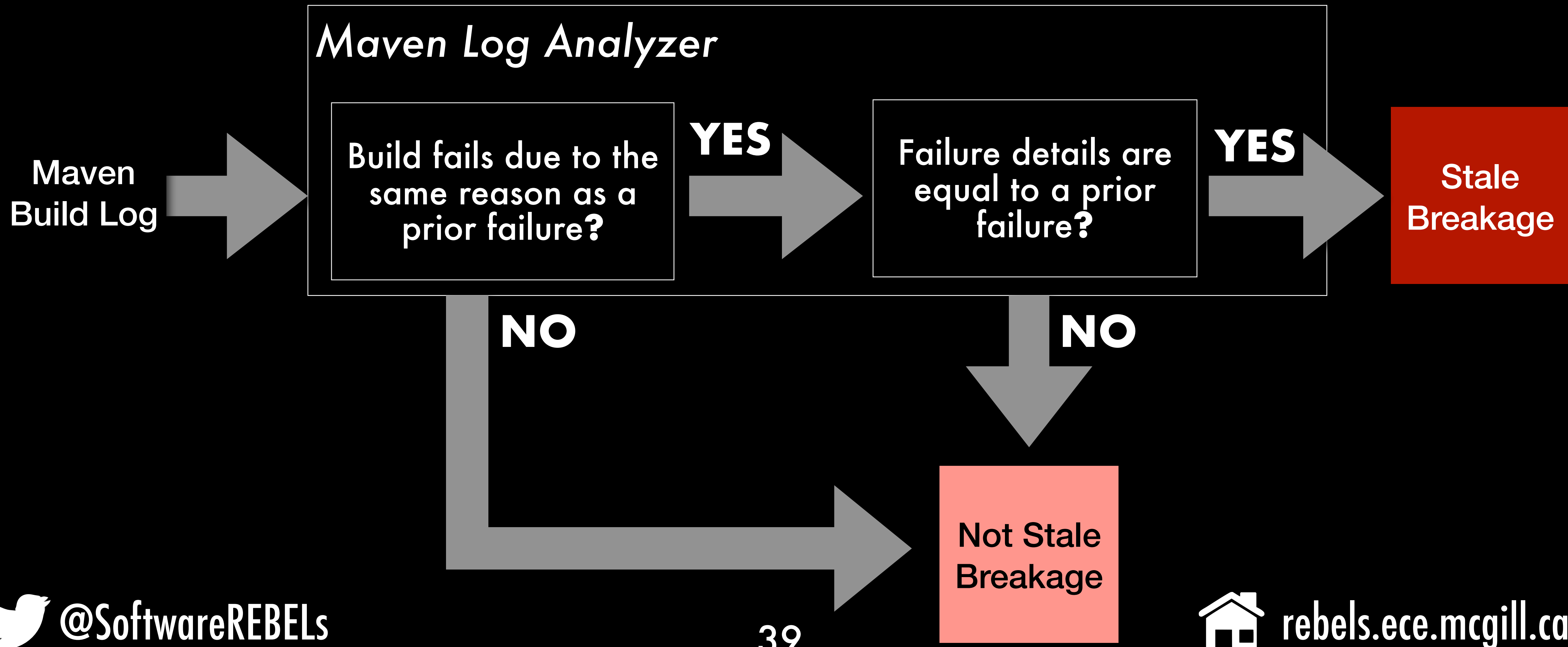
Reasons for ignoring a build breakage: Staleness



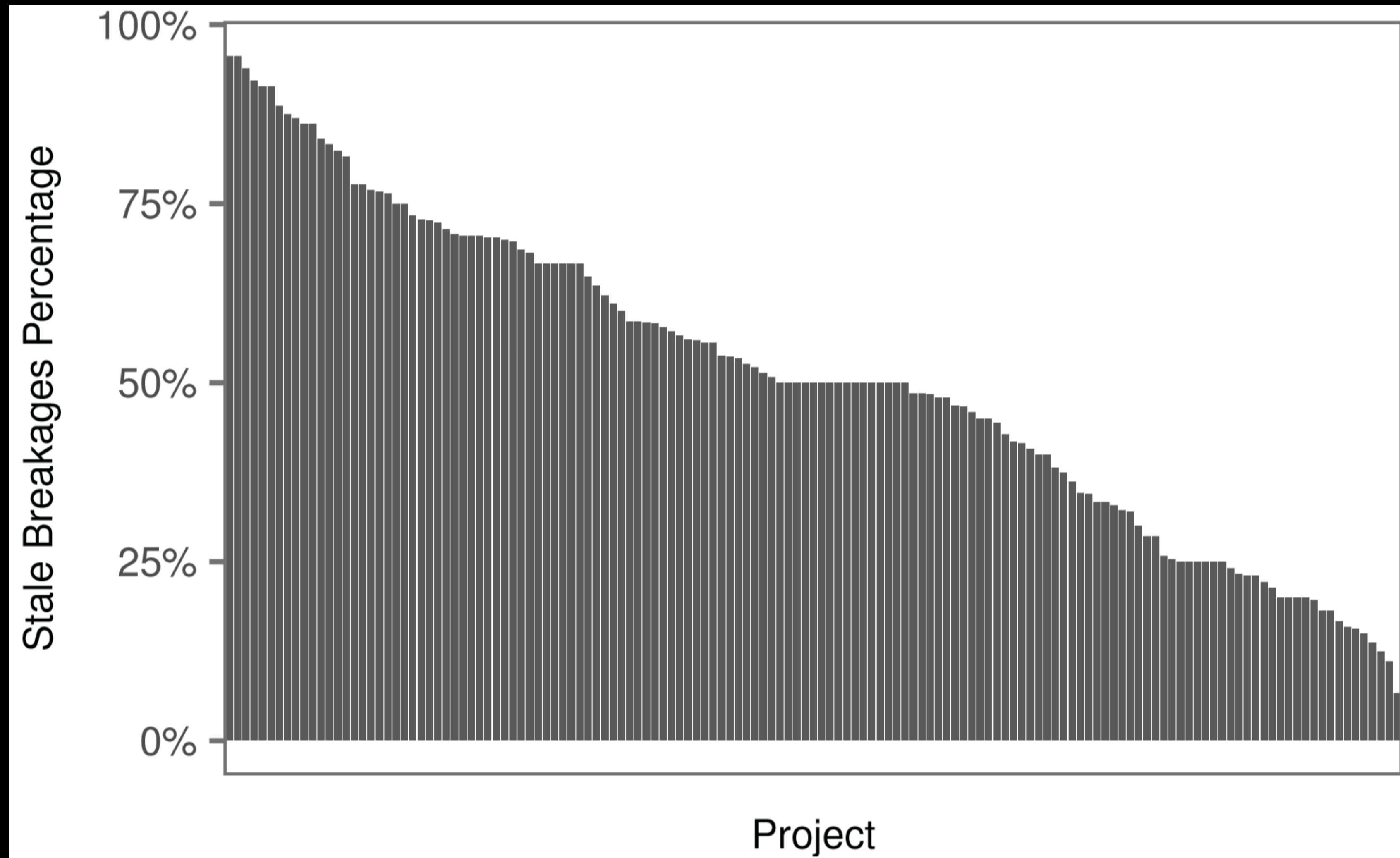
Developers become
desensitized to *stale*
(a.k.a., repeated)
breakages



Measuring staleness in Maven build breakages



Two of every three build breakages are **stale**



Noise may influence analyses based on build outcome data

Passing build outcomes do not always indicate that the build was entirely clean

Build breakages persist for up to 485 commits (423 days)

67% of build breakages we analyze are stale



Noise may influence analyses based on build outcome data

Passing build outcomes do not always indicate that the build was entirely clean

Build breakages persist for up to 485 commits (423 days)

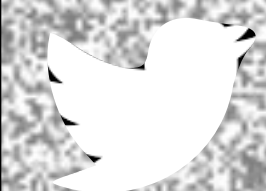
67% of build breakages we analyze are stale

