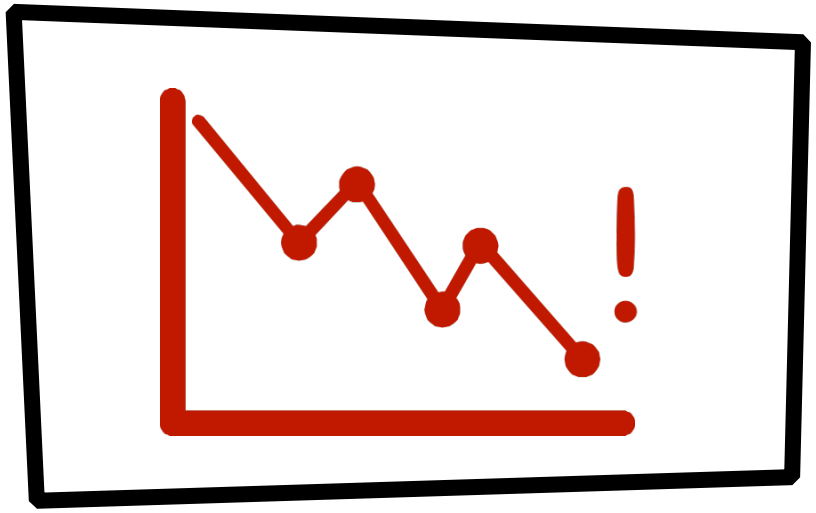
Shifting QA Left

Code Analysis at Google and Facebook

Stephen Magill

CEO, Muse Dev

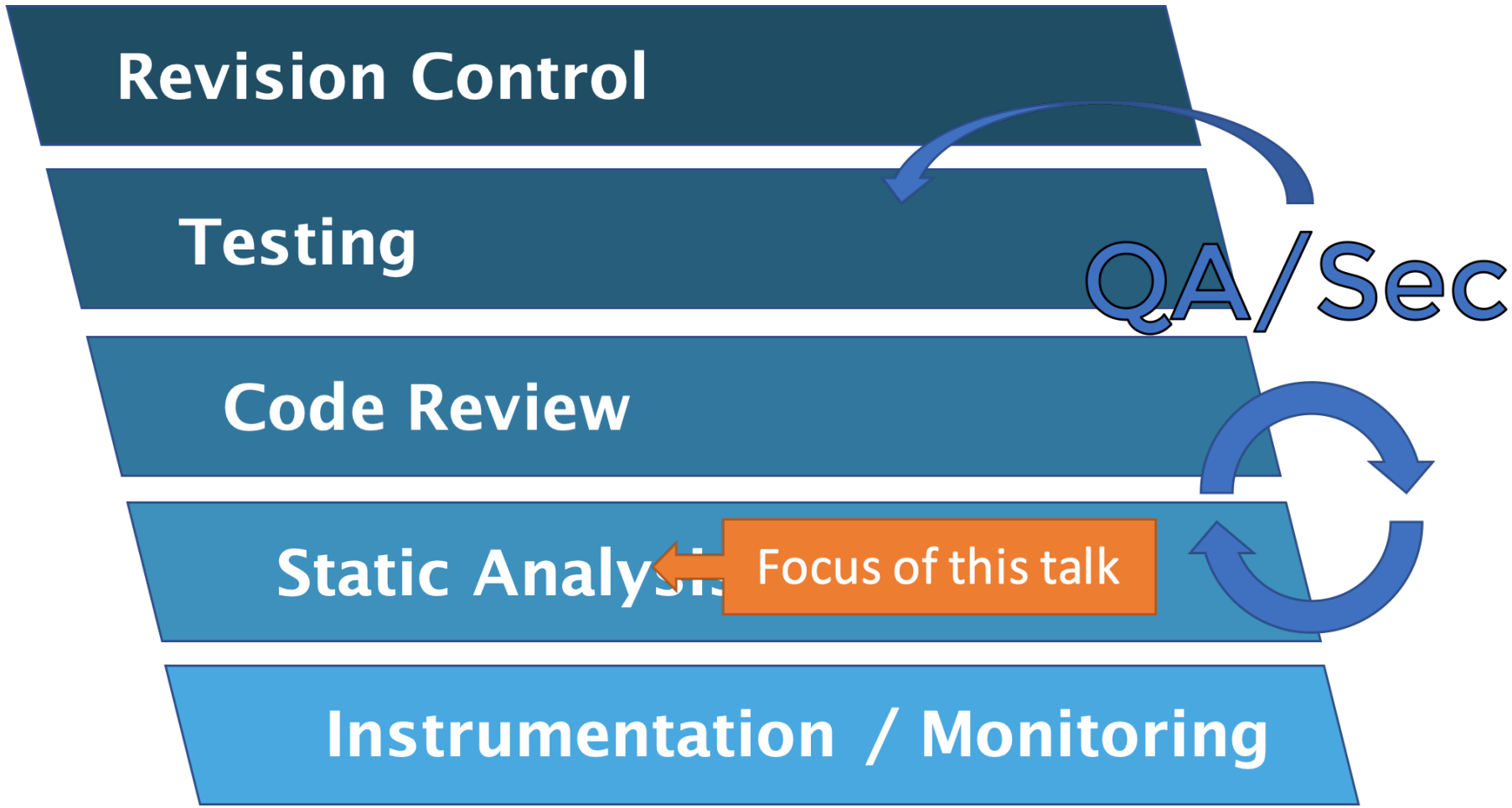
Slide on why code security and quality are important



Scary Graph

Scary Story

Code Quality Best Practices



MANY CHOICES

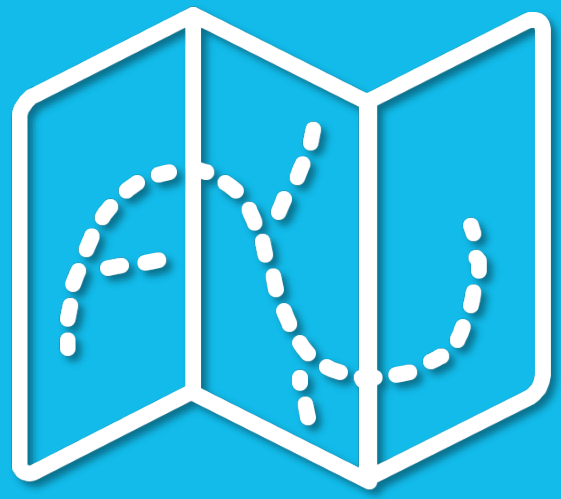
Some more effective than others

TODAY'S TALK

How to maximize outcomes
while minimizing process.

