



# Static Testing

TVVS

Ana Paiva

FEUP 2020/2021

João Vieira up201603190

Ricardo Ferreira up200305418

Susana Lima up201603634



# What is?

- Type of a Software Testing method in which the software is tested without actually executing the code of the application
  - Usually done by analyzing the code against a given set of rules or coding standards
- Can be incorporated at any stage after the development of code, but usually before any other testing phase
  - Some projects have static analysis reports in their CI/CD pipelines as a quality gate for code promotions
- It is associated to Linter software and all major languages have linters



# Static vs Dynamic

Static Testing	Dynamic Testing
No code execution (pré compilation)	Requires code execution (post compilation)
Does the verification process	Does the validation process
For prevention of defects	For finding and fixing the defects
Gives an assessment of code and documentation	Gives bugs/bottlenecks in the software system.
Involves a checklist and process to be followed	Involves test cases for execution
Cost of finding defects and fixing is less	Cost of finding and fixing defects is high
Requires loads of meetings	Comparatively requires lesser meetings

---

# Static Testing Techniques

Reviews

Static Analysis



# Review

- Find the potential defects in the design of the software.
- Detect and remove errors or defaults in the different support documents like requirements, design, test cases...
- One or several humans examine the documents and sort out errors, redundancies and ambiguities.



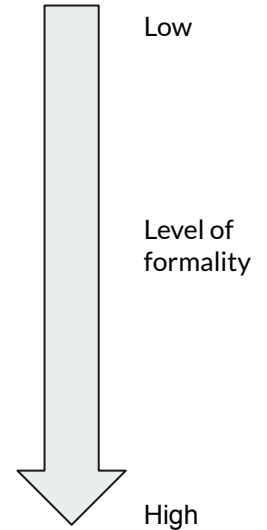
# Review Types

**Informal:** Coworkers perform reviews and give informal comments and opinions, allowing early detection of defects.

**Walkthrough:** The author of the document being reviewed explains it to their team. The reviewers ask questions and write down notes.

**Peer review:** A team of colleagues (peers) review the technical specifications in order to detect any errors.

**Inspection:** A designated moderator conducts a meticulous review as a process to find defects.





# Static Analysis

- Evaluation of the quality of the code written by developers.
- The code is analysed for structural defects (without being run) that may lead to defects.
- Generally done with a support tool.



# Why you should use static code analysis

*“When I feel I’ve finished a small piece of functionality, I want to get feedback about a number of things: Am I documenting what I’m doing? Am I following standards? I want to find anything that could cause a security vulnerability or the code to crash.”*

Joachim Herschman, research director at Gartner

*“Sometimes you have a mix of senior and junior developers. [Static code analysis] allows the junior developers to work independently without guardrails while making sure they’re developing along the lines of the team’s expectations.”*

Jasen Schmidt, director of solutions architecture at Avanade

*“The recommendation is make static code analysis part of your CI process. You want to stop corrupt code from deploying, so you want to stop it as early as possible”*

Gartner Herschmann



# Goals

## NON COMPLIANT CODE

```
int j;
while (true) { // Noncompliant; end condition omitted
    j++;
}

int k;
boolean b = true;
while (b) { // Noncompliant; b never written to in loop
    k++;
}
```

## COMPLIANT SOLUTION

```
int j;
while (true) { // reachable end condition added
    j++;
    if (j == Integer.MIN_VALUE) { // true at Integer.MAX_VALUE + 1
        break;
    }
}

int k;
boolean b = true;
while (b) {
    k++;
    b = k < Integer.MAX_VALUE;
}
```

- Identify weaknesses in code
- Adhere to development standards
- Help reduce potential production issues
- Save time by detecting errors while writing code  
(Gives immediate feedback)



