Icarus


Icarus
Over-
confidence


Icarus
Over-confidence
"Don't fly too close to the sun"


Icarus
Over-confidence
"Don't fly too close to the sun"
Daedalus

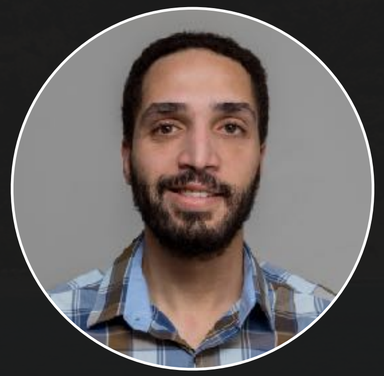

Icarus
Over-confidence
"Don't fly too close to the sun"
Daedalus
Master craftsman


Icarus
Over-confidence
"Don't fly too close to the sun"
Daedalus
Master craftsman
Inventions often have unintended consequences

The unintended consequences of mining software build systems

Shane McIntosh

Lead Rebel



UNIVERSITY OF WATERLOO

SOFTWARE REBELS

@SoftwareREBELs

rebels.cs.uwaterloo.ca

Artist: Markus Stadlober


I've got a plan
And some wax and some string
Some feathers I stole from the birds
Artist: Markus Stadlober

```java
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!");
  }
}
```

```java
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!");
  }
}
```

```xml
<project …>
  <groupId>rebels</groupId>
  <artifactId>helloworld</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <dependencies>
   <dependency>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
      <version>1.2.17</version>
    </dependency>
  </dependencies>
</project>
```

```
$ mvn compile
```

```
$ mvn compile

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

```
$ mvn compile

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

```
$ 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 ^%@#$!
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:
```

```
$ 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]
```

```java
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!");
  }
}
```

```xml
<project …>
  <groupId>rebels</groupId>
  <artifactId>helloworld</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <dependencies>
   <dependency>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
      <version>1.2.17</version>
    </dependency>
  </dependencies>
</project>
```

```java
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!");
  }
}
```

```xml
<project …>
  <groupId>rebels</groupId>
  <artifactId>helloworld</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <dependencies>
   <dependency>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
      <version>1.2.17</version>
    </dependency>
  </dependencies>
</project>
```

```java
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!");
  }
}
```

```xml
<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
```

```
$ mvn compile

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

```
$ mvn compile
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

Why can’t the build tool fix this mistake automatically?

Why can't build systems sustain themselves?

Why can’t build systems sustain themselves?

Un(der)specified dependency detection

Why can't build systems sustain themselves?

Un(der)specified dependency detection

+

Build change anticipation

Why can't build systems sustain themselves?

Un(der)specified dependency detection

+

Build change anticipation

+

Automated repair of build scripts

Why can't build systems sustain themselves?

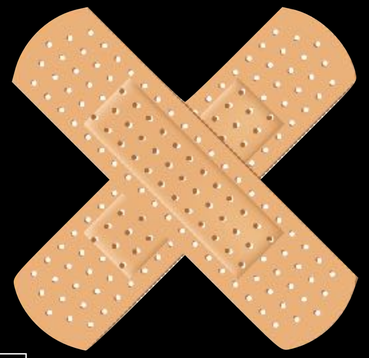
Un(der)specified dependency detection

+

Build change anticipation

+

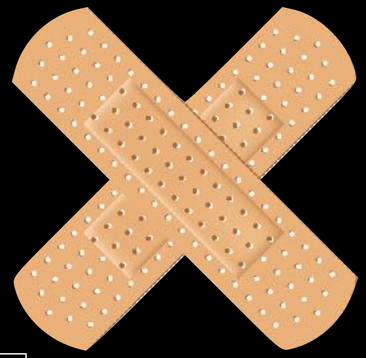
Automated repair of build scripts



Build Medic

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

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

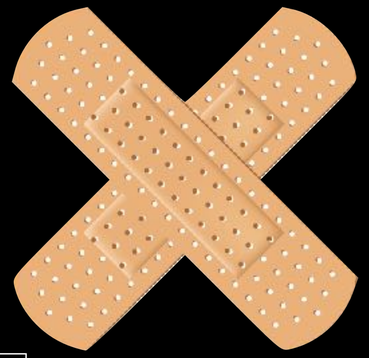


Build Medic

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

Detect Build Failure Type

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



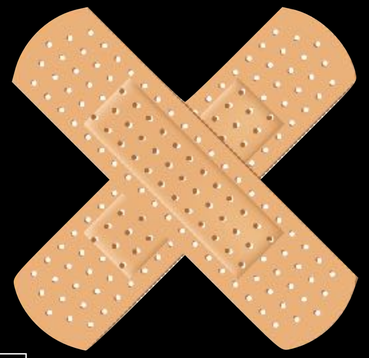
Build Medic

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

Detect Build Failure Type

Apply Repair Strategy

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



Build Medic

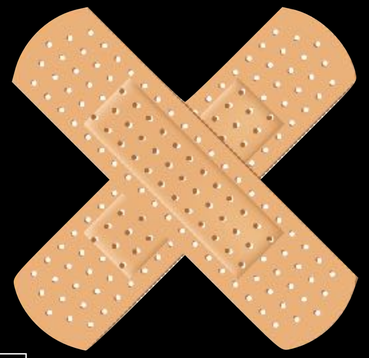
```
[INFO] ------------
[INFO] BUILD FAILURE
[INFO] ------------
…
```