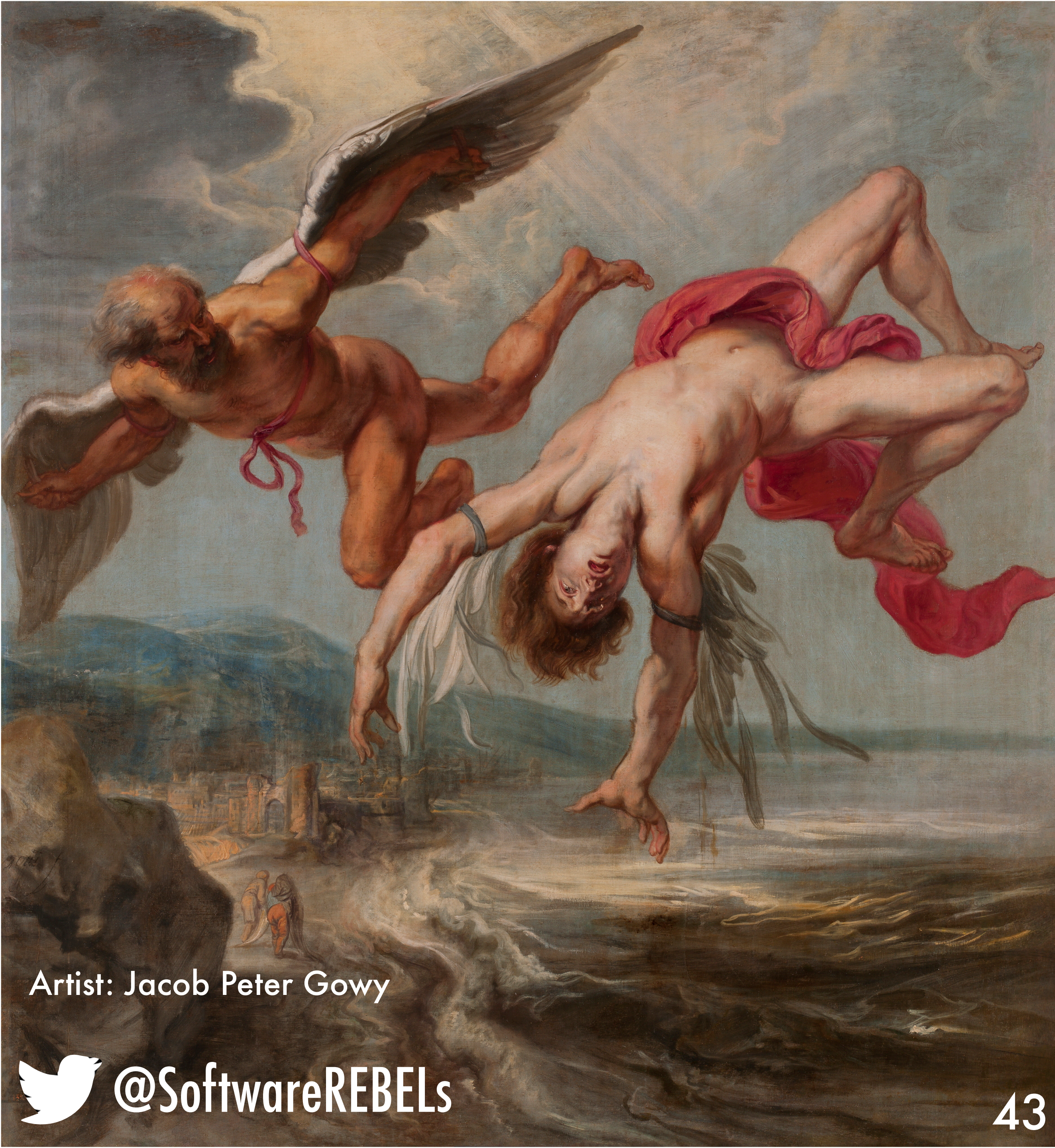
Build outcomes are heterogenous

Environment-specific breakage is common

Breakage due to build tool execution is rare

Future automatic breakage recovery techniques should tackle issues in CI scripts





Won't you **look** at your **wings**
They're coming **undone**
Splitting at the **seams**

