# End to End Quality
## with the
# Sonar Ecosystem
## and the
# Water Leak Metaphor

G. Ann Campbell

*@GAnnCampbell | ann.campbell@SonarSource.com*
*@SonarLint | @SonarQube | @SonarSource*

sonarsource

# SonarLint
# Leak Period
# Quality Gate

sonarsource

# 20+ Languages


sonarsource

# The <3 of the ecosystem

Static Analysis

# What is Static Analysis?

Analyzing code,

*without executing it!*

# A Means to an End

Detecting Bugs, Vulnerabilities, and Code Smells

# Why use Static Analysis

## Catch new problems ASAP

- the longer it takes to catch a bug, the more it costs
- no one writes perfect code *every time*
- rule description and precise issue location cut research time

# Why use Static Analysis

## Changing A might have added bugs in B

- peer review misses new issues in untouched code
- static analysis is machine-assisted code review; it looks at every file every time

# Why use Static Analysis

## Provide coaching

- language best practices
- team coding style

# SonarSource’s Toolbox


sonarsource

# Lexical Analysis

Only two things are infinite, the universe and human

stupidity, and I am not sure about the former.

# Syntactic Analysis

Only two things are infinite, the universe and human stupidity, and I am not sure about the former.

Subjects

Verbs

Albert E.

# Semantic Analysis

Only two things are infinite, the universe and human stupidity, and I am not sure about the former.

Albert E.

sonarsource

# Semantic Analysis

Only two things are infinite, the universe and human stupidity, and I am not sure about the former.

Albert E.

sonarsource

# Beyond Semantic: Symbolic Execution

```
Object myObject = new Object();



if(a) {  myObject = null; }
...
if( !a ) { ... }
else { myObject.toString(); }
```

sonarsource

# Beyond Semantic: Symbolic Execution

```
Object myObject = new Object();



if(a) {  myObject = null; }
...
if( !a ) { ... }
else { myObject.toString(); } //NPE
```

sonarsource

# Beyond Semantic: Symbolic Execution

```
Object myObject = new Object();
```

*Program State#0*
myObject != null

```
if(a) {  myObject = null; }
...
if( !a ) { ... }
else { myObject.toString(); } //NPE
```

sonarsource

# Beyond Semantic: Symbolic Execution

```
Object myObject = new Object();
```


```
if(a) {  myObject = null; }
...
if( !a ) { ... }
else { myObject.toString(); } //NPE
```

sonarsource

# Beyond Semantic: Symbolic Execution

*Program State#1*
myObject != null
a = false

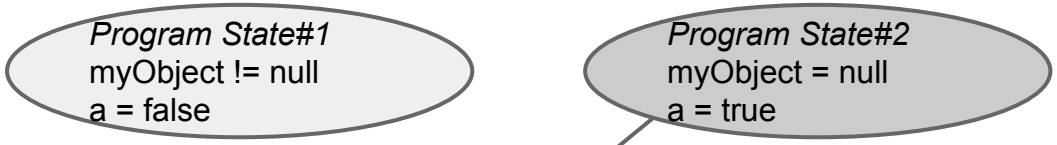
*Program State#2*
myObject = null
a = true

```
...
if( !a ) {
    ...
} else {

    myObject.toString(); // NPE
}
```

sonarsource

# Beyond Semantic: Symbolic Execution

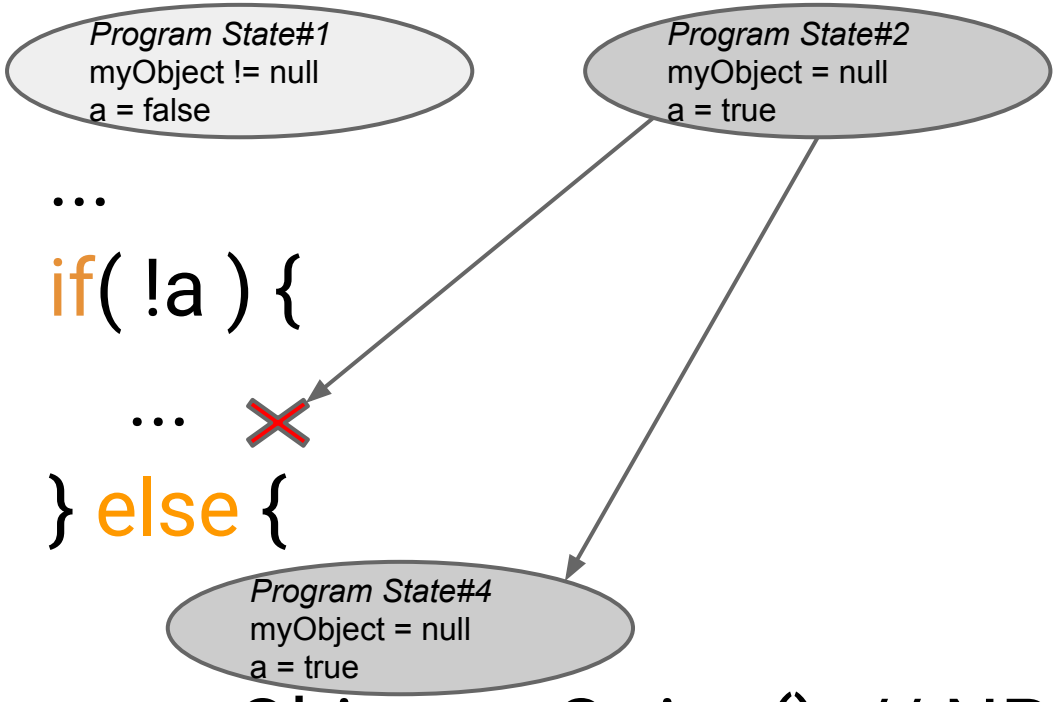



```
...
if( !a ) {
    ...
} else {

    myObject.toString(); // NPE
}
```

