
goto;
chicago
Emerging Trends in Code Quality and Security Automation
Dr. Stephen Magill
CEO, MuseDev
follow us @gotochgo

goto; chicago

Click ‘Rate Session’
to rate session
and ask questions.

follow us @gotochgo

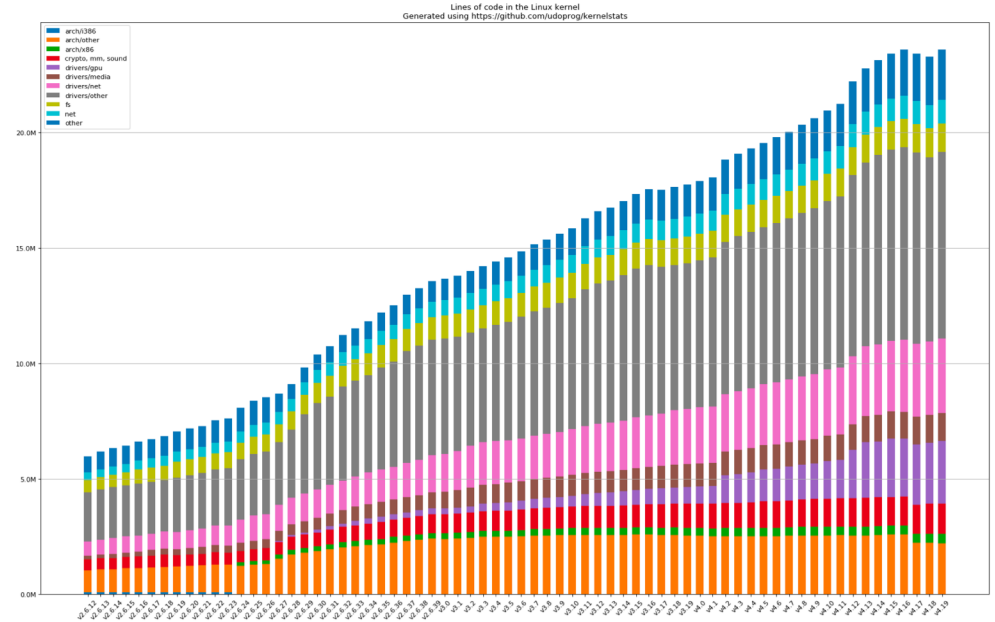
Code quality is essential

Today's software is essential.
Hidden errors can deeply impact business and cost millions.

Equifax – SSNs of 143 Million Americans were stolen.
45% of US Population!

Source: MarketWatch
https://www.marketwatch.com/story/the-equifax-data-breach-in-one-chart-2018-09-07

Codebases are large and complex.
Critical errors can span multiple files and functions. They often go undetected.



Source: https://github.com/udoprog/kernelstats

muse dev

What Can Help?

1. Applying More Tools *During* Development

2. Social / Cultural Processes To Drive Adoption

The focus of this talk!

Code Quality Best Practices

Revision Control (hg, git, svn)

Testing (unit tests, integration tests, UI tests, pen. tests)

Code Review (Pull Requests, Phabricator, Gerrit)

Static Analysis (main focus of this talk!)

Instrumentation / Monitoring

Traditionally
in this order

Code Quality Best Practices

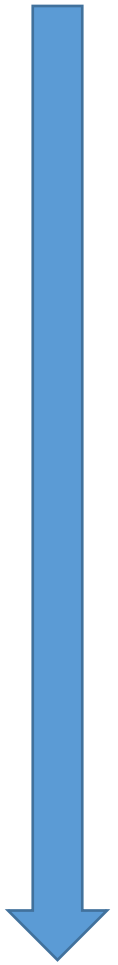
Revision Control (hg, git, svn)

Testing (unit tests, integration tests, UI tests, pen. tests)

Code Review (Pull Requests, Phabricator, Gerrit)

Static Analysis (main focus of this talk!)

Instrumentation / Monitoring



Traditionally
in this order

Code Quality Best Practices

Revision Control (hg, git, svn)

Testing (unit tests, integration tests, UI tests, pen. tests)

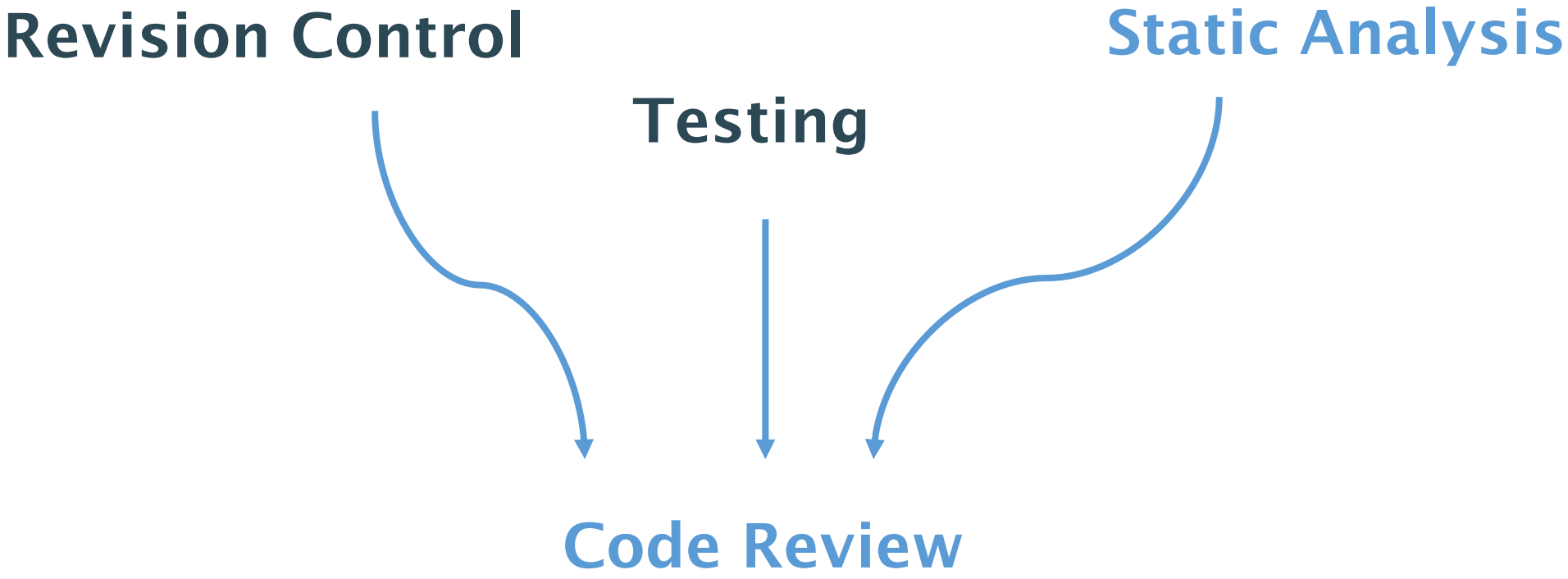
Code Review (Pull Requests, Phabricator, Gerrit)