Artist: Jacob Peter Gowy



@SoftwareREBELs

43



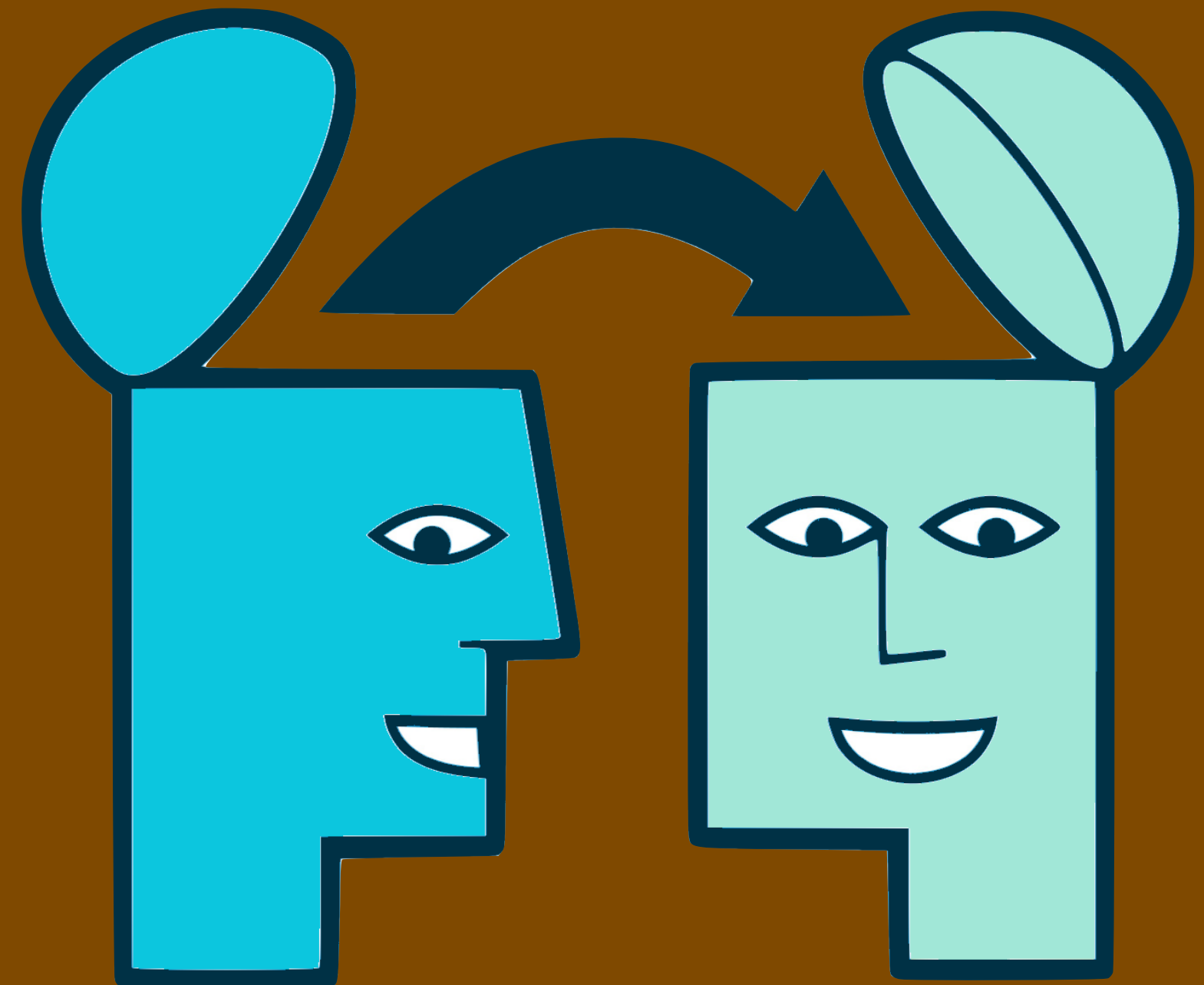
rebels.ece.mcgill.ca

Supporting

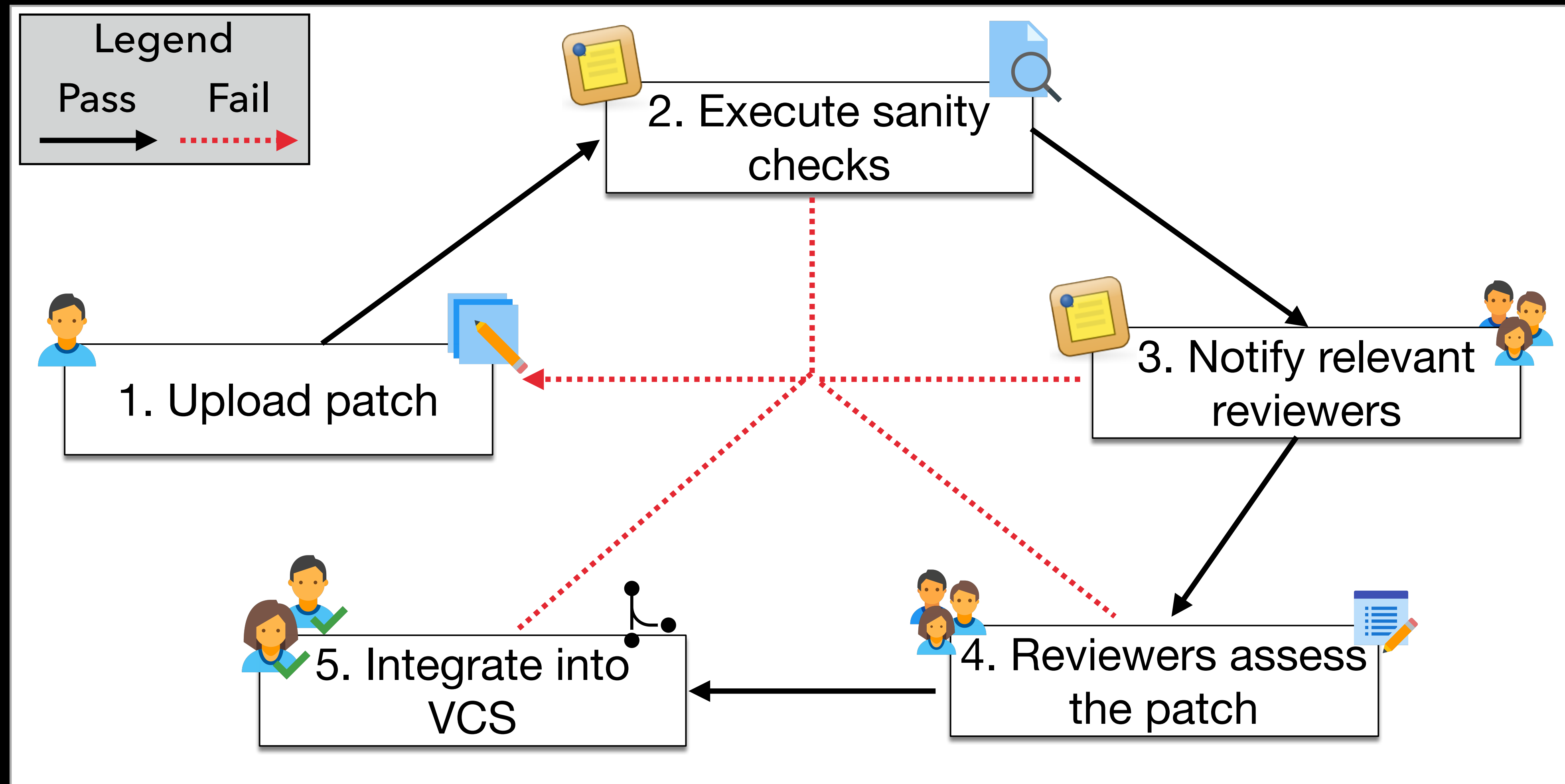


Watch out for noise
and heterogeneity in
build outcome data!

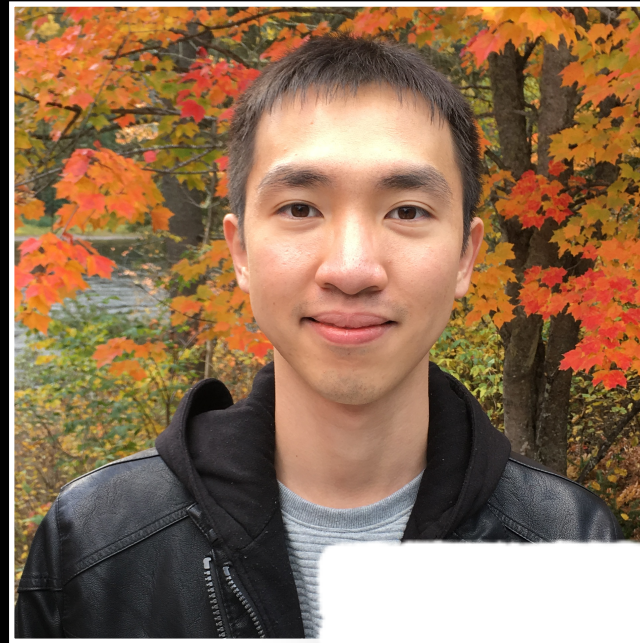
Leveraging



Code review: A Best Practice for SQA



Existence of a code review \neq code quality



sslConnection.c



ReviewBoard

```
sslConnection.c  
sslClientServerHandshake(...)
```

.....

```
+ if (err != 0)  
+     goto fail;  
+     goto fail;    /* MISTAKE! */
```

.....

Ship it!

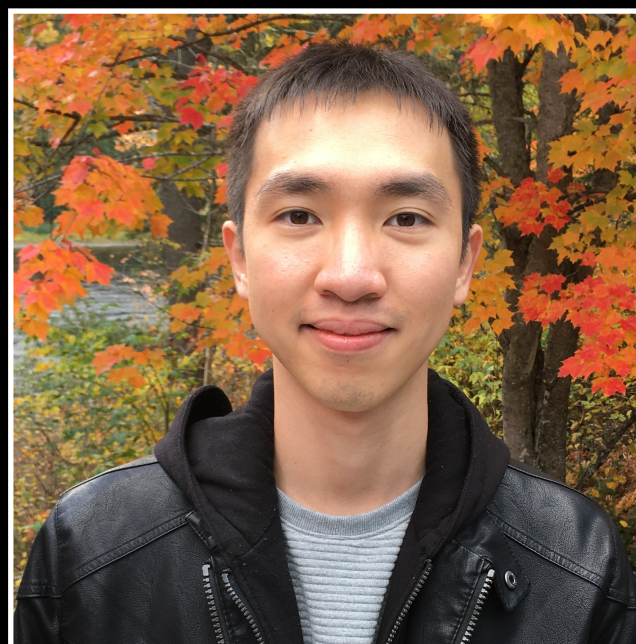


Why does “rubber stamping” happen?

Team Lead



The potential
impact of a
change matters
more!



BLIMP Tracer: Integrating Build Impact Analysis with Code Review

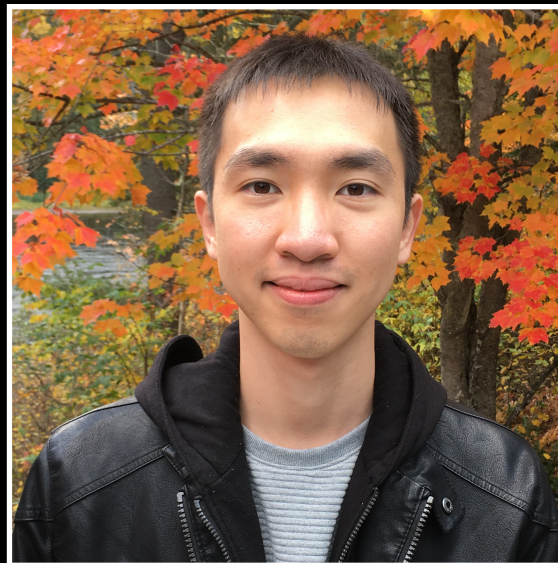
R. Wen, D. Gilbert, M. G. Roche, S. McIntosh
[ICSME 2018]

Smallest Patch

47

Example: how BLIMP Tracer can help

Novice Developer



REVIEW REQUEST #10

```
sslConnection.c
  sslClientServerHandshake(...)
.....
+ if (err != 0)
+   goto fail;
+   goto fail; /* MISTAKE! */
.....
```

BLIMP
TRACER

Most impactful



Least impactful
48

Priority Queue

REVIEW REQUEST #10

REVIEW REQUEST #1

...

REVIEW REQUEST #9



@SoftwareREBELs



rebels.ece.mcgill.ca

Context of the studied team

D~~E~~LL EMC



Backup & Recovery



Client software



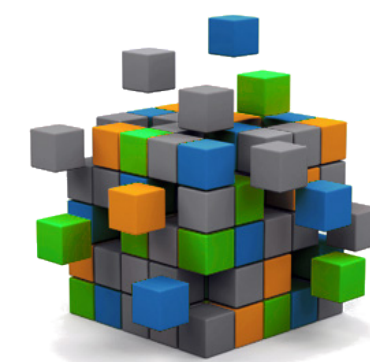
Server software



Administrator software



Storage node software

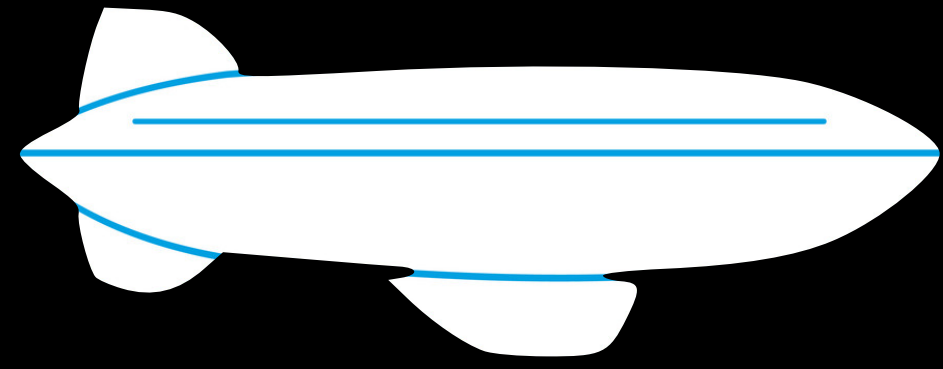


Modules for database



@SoftwareREBELs





BLIMP Tracer

Step 1: Identify changed files

All files in a sample system:

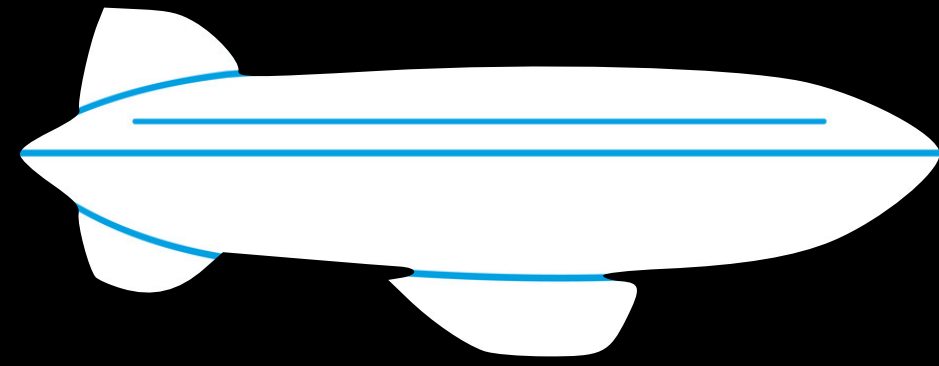
eg1.c

eg2.c

eg3.c

example.h

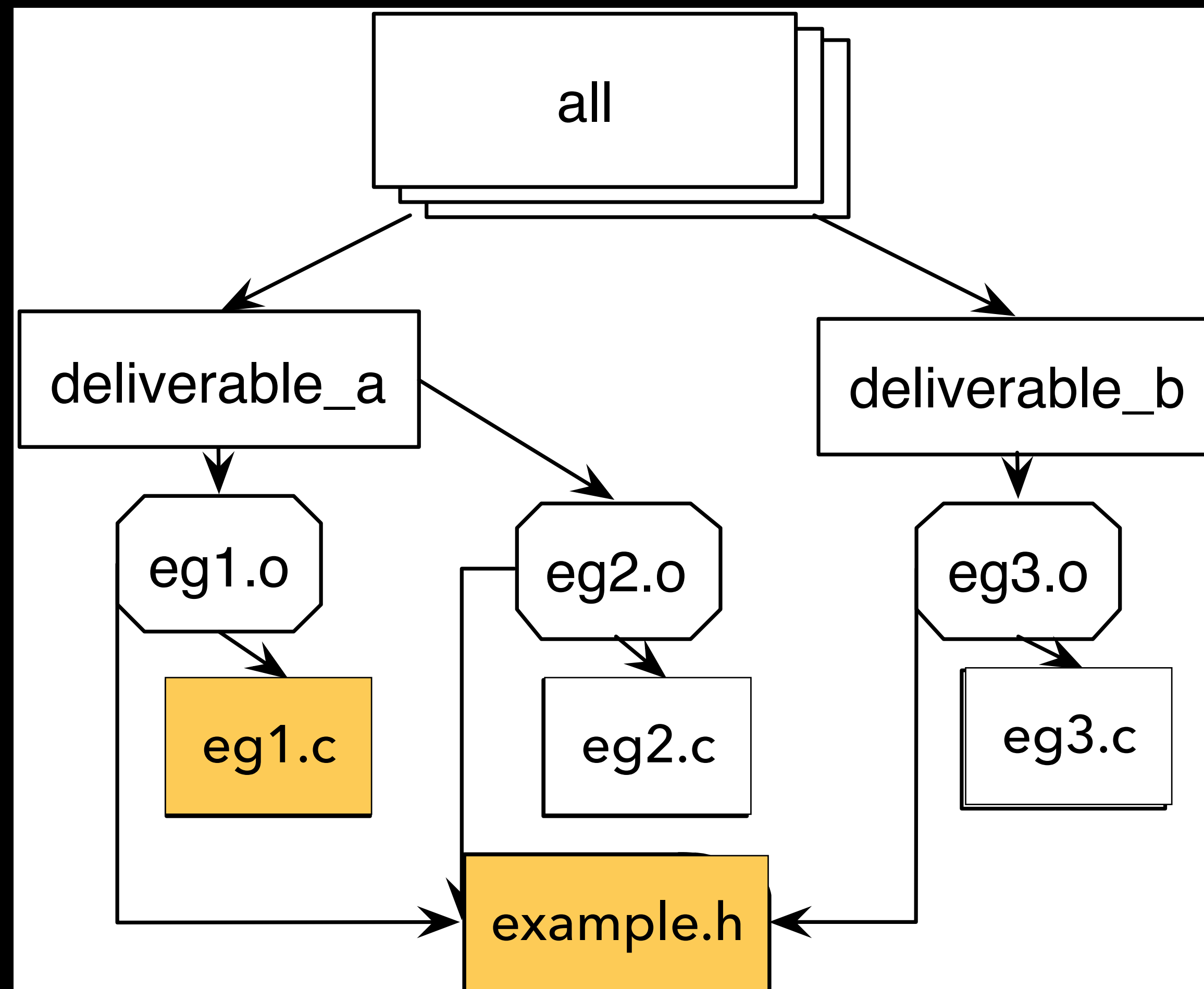
50

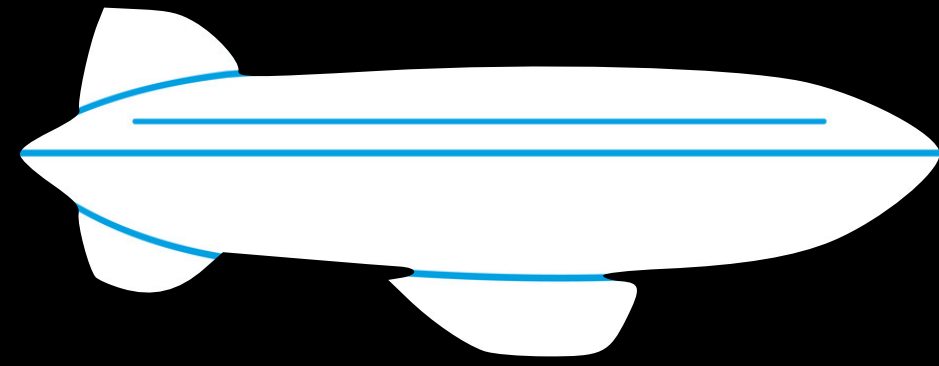


BLIMP Tracer

Step 2: Construct Build Dependency Graph (BDG)