sonarsource

# Beyond Semantic: Symbolic Execution




sonarsource

# Cross-Procedural Analysis


# What is Static Analysis ?

Analyzing code,

*without executing it.*

*by (symbolically) executing*

*all possible paths!*

sonarsource

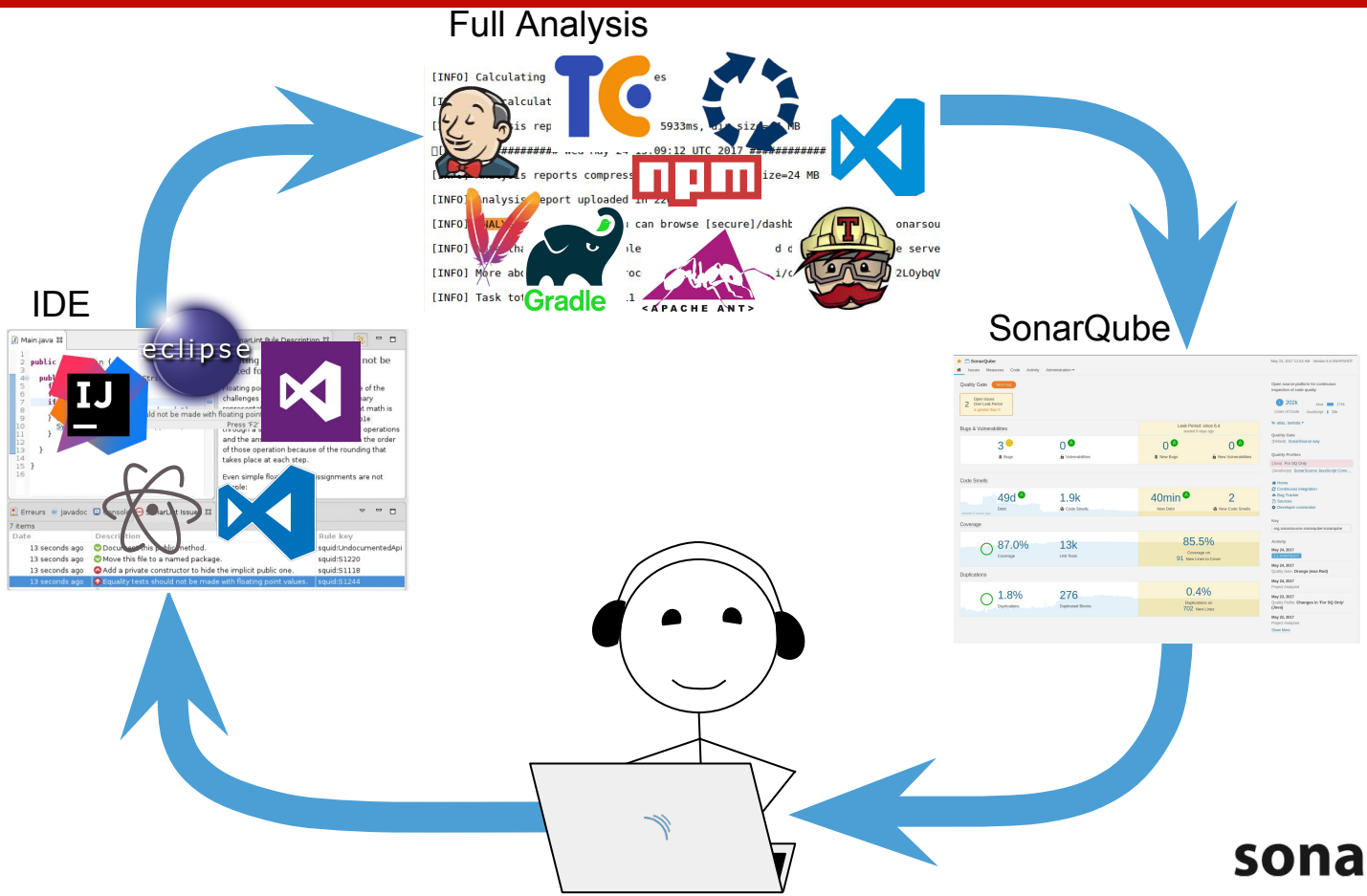
# Symbolic Execution Almost Everywhere

- SonarAnalyzers for C#, C/C++, Java, and JS
  - Dereferences of Null Pointers
  - Unconditionally True/False (sub)conditions
  - Division by zero
  - Resource leaks
    - Unclosed resources (Java)
    - Unreleased memory (C/C++)
  - Double free (C/C++)

sonarsource™

Fewer slides, more code!

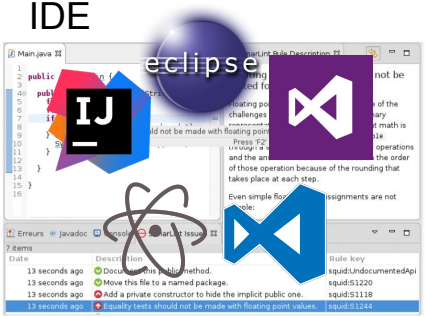
# Full Cycle



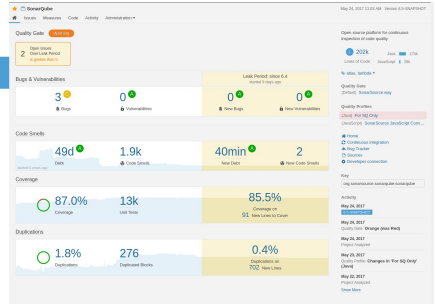

sonarsource

# Full Cycle

IDE




SonarQube




sonarsource

# Fix the Leak

- ✓ SonarLint
- Leak Period
- Quality Gate

sonarsource

# Reimbursing the Debt


sonarsource

# This is Hard

- Total amount of Technical Debt can be depressing
- How to get a budget to fix old Technical Debt?
- Risk of injecting functional regression
- This is not fun!

sonarsource

# Project Homepage

SonarQube

August 30, 2017, 1

Overview Issues Measures Code Activity Administration

Quality Gate Failed

83.6% Coverage on New Code is less than 85.0%

C Reliability Rating on New Code is worse than A

Bugs Vulnerabilities

Leak Period: since 6.5
started 2 months ago

8 C Bugs

0 A Vulnerabilities

5 C New Bugs

0 A New Vulnerabilities

Code Smells

74d A Debt
started 6 years ago

2.5k Code Smells

25d A New Debt

687 New Code Smells