Static Analysis (main focus of this talk!)

Instrumentation / Monitoring

New Trend

Code Quality Best Practices



Code Quality Best Practices

Revision Control

Testing

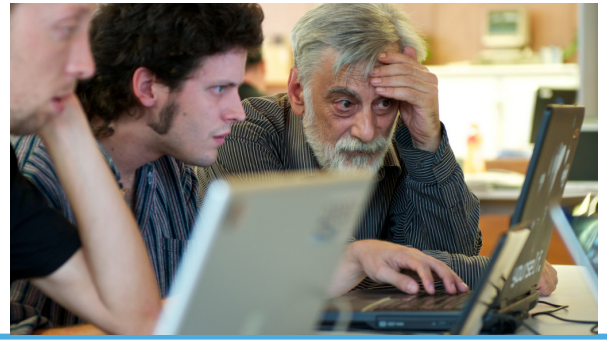
Static Analysis

Code Quality Best Practices

Revision Control

Testing

Static Analysis



Code Quality Best Practices

Revision Control

Testing

Static Analysis

Static Analysis

Techniques span a spectrum

"shallow" analyses **(seconds)**

(minutes)

"deep" analyses (hours)

Static Analysis

What range of techniques / tools exist?

How are they being applied by large tech companies?

First Explain…

What do you mean by *static analysis*?

Static Analysis

Say something about the behavior of the program

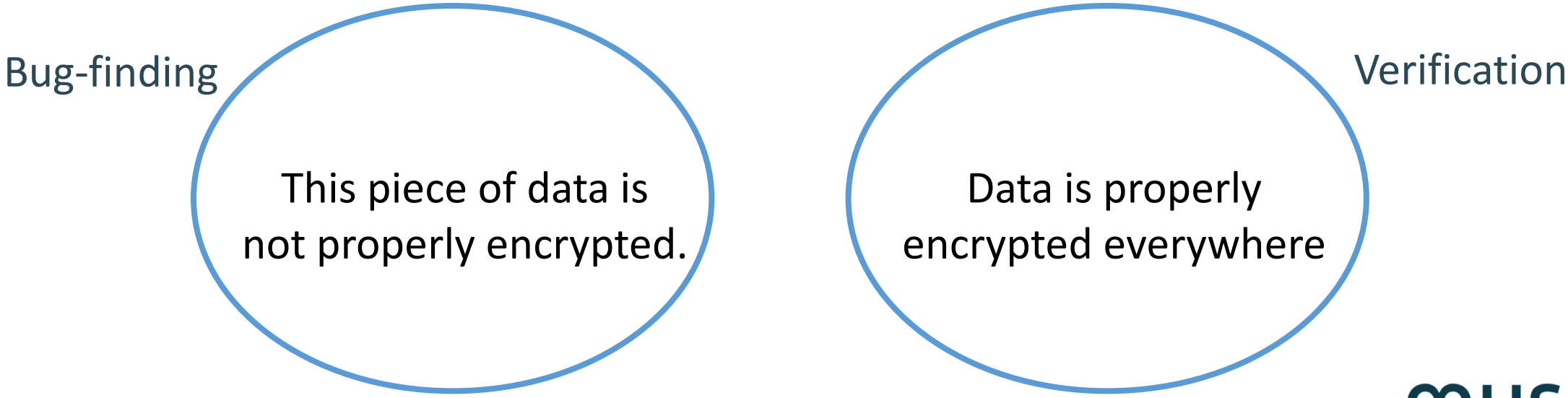
without running the program.

This piece of data is
not properly encrypted.

Data is properly
encrypted everywhere

Static Analysis

Say something about the behavior of the program without running the program.



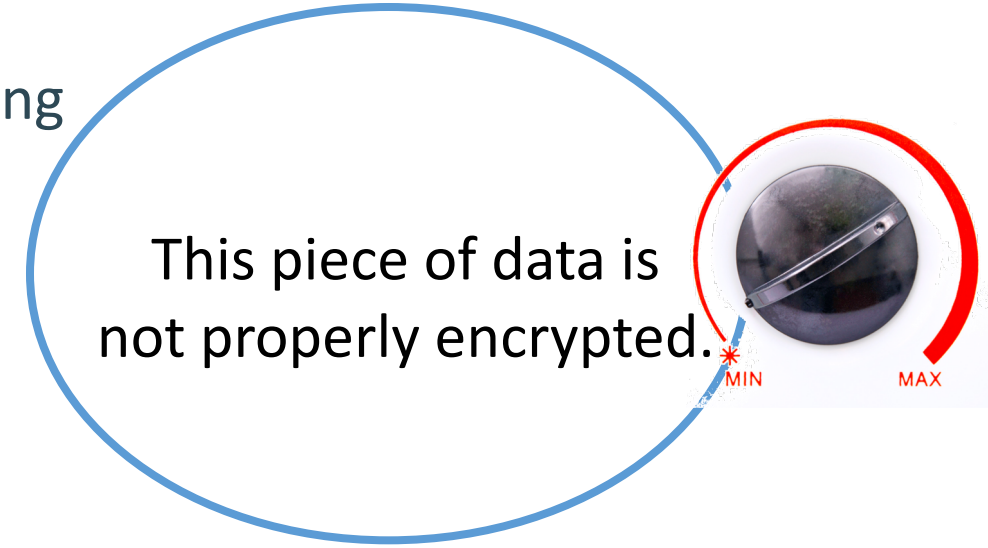
Static Analysis

Say something about the behavior of the program without running the program.