PRIMARY FOCUS

Orchestrating analysis
(vs. analysis itself)



WHAT ARE THE BIG TECH COMPANIES DOING

KEY PRINCIPLES

- Principle 1: Use Multiple Tools
- Principle 2: Integration Matters
- Principle 3: Cherish Developer Trust
- Principle 4: Tools Can Support Productivity

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.

Security	Readability	Reliability
Performance	Maintainability	Correctness

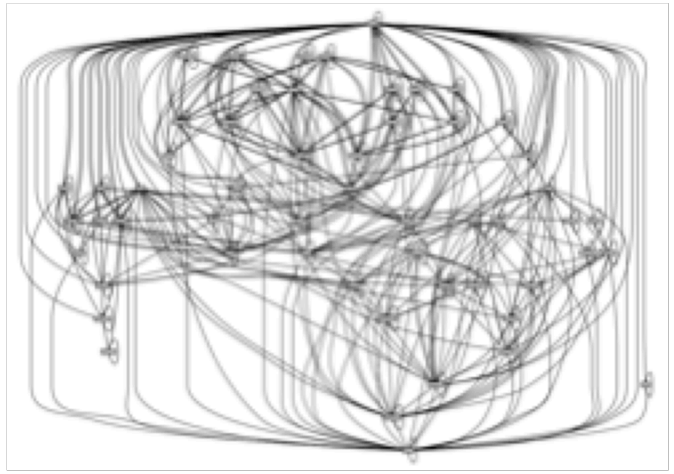
STATIC ANALYSIS

Simplest Example: **Grep**

```
init_connection(3DES, …)
```

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

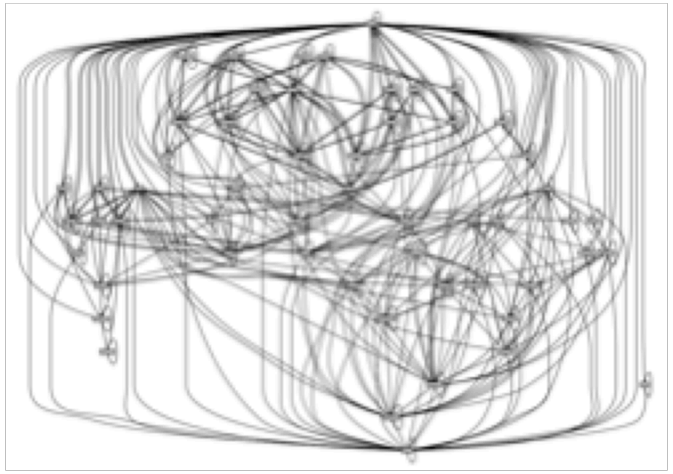
More Advanced: **Graph Analysis**



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

STATIC ANALYSIS

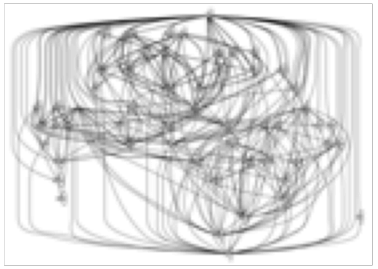
More Advanced: **Graph Analysis**



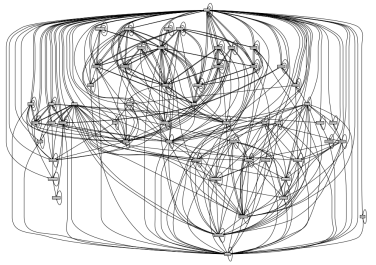
The value `3DES` does not flow to the `init_connection` method.

More Advanced Still: **Compositional Program Analysis**

UI Thread



Network Thread



The UI thread and Network thread are not properly synchronized.

STATIC ANALYSIS

Simple “Shallow” Analyses

Simple API misuse

Deprecated APIs

Leaked Authentication Tokens

More Advanced “Deep” Analyses

Memory Errors

Thread Safety Problems

Performance Issues

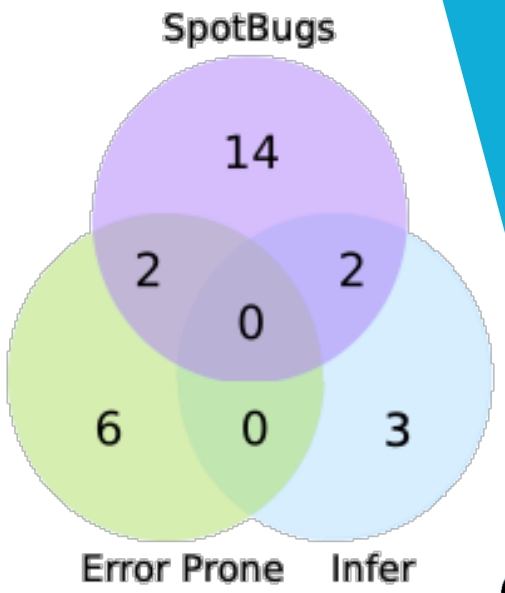
KEY PRINCIPLES

- Principle 1: Use Multiple Tools
- Principle 2: Integration Matters
- Principle 3: Cherish Developer Trust
- Principle 4: Tools Can Support Productivity

PRINCIPLE 1: USE MULTIPLE TOOLS

> Results also showed limited overlap between tool reports. The use of multiple tools can increase overall recall and boost confidence in overlapping results.

Delaitre, Aurelien M., et al. *SATE V Report: Ten Years of Static Analysis Tool Expositions*. No. Special Publication (NIST SP)-500-326. 2018.



Habib, Andrew, and Pradel, Michael. "How many of all bugs do we find? a study of static Bug detectors." *ASE*. 2018.

PRINCIPLE 1: USE MULTIPLE TOOLS

> As of January 2018, Tricorder included 146 analyzers, with 125 contributed from outside the Tricorder team...

Sadowski, Caitlin, Edward Aftandilian, Alex Eagle, Liam Miller-Cushon, and Ciera Jaspan. "Lessons from building static analysis tools at Google." (2018).

PRINCIPLE 1: USE MULTIPLE TOOLS

FROM GOOGLE

ErrorProne, ClangTidy

FROM FACEBOOK

Infer

FROM OSS COMMUNITY

PMD, SpotBugs, FindSecBugs,
Clang Static Analyzer

KEY PRINCIPLES

- Principle 1: Use Multiple Tools
- Principle 2: Integration Matters
- Principle 3: Cherish Developer Trust
- Principle 4: Tools Can Support Productivity

HOW TO DO STATIC ANALYSIS

1. Run static analysis tools and never look at the results.
2. Run static analysis tools and file bug reports that get ignored.
3. Evaluate QA on bugs fixed and developers on features shipped, setting up an epic ongoing battle.