Detect Build Failure Type

Apply Repair Strategy

Evaluate Revision

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



Build Medic

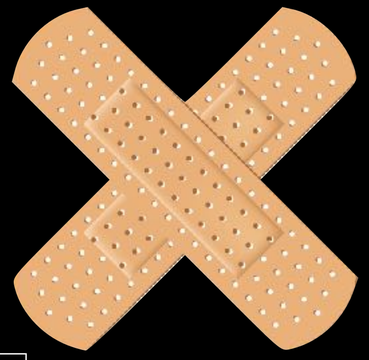
```
[INFO] ------------
[INFO] BUILD FAILURE
[INFO] ------------
…
```

Detect Build Failure Type

Apply Repair Strategy

Evaluate Revision

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



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]

Artist: Charles Paul Landon

Artist: Charles Paul Landon

@SoftwareREBELs

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

BuildMedic achieves promising results on 84 unseen breakage pairs

BuildMedic achieves promising results on 84 unseen breakage pairs

54%

of repair attempts are eventually successful

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 often identical or very similar to developer repairs

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

36%

of successful repairs are identical to the corresponding developer repairs

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

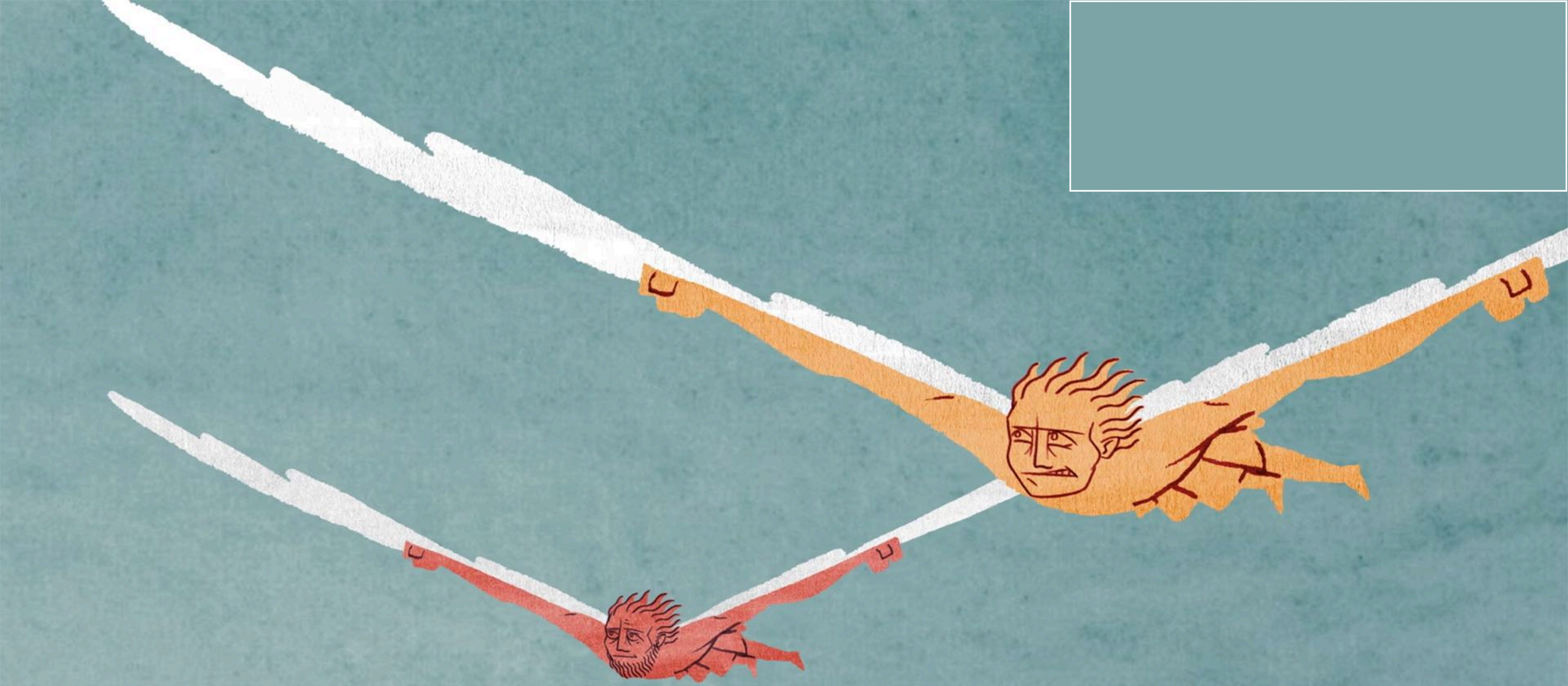
36%

of successful repairs are identical to the corresponding developer repairs

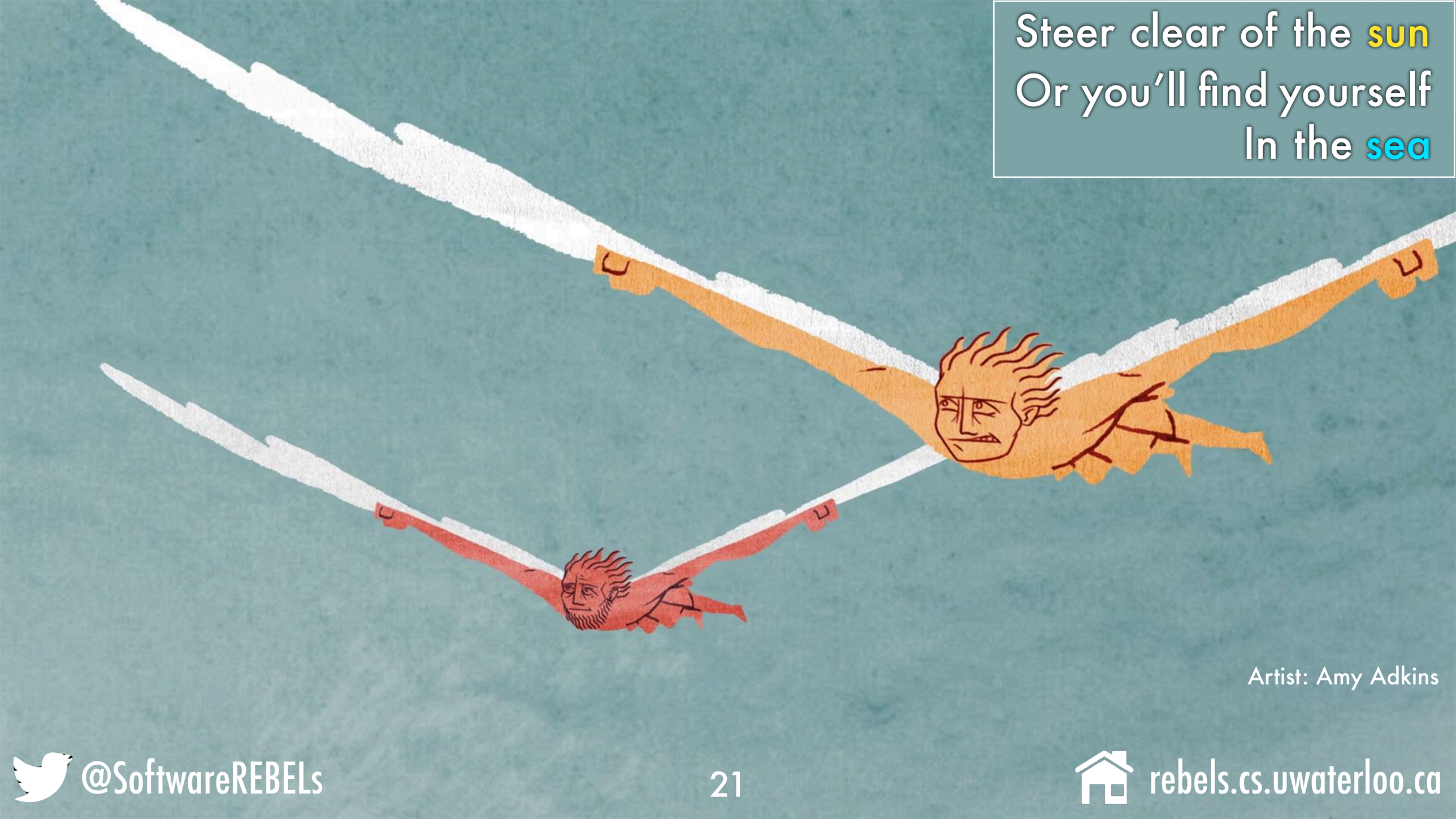
Another

44%

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



Artist: Amy Adkins



Artist: Amy Adkins

Operating Assumption: Historical breakages are important

Developer

Breakage

Can we rely on historical build records?

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

Can we rely on historical build records?

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

Actively ignored failures:

Can we rely on historical build records?

