Standard C++ toolset 

Anastasia Kazakova, JetBrains
@anastasiak2512



Core C++ 2023

About me

C++: Embedded, Telecom, 4G/LTE

@anastasiak2512

C++ Tools PMM

The Dutch C++ Group

Most frustrating points about C++

–

Major Frustration Points	major %
Managing libraries my application depends on	47 %
Build times	43 %
Setting up a CI pipeline from scratch	31 %
Managing CMake projects	31 %
Concurrency safety: Races, deadlocks, performance bottlenecks	28 %
Setting up a dev env from scratch	27 %
Parallelism support	23 %
Managing Makefiles	20 %
Memory safety: Bounds safety issues	19 %
Memory safety: Use-after-delete/free	19 %
Managing MSBuild projects	17 %
Debugging issues in my code	18 %
Security issues: disclosure, vulnerabilities, exploits	11 %
Type safety: Using an object as the wrong type	11 %
Memory safety: Memory leaks	11 %
Moving existing code to the latest language standard	8 %

What is language toolset?

—

Essential tools

- Compiler + standard library
- Debugger
- Build system
- Libraries & Dependency management

Complementary

- Analyzer
- Unit testing frameworks
- Coverage
- Profiling
- etc..

Java

1. javac, ECJ
2. Java Class Library: concurrency, networking, AWT / Swing, generics, internationalization
3. Maven (67-72%), Gradle (20-49%), Ant (8-11%)
4. JUnit (85%)

Ruby

1. “Compiler”:
   a. Interpreter: MRI (CRuby), YARV
   b. JIT: JRuby (JVM), Rubinius (LLVM)
   c. IronRuby (.NET), new mruby interpreter, …
2. RubyGems (Bundler 90%)
3. RSpec (77%)

Swift

—

1. Swift toolchain from Apple
2. Swift compiler (LLVM)
3. SourceKit (LSP)
4. LLDB, Read Eval Print Loop (REPL), Playground
5. SPM, CocoaPods
6. XCTest (87%)

Rust

1. rustc
2. Cargo build + packages
3. Rustfmt
4. rust-analyzer (LSP)
5. 60% debug via `println!` or `dbg!` macros
6. test module / cargo test

Does the toolset help?

–

1. Help newbies
2. Onboarding via dev environments
3. Code unification
4. Best practices adoption
5. Code sharing (libraries)

Variety is good or?

–

GCC / Clang / MSVC / Intel

Example: Compiler predefined macros

—

```
gcc/clang -E -dM -O3 -x c++ /dev/null
```

```
cl /EP /Zc:preprocessor /PD empty.cpp 2>nul
```

Example: Syntax style

–

Almost Always Auto or when type is evident or never for numeric types.

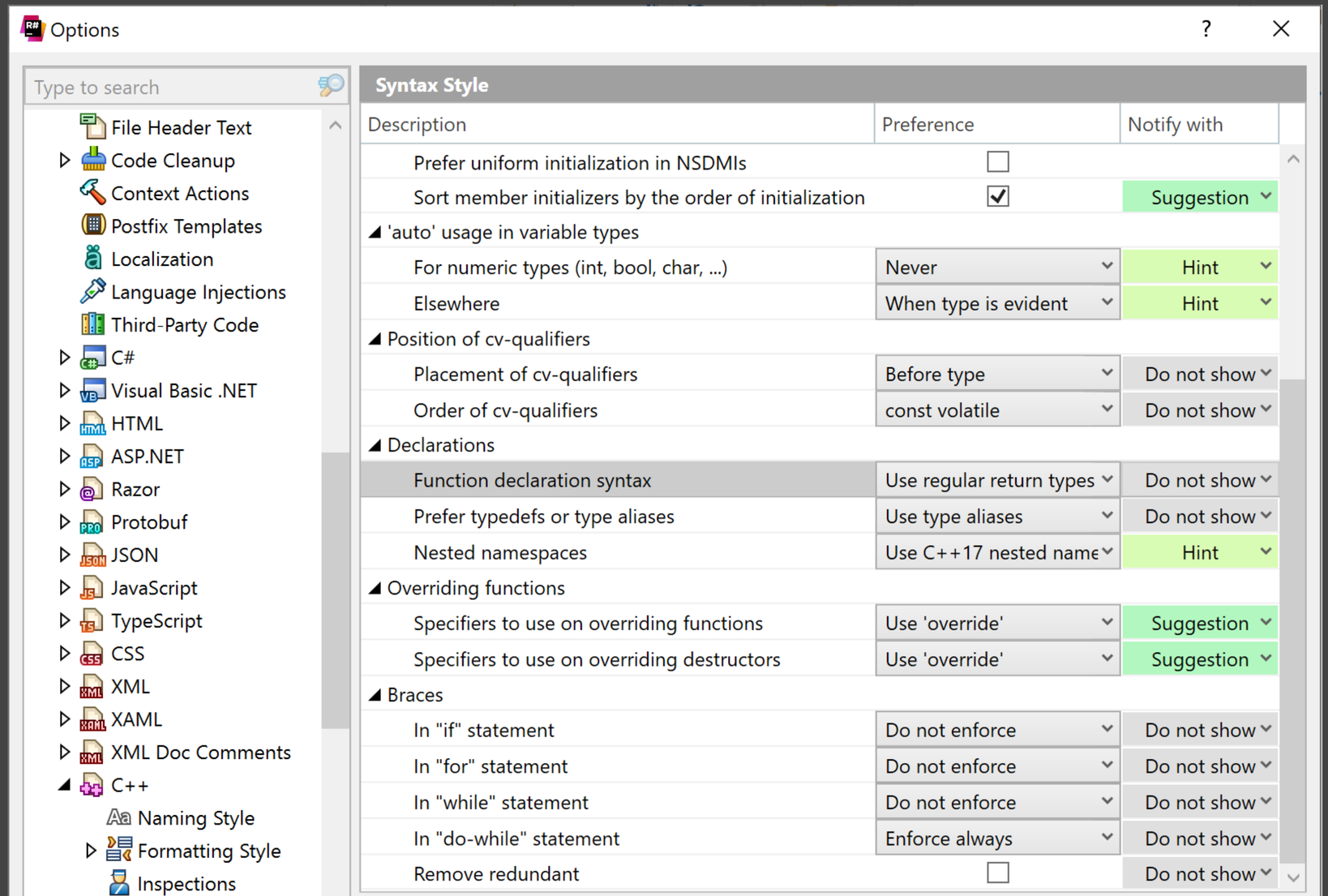
Const before or after the type it applies to.

