Program analysis survey - Final

What is an ideal program analyzer?

We are researchers at ORGANIZATION trying to understand what software engineers consider an ideal program analyzer. We are conducting this survey that should take about 15 minutes to complete. Your answers will help us suggest future investment directions for the company in the area of program analysis.

What does program analysis mean in this survey?
By program analysis we mean the process of automatically analyzing the behavior of a program *without running it*. Program analysis detects potential issues in your code and gives you feedback. Feedback is in the form of warnings that are either true or false positives. *True positives* flag real code issues in your program, whereas *false positives* are mistakes, in other words, they warn about issues that do not occur in practice.

For the purposes of this survey, *we do not consider the compiler to be a program analyzer*.

This survey is anonymous and no personal information will be collected. Aggregate information may be used in a publication. Please feel free to contact us if you have any questions.

As a thank you for your time, you can enter your name into a raffle of four $50 Amazon.com Gift Certificates after completion of the survey.

Thank you,

Experience with program analyzers

Logic: Show/hide trigger exists.

1) Have you heard of ReSharper?

○ Yes

○ No

Logic: Show/hide trigger exists. Hidden unless: Question "Have you heard of ReSharper?" #1 is one of the following answers ("Yes")

2) Are you using ReSharper?

○ Yes

○ No

Logic: Hidden unless: Question "Are you using ReSharper?" #2 is one of the following answers ("Yes")

3) What do you like about ReSharper? Rank the features from your most to least favorite. Don't stress over the exact ordering and don't move over items that you don't use or don't particularly like.

☐ Code analysis

☐ Navigation and search

☐ Coding assistance

☐ Refactorings

☐ Project level features

☐ Code generation

☐ Code templates

☐ Code style

☐ Unit testing

☐ Internationalization

☐ Cross language functionality

☐ Open API

☐ Command line tools

Logic: Show/hide trigger exists.

4) How often do you run program analyzers?

○ At least once a day

○ At least once a week

○ At least once a month

○ At least once a year

○ I used to run program analyzers, not anymore

○ Never

Logic: Hidden unless: Question "How often do you run program analyzers?" #4 is one of the following answers ("I used to run program analyzers, not anymore")

5) Why did you stop running program analyzers?

○ I didn't make the decision, the team policy changed

○ Some of my team members dislike program analyzers

○ The disadvantages were more than the benefits of running program analyzers

- ○ There was no benefit from running program analyzers
- ○ I wasn't able to find a program analyzer for my needs
- ○ I don't code anymore
- ○ Other - Please specify: ______

Logic: Hidden unless: Question "How often do you run program analyzers?" #4 is one of the following answers ("At least once a day","At least once a week","At least once a month","At least once a year","I used to run program analyzers, not anymore")

6) Did you choose the program analyzers you run or is running these analyzers team policy?

- ○ I chose the program analyzers
- ○ Running the program analyzers is team policy

Logic: Hidden unless: Question "How often do you run program analyzers?" #4 is one of the following answers ("At least once a day","At least once a week","At least once a month","At least once a year","I used to run program analyzers, not anymore")

7) Which program analyzers have you run the most?

- ☐ ApiScan
- ☐ BinScope/BinSkim
- ☐ CheckMarx
- ☐ Code Contracts
- ☐ CodeRush
- ☐ Coverity
- ☐ CppCheck
- ☐ Fortify

☐ FxCop

☐ OACR

☐ PoliCheck

☐ PREfast (and optionally plugins)

☐ PREfix

☐ Semmle

☐ StyleCop

☐ ReSharper

☐ Veracode

☐ VTune

☐ Other - Please specify: ______

Logic: Hidden unless: Question "How often do you run program analyzers?" #4 is one of the following answers ("At least once a day","At least once a week","At least once a month","At least once a year","I used to run program analyzers, not anymore")

8) What are pain points, obstacles, or annoyances you encountered when using program analyzers? Rank them in order of inconvenience to you (up to 5).

☐ Bad phrasing of warnings

☐ Bad visualization of warnings

☐ Complex user interface

☐ Difficult to integrate in the workflow

☐ Irrelevant checks are turned on by default

☐ Miss too many issues

☐ No suppression of warnings

☐ No ranking of warnings

☐ No suggested fixes

☐ Not cross platform

☐ No support for custom rules

☐ No support for all language features

☐ No support for selectively turning analysis off

☐ Too many false positives

☐ Too slow

Types of program analysis

9) Which types of code issues would you like program analyzers to detect? Rank them in order of importance to you (up to 5).

☐ Best practices violations

☐ Compliance violations

☐ Concurrency errors

☐ Dependency issues

☐ Maintainability issues

☐ Memory consumption issues

☐ Memory corruption issues

☐ Performance issues

☐ Portability issues

☐ Power consumption issues

☐ Reliability errors

☐ Security errors

☐ Style inconsistencies

10) Program analysis can analyze most code without problems. However, certain types of code or checking certain properties can be difficult, time consuming, and yield many false positives. We want to know which of the following SHOULD NOT be overlooked (i.e., which are most helpful to you). Rank those that you would like program analysis to check in order of importance to you (up to 5), if any.