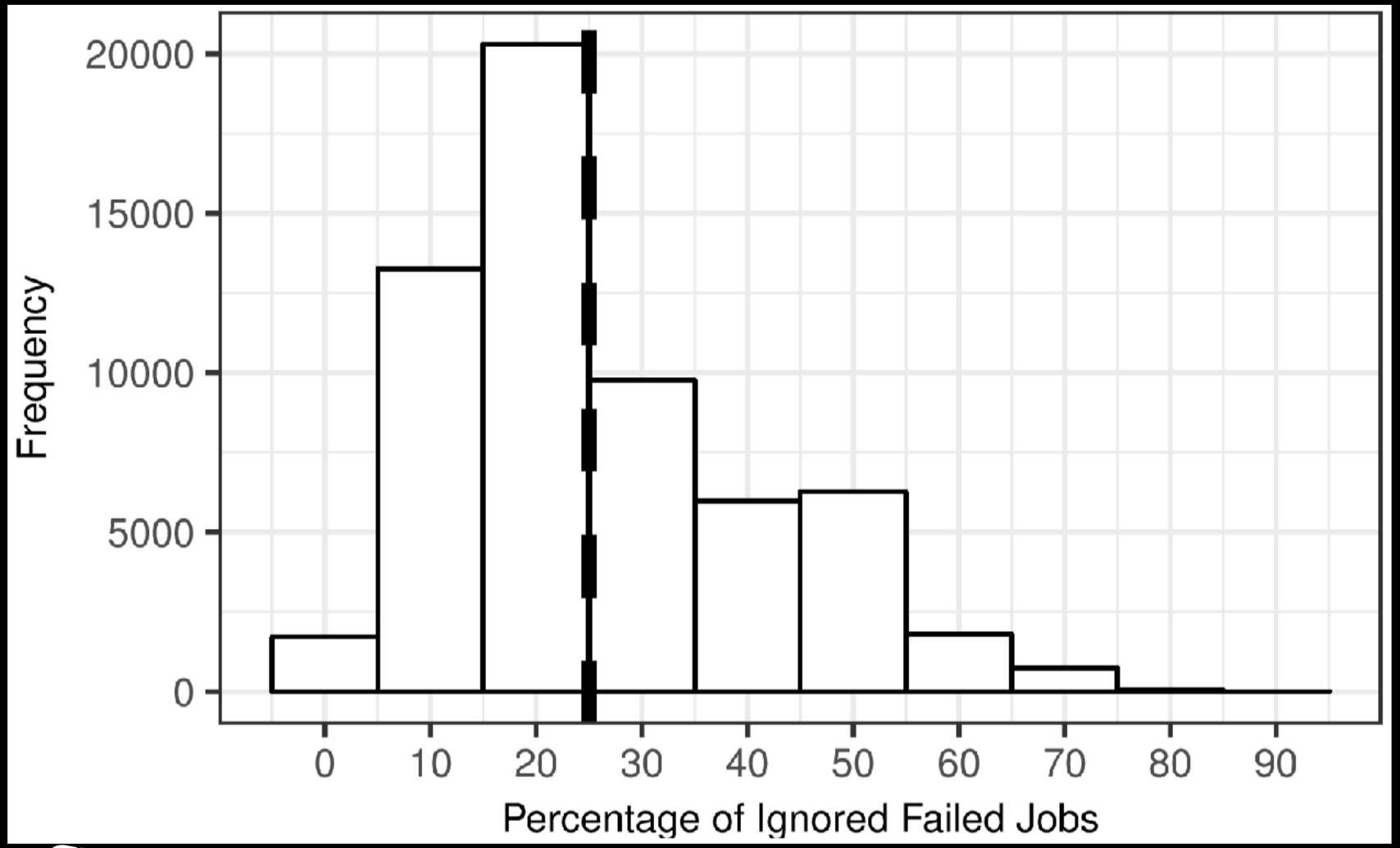
Noise and Heterogeneity in Historical Build Data: An Empirical Study of Travis CI
K. Gallaba, C. Macho, M. Pinzger, S. McIntosh
[ASE 2018]

Actively ignored failures:

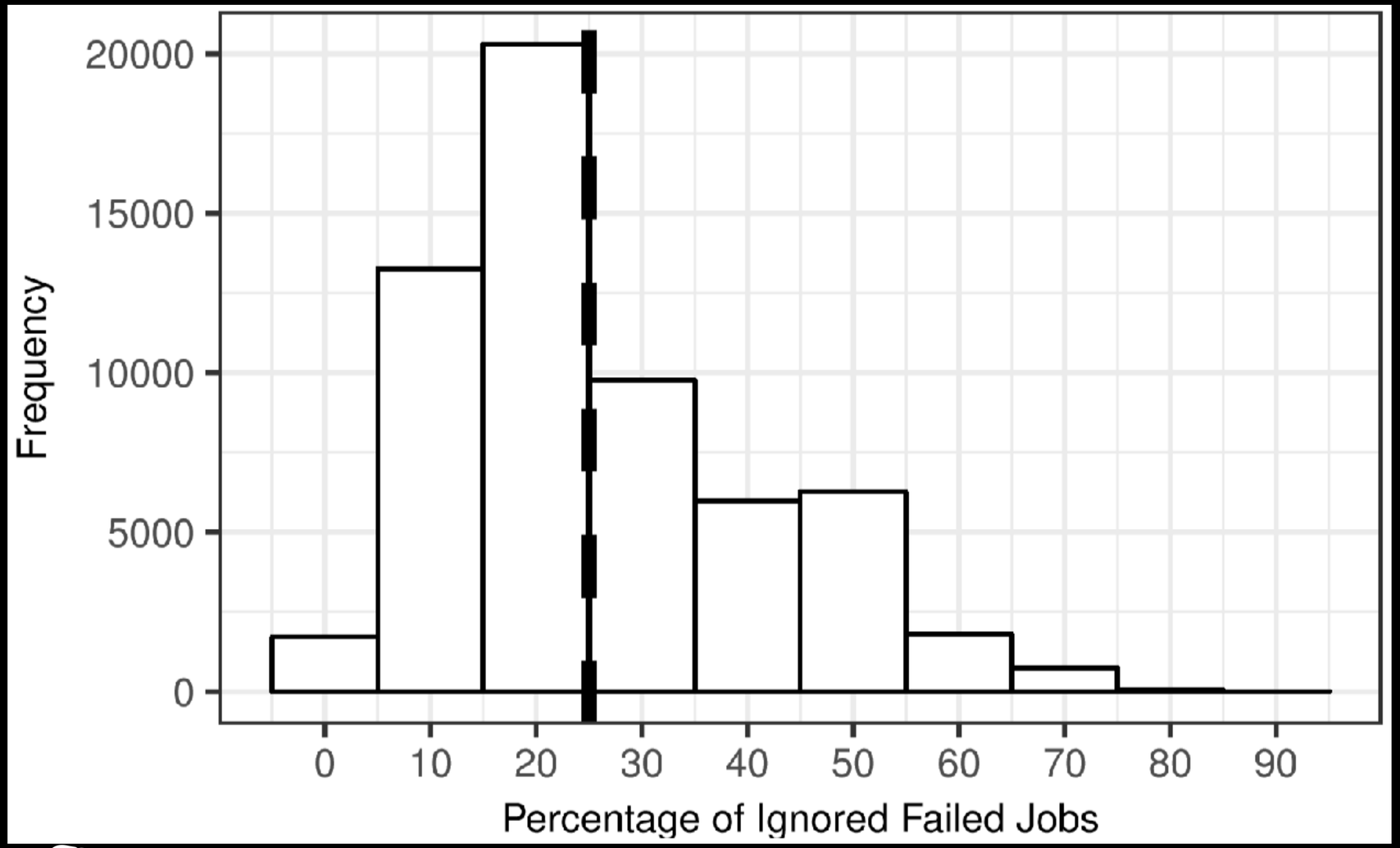
12%

of passing builds have at least one actively ignored failure

In those passing builds with actively ignored failures…



In those passing builds with actively ignored failures…

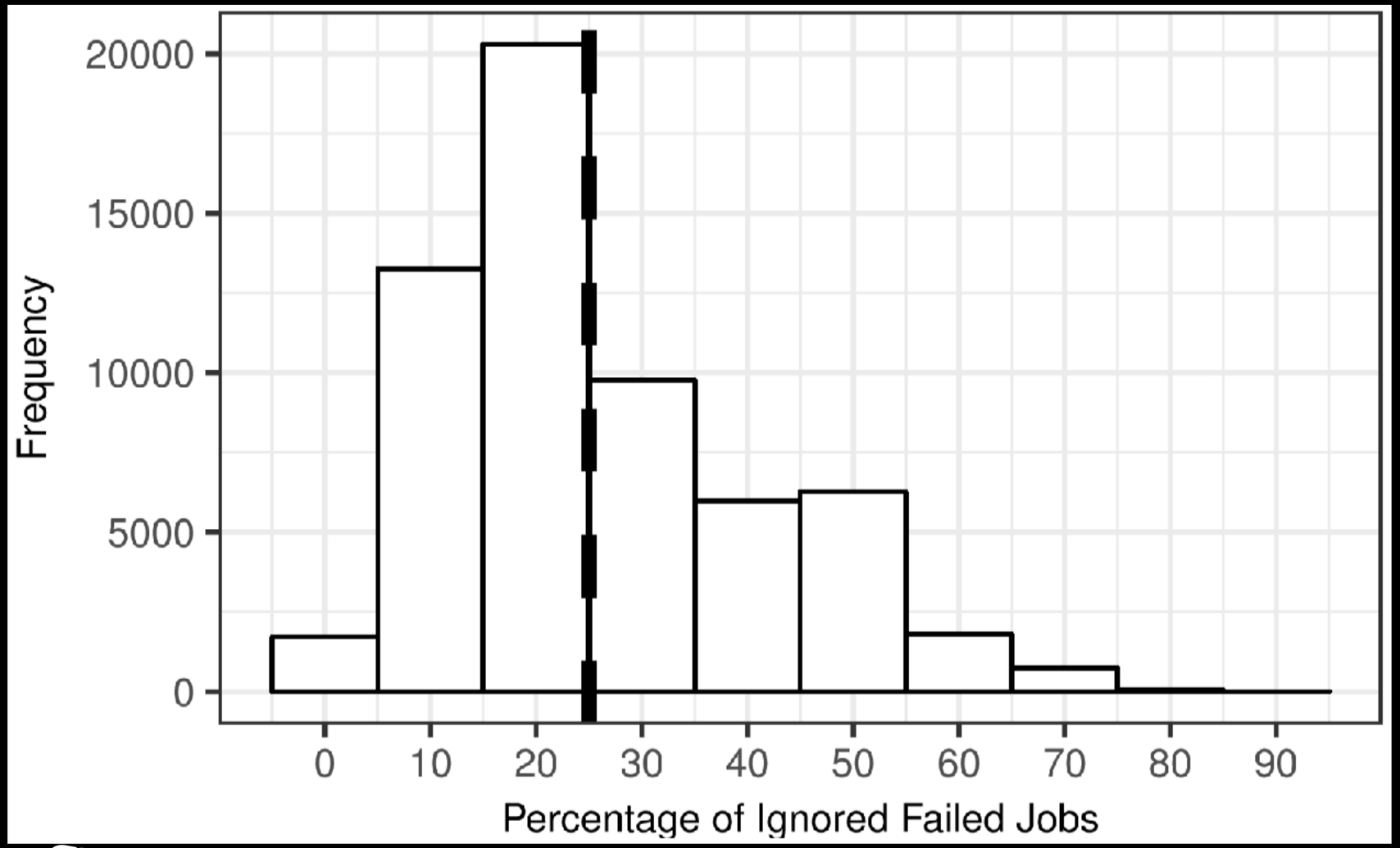


A median of

25%

of jobs failed

In those passing builds with actively ignored failures…



A median of

25%

of jobs failed

Up to

87%

of jobs are actively ignored

Can we rely on historical build records?

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