Trailing return type for lambdas or always.

Virtual explicitly (UE) or override/final and no virtual (C++ Core Guidelines).

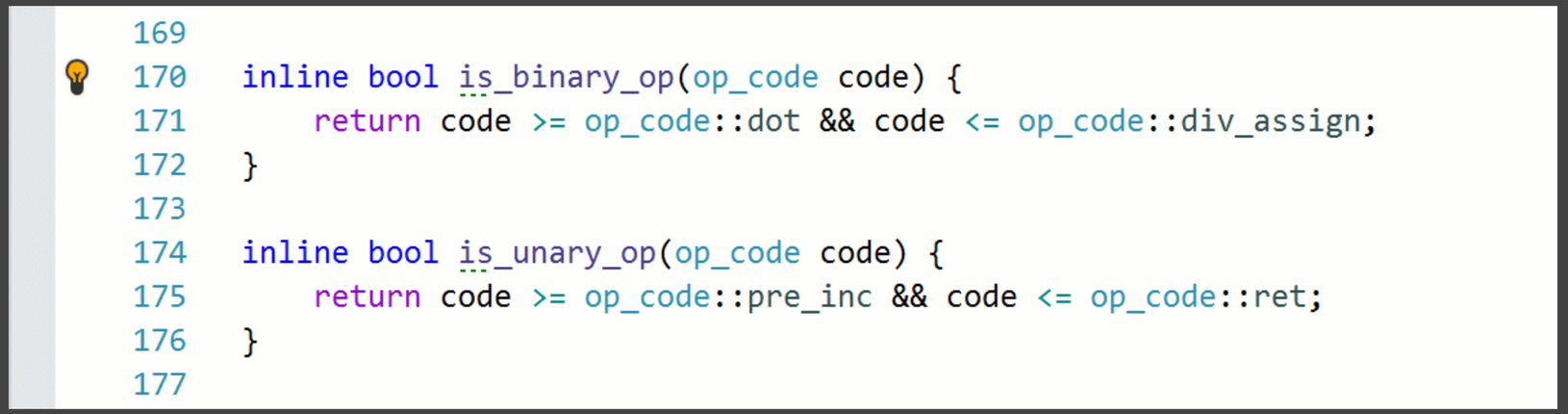
Example: Syntax Style

–



Example: Syntax Style

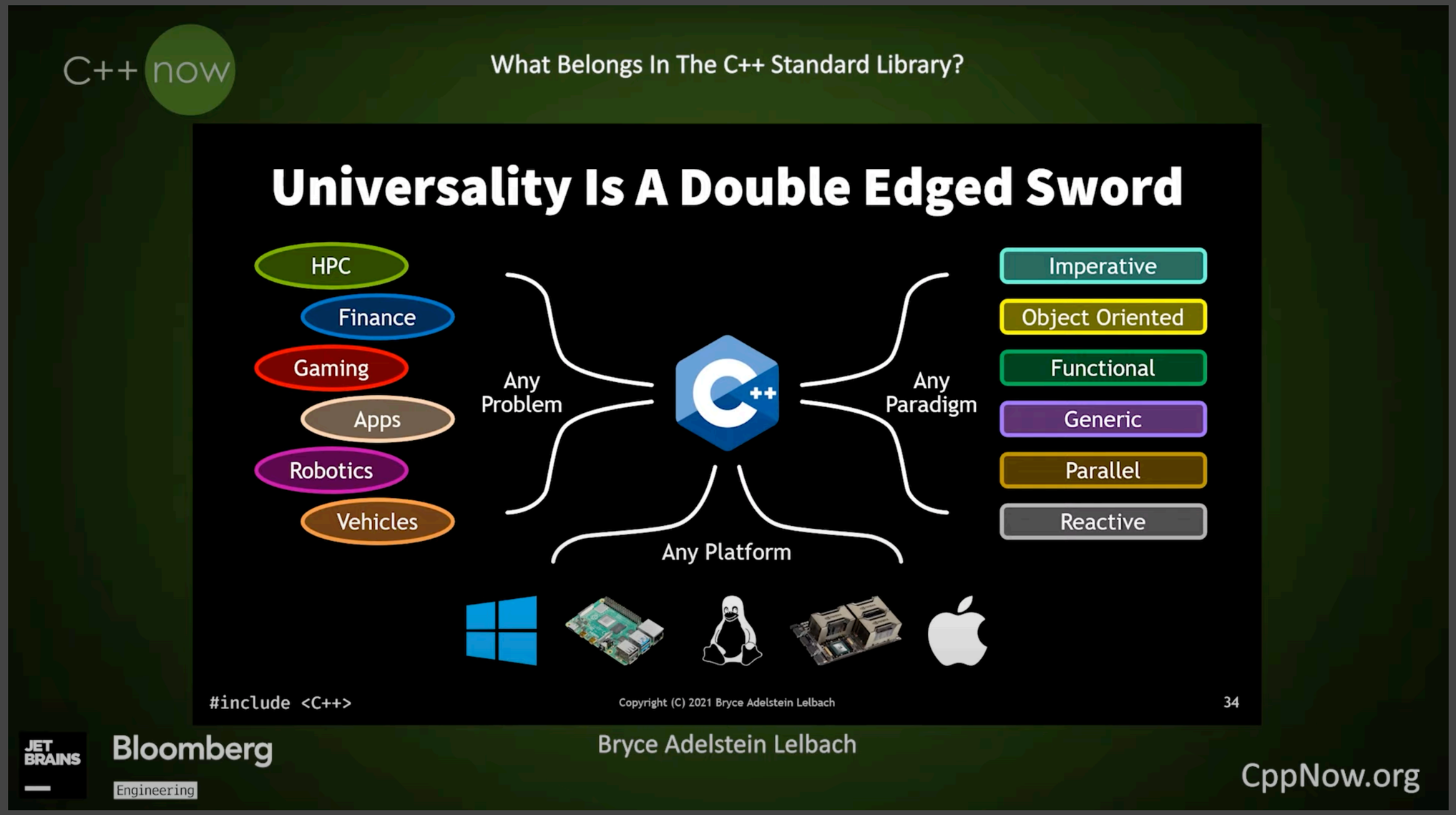
—



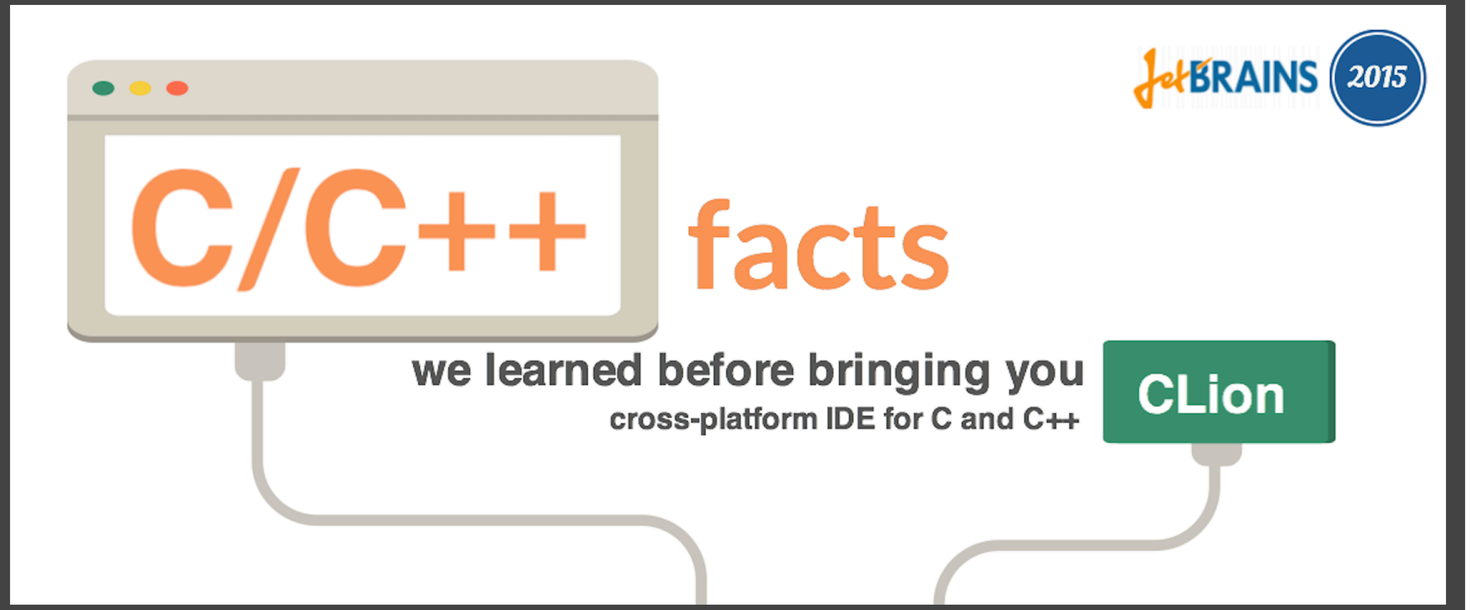
```
169
170  inline bool is_binary_op(op_code code) {
171      return code >= op_code::dot && code <= op_code::div_assign;
172  }
173
174  inline bool is_unary_op(op_code code) {
175      return code >= op_code::pre_inc && code <= op_code::ret;
176  }
177
```

C++ toolset. Where are we now?

