BMW GROUP

HOW TO DEAL WITH STATIC ANALYSIS FINDINGS: MISRA

XAVIER BONAVENTURA

Meeting C++ 2023

ABOUT ME

- Where to find me:
  - @limdor
  - @xbonaventurab

- Working in C++
  - since 2011
  - in safety critical systems since 2018

- I represent BMW at the MISRA C++ working group and at the C++ committee meetings

ABOUT ME

- **Disclaimer:**
  - I'm not a Functional Safety expert
  - I'm not an static analysis expert
  - I'm not a MISRA expert
  - I'm just the technical lead of a SW development team
    - that has to deal with functional safety,
    - static analysis,
    - and MISRA every day.
  - I'm not representing MISRA

ABOUT YOU

- How many of you use static analysis tools?
- How many of you know MISRA?
- How many of you have to deal with MISRA?

WHAT IS THIS ABOUT?

How to deal with static analysis findings: MISRA

WHAT IS STATIC ANALYSIS?

"Computer programs can be checked for errors statically and dynamically.

- Static analysis looks for structural and semantic faults in programs.
- Dynamic analysis affirms proper operation—and helps identify errors."

- R. E. Fairley, "Tutorial: Static Analysis and Dynamic Testing of Computer Software," in Computer, vol. 11, no. 4, pp. 14-23, April 1978, doi: 10.1109/C-M.1978.218132.

WHAT IS STATIC ANALYSIS?

Static analysis (without executing the program)

- Control flow
- Data flow
- Code review
- Manual code inspections
- ...

Dynamic analysis (during runtime)

- Unit tests
- Thread sanitizers
- Address sanitizers
- Undefined behaviour sanitizers
- Memory leak detectors
- ....

WHAT IS STATIC ANALYSIS?

	Static analysis (without executing the program)	Dynamic analysis (during runtime)
Automatic	• Control flow • Data flow • Compiler warnings/errors	• Unit tests • Thread sanitizers • Address sanitizers • Undefined behaviour sanitizers • Memory leak detectors • UI tets • Acceptance tests
Manual	• Code review • Manual code inspections	• UI tests • Acceptance tests

WHAT IS THIS ABOUT?

How to deal with static analysis findings: MISRA

WHAT ARE FINDINGS?

*Meaning of **finding** in English*

finding

noun

UK /ˈfaɪn.dɪŋ/ US /ˈfaɪn.dɪŋ/

finding *noun* (DISCOVERY)

Add to word list

[C]

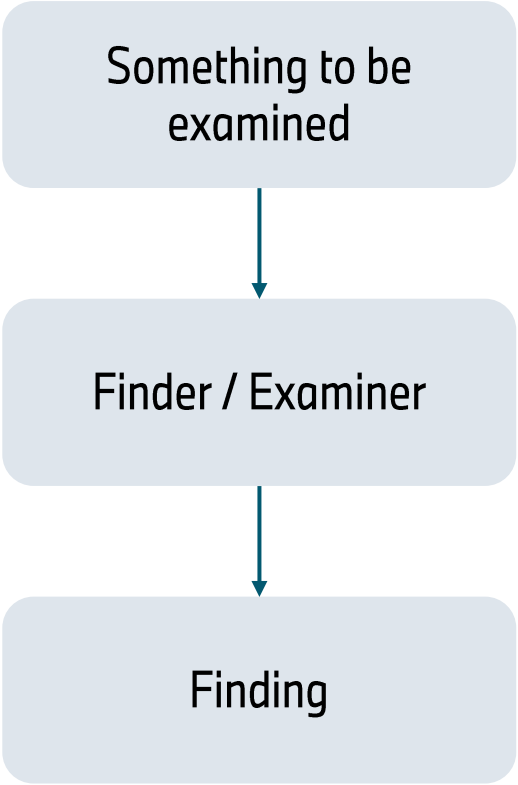
a piece of information that is discovered during an official examination of a problem, situation, or object:

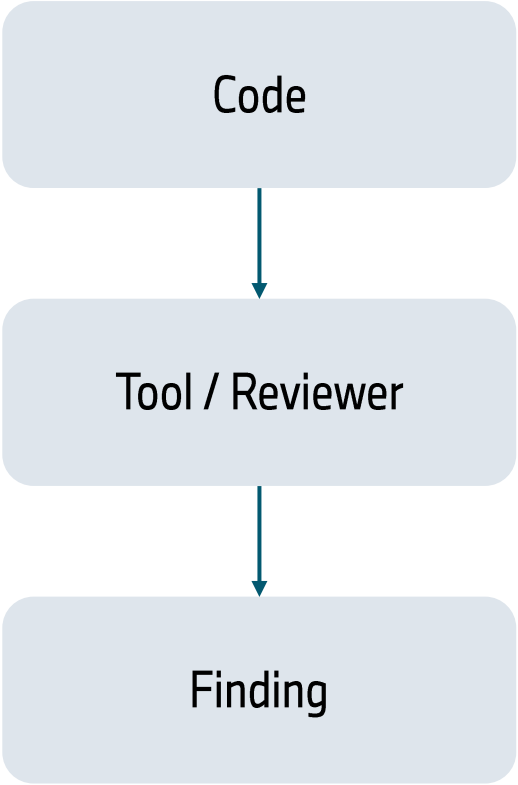
- *The report's finding on the decrease in violent crime supports the police chief's claims.*

– Fewer examples

- *These new findings turn the accepted theories on their head.*
- *The findings of the survey puzzle me - they're not at all what I would have expected.*
- *The methodology and findings of the research team have been criticized.*
- *These findings are inconsistent with those of previous studies.*
- *The survey's finding is likely to increase the pressure on the local authorities.*

WHAT ARE FINDINGS?





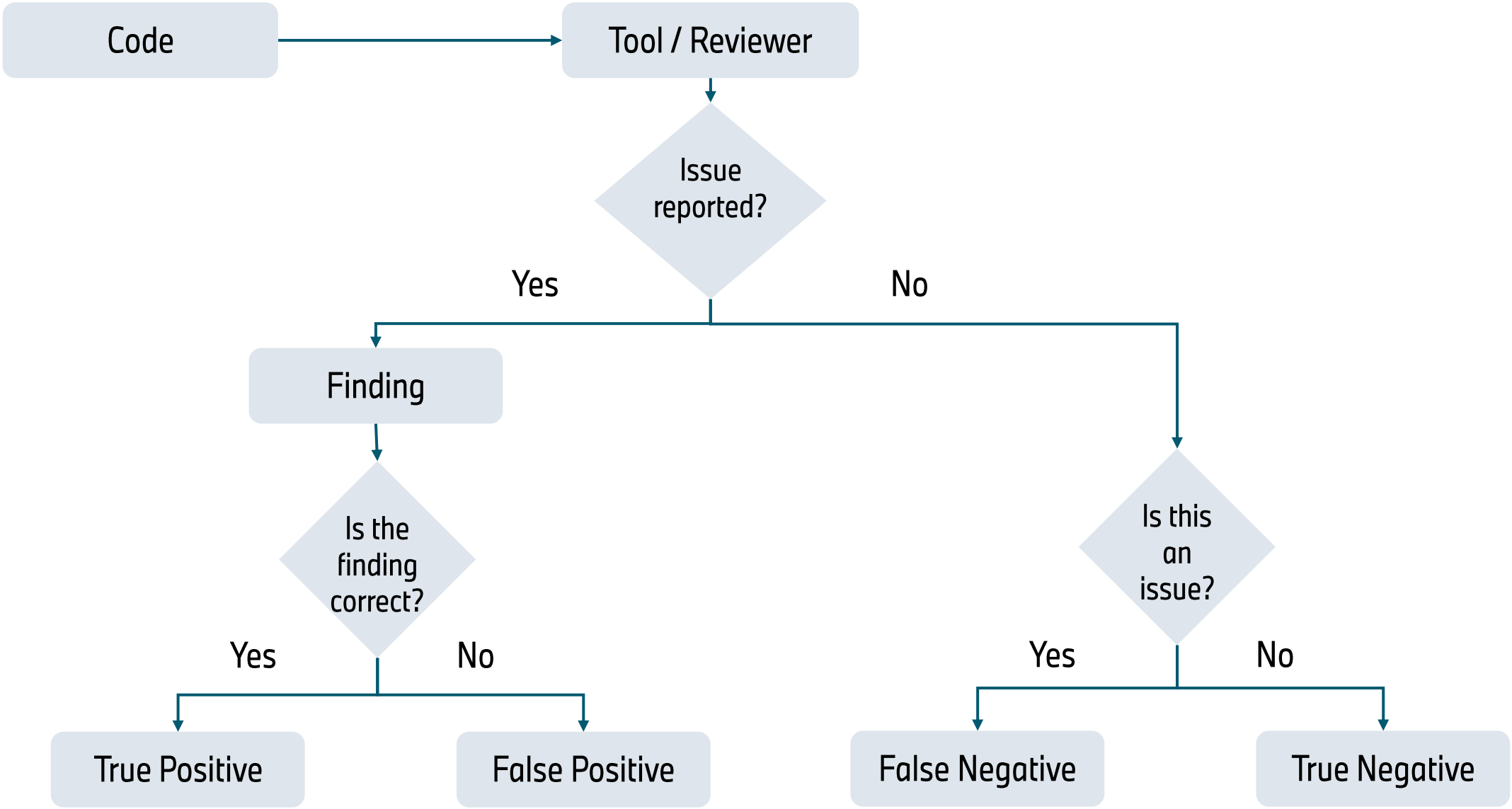
WHAT ARE FINDINGS?