Static Analysis

Say something about the behavior of the program without running the program.

Bug-finding



Static Analysis

Say something about the behavior of the program

without running the program.

Security	Readability	Reliability
Performance	Maintainability	Correctness

What is Static Analysis?

Simplest Example: **String Search**

```
init_connection(3DES, …)
```

Code does not contain
`“init_connection(3DES”`
(maybe some allowance for white space)

What is Static Analysis?

Simplest Example: **Grep**

```
init_connection(3DES, …)
```

Code does not contain
`“init_connection(3DES”`
(maybe some allowance for white space)

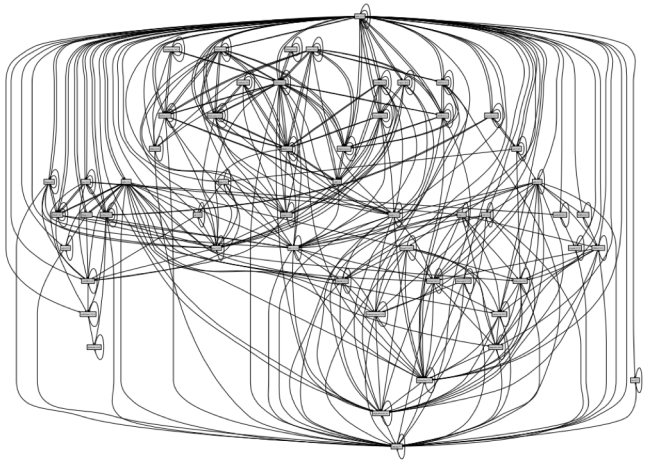
What is Static Analysis?

Simplest Example: **Grep**

```
init_connection(3DES, …)
```

Code does not contain
“`init_connection(3DES`”
(maybe some allowance for white space)

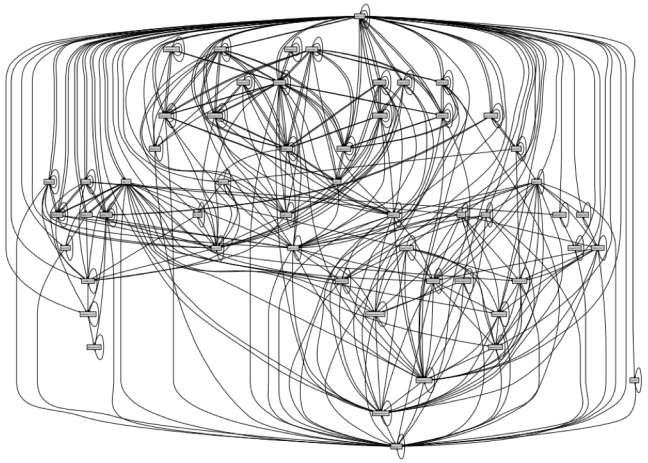
More Advanced: **Graph Analysis**



The value `3DES` does not flow to the `init_connection` method.

What is Static Analysis?

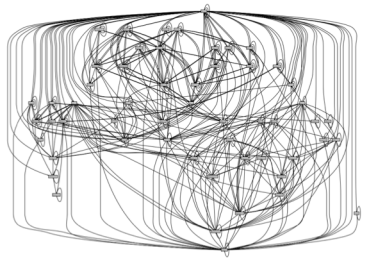
More Advanced: **Graph Analysis**



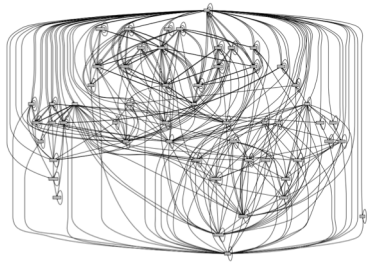
The value `3DES` does not flow to the `init_connection` method.

More Advanced Still:
Compositional Program Analysis

UI Thread



Network Thread



The UI thread and Network thread are properly synchronized.

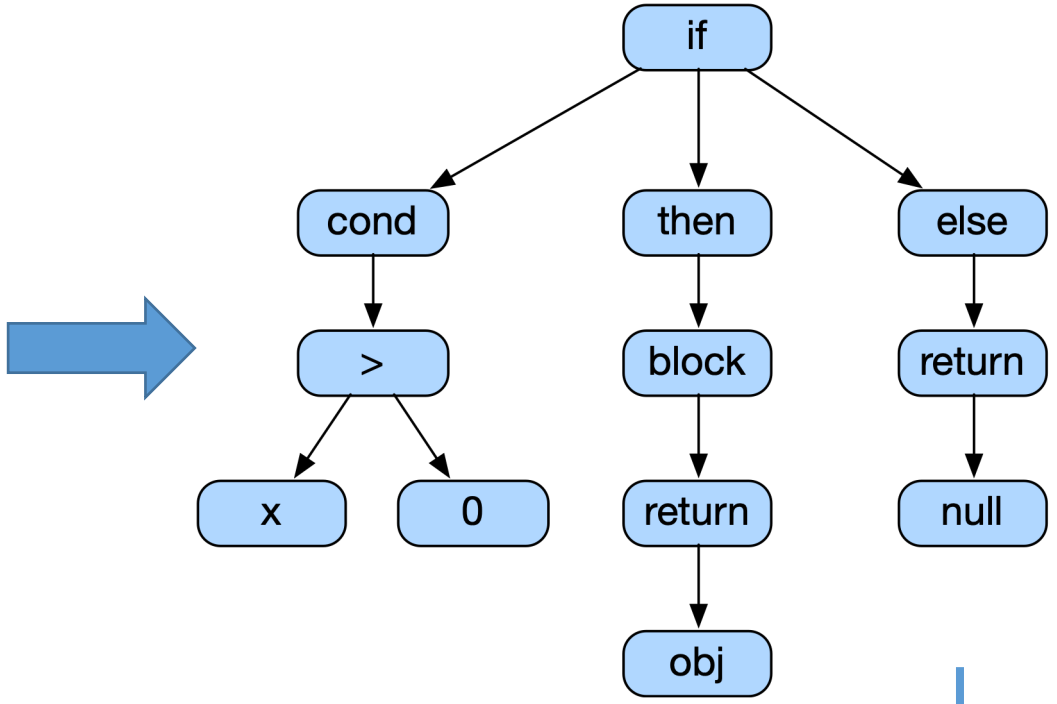
What is Static Analysis?