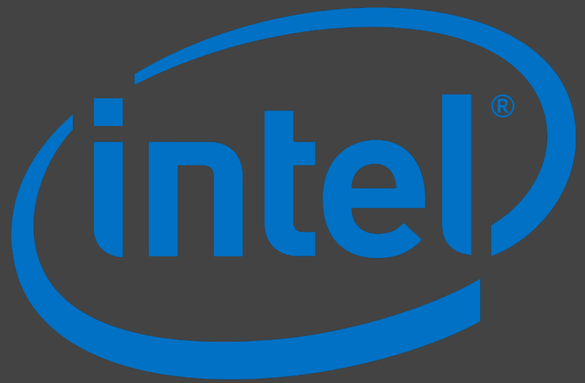
C++ toolset. Where are we now?



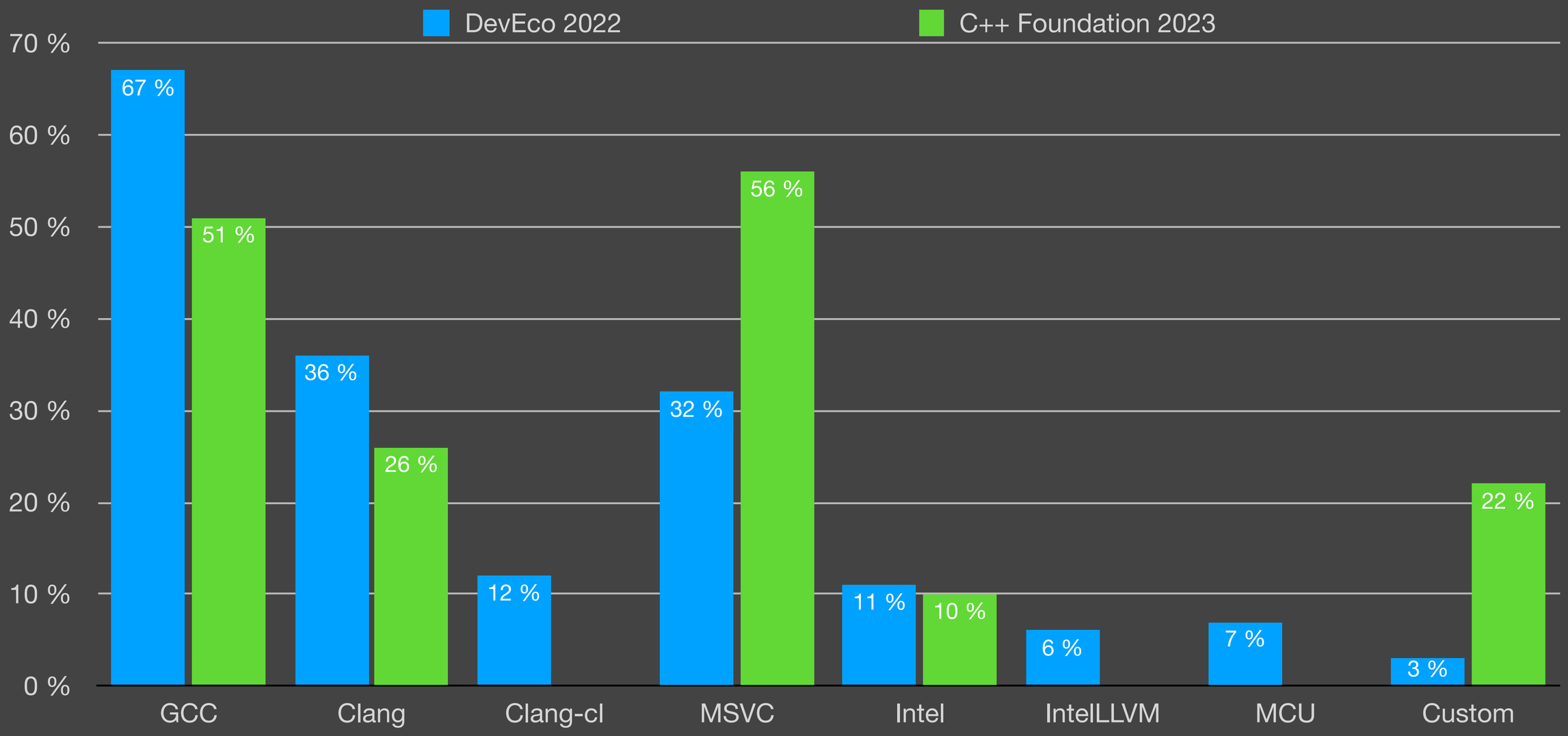
C++ toolset. Back in 2015
–



C++ compilers
–



C++ compilers: what do developers use?