- What information should a finding contain?
  - Where?
    - Line of code
    - Function
    - Variable
  - What?
    - Description of the problem
  - Why?
    - Reason why it is reported
    - Why it is a problem
  - How?
    - Availability of an automatic fix
    - Suggestion on how to fix it if available

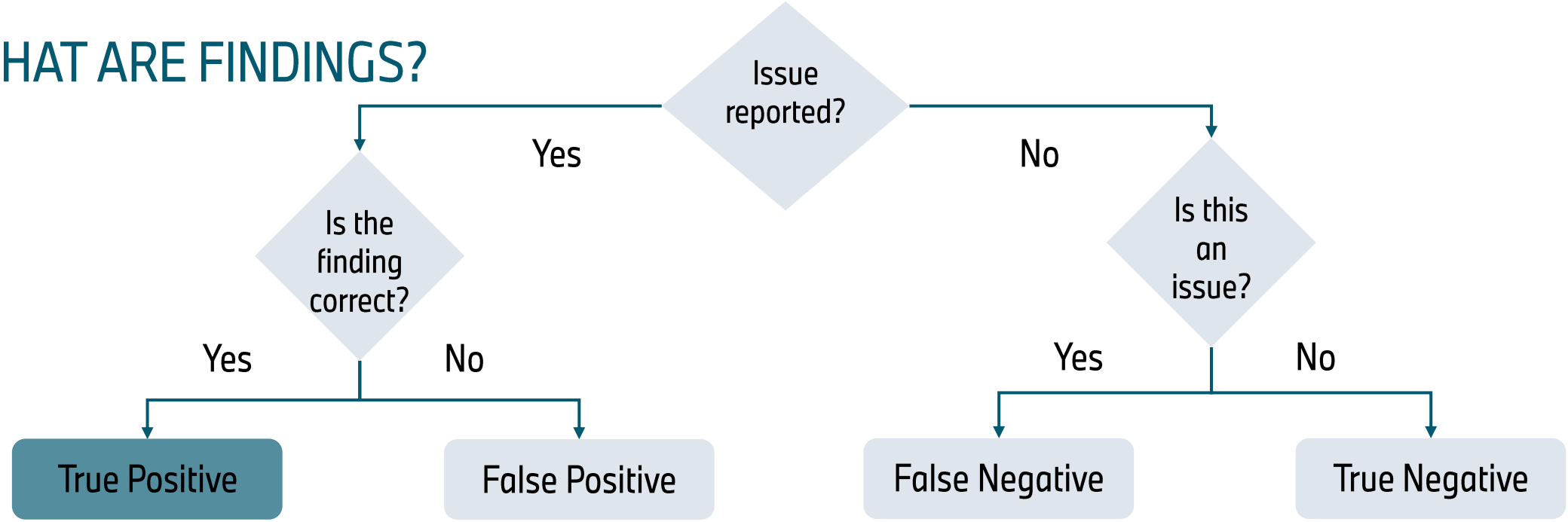
WHAT IS MISRA?