---

# Analysis Types

Data Flow

Control Flow

Lexical Analysis

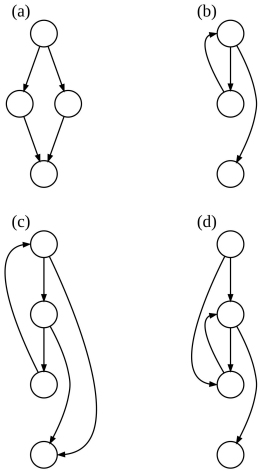


## Static Analysis Types: **Data Flow**

- Focuses on the data **variables** and their **values**, how they are **accessed**, **modified** and **defined** in the program.
- Detects, for example:
  - Variables that are declared but never used.
  - Variables that are used but never declared.
  - Variables that are used but have no value assigned.
  - Variables that are defined multiple times before being used.

# Static Analysis Types: Control Flow

Control Flow Graph (CFG)



- a) if-then-else
- b) while
- c) loop with two exits
- d) loop with two entry points

- Expresses the flow of a program in a CFG.
- Determine the order of program statements and predict and specify the set of execution traces.
- Most of the execution time is spent in loops.
- Detects, for example:
  - Loop invariant (remove from loop code that produces always the same result).
  - Infinite Loops.
  - Dead Code elimination.
  - Induction Variable Elimination (Variable that are only used in test conditions and their own definitions).
  - Cyclomatic complexity.

---

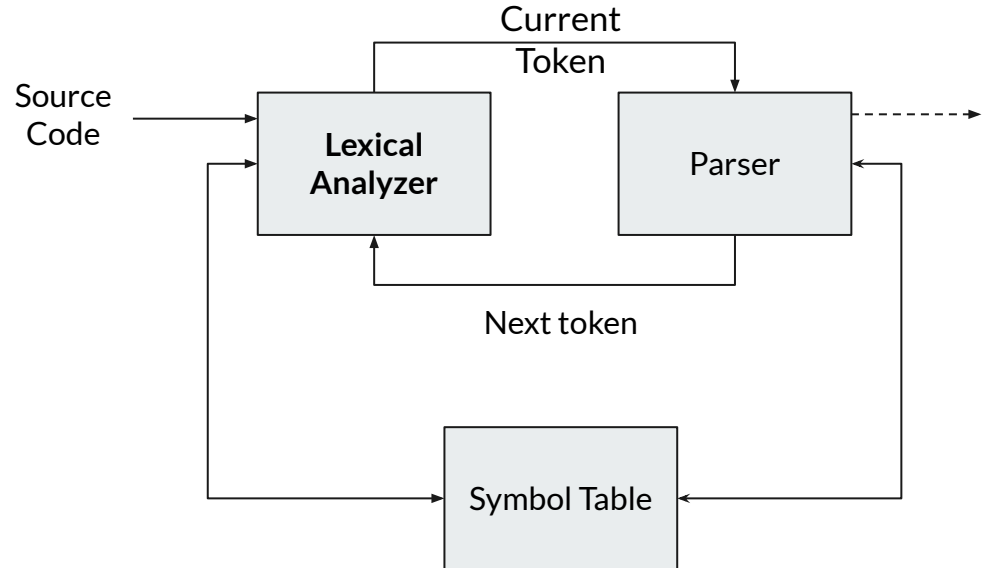
## Control Flow: Taint Analysis

- Identify variables that have been *tainted* with user controllable input.
- Traces them to possible vulnerable functions.
- If the tainted variable is not sanitized it is flagged as a vulnerability.
- ex: SQL injection



# Static Analysis Types: **Lexical**

- Equivalent to first step of compilation.
- Refers to the tokenization of code to make its manipulation easier.