4. Present results to developers **when they want to see them.**

TIMING MATTERS

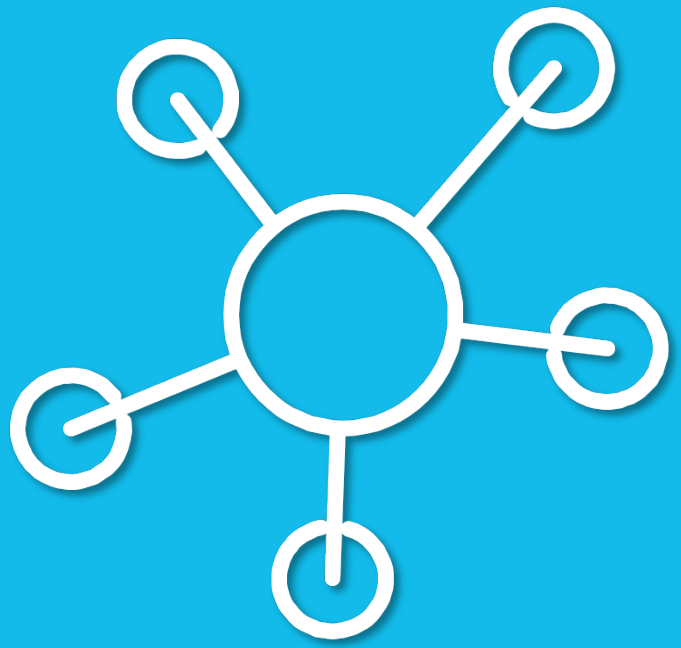
Best times to display results:

- Right after you wrote the code
- Right after you modified the code
- When you have the code “paged in”

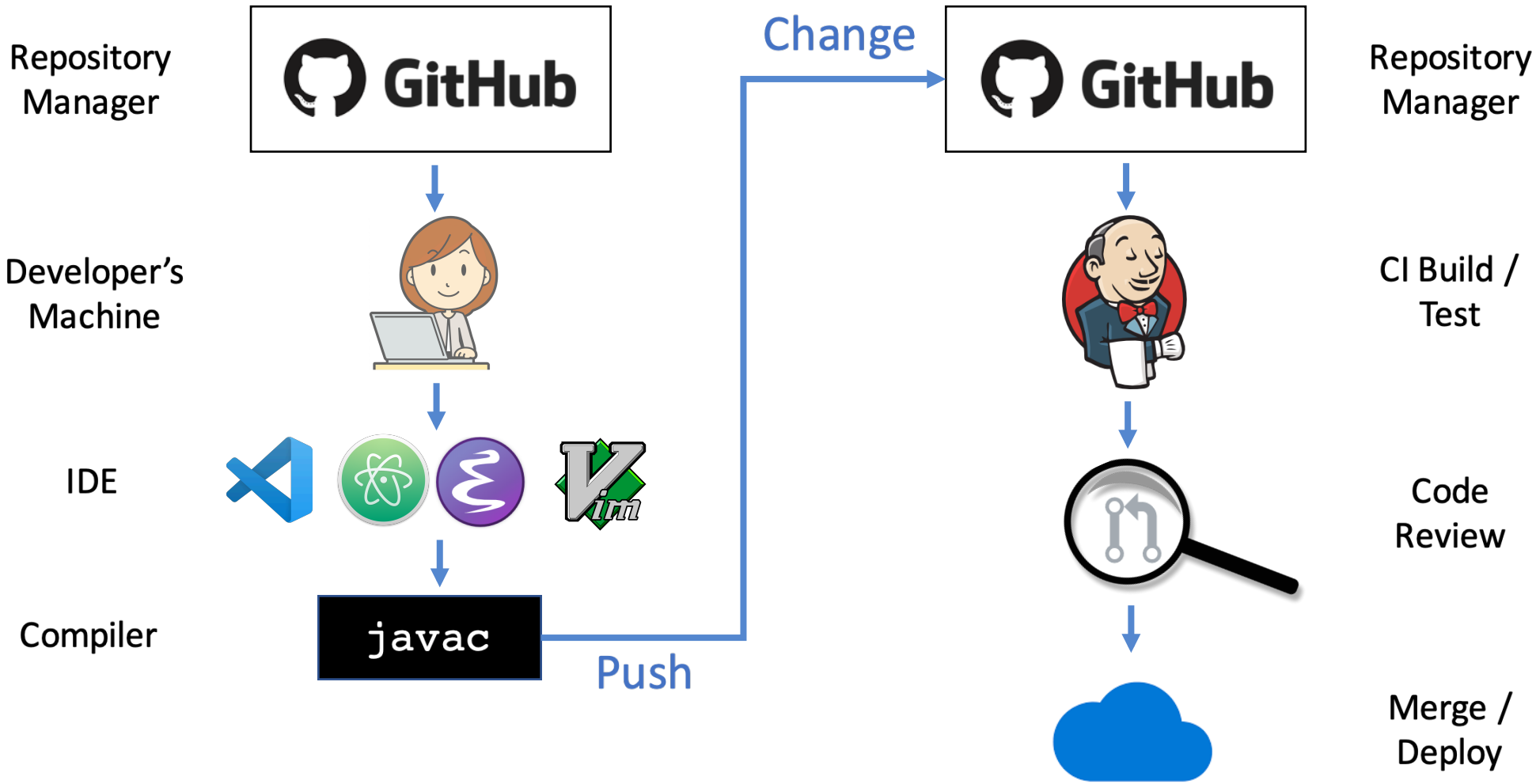
GOOD INTEGRATION

- Timely
- Part of an existing **developer** workflow
- Using existing **developer** tools

Developers are the ones who fix the bugs!



Use Existing Developer Workflow / Tools



FOLLOW ME, TO A PLACE OUTSIDE YOUR WORKFLOW...