- What information should a finding contain?
  - References
  - Contact
    - Expert for questions
    - Process to report false positives

WHAT ARE FINDINGS?

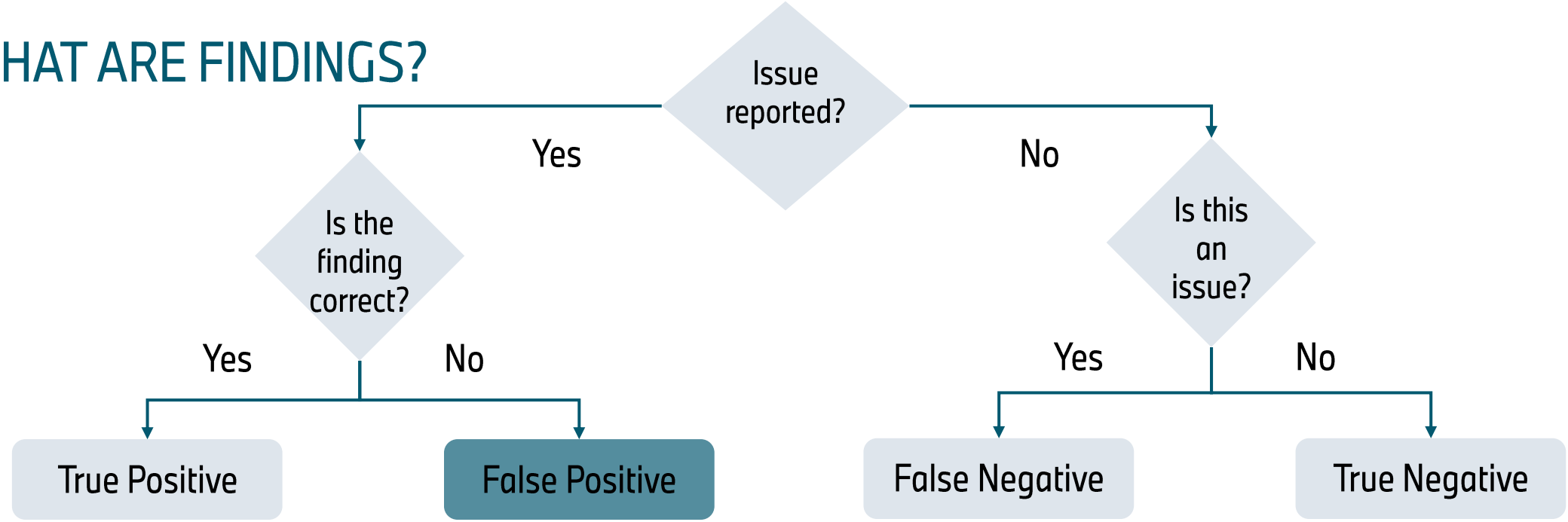


WHAT ARE FINDINGS?