Compatibility: compiler options

—

```
if (MSVC)
    add_compile_options(/W4 /WX)
else()
   add_compile_options(-Wall -Wextra -Werror)
endif()
```

Compatibility: C++ standard support –

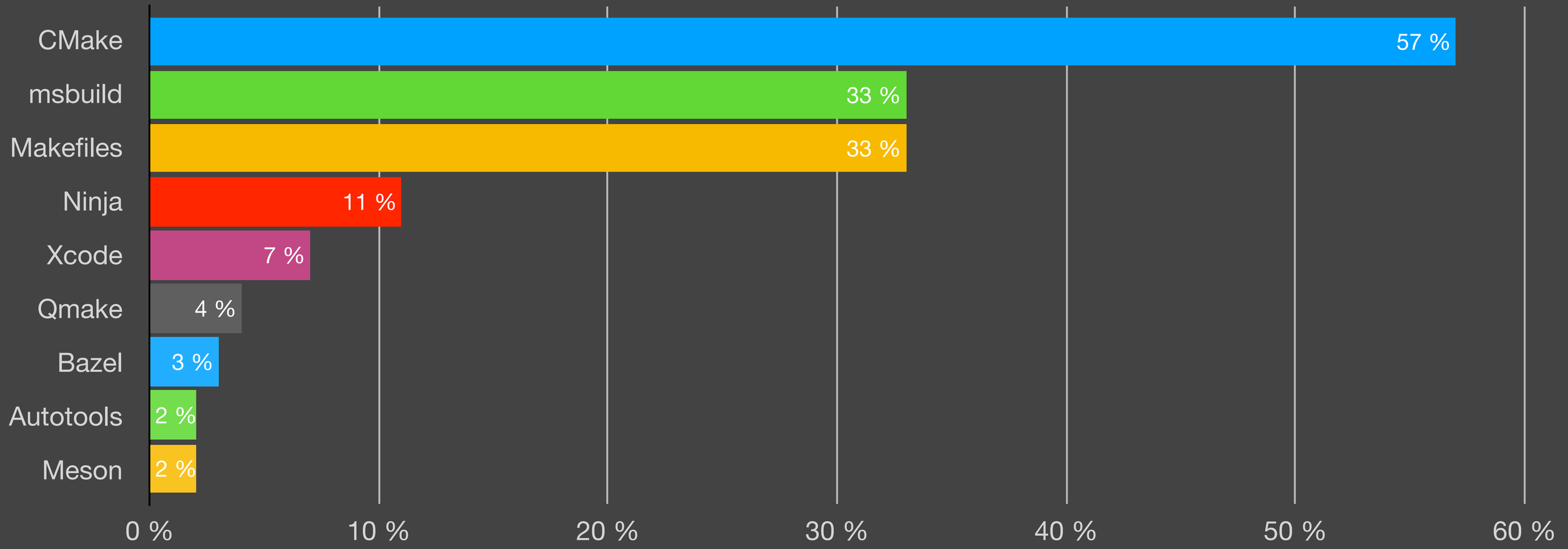
		GCC	Clang	MSVC	Apple Clang	EDG eccp	Intel C++	IBM XLC++	Sun/Oracle C++	Embarcadero C++ B	Cray	Nvidia HPC C++ (ex Portland Group)	Nvidia nvcc			
Concepts	P0734R0	6 (TS only) 10		10		19.23* (partial)* 19.30*	12.0.0* (partial)		6.1	was supported in 2021.6, removed in 2021.7				11.0	20.11	12.0
Modules	P1103R3	11 (partial)		8 (partial)		19.0 (2015)* (partial) 19.10* (TS only) 19.28 (16.8)*	10.0.1* (partial)									
Coroutines	P0912R5	10		8 (partial)		19.0 (2015)* (partial) 19.10* (TS only) 19.28 (16.8)*	10.0.1* (partial)		5.1	2021.1						12.0 (host code only)
auto(x) and auto{x}	P0849R8	12	15		14.0.3*	6.4										