Actively ignored failures:

12%

of passing builds have at least one actively ignored failure

Can we rely on historical build records?

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

Actively ignored failures:

12%

of passing builds have at least one actively ignored failure

Passively ignored failures:

Can we rely on historical build records?

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

Actively ignored failures:

12%

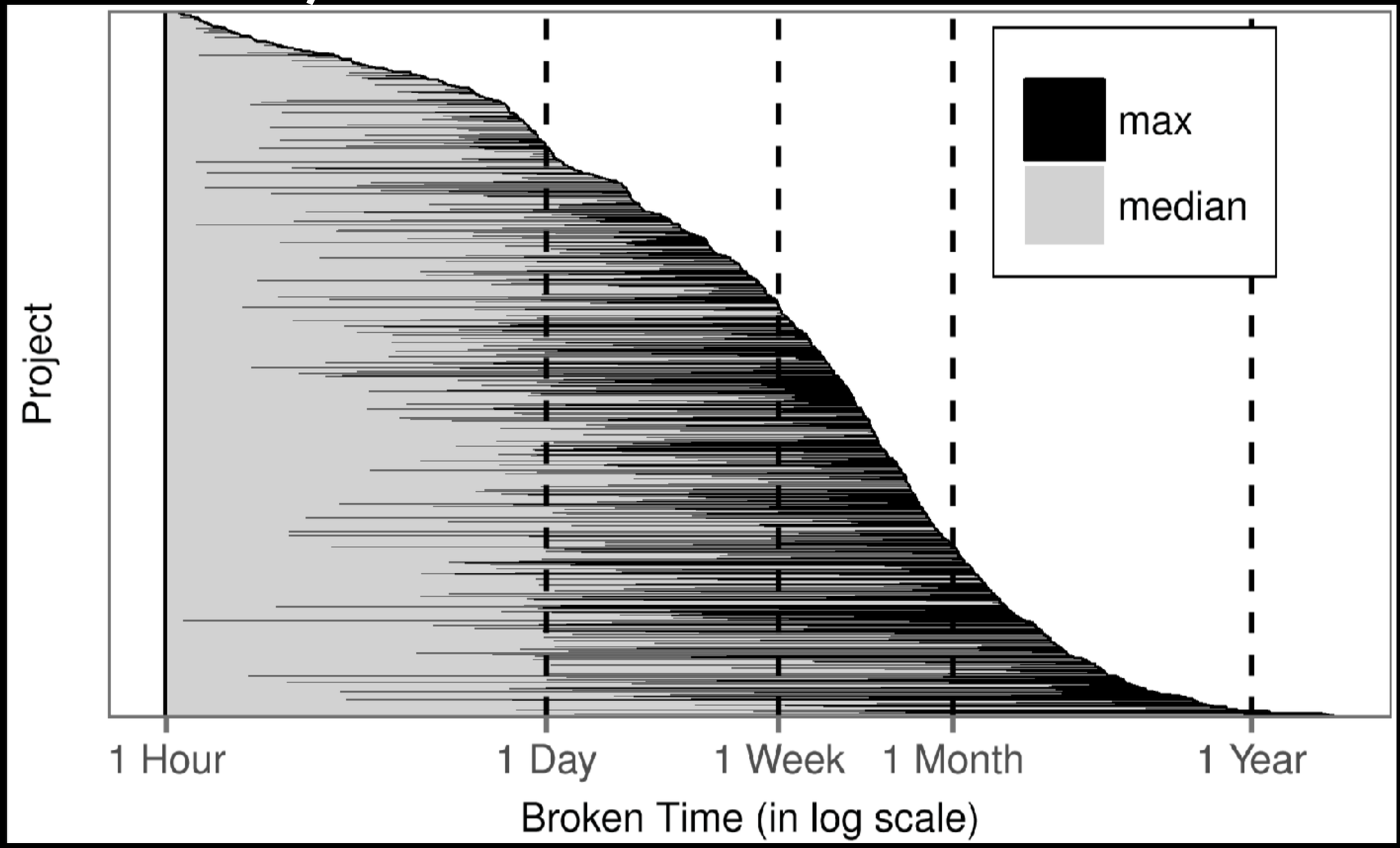
of passing builds have at least one actively ignored failure