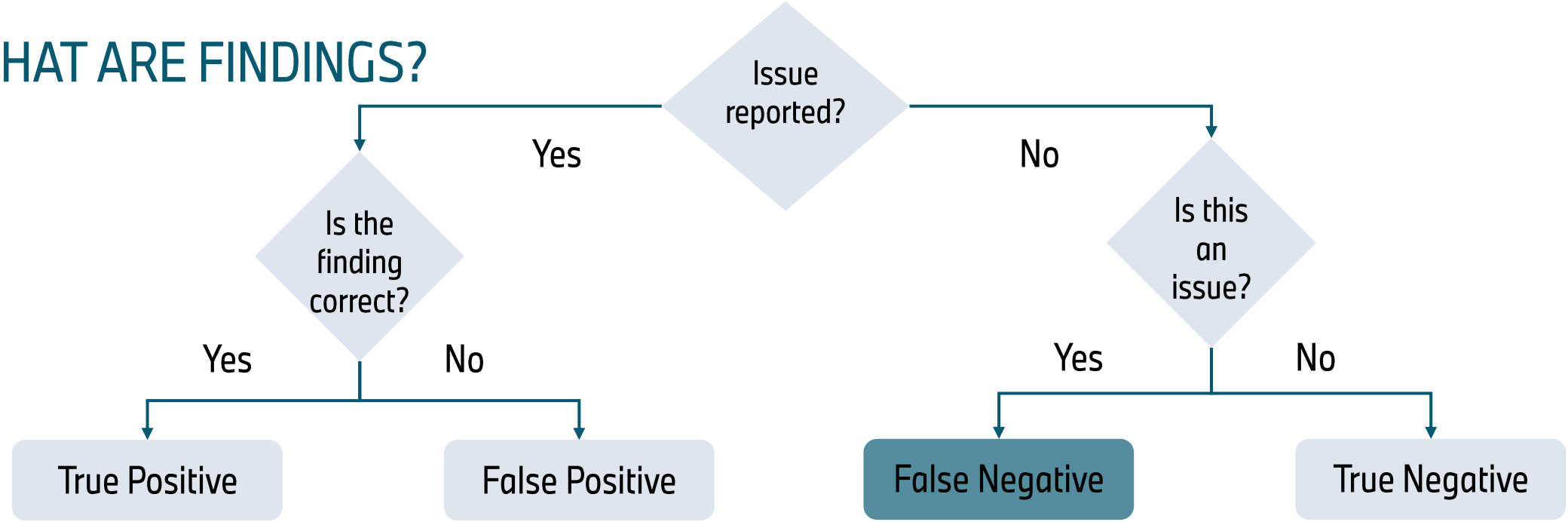
- Fix the issue
- Supress the issue and explain why it is fine to have such code
- Change the rule / guideline that led to that finding

WHAT ARE FINDINGS?



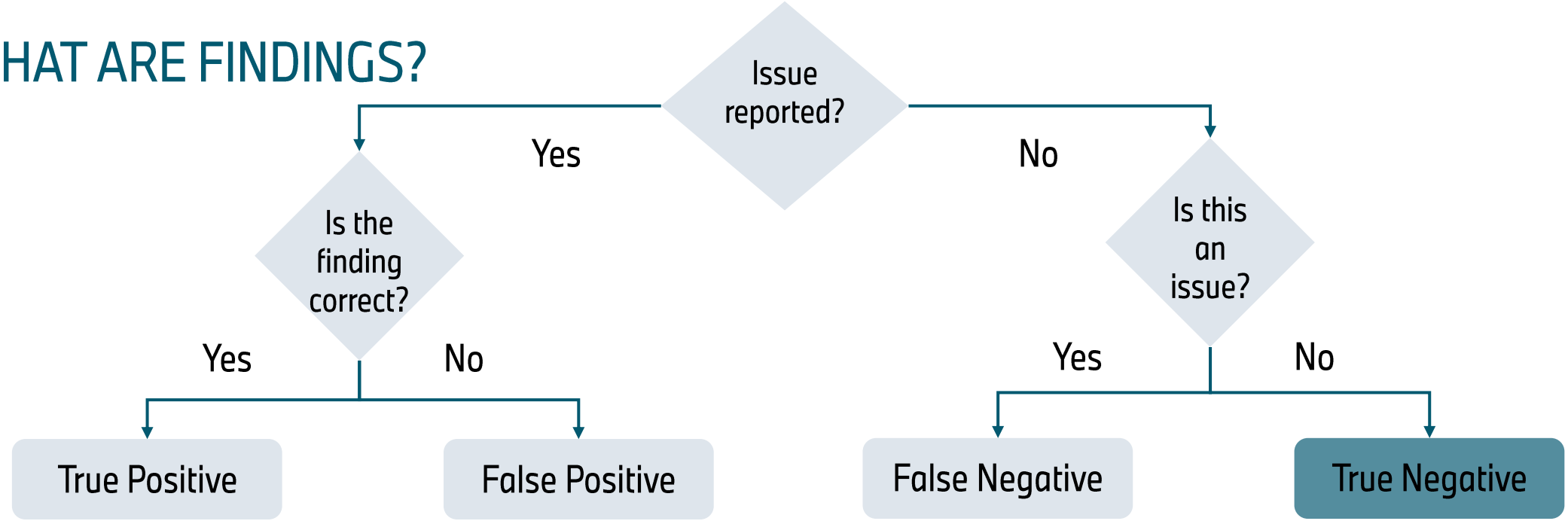
- Report the false positive to the tool / reviewer
- Supress the issue
- Explain why it is a false positive
- Create a test to detect when the false positive is fixed