# Tools

---

# (Facebook) Infer

Java, C, C++ and Objective-C

```
class Infer {  
    String mayReturnNull(int i) {  
        if (i > 0) {  
            return "Hello, Infer!";  
        }  
        return null;  
    }  
  
    int mayCauseNPE() {  
        String s = mayReturnNull(0);  
        return s.length();  
    }  
}
```

```
Found 1 issue  
Infer.java:12: error: NULL_DEREFERENCE  
object s last assigned on line 11 could be null and is dereferenced at line 12  
10.     int mayCauseNPE() {  
11.         String s = mayReturnNull(0);  
12. >         return s.length();  
13.     }  
14.
```


- Android and Java code
  - Checks for null pointer exceptions, resource leaks, annotation reachability, missing lock guards, and concurrency race conditions in.
- C, C++ and Objective C
  - Checks for null pointer problems, memory leaks, coding conventions and unavailable API's in
- Approach to the problem is completely different from norm (see [this](#) link)



# JSLint

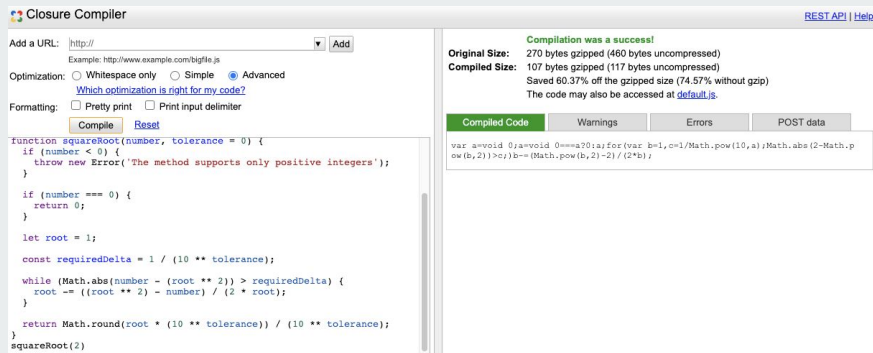
## JavaScript

```
8 function promiseAlwaysResolve(promise) {
9   return new Promise((resolve, reject) => {
10     promise.then(...args => resolve(...args))
11       .catch(...args => resolve(...args));
12   });
13 }
14
15 function findMissingCounselorIDs(getState, orgID) {
16   return getCounselorsByOrgID(orgID);
17 }
18
19 function findMissingParticipantIDs(getState, chats) {
20   const { participants } = getState();
21   const currentParticipantIDs = Object.keys(participants.byID);
22
23   return Promise.all(map(chats, (chat) => {
24     let { chatType } = chat;
25     if (chatType === 'org') {
26       const { participantIDs } = chat;
27       let { orgID } = chat.orgData;
28
29       return findMissingCounselorIDs(getState, orgID)
30         .then(counselorIDs => {
31           const allParticipants = [...participantIDs.filter((p) => p !== orgID), ...counselorIDs];
32           return allParticipants.filter(p => currentParticipantIDs.indexOf(p) === -1);
33         });
34     .catch((error) => {
35       // ...
36     });
37   }));
38 }
39
40 NORMAL app/browser/dashboard/actions/chats.js javascript.jsx 17% 14/79 1
41 app/browser/dashboard/actions/campaigns.js Expected to return a value in this function. (array-callback-return)
42 app/browser/dashboard/actions/campaigns.js Block must not be padded by blank lines. (padded-blocks)
43 app/browser/dashboard/actions/campaigns.js Expected parentheses around arrow function argument having a body with curly braces. (arrow-parens)
44 app/browser/dashboard/actions/campaigns.js Expected parentheses around arrow function argument having a body with curly braces. (arrow-parens)
45 app/browser/dashboard/actions/campaigns.js Missing semicolon. (semi)
46 app/browser/dashboard/actions/campaigns.js Missing semicolon. (semi)
47 app/browser/dashboard/actions/campaigns.js 'chatAdapter' is defined but never used. (no-unused-vars)
48 app/browser/dashboard/actions/campaigns.js Missing semicolon. (semi)
49 app/browser/dashboard/actions/campaigns.js Missing semicolon. (semi)
50 app/browser/dashboard/actions/campaigns.js Unexpected console statement. (no-console)
51 app/browser/dashboard/actions/campaigns.js Unexpected console statement. (no-console)
52 app/browser/dashboard/actions/chats.js9 col 52 ERROR 'reject' is defined but never used. (no-unused-vars)
53
```