Simple “Shallow” Analyses	More Advanced “Deep” Analyses
Simple API misuse	Memory Errors
Deprecated APIs	Thread Safety Problems
Leaked Authentication Tokens	Performance Issues

“Shallow” example (PMD Analyzer)

1. Construct abstract syntax tree

```
if(x > 0) {
  return obj;
}
else
  return null;
```



”grep”

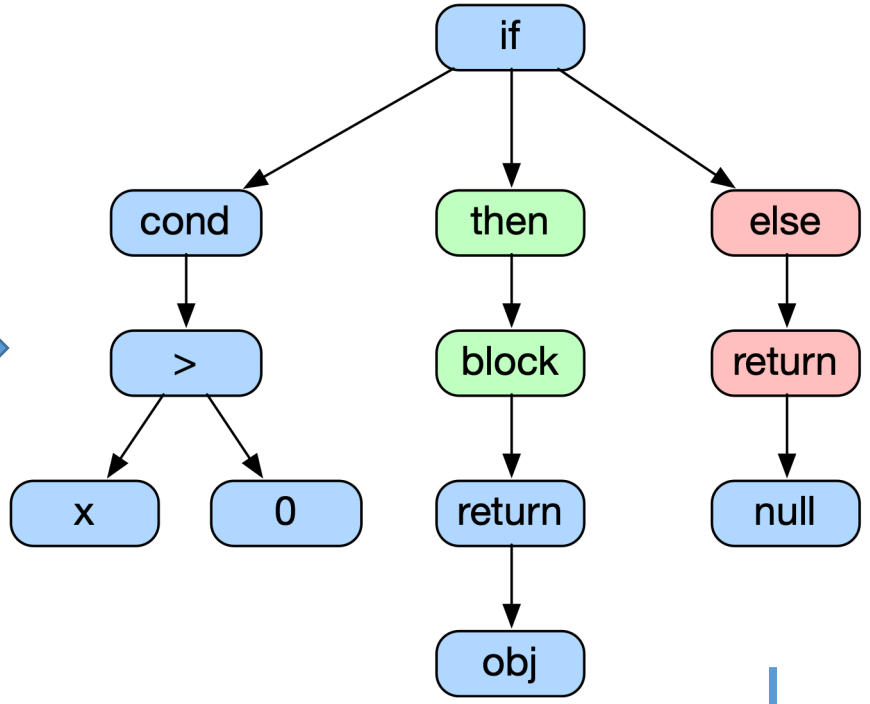
PMD

“Shallow” example (PMD Analyzer)

2. Pattern match on tree

Find “if” stmt
Check “then” branch contains “block”
Check “else” branch contains “block”

“If statements must use braces”



”grep”

PMD

“Shallow” example (PMD Analyzer)

Can find:

- style violations
- local bug patterns

Examples:

- OverrideBothEqualsAndHashcode
- CloneMethodMustBePublic
- UseEqualsToCompareStrings

”grep”

PMD

“Shallow” example (PMD Analyzer)

“UseEqualsToCompareStrings”

```
if(com == “end”)
```

Bad

```
if(“end”.equals(com))
```

Good

”grep”

PMD

“Shallow” example (PMD Analyzer)

“UseEqualsToCompareStrings”

```
void equalsIgnoreCase(String x, String y) {
  return x.toLowerCase() == y.toLowerCase();
```

Not Flagged

”grep”

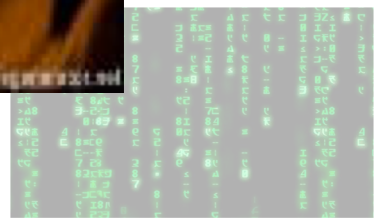
PMD

“Shallow” example (PMD Analyzer)

UseEquals

”grep”

PMD



“Medium” example (ErrorProne Analyzer)

“StringEquality” rule

```
void equalsIgnoreCase(String x, String y) {
  return x.toLowerCase() == y.toLowerCase();
```

Bad

”grep” PMD ErrorProne

“Medium” example (ErrorProne Analyzer)

1. Construct abstract syntax tree
2. Construct symbol table
3. Resolve names
4. Propagate types
5. Perform dataflow analysis
6. Check for errors

Built into the compiler
Inherits compiler analysis passes

(same approach taken by Clang static analyzer)

”grep” PMD ErrorProne

“Medium” example (ErrorProne Analyzer)

1. Construct abstract syntax tree
2. Construct symbol table
3. Resolve names
4. Propagate types
5. Perform dataflow analysis
6. Check for errors.