WHAT ARE FINDINGS?



- Check if this is an issue according to the rules / guidelines
- Adapt the rules / guidelines
- Report the false negative to the tool / reviewer
- Create a test to detect when the false negative is fixed

WHAT ARE FINDINGS?



- Everything is fine!

How to deal with static analysis findings: MISRA

WHAT IS MISRA?

- MISRA (Motor Industry Software Reliability Association)
  - Started in the early 1990s developing guidelines for the creation of embedded software in road vehicle electronic systems

- Provides guidelines for the safe and secure application for C and C++

- Publications:
  - MISRA C:2023 for C11 and C18
  - MISRA C++:2008 for C++03
  - MISRA Compliance:2020

- Web: https://www.misra.org.uk/

WHY MISRA?

- ISO26262: Road vehicles – Functional Safety

- Criteria for suitable modelling, design or programming languages (see 5.4.2) that are not sufficiently addressed by the language itself **shall be covered by the corresponding guidelines**, or by the development environment, considering the topics listed in Table 1.
  - ISO 26262:2018 Part 6 (Paragraph 5.4.3)

1a	Enforcement of low complexity[a]
1b	Use of language subsets[b]
1c	Enforcement of strong typing[c]
1d	Use of defensive implementation techniques[d]
1e	Use of well-trusted design principles[e]
1f	Use of unambiguous graphical representation
1g	Use of style guides
1h	Use of naming conventions
1i	Concurrency aspects[f]

WHY MISRA?

- It was created before ISO26262 in order to achieve functional safety

- AUTOSAR has guidelines for the use of C++14 in critical and safety-related systems
  - They will get replaced by the next version of MISRA C++ for C++17

- More about MISRA:
  - Safer C++: MISRA-C++ : 202X Rules and Beyond - Peter Sommerlad [ACCU 2021] https://www.youtube.com/watch?v=SAK2IyYtMBE
  - MISRA C++202x: It ain't your grandpa's MISRA any more - Loïc Joly [NDC TechTown 2022] https://www.youtube.com/watch?v=RwSaDVubdKk

HOW IS MISRA DOCUMENT STRUCTURED?

- MISRA document contains more than just rules

- MISRA C++:2008
  - 1. Background
  - 2. The vision
  - 3. Scope
  - 4. Using MISRA C++
  - 5. Introduction to the rules
  - 6. Rules
  - 7. References
  - Appendix A: Summary of rules
  - Appendix B: C++ vulnerabilities
  - Appendix C: Glossary

  - 6.5. Expressions
    - 6.5.0. General
    - 6.5.2. Postfix expressions
    - 6.5.3. Unary expressions

WHAT A MISRA RULE CONTAINS

- A MISRA rule contains
  - Rule
  - Rule category: Mandatory / Required / Advisory
  - Decidability: Decidable / Undecidable
  - Analysis scope: Single translation unit / System
  - References to the C++ standard
  - Amplification
  - Rationale
  - Example
  - Reference to other rules

COMPLIANCE

- Publications:
  - MISRA C:2023 for C11 and C18
  - MISRA C++:2008 for C++03
  - MISRA Compliance:2020

- About MISRA Compliance:2020
  - Achieving compliance with MISRA Coding Guidelines
  - Available for free in the MISRA page

HOW IS MISRA COMPLIANCE DOCUMENT STRUCTURED?