- Checks following of code style guide
- First syntax checker
- Derived tools
  - JSHint
  - TSLint
  - ESLint 
- Basically a highly configurable JSLint

# Google Closer Tools

## JavaScript



- Developed mainly for web apps
- Compiles JavaScript files into more efficient ones, compromising readability in the process
- Used in popular tools
  - Google Maps
  - Google Docs
  - Gmail



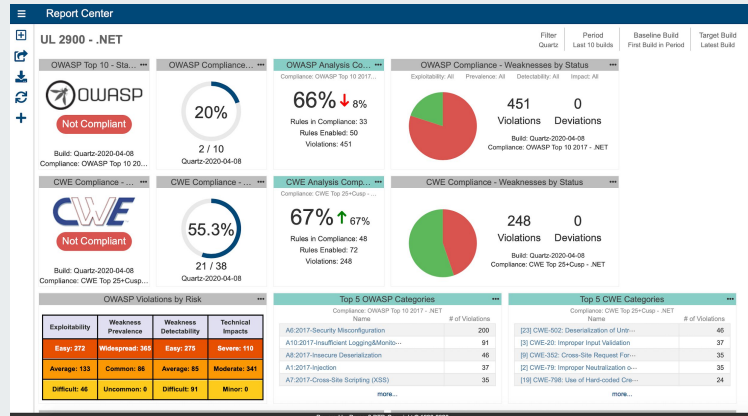
Python

```
manim@mani-laptop: ~  
File Edit View Terminal Help  
manim@mani-laptop:~$ pylint simple_calc.py  
No config file found, using default configuration  
***** Module simple_calc  
R: 16:Maths.is even: Method could be a function  
R: 27:Maths.xcube: Method could be a function  
R: 38:Maths.square: Method could be a function  
R: 49:Maths.power: Method could be a function  
R: 60:Maths.square root: Method could be a function  
R: 71:Maths.factorial number: Method could be a function  
R: 82:Maths.check prime: Method could be a function  
Exception RuntimeError: 'maximum recursion depth exceeded while calling a Python  
object' in <type 'exceptions.RuntimeError'> ignored  
  
Report  
=====  
24 statements analysed.  
  
Raw metrics  
-----  
  
+-----+-----+-----+-----+  
|type|number| % |previous|difference|  
+-----+-----+-----+-----+  
|
```

- Messages according to the PEP 8
- Similar tools may be
  - pycheckers
  - pyflakes
- Additional functionalities/checks
  - Line length
  - Implementation of declared interfaces
  - Variable naming convention following



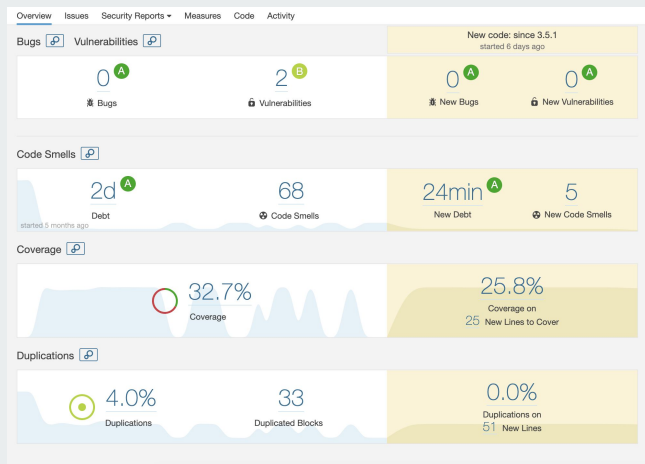
C/C++, Java, .NET



- Has a multitude of tools for this languages, like Jtest
  - Built-In Test Configurations
  - Rule Categories: Each rule belongs to a rule category (i.e. Optimization, Security, Exceptions, API )
  - Severity Levels: Each rule is assigned with a severity level