Clang - beyond the compiler

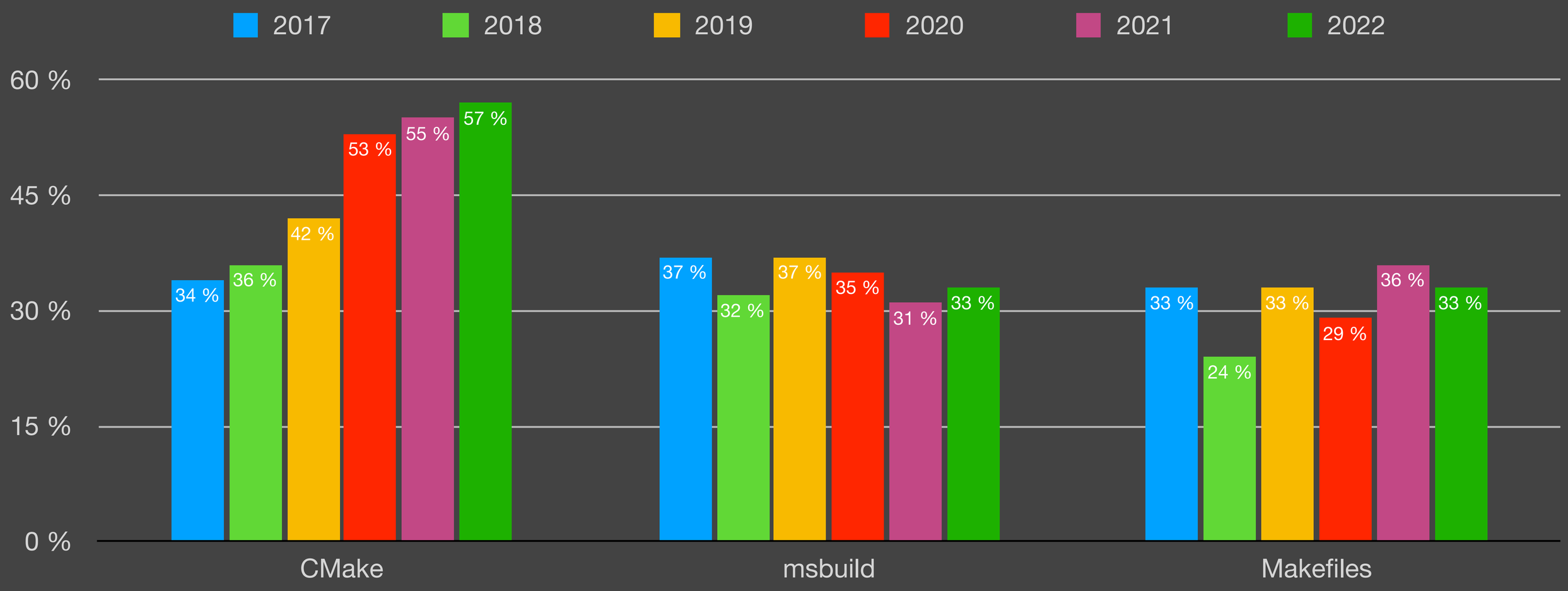
1. Clang, libclang, clangd, LSP
2. Clang Analyzer, Clang-Tidy
3. ClangFormat
4. IDEs
5. Every tool that needs AST

Clang - the basis for many C++ tools

Build system / project model



Build system / project model



Build system / project model

—

You wanna standard C++ Build System?

You got One! It’s called CMake!

CMake adoption

1. Libraries, package managers, IDEs
2. Qt6: qmake → CMake
3. Embedded: Zephyr RTOS, Nordic’s nRF5 SDK
4. CMake-buildable Boost effort
5. CMake File API
6. CMake Presets
7. …and even more!