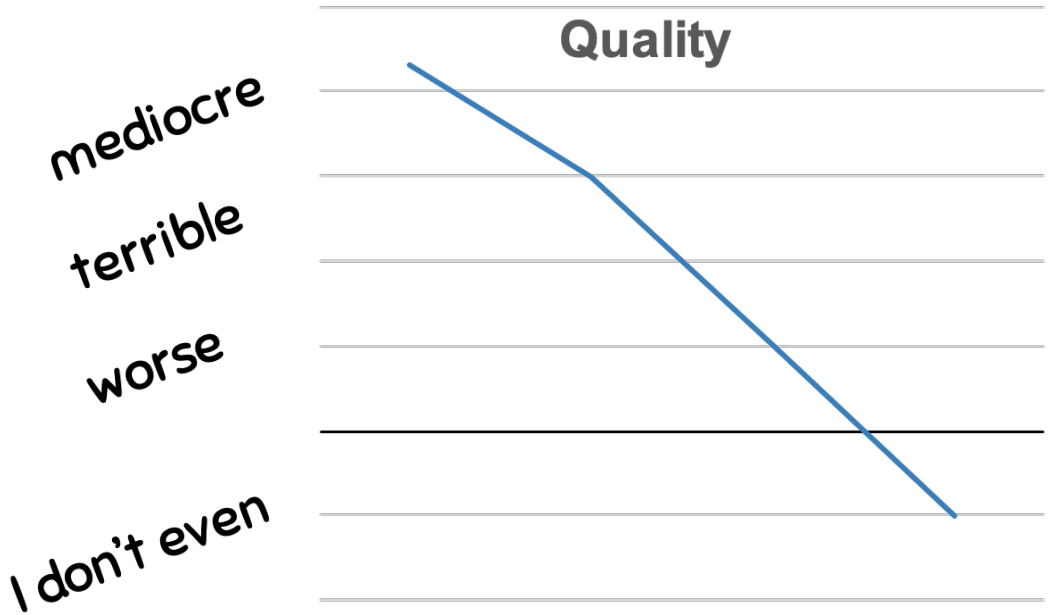
MY AWESOME BUG DASHBOARD

Error in src/villians/thanos.c, line 182

Didn't use `Ops::balance` correctly, click here for more info.

Error in src/utopias/brazil.c, line 8287

`CentralServices::repair` always results in an exception.



AND IT'S ALL JOE'S FAULT!

Performance improvement in RequestMappingInfo #22598

Open **aftersss** wants to merge 3 commits into `spring-projects:master` from `aftersss:master`

Conversation 6 | Commits 3 | Checks 0 | Files changed 3 | +45 −12

aftersss commented 7 days ago • edited — Contributor

We are using `spring-cloud-gateway` integrated with `spring-boot-starter-actuator`, we run a load test on it(using jmeter, and my gateway program runs on a linux server with 4 cores/2GB) and found that `RequestMappingInfo` have some performance problem.

Scene 1: We added a `Http Header Manager` into `jmeter` with the header `Accept:text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8`,(default value of chrome)

then I run jmeter and let my gateway program run to 2000 qps

cpu usage is about 60%

```
procs -----------memory---------- ---swap-- -----io---- -s
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 9  0      0  88152      0 737520    0    0     0     0 13309 12464 58  6 36  0  0
 5  0      0  88216      0 737360    0    0     0     0 12554 10777 53  5 43  0  0
 5  0      0  87228      0 737296    0    0     0     0 11844 10975 46  4 50  0  0
 7  0      0  87840      0 737360    0    0     0     0 11759 10714 51  5 45  0  0
```

Reviewers

No reviews

Assignees

rstoyanchev

Labels

in: web

type: enhancement

Projects

None yet

```
10  0      0  87592      0 737296    0    0     0    20 13214 11110 54  5 41  0  0
 5  0      0  87468      0 737396    0    0     0     0 13202 9999 62  6 32  0  0
```

Scene 2: We added a `Http Header Manager` int , then I run jmeter and let my gateway program run to 2000 qps, t

cpu usage is about 35%

```
procs -----------memory---------- ---swap-                    u-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 4  0      0 139732      0 796180    0    0     0     4 10783 16462 23  4 72  0  0
 0  0      0 139512      0 796180    0    0     0    12 13785 19343 28  6 66  0  0
 0  0      0 139732      0 796180    0    0     0    25 10906 15841 22  4 75  0  0
 3  0      0 139356      0 796212    0    0     0     0 10927 15714 22  4 73  0  0
 5  0      0 139704      0 796216    0    0     0     0 12365 16278 32  5 62  0  0
 5  0      0 139668      0 796216    0    0     0     0 11963 16818 28  5 67  0  0
 0  0      0 139736      0 796184    0    0     0     0 12657 16375 30  5 64  0  0
 1  0      0 139640      0 796184    0    0     0     0 11530 15459 28  5 68  0  0
 6  0      0 139424      0 796188    0    0     0     0 13327 18703 29  4 67  0  0
 6  0      0 139452      0 796188    0    0     0     0 12045 16617 26  6 68  0  0
 1  0      0 139512      0 796220    0    0     0     4 12067 16385 27  5 69  0  0
 1  0      0 139484      0 796188    0    0     0    40 12101 17225 24  4 71  0  0
 0  0      0 139420      0 796188    0    0     0     0 12447 16834 27  6 67  0  0
 1  0      0 139140      0 796220    0    0     0     0 10916 15704 23  4 73  0  0
```

As you can see, the performance is affected by the length of `Accept` header, maybe hackers can use this performance problem to make ddos attack easier.

I use `async-profiler` (https://github.com/jvm-profiling-tools/async-profiler) to profile my program, and this is the result:
aaa.txt
there are many stacktraces like the following costs too much cpu:

```
Total: 2777057399 (3.74%)  samples: 2766
```


Human Side of Code Review

jhoeller commented 7 days ago — Contributor

Would it make sense to immediately return `null` after each individual condition check even?

👍 2

`Immediately return `null` after each individual condition check` ✔ c7564f7

aftersss commented 7 days ago — Author — Contributor

> Would it make sense to immediately return `null` after each individual condition check even?

I think it make sense, and the modification is committed.

jhoeller merged commit **29f382b** into `spring-projects:master` 8 days ago

1 check passed

View details

muse

muse-dev bot reviewed on Jan 30

View changes

src/acvp.c

```
4219  +         cap_entry->cap_type = ACVP_ECDSA_SIGVER_TYPE;
4220  +         break;
4221  +     default:
4222  +         return ACVP_INVALID_ARG;
```

muse

muse-dev bot on Jan 30

MEMORY_LEAK: memory dynamically allocated to `return` by call to `calloc()` at line 4197, column 17 is not reachable after line 4222, column 9.