Built into the compiler
Inherits compiler analysis passes

```
void equalsIgnoreCase(String x, String y) {
  return x.toLowerCase() == y.toLowerCase();
```

”grep” PMD ErrorProne

“Medium” example (ErrorProne Analyzer)

“NullTernary” rule

```
int x = flag ? foo : null;
```

”grep” PMD ErrorProne

“Medium” example (ErrorProne Analyzer)

“NullTernary” rule

```
Integer y = null;
int x = flag ? foo : y;
```

”grep” PMD ErrorProne

“Medium” example (ErrorProne Analyzer)

“NullTernary” rule

```
Integer y = someFnThatMayReturnNull();
int x = flag ? foo : y;
```

”grep” PMD ErrorProne

“Medium” example (ErrorProne Analyzer)

“NullTerr

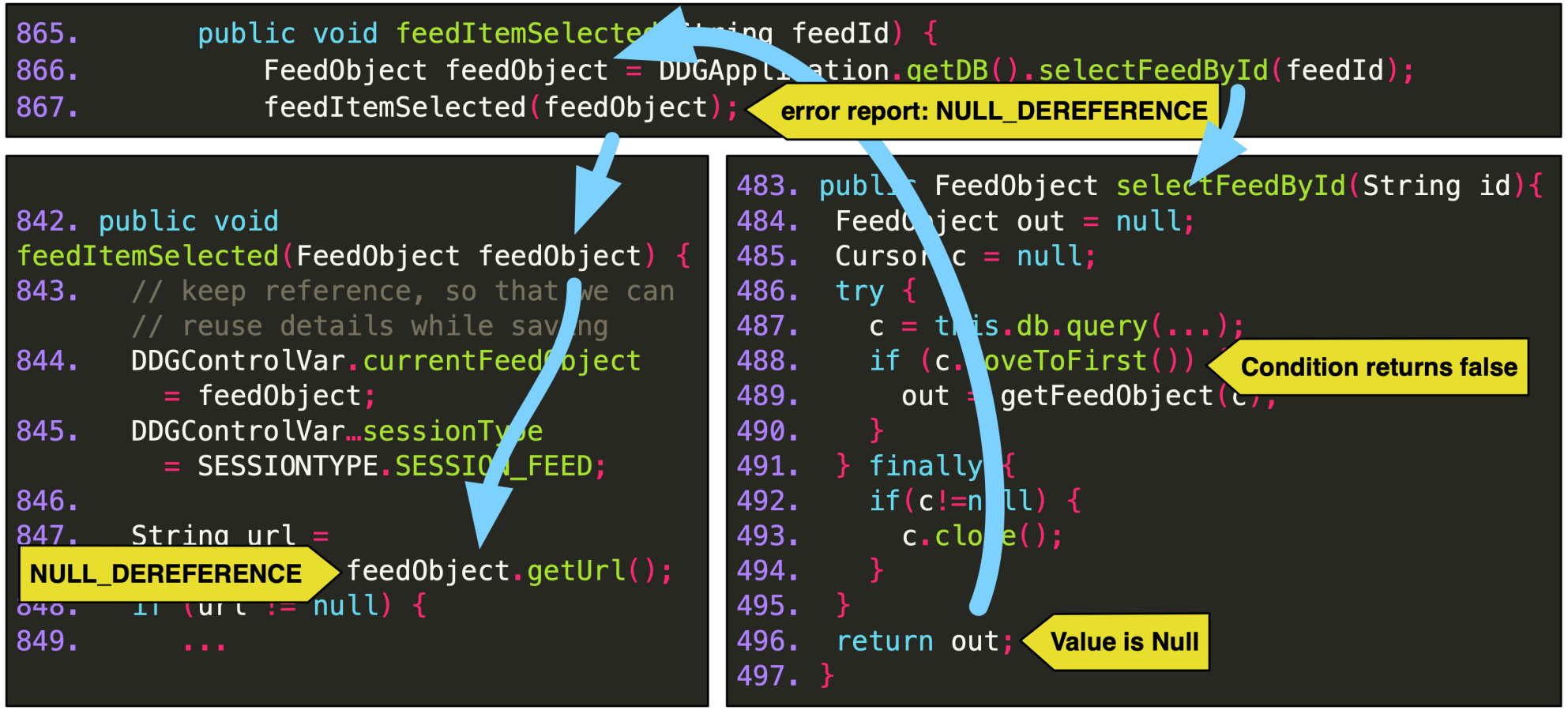
”grep”

“Deep” example (Infer)

Goals:

- Catch important errors that programmers have trouble finding.
- Reason inter-procedurally.

“Deep” example (Infer)



“Deep” example (Infer)

Approach:

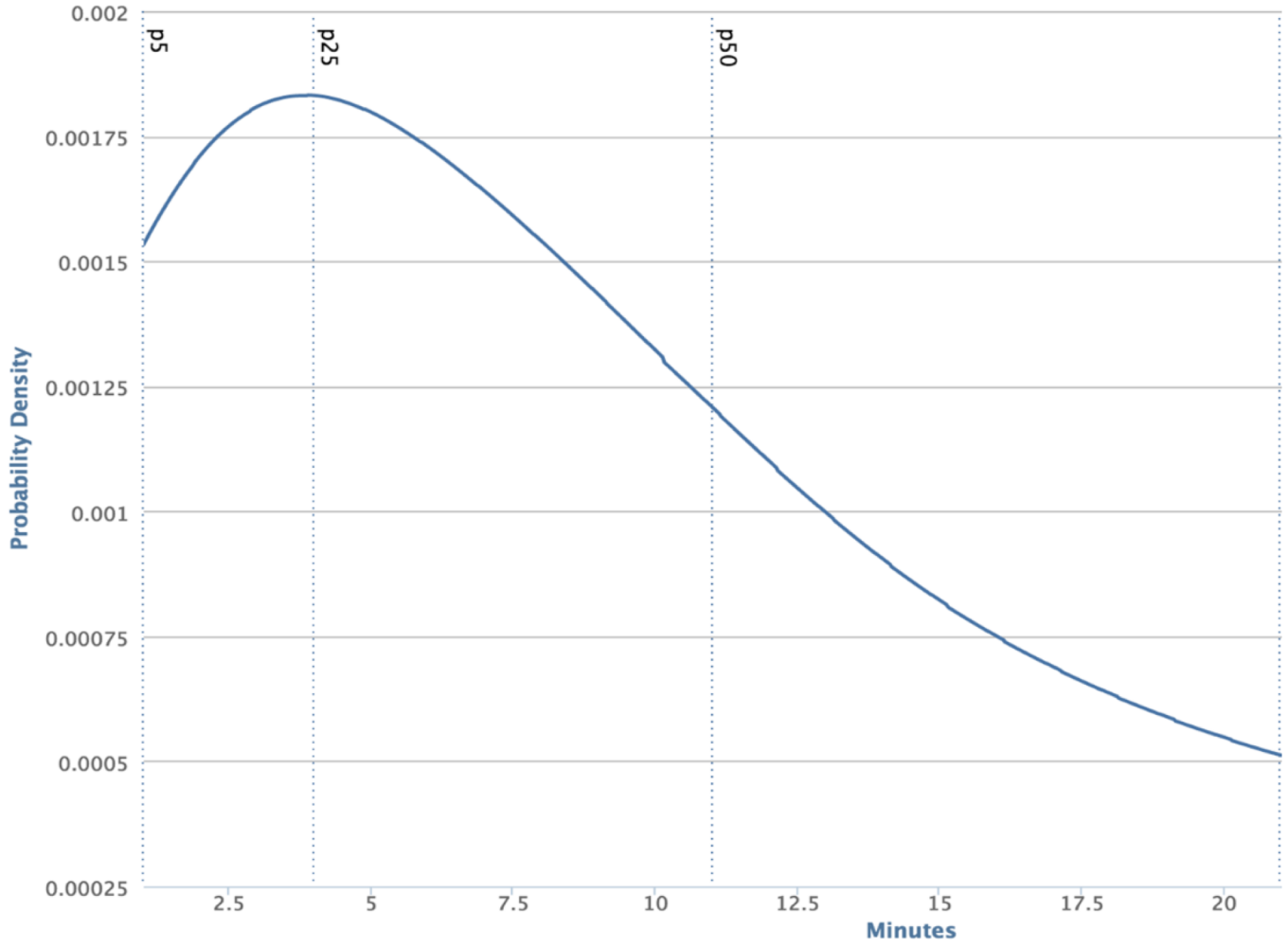
- For each procedure
  - Perform a **problem-specific analysis**

    (track null assignments)
  - Store a summary that captures info for that procedure

    (nullability of arguments / return)
  - At procedure calls, use or generate the summary

”grep” PMD ErrorProne Infer

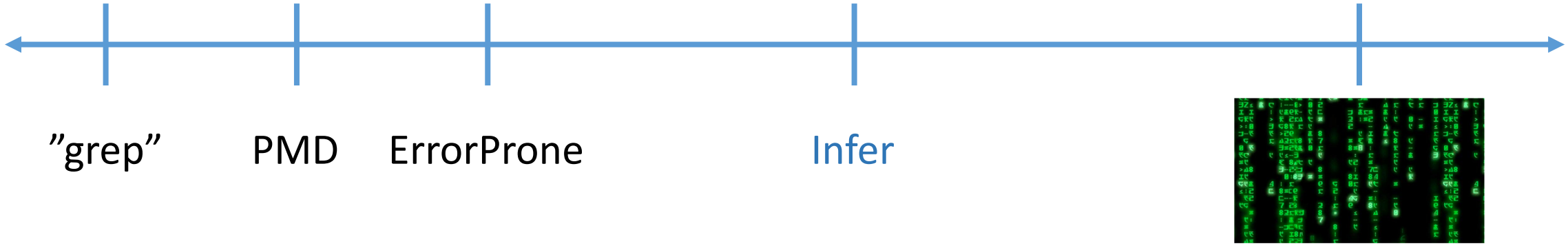
Extremely Scalable (Infer)



“Deep” example (Infer)

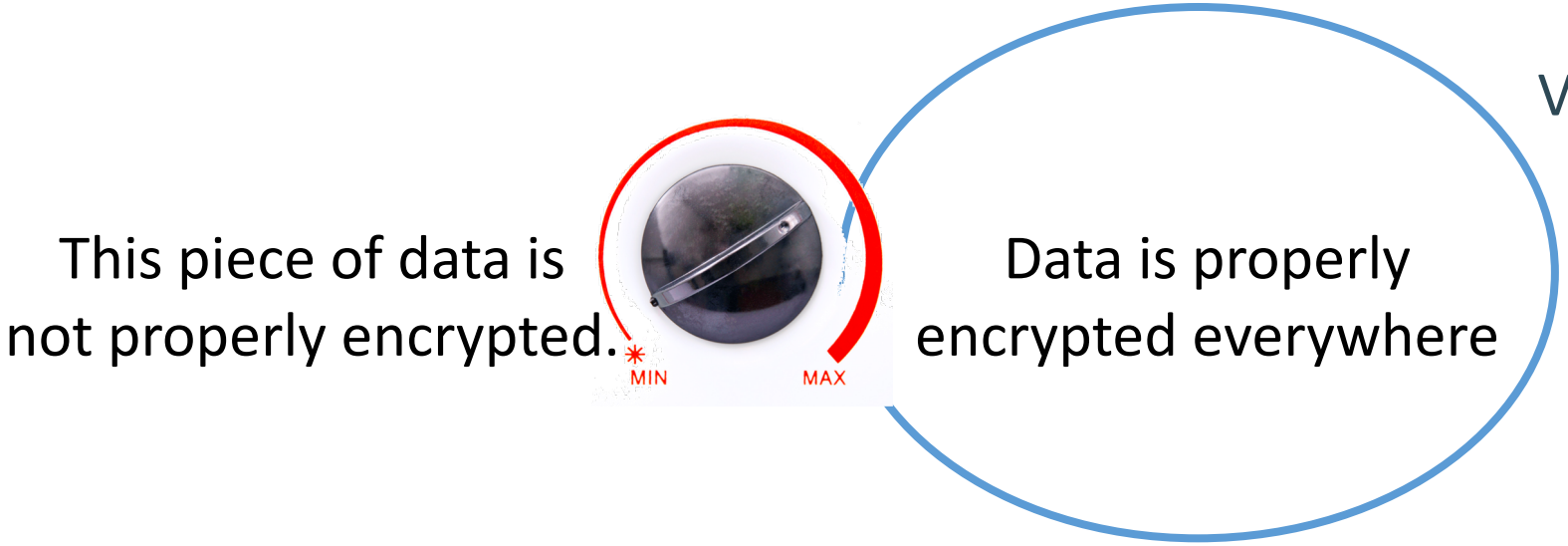
Can also handle:

- Safe use of multi-threading
- Array bounds checks
- Performance analysis



Getting to verification (NullAway)

With the right annotations, we can be sure the property holds.