Passively ignored failures:

67%

of failing builds are stale

In some cases, builds remain broken for 423 days



Artist: Paul Lee

Artist: Paul Lee

I will never  again!

I will hang up my 

O gods!

Why can’t build systems sustain themselves?

Un(der)specified dependency detection

+

Build change anticipation

+

Automated repair of build scripts

Why can't build systems sustain themselves?

Un(der)specified dependency detection

+

Build change anticipation

+

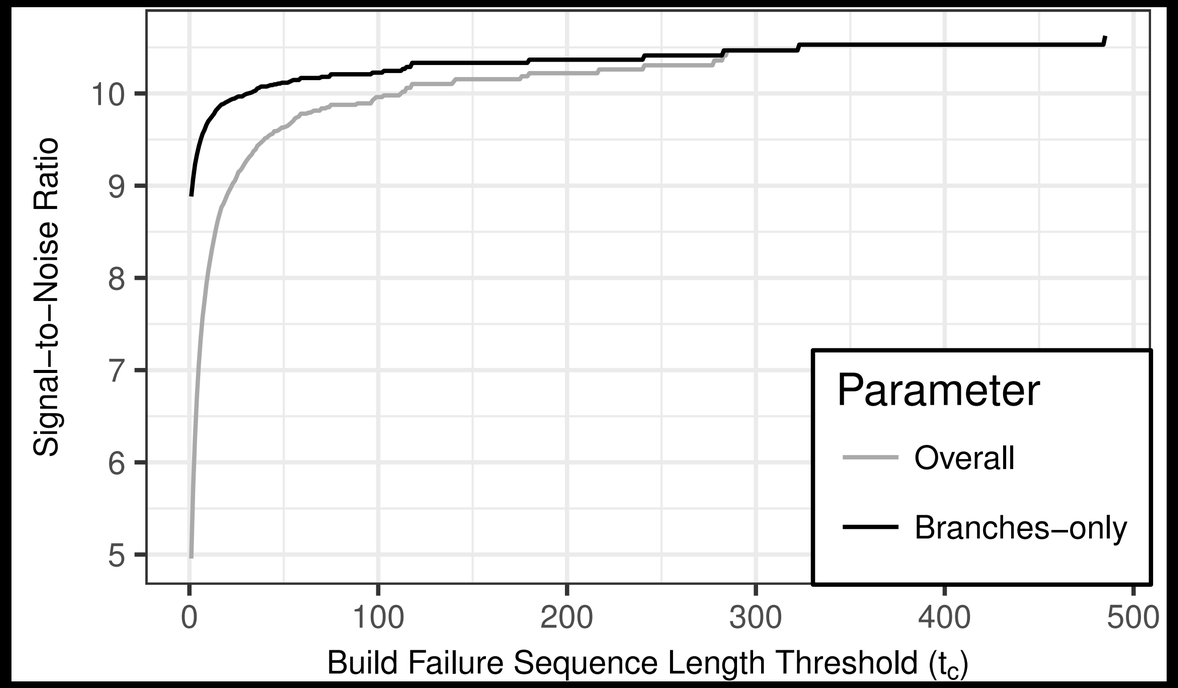
Automated repair of build scripts

Conclusion 1:

Our evaluation is correct! The data is broken!