👍 1

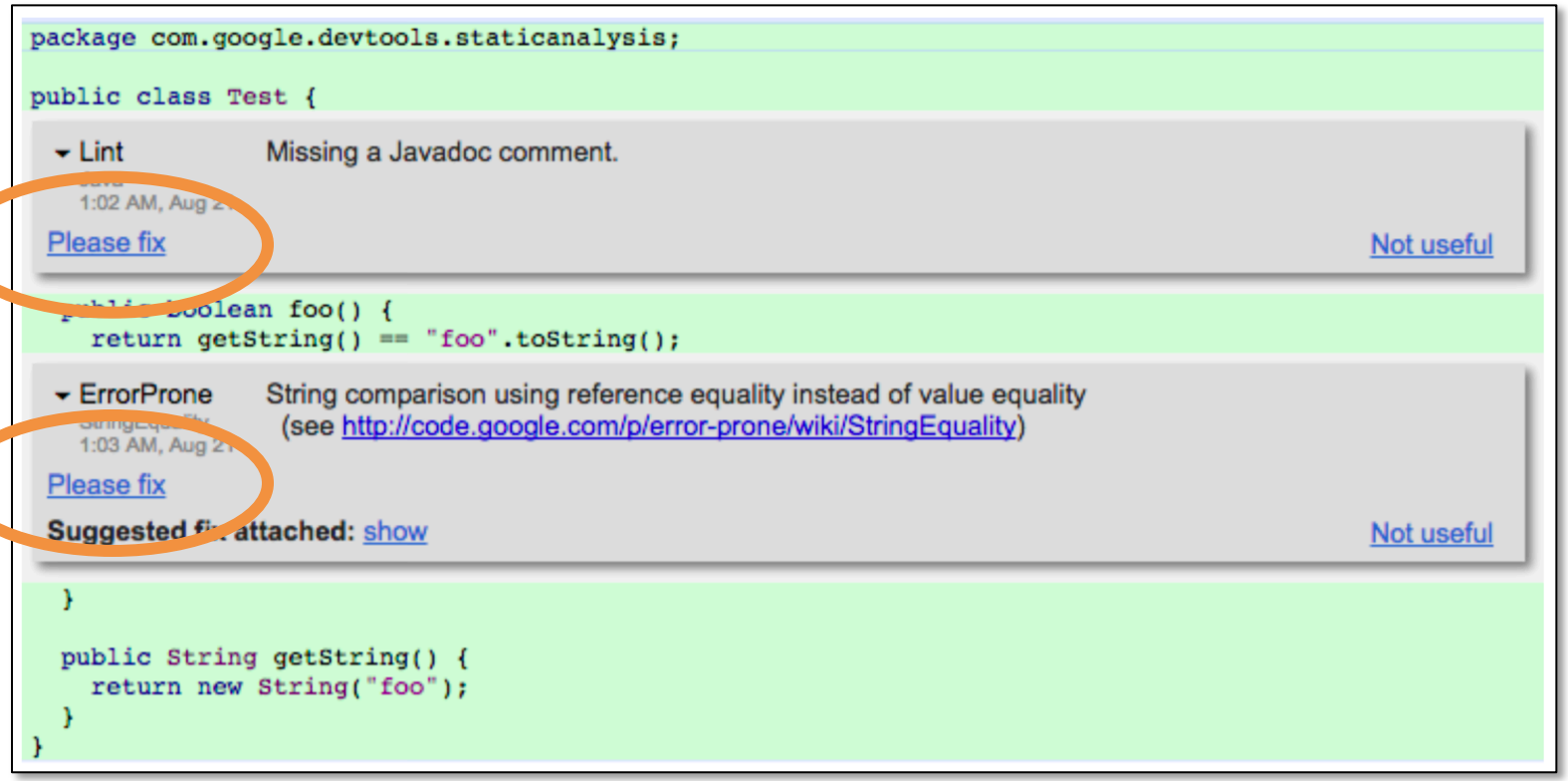
Reply...

Resolve conversation

PRINCIPLE 2: INTEGRATION MATTERS

Google

Tricorder



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

PRINCIPLE 2: INTEGRATION MATTERS

...a striking situation where the diff time deployment saw a 70% fix rate, where a more traditional "offline" or "batch" deployment (where bug lists are presented to engineers, outside their workflow) saw a 0% fix rate.

Distefano, Dino, et al. "Scaling static analyses at Facebook." *Communications of the ACM* 62.8 (2019): 62-70.

KEY PRINCIPLES

- Principle 1: Use Multiple Tools
- Principle 2: Integration Matters
- Principle 3: Cherish Developer Trust
- Principle 4: Tools Can Support Productivity

THINGS THAT CAN GO WRONG

- There are more automated results than developer comments
- Most tool results are useless status updates or style suggestions
- The results that report errors were false positives
- I start looking for browser extensions to filter out tool results.

DANGER OF LOW SIGNAL TO NOISE

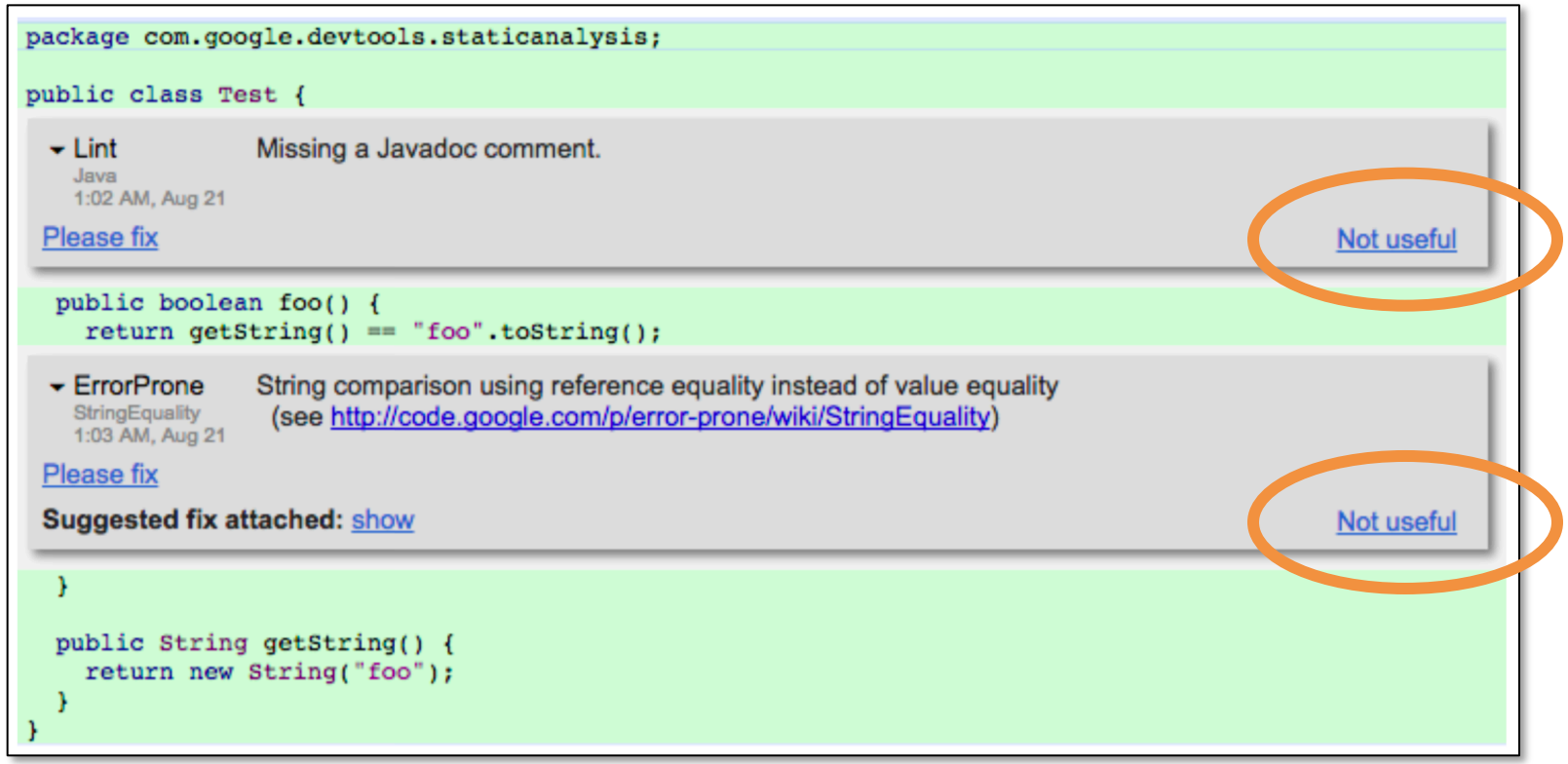
- As more analyzers are added, the risk of “tool fatigue” increases.
- Makes it even more important to maintain a high standard of quality for error reports.