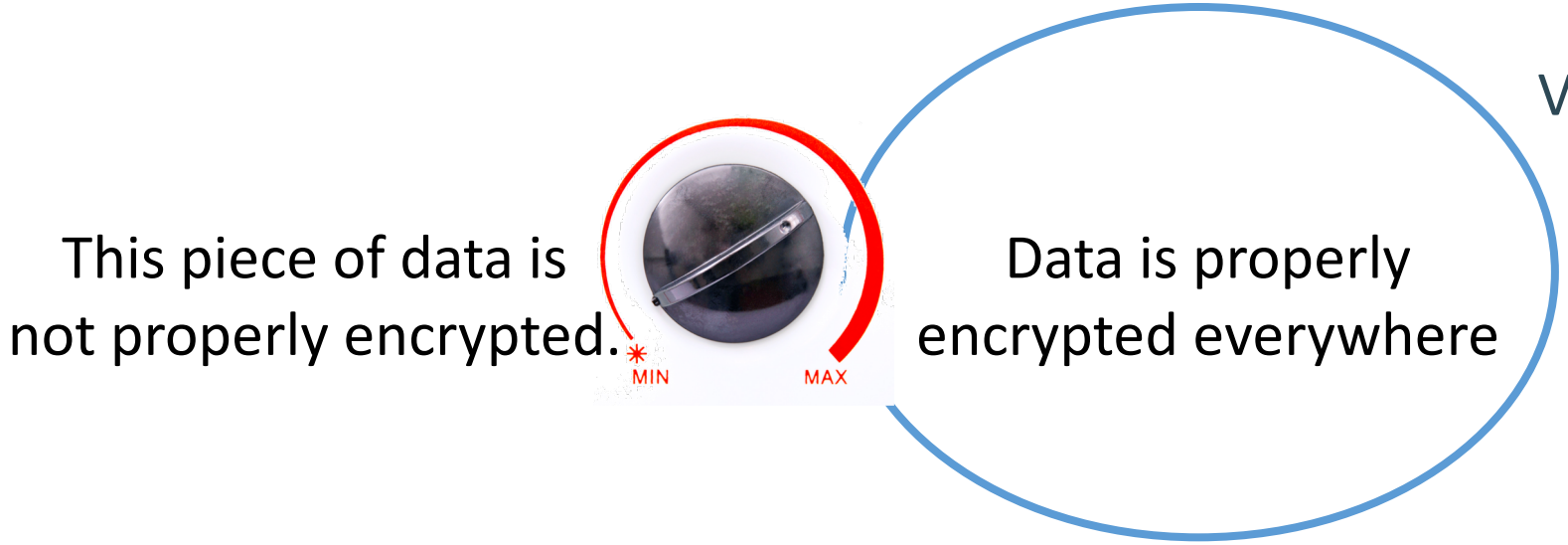


Getting to verification (NullAway)

`@Nullable`
`@NotNull`

Check that all @NotNull variables are assigned @NotNull results.

Shifts burden from analysis tool to developer, but increases confidence.



What enables this?

Successful transition of academic research results to industry.

(via transition of researchers to industry)

How Well Does It Work?

Google

"As of January 2018, Tricorder had analyzed approximately 50,000 code review changes per day." Of the more than 5,000 code review reports per day, 95% were deemed "useful" by developers.

Sadowski, Caitlin, et al. "Lessons from Building Static Analysis Tools at Google." *Communications of the ACM* 61.4: 58-66.

facebook

"RacerD has been running in production for 10 months on our Android codebase and has caught over 1000 multi-threading issues which have been fixed by Facebook developers before the code reaches production."

Sam Blackshear, Peter O'Hearn. "Open-sourcing RacerD: Fast static race detection at scale". https://code.fb.com/android/open-sourcing-racerd-fast-static-race-detection-at-scale/

Integration is Key

Even simple checks have required analysis infrastructure supporting workflow integration to make them successful… Initially, in 2006, FindBugs was integrated as a centralized tool that ran nightly… Although FindBugs found hundreds of bugs in Google's Java codebase, the dashboard saw little use because a bug dashboard was outside the developers' usual workflow.