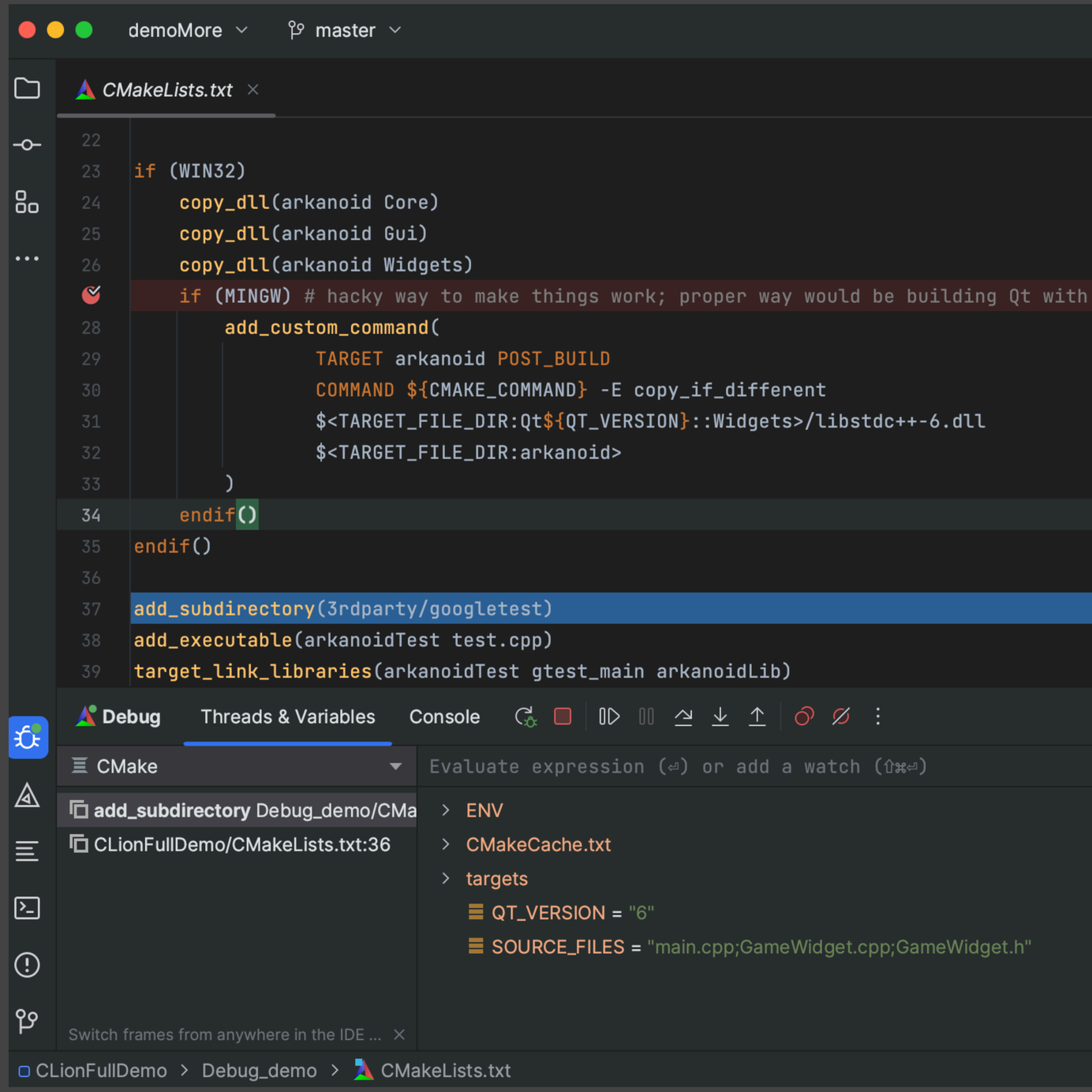
CMake as a language

—

1. Good for quizzes

   CMake + Conan: 3 Years Later - Mateusz Pusz, CppNow 2021

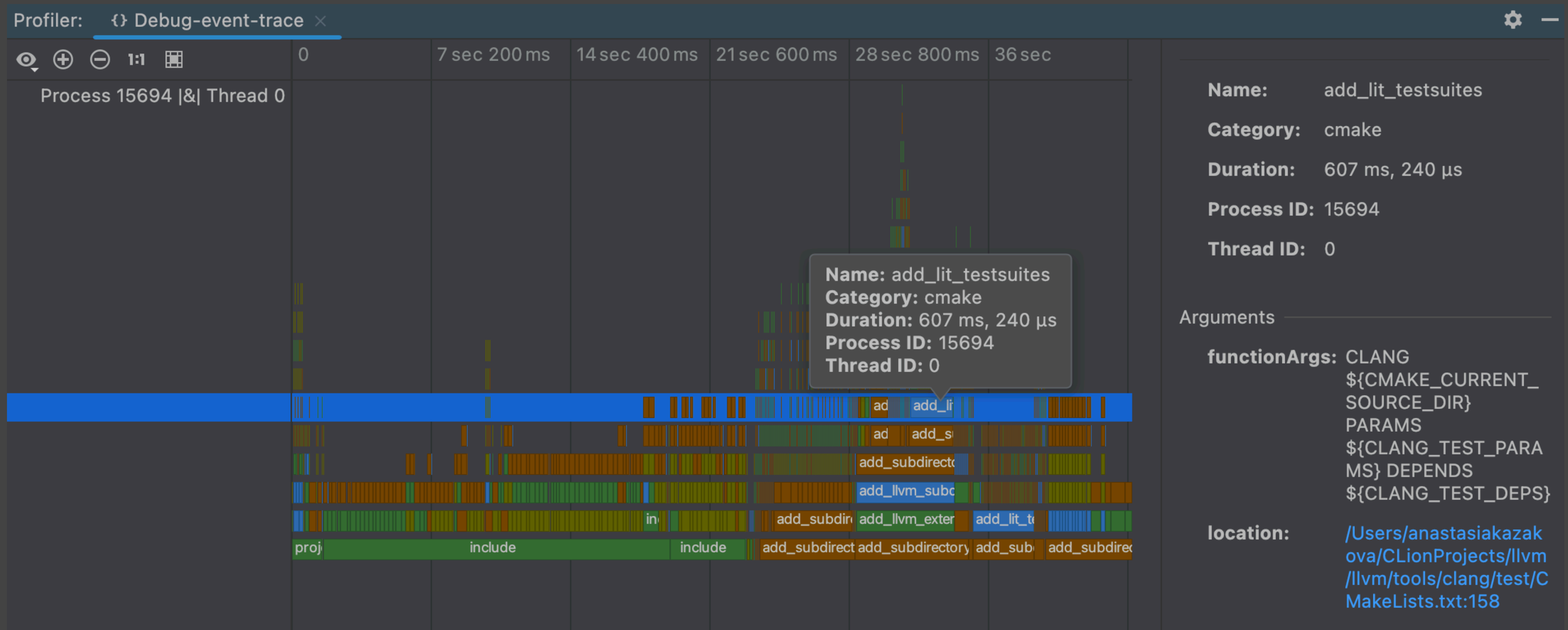
2. Needs a debugger
3. CMake 3.27: https://gitlab.kitware.com/cmake/cmake/-/merge_requests/8338