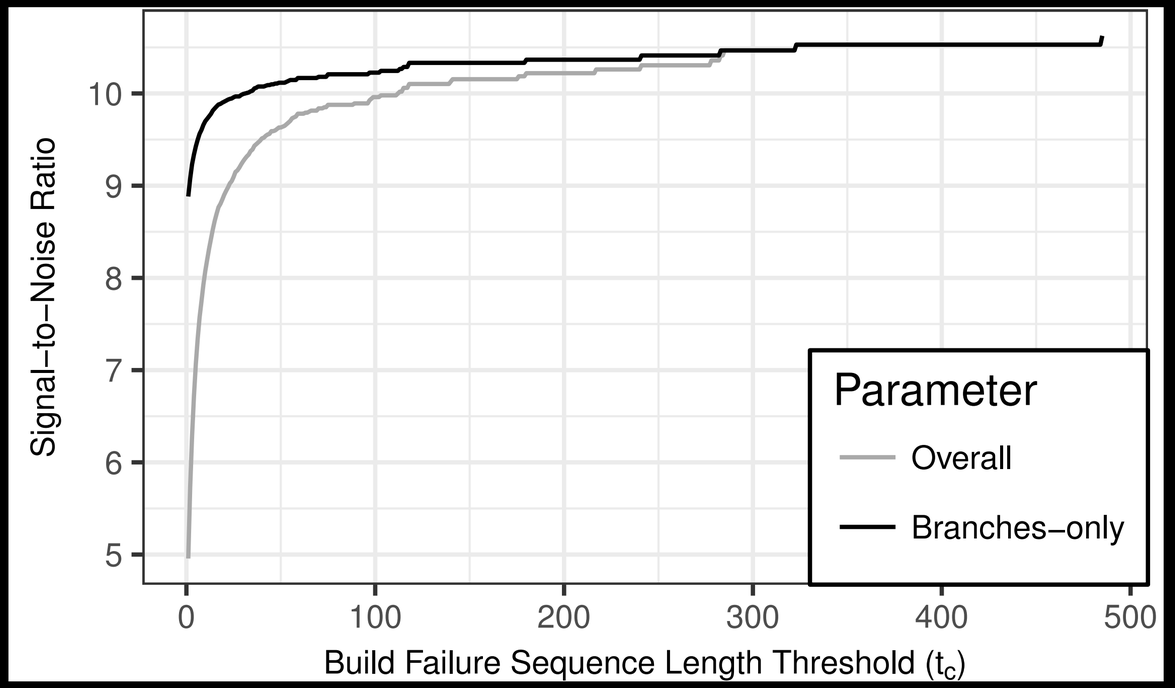
Conclusion 1:

Our evaluation is correct! The data is broken!



Conclusion 1:

Our evaluation is correct! The data is broken!

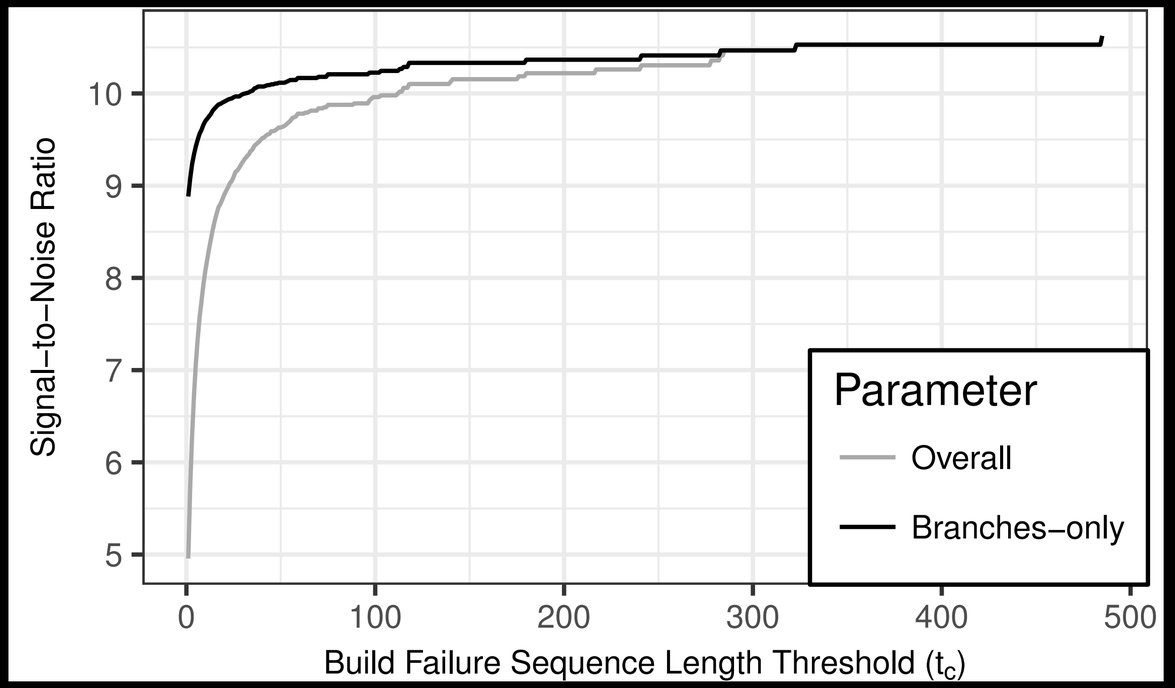


Conclusion 2:

Our evaluation is incorrect! Please help us to fix it!

Conclusion 1:

Our evaluation is correct! The data is broken!



Conclusion 2:

Our evaluation is incorrect! Please help us to fix it!

shane.mcintosh@uwaterloo.ca

/in/shane-mcintosh

@shane_mcintosh

Thrice, “Daedalus,”
https://open.spotify.com/track/42lZpbV1P8KbzDl1duWT25