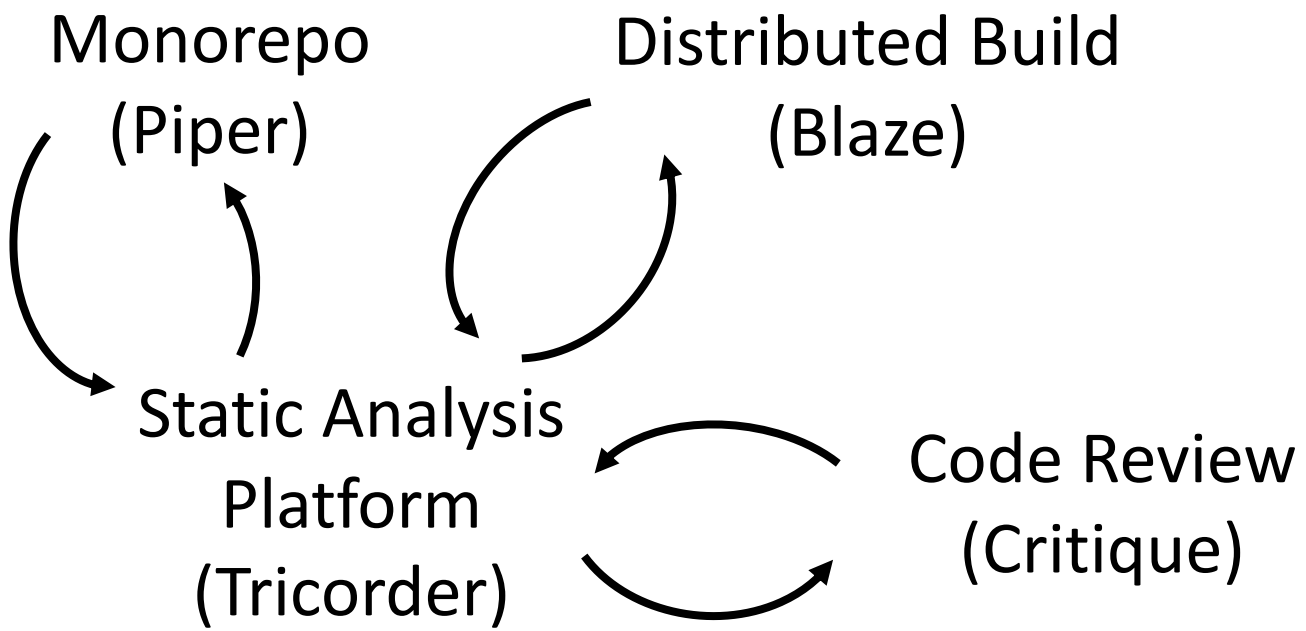
Sadowski, Caitlin, et al. "Lessons from Building Static Analysis Tools at Google." *Communications of the ACM* 61.4: 58-66.

Fix rate for reports went from almost 0% to 70% following code review integration.

Peter W. O'Hearn. 2018. Continuous Reasoning: Scaling the impact of formal methods. Symposium on Logic in Computer Science (LICS '18).

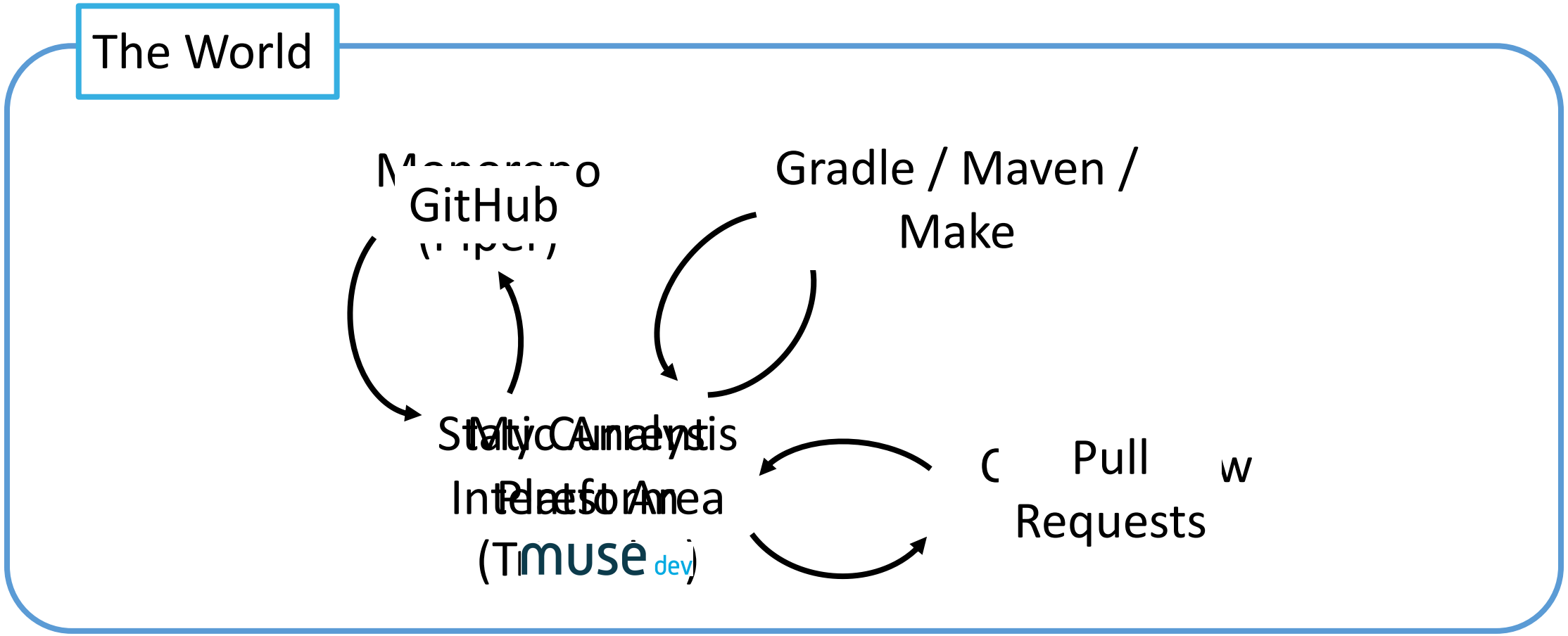
Making It Effective
Integration is Key

Google



Sadowski, Caitlin, et al. "Tricorder: Building a program analysis ecosystem." *Proceedings of the 37th International Conference on Software Engineering-Volume 1*. IEEE Press, 2015.

Making It Effective
Integration is Key



Sadowski, Caitlin, et al. "Tricorder: Building a program analysis ecosystem." *Proceedings of the 37th International Conference on Software Engineering-Volume 1*. IEEE Press, 2015.

Supported Integration is Key

Lots of open-source tools that deliver substantial value, if

- Configured properly
- Integrated into developer workflow
- Kept up-to-date

Try It Out

PMD: https://pmd.github.io/
Error Prone: http://errorprone.info/
Infer: https://fbinfer.com/
NullAway: https://github.com/uber/NullAway

Muse Dev Private Beta
https://muse.dev

muse dev