- Once lost, developer trust is never regained and effectiveness plummets.

PRINCIPLE 3: CHERISH DEVELOPER TRUST

Google

Tricorder



Data-driven continual assessment of value.

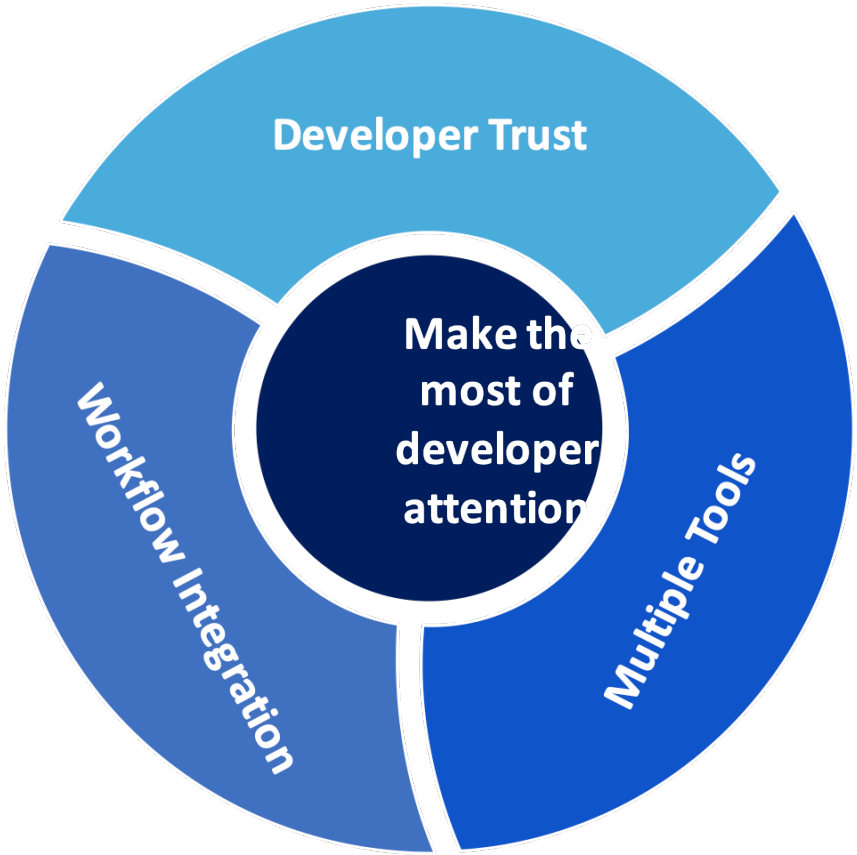
Analyses with >10% effective false positive rate are pulled.

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

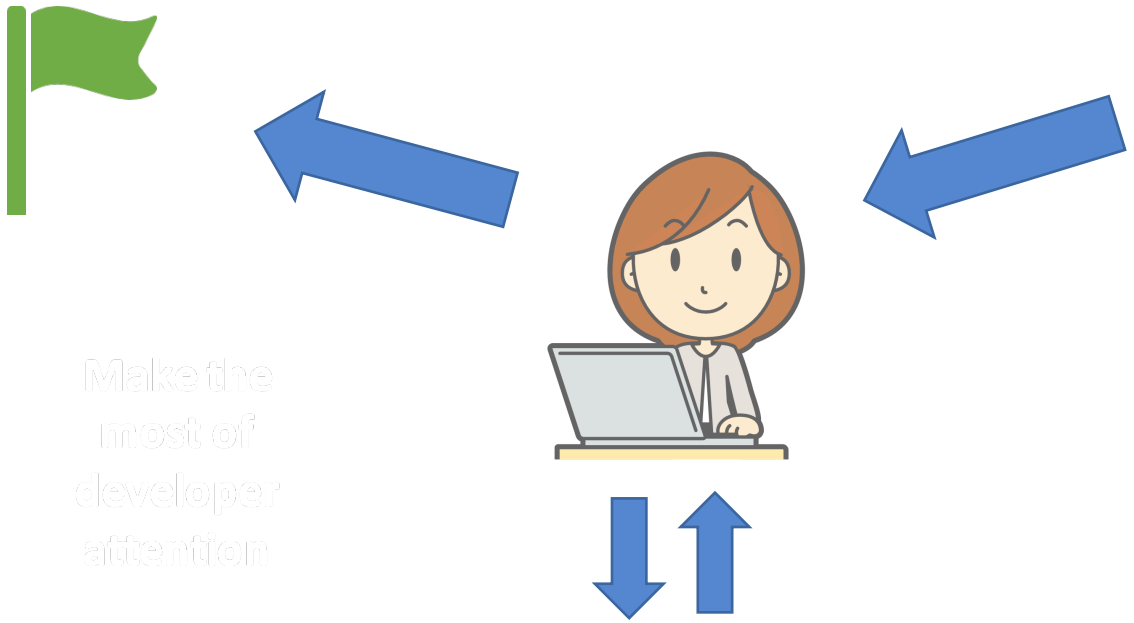
KEY PRINCIPLES

- Principle 1: Use Multiple Tools
- Principle 2: Integration Matters
- Principle 3: Cherish Developer Trust
- Principle 4: Tools Can Support Productivity