Coverage

86.5% Coverage

14k Unit Tests

83.6% Coverage on 3.3k New Lines to Cover

Duplications

1.9% Duplications

1.2k Duplicated Blocks

0.0% Duplications on 44k New Lines

sonarsource

# Project Homepage: Leak Period

Leak Period: since 6.5
started 2 months ago

5 C
New Bugs

0 A
New Vulnerabilities

25d A
New Debt

687
New Code Smells

83.6%
Coverage on
3.3k New Lines to Cover

0.0%
Duplications on
44k New Lines

sonarsource

# Fix the Leak

- ✓ SonarLint
- ✓ Leak Period Quality Gate

sonarsource

# Quality Gate

SonarQube way

**Conditions**

Only project measures are checked against thresholds. Sub-projects, directories and files are ignored. More

| Metric | Over Leak Period | Operator | Warning | Error |
|---|---|---|---|---|
| Coverage on New Code | Always | is less than | | 80.0% |
| Maintainability Rating on New Code | Always | is worse than | | A |
| Reliability Rating on New Code | Always | is worse than | | A |
| Security Rating on New Code | Always | is worse than | | A |

sonarsource

# Project Homepage: Quality Gate


sonarsource

# Quality Gate

Quality Gate Passed

Quality Gate Warning

1 Skipped Unit Tests Over Leak Period
is greater than 0

Quality Gate Failed

83.6% Coverage on New Code
is less than 85.0%

C Reliability Rating on New Code
is worse than A

sonarsource

# Fix the Leak

- ✓ SonarLint
- ✓ Leak Period
- ✓ Quality Gate

sonarsource™

# Thanks!

G. Ann Campbell

*@GAnnCampbell | ann.campbell@SonarSource.com*
*@SonarLint | @SonarQube | @SonarSource*

sonarsource