Aliasing - The analyzer ignores certain side effects due to reference equality. As an example, if there are two references to the same object and the object is modified via one reference, the analyzer will not be aware of the change through the other reference.
Non-nullness of arguments to main(...) - The analyzer does not check non-nullness of arguments to main(...).
Arithmetic overflow - The analyzer assumes no arithmetic overflow ever happens (in arithmetic operations and conversions).
Exceptional control flow - The analyzer ignores exceptional control flow, such as catch blocks.
Floating point numbers - The analyzer assumes properties about floating point numbers that may not always hold, like associativity.
Iterators - The analyzer does not check iterator methods.
Multiple loop iterations - The analyzer assumes that there is always a fixed number of loop iterations.
Object invariants - The analyzer does not thoroughly check the correctness of object invariants.
Purity - The analyzer assumes that all methods annotated as pure (for example using .NET Code Contracts) actually make no change to state visible outside the method.
Reflection - The analyzer assumes that the code does not use reflection.
Static initialization - The analyzer assumes that the code runs without interference from a static initializer.

☐ Aliasing

☐ Non-nullness of arguments to main(...)

☐ Arithmetic overflow

☐ Exceptional control flow

☐ Floating point numbers

☐ Iterators

☐ Multiple loop iterations

☐ Object invariants

☐ Purity

☐ Reflection

☐ Static initialization

11) Which types of code issues that you encounter do you estimate could have been caught by a program analyzer? Select all that apply.

☐ Best practices violations

☐ Compliance violations

☐ Concurrency errors

☐ Dependency issues

☐ Maintainability issues

☐ Memory consumption issues

☐ Memory corruption issues

☐ Performance issues

☐ Portability issues

☐ Power consumption issues

☐ Reliability errors

☐ Security errors

☐ Style inconsistencies

Tradeoffs

Program analysis includes many tradeoffs. Here, we are asking about tradeoffs such as how long the analysis takes, how intricate (complex, subtle, etc.) the issues found are, and how many false positives are reported. A false positive in

this context is an issue reported by a program analyzer that is incorrect or not really an issue. For example, a simple analyzer might suggest that you do a null check on a parameter when in fact there is no way for the parameter to ever be null based on all the calls to the method. This would be a false positive.

12) If you had to pick, what would you prefer in a program analyzer?

○ Be fast and find superficial issues, like run in a few minutes and find style inconsistencies

○ Be slow and find intricate issues, like run in a few hours and find security or concurrency errors

13) If you had to pick, what would you prefer in a program analyzer?

○ Emit many false positives and miss few code issues, like emit 25% of false positives and check 90% of my code

○ Emit few false positives and miss many code issues, like emit 10% of false positives and check 70% of my code

14) If you had to pick, what would you prefer in a program analyzer?

○ Be fast and emit many false positives, like run in a few minutes and emit 25% of false positives

○ Be slow and emit few false positives, like run in a few hours and emit 10% of false positives

15) If you had to pick, what would you prefer in a program analyzer?

○ Be fast and miss many code issues, like run in a few minutes and check 70% of my code

○ Be slow and miss few code issues, like run in a few hours and check 90% of my code

16) How long are you willing to wait for a program analyzer to check a changeset?

○ Seconds

○ Minutes

○ Hours

○ Days

○ Weeks

17) What would you prefer: that you come up with a fix for a code issue yourself, or that you go through 10 fixes suggested by the program analyzer to find the right one for your code?

○ I prefer to come up with a fix myself

○ I prefer to go through the fixes suggested by the analyzer

18) If a program analyzer emits false positives, when are you willing to tolerate them? Assume that you have the ability to indicate that a warning is a false positive when running the analyzer so that you don't see it again on subsequent runs. This question is about when you are more willing to go through the false positives to begin with.

○ Only when it's easy to tell whether a warning is a false positive

○ Only when analyzing critical code, finding all potential issues in such code is important to me

Validation: Min = 5 Max = 100

19) Assume that a program analyzer emits false positives only when you are willing to tolerate them. Up to what percentage of all warnings may be false positives so that you do not lose trust in the analyzer?

5 ________________________[__]_____________________________ 100

20) Are you willing to annotate your code with specifications/contracts/annotations in order to improve the results of a program analyzer? Note that these specifications would be no-ops at runtime, in other words, they would have no runtime overhead.

- I am willing to write assertions
- I am willing to write assertions and preconditions
- I am willing to write assertions, preconditions, and postconditions
- I am willing to write assertions, preconditions, postconditions, and invariants
- I don't want to write any annotations
- I have no idea what you're talking about

21) Would you be more willing to annotate your code with specifications if these were part of the language, for instance, in the form of non-nullable reference types or an assert keyword?