MAKAO

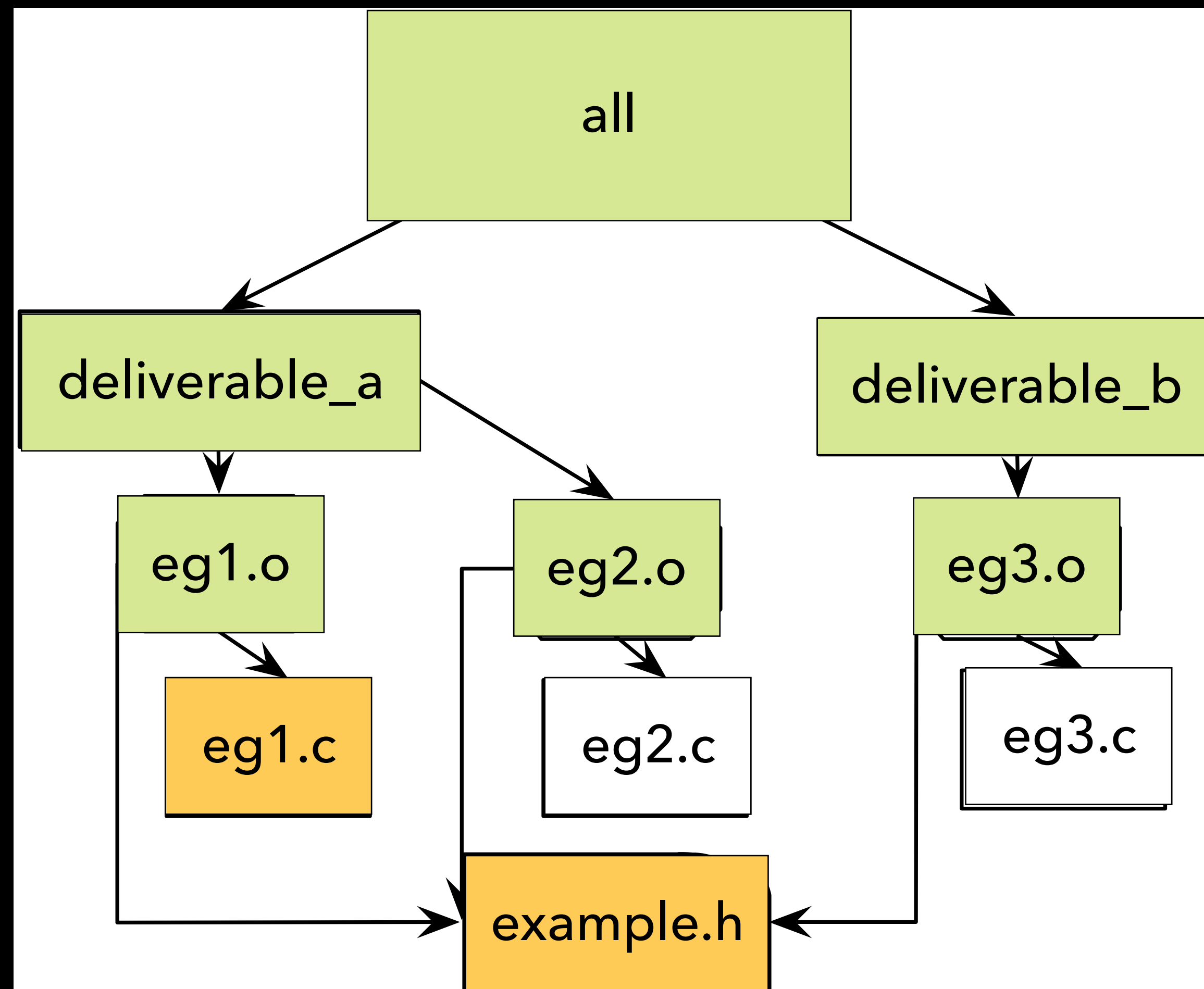


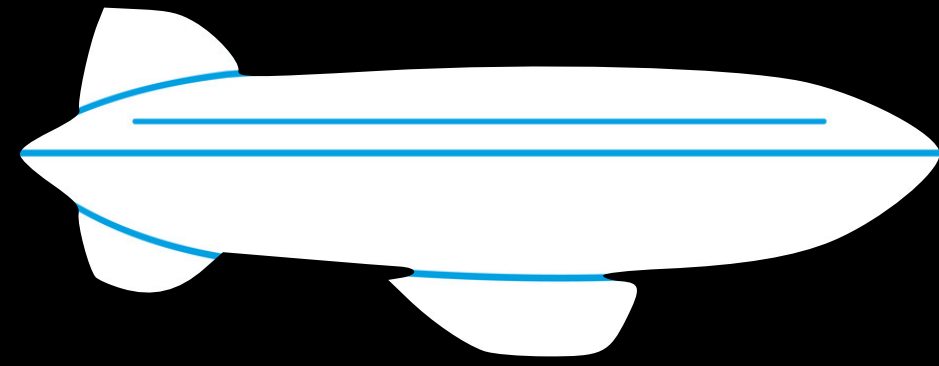


BLIMP Tracer

Step 3: BDG Traversal and Filtering

MAKAO





BLIMP Tracer

Step 4: Present results within the reviewing interface

View Diff

Ship It!

Summary: Bug #12345 Fix compatibility problems

Review Request: [735](#), Created Jun 1, 2018

Submitter: Adam

Bugs: [12345](#)

Description:

Reviewers

Groups: [project_linux](#)

People: Becky

12345: Fix compatibility problems

These are the changes for fixing the problems in bug [12345](#)

Summary of changes:

Testing Done:

Manual testing and unit testing are done by the QA team.

Review request created

Jun 1, 2018 10:42AM

BLIMP Tracer

Jun 1, 2018 10:49AM

BLIMP Tracer: Impact Analysis Summary

Detailed report can be found at: <https://secure.comp.com/blimp/735>

>> linux/app.c impacts on 10 deliverables

>> linux/foo.c impacts on 2 deliverables

<https://secure.comp.com/blimp/735>

BLIMP Tracer

Impact analysis report for RR [735](#)

Generated at Jun 1, 2018, 10:49:01 EDT

Original Review Request URL: <https://reviews.secure.comp.com/735>

[1] linux/app.c impacts on 10 deliverables within 2 components

[2] linux/foo.c impacts on 2 deliverables within 1 component

[Component] ProjectX.Foo

[Deliverable] linux/bar1

[Deliverable] linux/bar2

Icarus climbs
Higher still in the sky
Maybe I've spoken too soon?

Key Assumption:
Each review evolves independently



In reality, code reviews can contain links

To what extent do review links impact code review analytics?



**Toshiki
Hirao**

Ph.D.
Student

#MSR

#codereview #linkage

The Review Linkage Graph for Code Review Analytics: A Recovery Approach and Empirical Study

T. Hirao, S. McIntosh, A. Ihara, K. Matsumoto

[ESEC/FSE 2019]

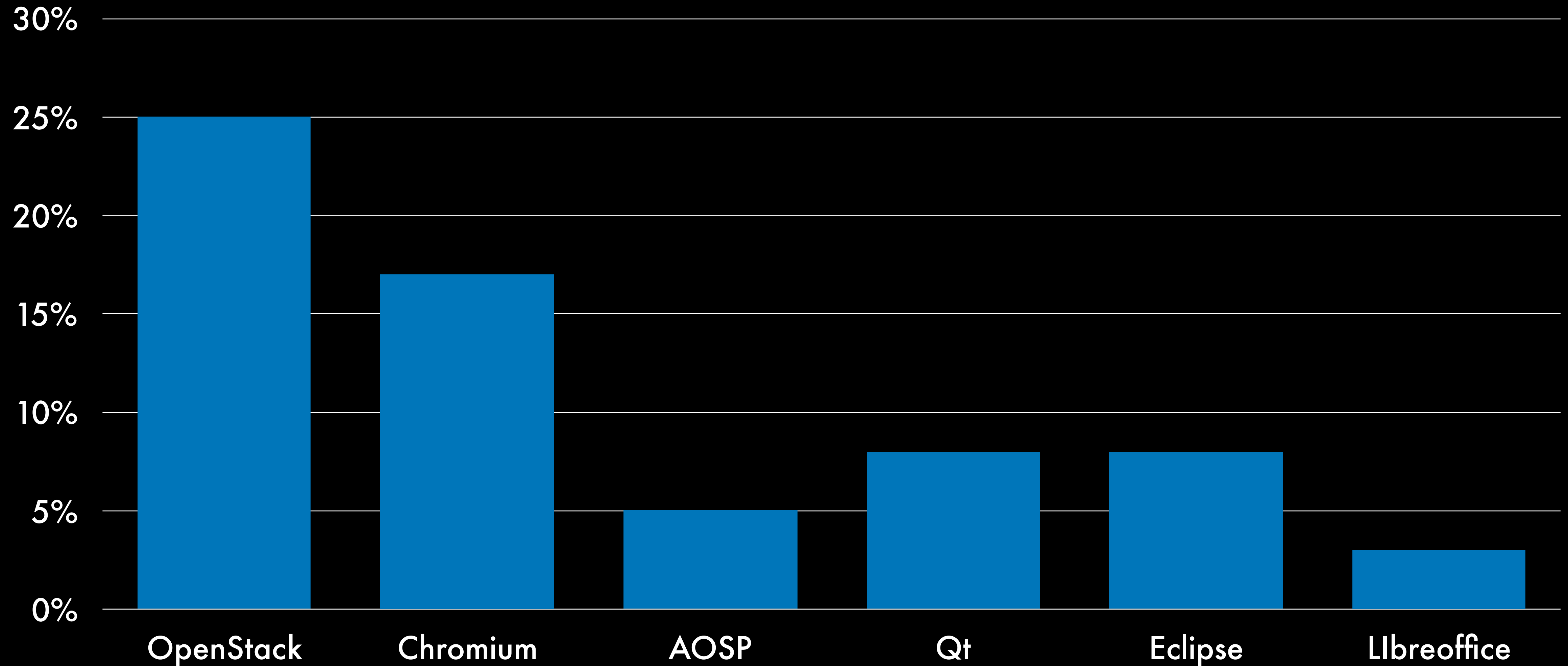


@SoftwareREBELs



rebels.ece.mcgill.ca

Links between reviews are not rare



Manually labelling reviews

Linked reviews
from OpenStack



Duplicated
Solution



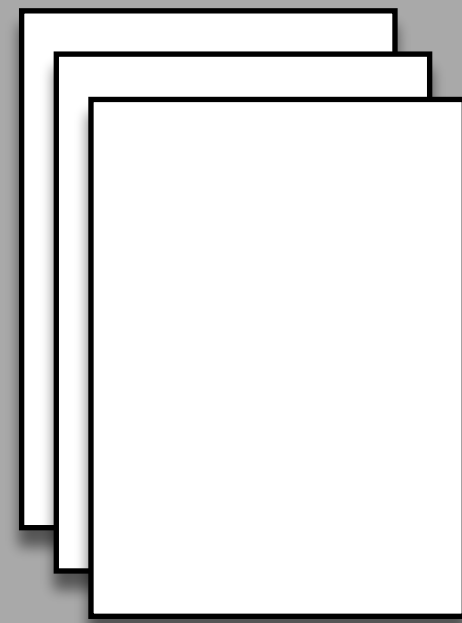
Looks like a
duplicated solution



I agree!