PRINCIPLE SYNERGIES



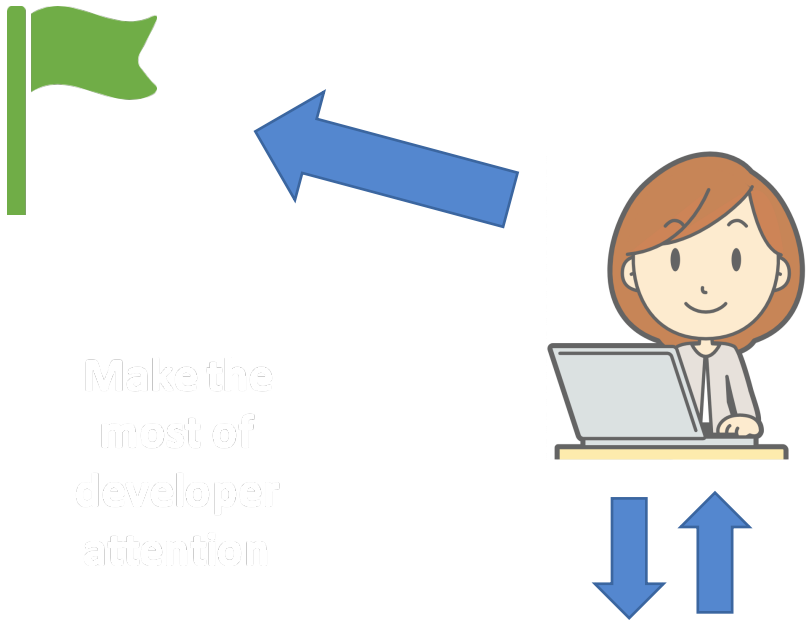
DEVELOPER ATTENTION



Change Request /
Bug Report /
Analysis Tool Result

Make the most of developer attention

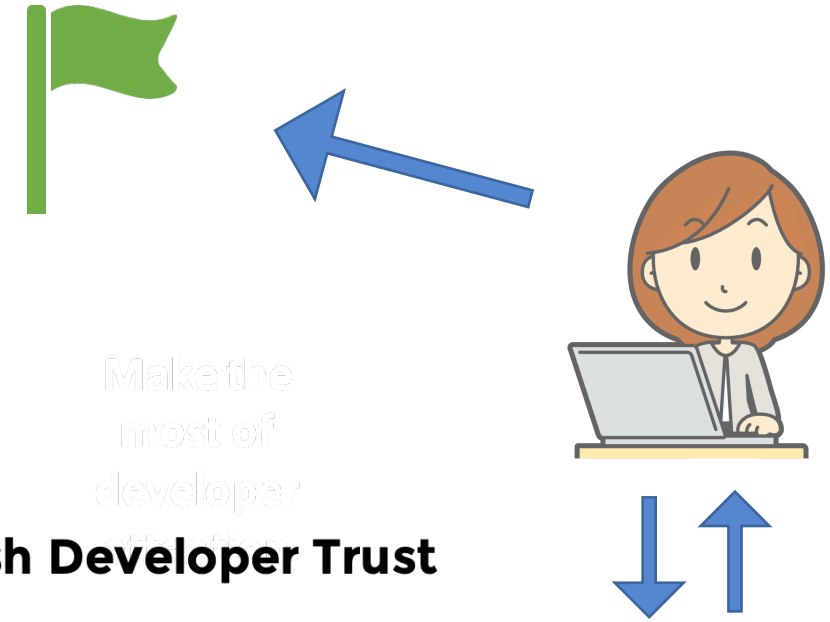
DEVELOPER ATTENTION



Change Request /
Bug Report /
Analysis Tool Result

Make the most of developer attention

DEVELOPER ATTENTION

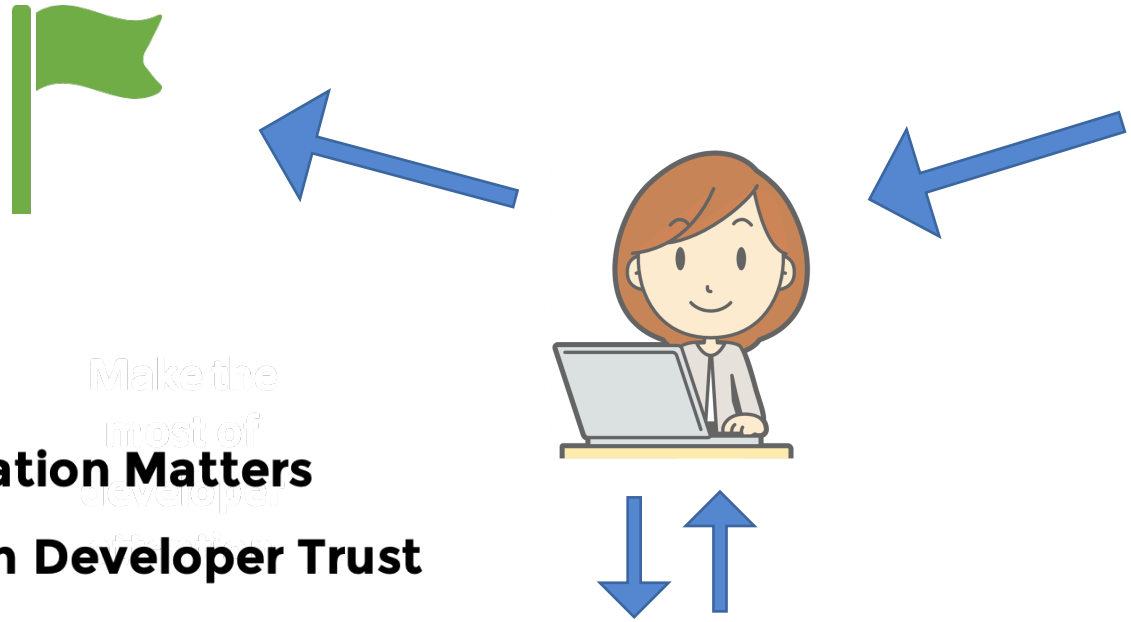


Change Request /
Bug Report /
Analysis Tool Result

Make the most of developer attention

Principle 3: Cherish Developer Trust

DEVELOPER ATTENTION

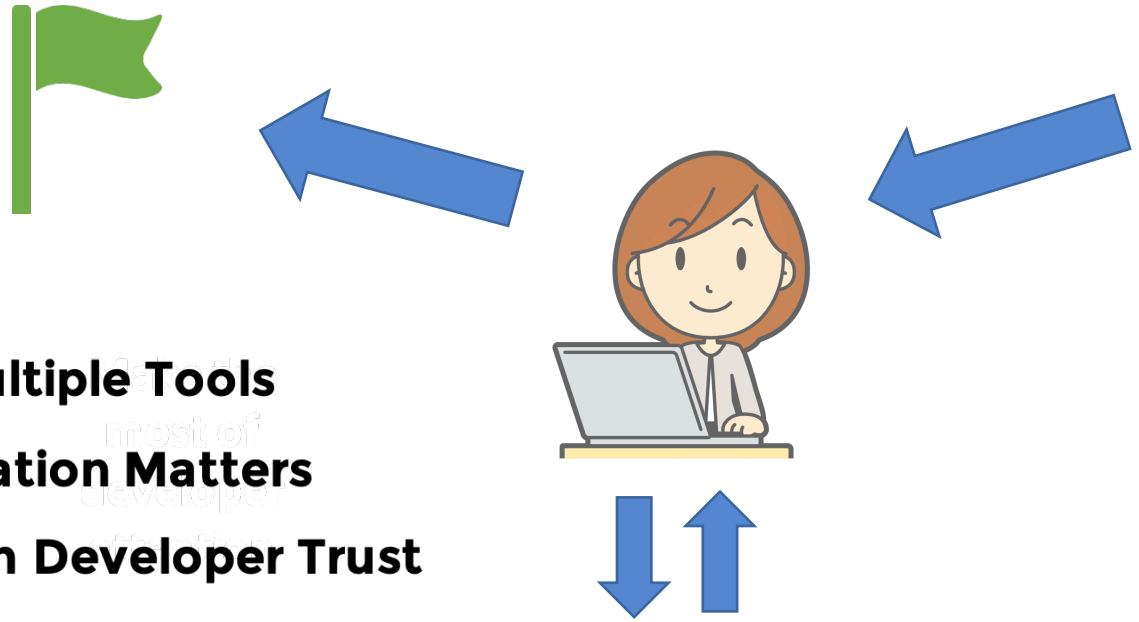


Change Request /
Bug Report /
Analysis Tool Result

Principle 2: Integration Matters

Principle 3: Cherish Developer Trust

DEVELOPER ATTENTION



Change Request /
Bug Report /
Analysis Tool Result

Principle 1: Use Multiple Tools

Principle 2: Integration Matters

Principle 3: Cherish Developer Trust

SYNERGY OF PRINCIPLES 1-3

Google

- Approximately 50,000 code review changes analyzed per day
- “Please Fix” clicked more than 5,000 times per day
- “prevents hundreds of bugs per day from entering the Google codebase”

Sadowski, Caitlin, Edward Aftandilian, Alex Eagle, Liam Miller-Cushon, and Ciera Jaspan. "Lessons from building static analysis tools at Google." (2018).

SYNERGY OF PRINCIPLES 1-3

- Static analysis prevented thousands of vulnerabilities from being introduced over the last two years.