- MISRA Compliance:2020
  - 1. Introduction
  - 2. The software development process
  - 3. Fundamental elements of compliance
  - 4. Deviations
  - 5. The guideline re-categorization plan
  - 6. Adopted Code
  - 7. Claiming MISRA compliance
  - Appendix A: Process and tools checklist
  - Appendix B: Example deviation record
  - Appendix C: Example deviation permit
  - Appendix D: Glossary

COMPLIANCE

- Justifying a deviation:
  - Some of the cases where a deviation must not be permitted:
    - Simply to satisfy the convenience of the developer
    - When a reasonable alternative coding strategy would make the need for a violation unnecessary
    - Without considering the consequences on other guidelines
    - Without the support of a suitable process
    - Without the consent of a designated technical authority

- Note: For the full information on how to justify a deviation see MISRA Compliance:2020 document

EXAMPLE OF A MISRA RULE

- Rule:
  - Do not abruptly terminate a program
- Amplification:
  - Do not call abort, exit or terminate implicitly or explicitly
- Rationale:
  - In some cases, stack unwinding is not happening. In other cases, it is implementation-defined whether the call stack is unwound.

- Note: This is just a simplification of how the rule looks like, for the exact definition refer to the MISRA document

EXAMPLE

```
int my_integer_division(int a, int b){
      if(b == 0){
            std::terminate(); // Finding. Correct?
      }
      return a/b;
}
```

https://godbolt.org/z/M1sPr7h8f

EXAMPLE

```
int my_integer_division(int a, int b){
    if(b == 0){
        // Ignore finding:
        std::terminate();
    }
    return a/b;
}
```

https://godbolt.org/z/M1sPr7h8f

EXAMPLE

```
int my_integer_division(int a, int b){
    if(b == 0){
        // Ignore finding: It is fine.
        std::terminate();
    }
    return a/b;
}
```

https://godbolt.org/z/M1sPr7h8f

EXAMPLE

```
int my_integer_division(int a, int b){
    if(b == 0){
        // Ignore finding: I agreed with the reviewer that we accept it.
        std::terminate();
    }
    return a/b;
}
```

https://godbolt.org/z/M1sPr7h8f

EXAMPLE

```
int my_integer_division(int a, int b){
    if(b == 0){
        // Ignore finding:
        // The project agreed to address precondition violations with
        // a call to terminate. In <url_link> you can read why it is not
        // a problem if the stack is not unwound
        std::terminate();
    }
    return a/b;
}
```

An example of a deviation record can be found in the MISRA Compliance:2020 document

https://godbolt.org/z/M1sPr7h8f

EXAMPLE

```
bool should_we_terminate(const Error& error){
    auto const terminate = error.is_fatal(); // Finding. Correct?
    return terminate; // Finding. Correct?
}

class Error{
    public:
        bool is_fatal() const{
            std::terminate();
            return true;
        }
};
```

https://godbolt.org/z/s65xdaTh4

EXAMPLE

```
bool should_we_terminate(const Error& error){
    auto const terminate = error.is_fatal(); // Suppress: False positive
    return terminate; // Suppress: False positive
}
```

SUMMARY

- Don't fix findings without thinking
- Don't suppress findings without thinking
- Keep in mind that false negatives exist
- Fixing or suppressing a finding are not the only the two ways to proceed

BMW GROUP

HOW TO DEAL WITH STATIC ANALYSIS FINDINGS: MISRA

XAVIER BONAVENTURA

Meeting C++ 2023