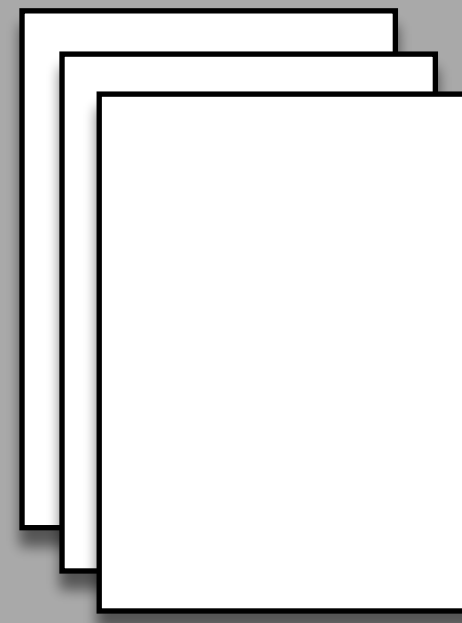
Group review linkage patterns into categories

Duplicated
Solution



Category:
Alternative
Solution

Shallow
Fix



Root
Cause



Category:
Patch
Dependency

The five discovered categories and their link to review analytics

	Nova	Neutron	Example
Patch Dependency	55%	56%	It's used in the dependent patch: <review 100383>
Broader Context	20%	1%	<review 103174>.
Alternative Solution	14%	1%	<review 99522>.
Version Control Issues	6%	6%	Unfortunately this patch is conflict to <review 94208>
Feedback Related	5%	4%	I split this ... after comment... on my proposed fix <review 93903 >

Reviewer recommendation

Review outcome prediction

The five discovered categories and their link to review analytics

	Nova	Neutron	Example
Patch Dependency	55%	56%	It's used in the dependent patch: <review 100383>
Broader Context	20%	19%	Have a look at <review 103174>.
Alternative Solution	14%	1	<div>Review outcome prediction</div> <review 99522>.
Version Control Issues	6%	6%	Unfortunately this patch is conflict to <review 94208>
Feedback Related	5%	4%	I split this ... after comment... on my proposed fix <review 93903 >

The five discovered categories and their link to review analytics

	Nova	Neutron	Example
Patch Dependency	55%	56%	It's used in the dependent patch: <review 100383>
Broader Context	20%	19%	Have a look at <review 103174>.
Alternative Solution	14%	15%	It looks same as <review 99522>.
Version Control Issues	6%	6%	which is conflict to
Feedback Related	5%	4%	ment... on my proposed fix <review 93903 >

Reviewer recommendation

Review outcome prediction

The five discovered categories and their link to review analytics

	Nova	Neutron	Example
Patch Dependency	55%	56%	It's used in the dependent patch: <review 100383>
Broader Context	20%	19%	Have a look at <review 103174>.
Alternative Solution	14%	15%	It looks same as <review 99522>.
Version Control Issues	6%	6%	Unfortunately this patch is conflict to <review 94208>
Feedback Related	5%	4%	I split this ... after comment... on my proposed fix <review 93903 >

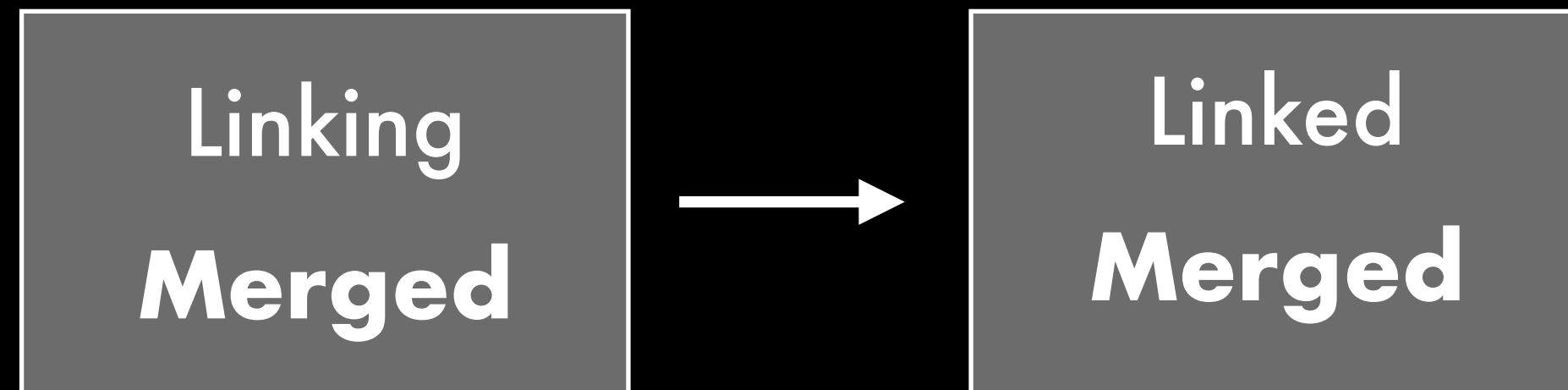
The five discovered categories and their link to review analytics

	Nova	Neutron	Example
Patch Dependency	55%	56%	It's used in the dependent patch: <review 100383>
Broader Context	20%	19%	Have a look at <review 103174>.
Alternative Solution	14%	15%	It looks same as <review 99522>.
Version Control Issues	6%	Reviewer recommendation	... which is conflict to <review 94208>
Feedback-Related	5%	4%	I split this ... after comment... on my proposed fix <review 93903>