- Catches more severe security bugs than either manual security reviews or bug bounty reports.

Distefano, Dino, Manuel Fähndrich, Francesco Logozzo, and Peter W. O'Hearn. "Scaling static analyses at Facebook." Communications of the ACM 62, no. 8 (2019).

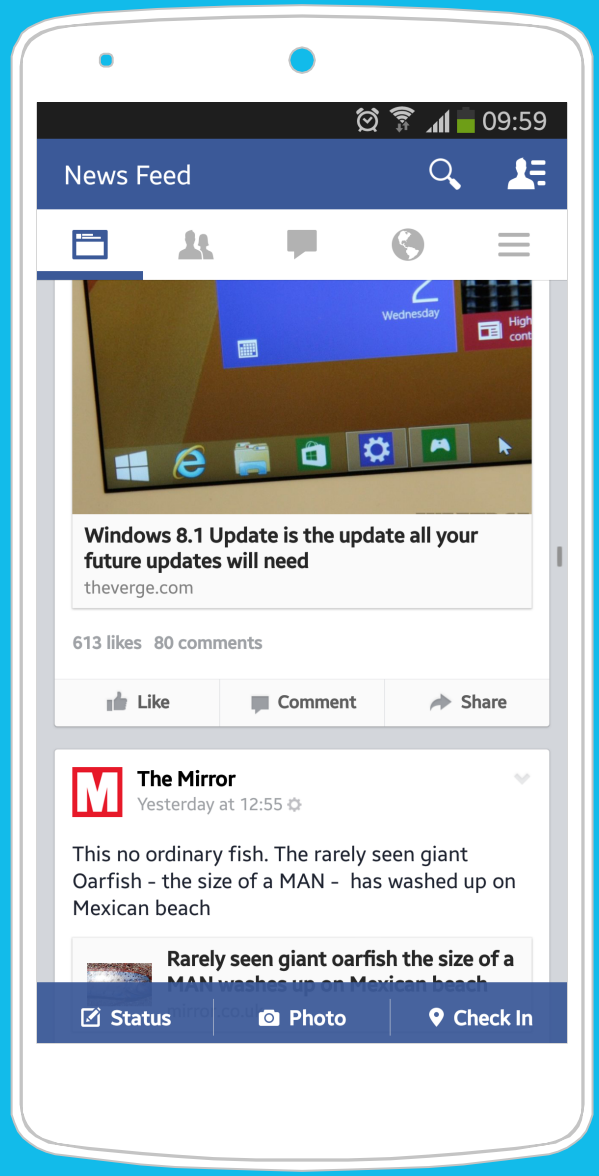
KEY PRINCIPLES

- Principle 1: Use Multiple Tools
- Principle 2: Integration Matters
- Principle 3: Cherish Developer Trust
- Principle 4: Tools Can Support Productivity

FACEBOOK NEWS FEED

- 2016: “News Feed” team wants to add multi-threading.
- Hundreds of classes impacted.
- Collaborates with program analysis team to help analysis support the dev effort.

Distefano, Dino, et al. "Scaling static analyses at Facebook." Communications of the ACM 62.8 (2019): 62-70.



PRINCIPLE 4: Tools Can Support Productivity

> Without Infer, multithreading in News Feed would not have been tenable.
>
> –Android Engineer at Facebook

Distefano, Dino, Manuel Fähndrich, Francesco Logozzo, and Peter W. O'Hearn. "Scaling static analyses at Facebook." Communications of the ACM 62, no. 8 (2019).

SHIFTING QA LEFT

- Static analysis has always been a way to automate QA
- As analysis efficiency improves, these automated tools can be brought left.
- Doing this in the right way can simultaneously improve code quality and developer productivity

THE HELP I'M LOOKING FOR

Tell me all the reasons this wouldn't
work in your environment.