- Yes
- No

Features

22) Assume that a program analyzer targets the programming languages you code in, and it is free or the company has a license for it. What are the minimum requirements that the analyzer must satisfy for you to start using it? Rank them in order of importance to you (up to 5).

- Cross platform
- Detect issues that are important to me
- Easy to integrate in the workflow
- Fast
- Few false positives

- Good phrasing of warnings
- Good visualization of warnings
- Miss as few issues as possible
- Easy user interface
- With suppression of warnings
- With ranking of warnings
- With suggested fixes
- With support for all language features
- With support for custom rules
- With support for selectively turning analysis off

Logic: Show/hide trigger exists.

23) Do you currently have the functionality to direct a program analyzer toward the parts of your code that are more critical?

- Yes, and I am using it
- Yes, but I am not using it
- No, but it would be important to me
- No, I don't care

Logic: Hidden unless: Question "Do you currently have the functionality to direct a program analyzer toward the parts of your code that are more critical?" #23 is one of the following answers ("Yes, and I am using it","No, but it would be important to me")

24) For directing a program analyzer toward the parts of your code that are more critical, which level of granularity is more suitable for you?

- Assembly

○ File

○ Function or method

○ Program path

25) Do you currently have the functionality to analyze a changeset instead of the entire codebase?

○ Yes, and I am using it

○ Yes, but I am not using it

○ No, but it would be important to me

○ No, I don't care

26) Do you currently have the functionality to write your own program analysis rules?

○ Yes, and I am using it

○ Yes, but I am not using it

○ No, but it would be important to me

○ No, I don't care

27) Do you currently have the functionality to up or down vote program analysis warnings to train the analyzer to your or your team's needs?

○ Yes, and I am using it

○ Yes, but I am not using it

○ No, but it would be important to me

○ No, I don't care

Logic: Show/hide trigger exists.

28) Do you currently have the functionality to suppress a program analysis warning?

○ Yes, and I am using it

○ Yes, but I am not using it

○ No, but it would be important to me

○ No, I don't care

Logic: Hidden unless: Question "Do you currently have the functionality to suppress a program analysis warning?" #28 is one of the following answers ("Yes, and I am using it")

29) How do you suppress a program analysis warning and do you like this way of doing it? Answer only for the ways that you have been using.

	I like it	**I dislike it**
Global configuration	○	○
Source code annotations	○	○
Comment annotations	○	○
External suppression file	○	○
Fixing a code version as base	○	○

Fitting into your work

30) If you work on different codebases (different projects, modules in the same project, front-end versus back-end development, etc.), how different are your program analysis needs across the codebases?

- ○ Extremely different
- ○ Fairly different
- ○ Slightly different
- ○ Not different at all
- ○ I only work on one codebase

31) How would you like the results of a program analyzer to be shown to you? Rank them in order of importance to you.

- ☐ In my editor
- ☐ In the build output
- ☐ In the code review
- ☐ In a web viewer

32) If you had a program analyzer that met your criteria for use, when in your workflow would you be most likely to run it?

- ○ All the time
- ○ Every time I compile
- ○ During nightly builds
- ○ Every time I run unit tests
- ○ Right before I submit a code review

○ Right before I commit

○ During integration testing

○ During functional testing

○ Right before a release

○ When I remember

○ I don't want to run a program analyzer in my workflow, but I want its results to always be available in case I choose to look at them

Other forms of software QA

33) How confident are you that your code reviewers do not miss issues in your code changes?

○ My code reviewers exhaustively check all critical parts of my code

○ My code reviewers check many parts of my code but certainly not exhaustively

○ My code reviewers check a few parts of my code

○ I don't use code reviewing

34) How confident are you that your unit tests check your code changes?

○ My unit tests exhaustively check all critical parts of my code

○ My unit tests check many parts of my code but certainly not exhaustively

○ My unit tests check a few parts of my code

○ I don't write unit tests

35) In which languages are you currently programming?

☐ Assembly

☐ C

☐ C++

☐ C++/CLI

☐ C#

☐ F#

☐ Java

☐ JavaScript

☐ Objective-C

☐ Python

☐ Ruby

☐ VB.NET

☐ Other - Please specify: ____

Demographics

Validation: Min = 0 Max = 50 Must be numeric

36) How many years of experience do you have in software engineering?

____

37) Which of the following describes your knowledge and/or experience with program analysis?

○ I consider myself to be an expert

○ I understand program analysis

○ I am familiar with program analysis

○ I have a vague understanding of program analysis

○ I know very little about program analysis

38) If there is anything else that you would like to tell us or you think would be useful for us to know, please enter it here.

Thank You!

Thank you for taking our survey. Your response is very important to us.