A closer look at the potential impact on review outcome prediction

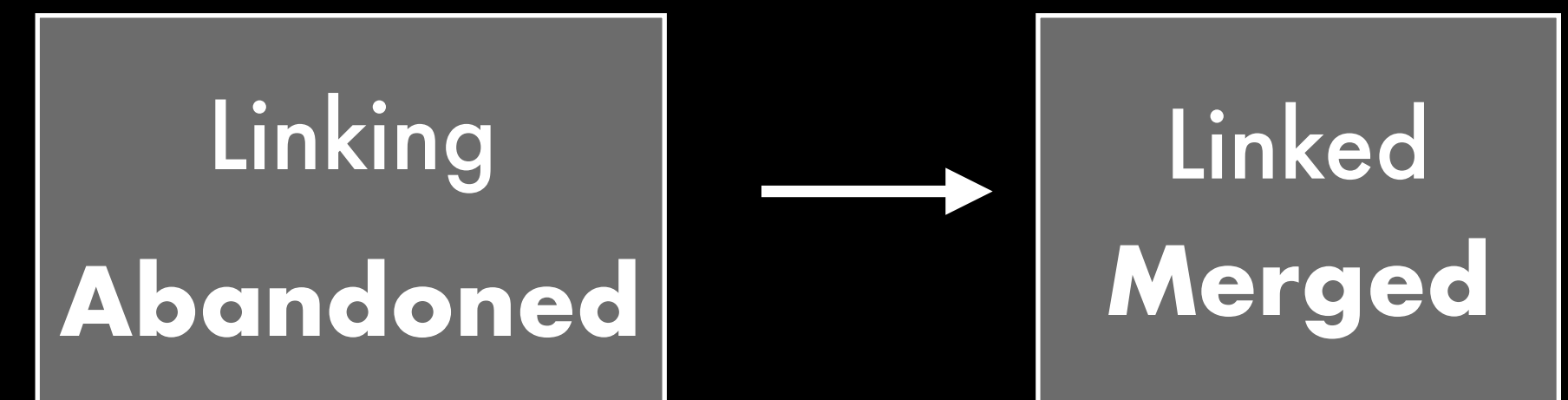
Patch Dependency

The outcome depends on the other review



Alternative Solution

One is likely no longer necessary



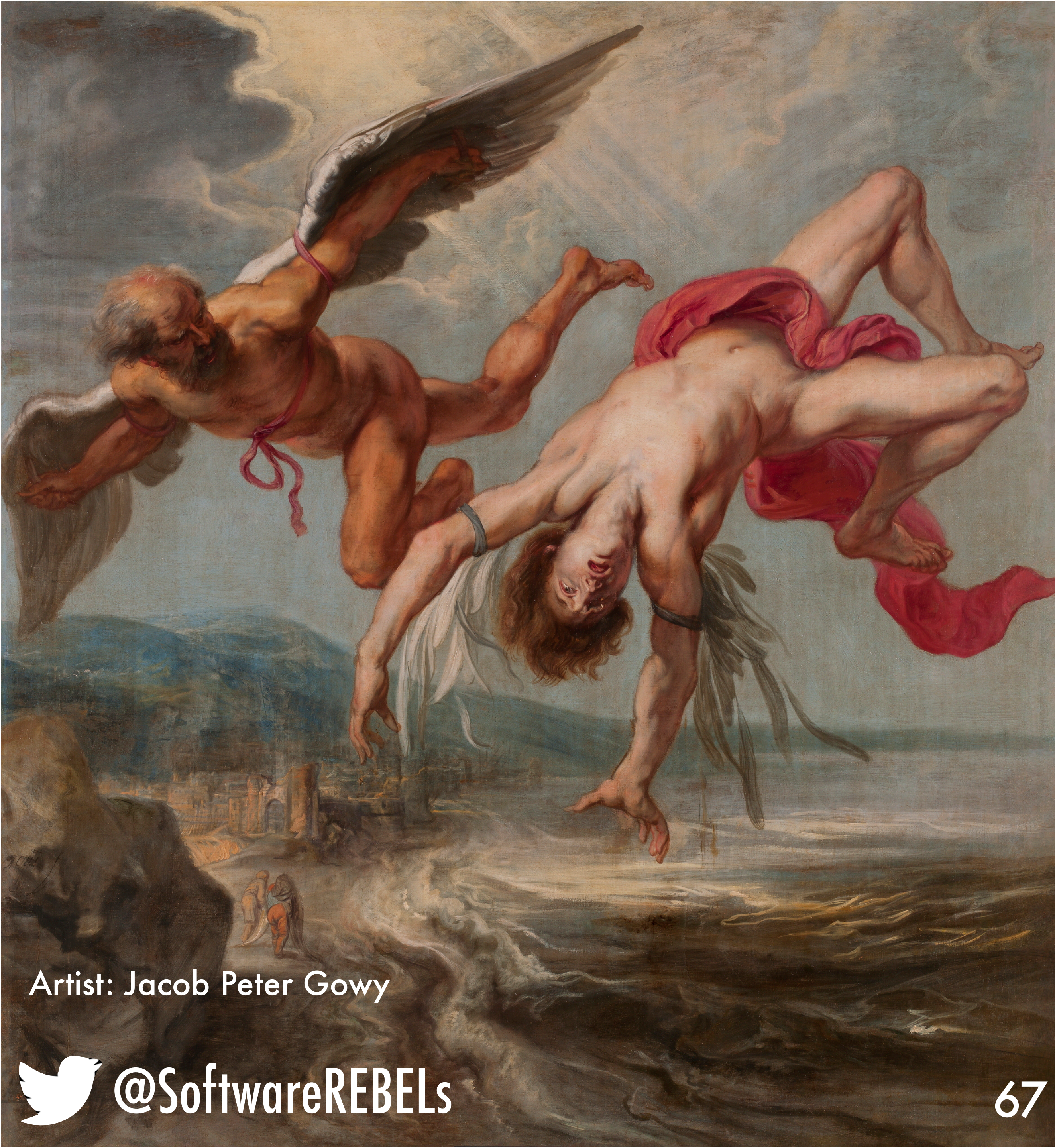
Patch dependency links tend to have the same outcome; alternative solutions do not

Identical Outcome Rate (IOR)		Nova	Neutron
Patch Dependency	IOR (Merged)	87%	71%
	IOR (Abandoned)	86%	86%
Alternative Solution	IOR (Merged)	18%	26%
	IOR (Abandoned)	46%	62%



A closer look at the potential impact on reviewer recommendation

Overlapping Reviewer Rate (ORR)	Nova	Neutron
Patch Dependency	50%	51%
Alternative Solution	65%	77%



Artist: Jacob Peter Gowy



@SoftwareREBELs

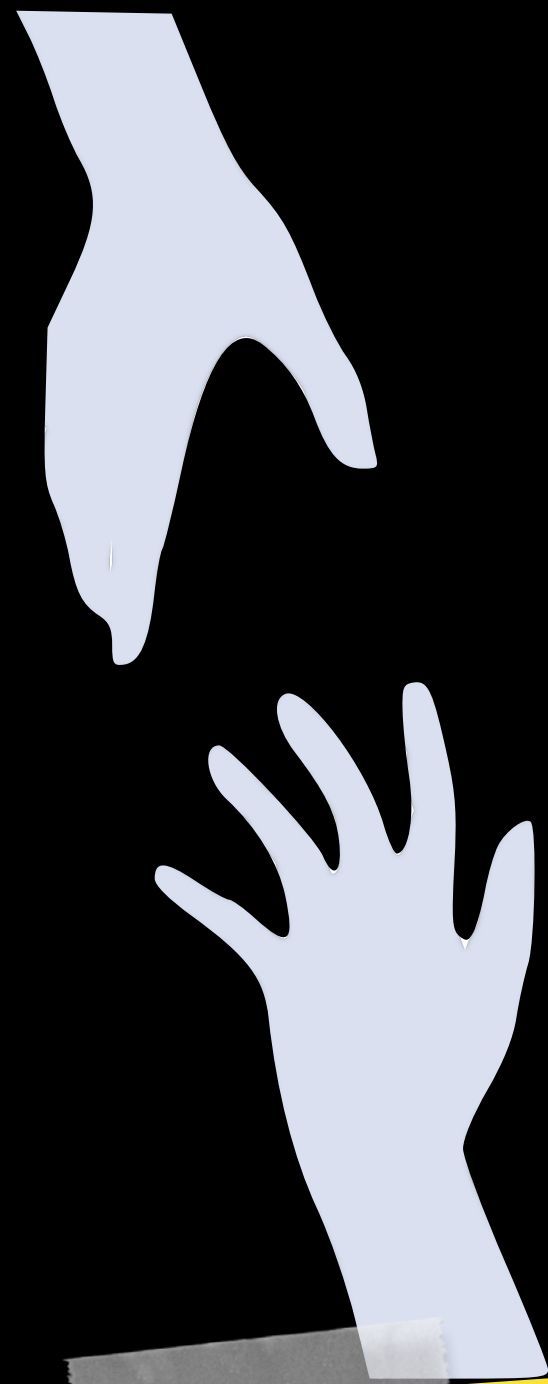
67

O gods!
Why is this happening to me?
All I wanted was a new life
For my **SOFTWARE REBELS** to grow up free



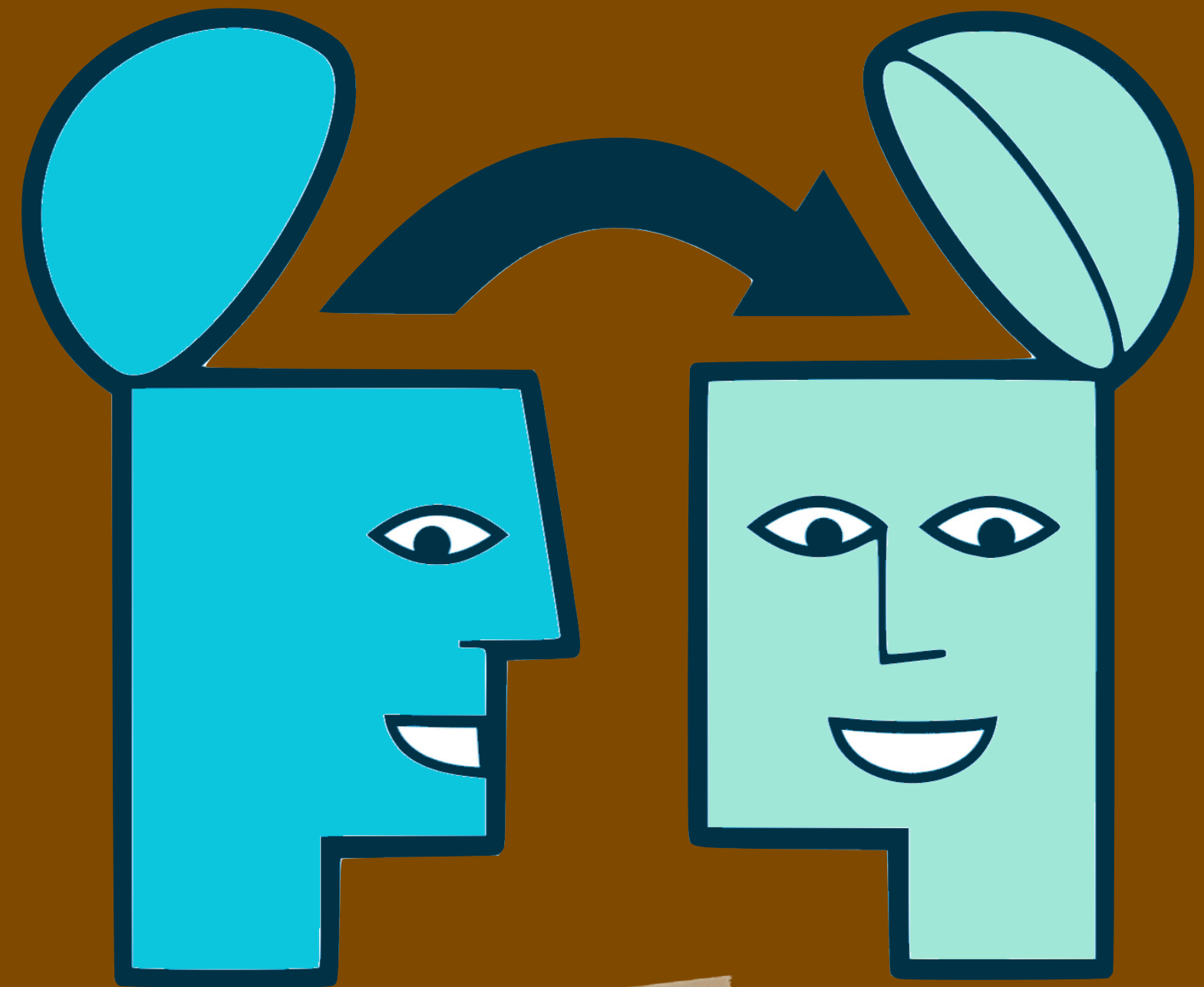
rebels.ece.mcgill.ca

Supporting



Watch out for noise
and heterogeneity in
build outcome data!