Multi-language



- Static analysis of code to code smells, detect bugs, and security vulnerabilities
- Open-source and available on +20 languages
- Reports on code coverage, code complexity, duplicated code, comments, unit tests, bugs, coding standards, and security vulnerabilities
- Integration with commonly used tools like Maven and Jenkins

# SonarLint



## Multi-language

```
1 // return the op between two POSITIVE numbers
2 // if the numbers are not >= 0, returns -1
3 // if op is not SUM, SUB, PROD or DIV, returns -1
4 function ex1(a, b, op, op2) {
5   let res;
6   if (a < 0 || b < 0) {
7     return -1;
8   }
9   switch (op) {
10     case 'SUM':
11       res = a + b;
12       break;
13     case 'SUB':
14       res = a - b;
15     case 'PROD':
16       res = a * b;
17     case 'DIV':
18       res = a / b;
19       break;
20     default:
21       return -1;
22   }
23 }
```

**"switch" statements should have "default" clauses (javascript:S131)**

Code Smell Critical

The requirement for a final default clause is defensive programming. The clause should either take appropriate action, or contain a suitable comment as to why no action is taken.

**Noncompliant Code Example**

```
switch (param) { //missing default clause
  case 0:
    doSomething();
    break;
  case 1:
    doSomethingElse();
    break;
}
```

**switch (param) {**  
default: // default clause should be the last one  
error();  
}

**PROBLEMS 21**

- ex1.js: 9, 3: Add a "default" clause to this "switch" statement. sonarlint(javascript:S131) [9, 3]
- ex1.js: 13, 5: End this switch case with an unconditional break, continue, return or throw statement. sonarlint(javascript:S128) [13, 5]
- ex1.js: 14, 5: "value\_" does not exist. Change its name or declare it so that its usage doesn't result in a "ReferenceError". sonarlint(javascript:S3827) [40, 14]
- ex1.js: 15, 5: Add the missing "else" clause. sonarlint(javascript:S126) [42, 5]
- ex1.js: 16, 5: "value\_" does not exist. Change its name or declare it so that its usage doesn't result in a "ReferenceError". sonarlint(javascript:S3827) [52, 7]
- ex1.js: 17, 5: "c" does not exist. Change its name or declare it so that its usage doesn't result in a "ReferenceError". sonarlint(javascript:S3827) [57, 10]

- Static analysis of code to code smells, detect bugs, and security vulnerabilities
- Detect and fix quality issues as you write code.
- IDE extension
  - Visual Studio Code
  - Eclipse
  - IntelliJ
  - Visual Studio

# DEMO

<https://bit.ly/39pA77Z>

---



# Links

<https://www.geeksforgeeks.org/software-testing-static-testing/>

[https://www.tutorialspoint.com/software testing dictionary/static testing.htm](https://www.tutorialspoint.com/software_testing_dictionary/static_testing.htm)

<https://searchsoftwarequality.techtarget.com/definition/static-analysis-static-code-analysis>

[https://en.m.wikipedia.org/wiki/List\\_of\\_tools\\_for\\_static\\_code\\_analysis](https://en.m.wikipedia.org/wiki/List_of_tools_for_static_code_analysis)

<https://searchsoftwarequality.techtarget.com/definition/static-testing>

<https://web.fe.up.pt/~ei05021/TQSO%20-%20An%20overview%20on%20the%20Static%20Code%20Analysis%20approach%20in%20Software%20Development.pdf>

[https://owasp.org/www-community/controls/Static\\_Code\\_Analysis](https://owasp.org/www-community/controls/Static_Code_Analysis)

<https://www.guru99.com/static-dynamic-testing.html>

<https://www.sonarlint.org/>