Leveraging



Watch out for
interconnected
artifacts of interest!





Artist: Paul Lee



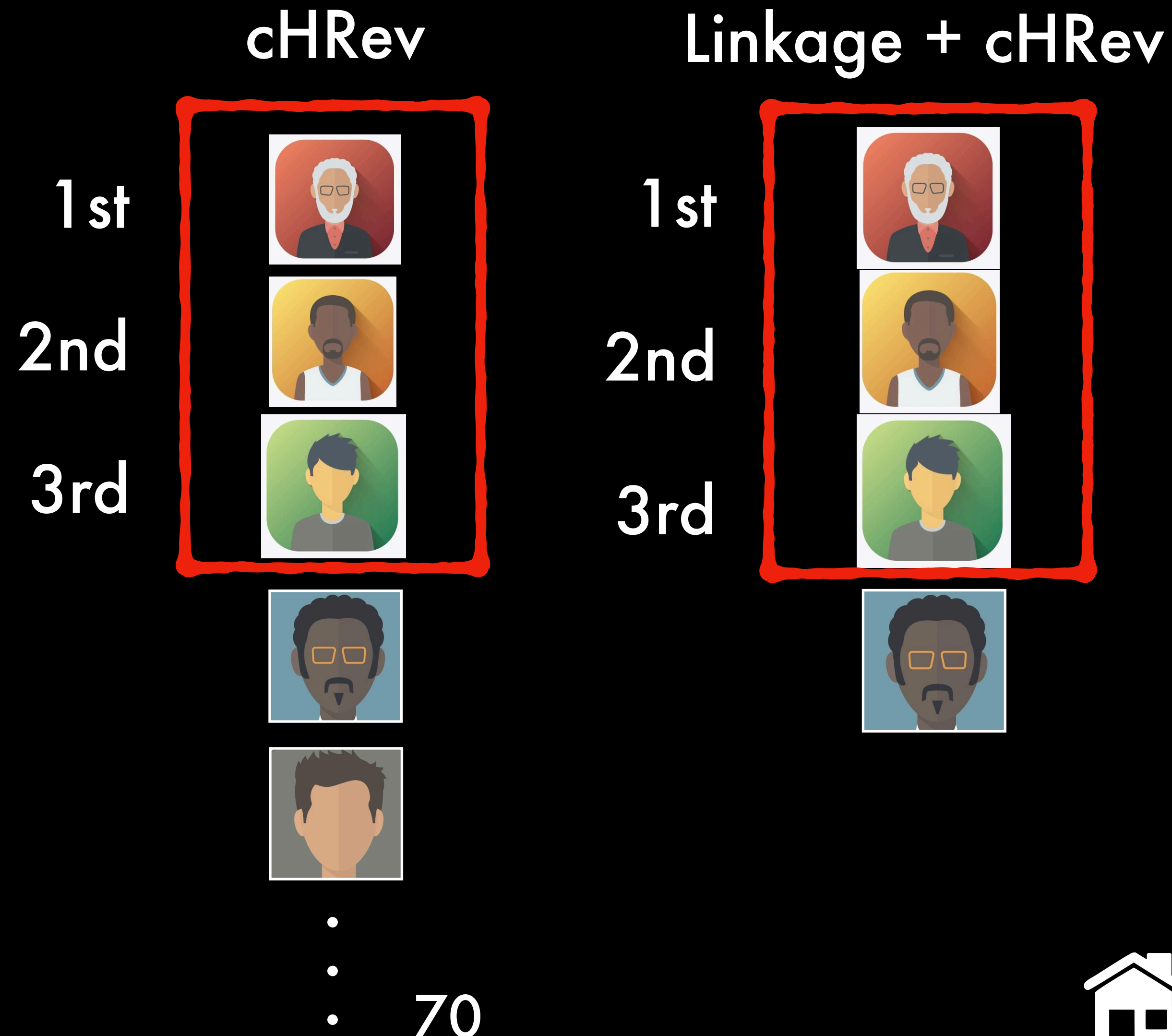
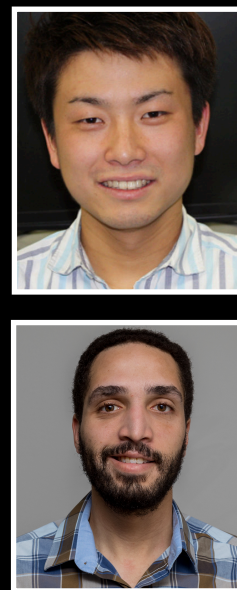
@SoftwareREBELs

I will never  again!
I will hang up my 
O gods!



If overlapping reviewers are added at the top of the reviewer candidate list...

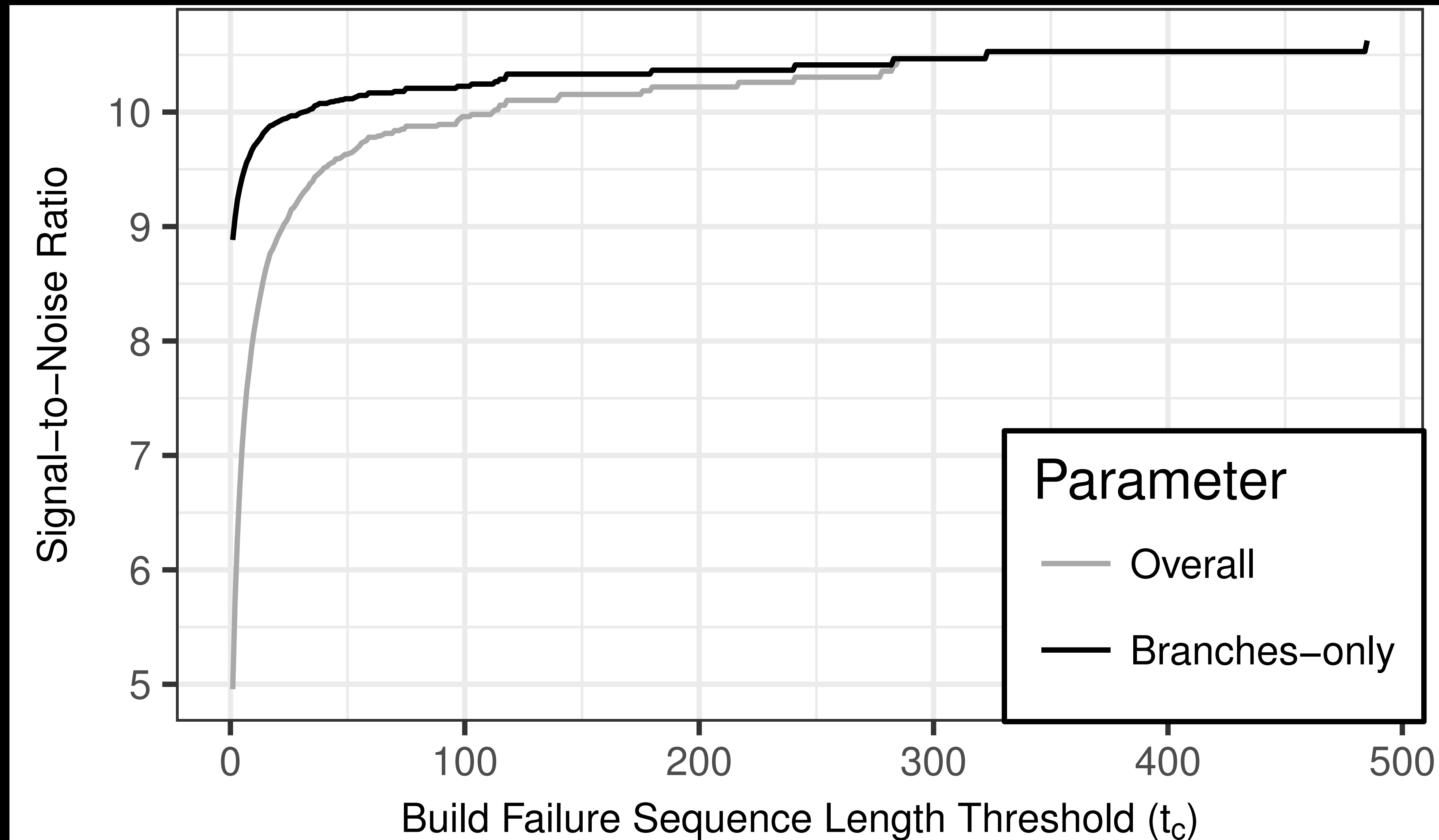
Overlapping reviewers



...Substantially better results are achieved!

	Nova			Neutron		
	Prec.	Rec.	F1.	Prec.	Rec.	F1.
cHRev	0.30	0.32	0.30	0.28	0.31	0.29
cHRev+ Linkage	0.40	0.43	0.41	0.42	0.45	0.43

Signal-to-Noise Ratio (SNR) for CI data





Artist: Paul Lee

I will never  again!
I will hang up my 
O gods!



@SoftwareREBELs
rebels.ece.mcgill.ca