CMake as a language

–

--profiling-format=google-trace and --profiling-output=<path>



CMake and C++20 modules

1. CMake + VS generator

```
add_executable(ModuleSample a.ixx main.cpp)
```

CMake and C++20 modules
–

2. CMake + GCC/Clang compiler flags

```
function(add_module name)
    file(MAKE_DIRECTORY ${PREBUILT_MODULE_PATH})
    add_custom_target(${name}.pcm
            COMMAND
            ${CMAKE_CXX_COMPILER}
            -std=gnu++20
            -x c++
            -fmodules
            -c
            ${CMAKE_CURRENT_SOURCE_DIR}/${ARGN}
            -Xclang -emit-module-interface
            -o ${PREBUILT_MODULE_PATH}/${name}.pcm
            )
endfunction()
```

CMake and C++20 modules
–

3. Natively in CMake 3.25 (CMake examples)

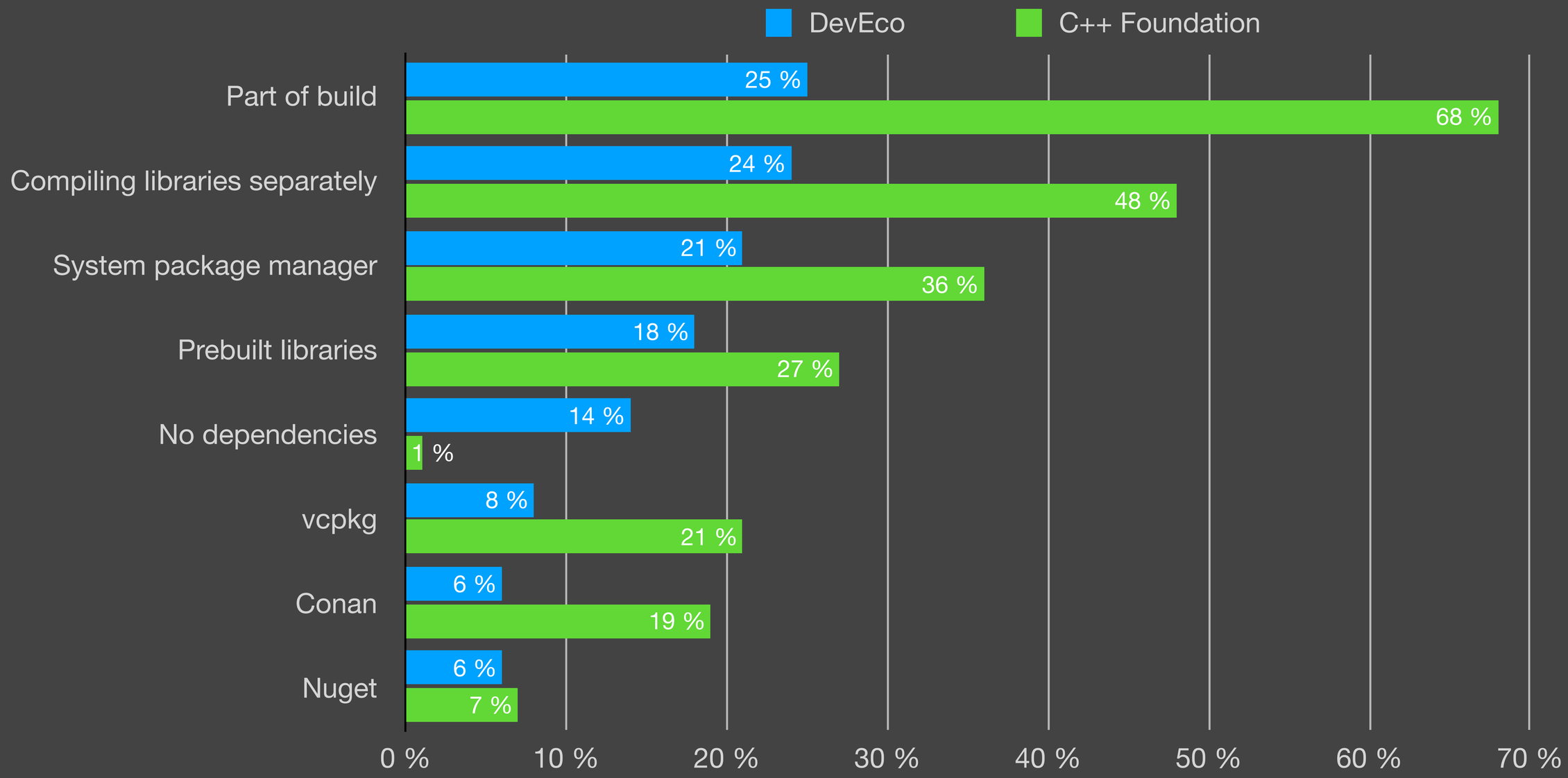
```
add_executable(simple)
target_sources(simple
        PRIVATE
          main.cxx
        PRIVATE
         FILE_SET CXX_MODULES
          BASE_DIRS
            "${CMAKE_CURRENT_SOURCE_DIR}"
          FILES
            importable.cxx)
target_compile_features(simple PUBLIC cxx_std_20)

add_test(NAME simple COMMAND simple)
```

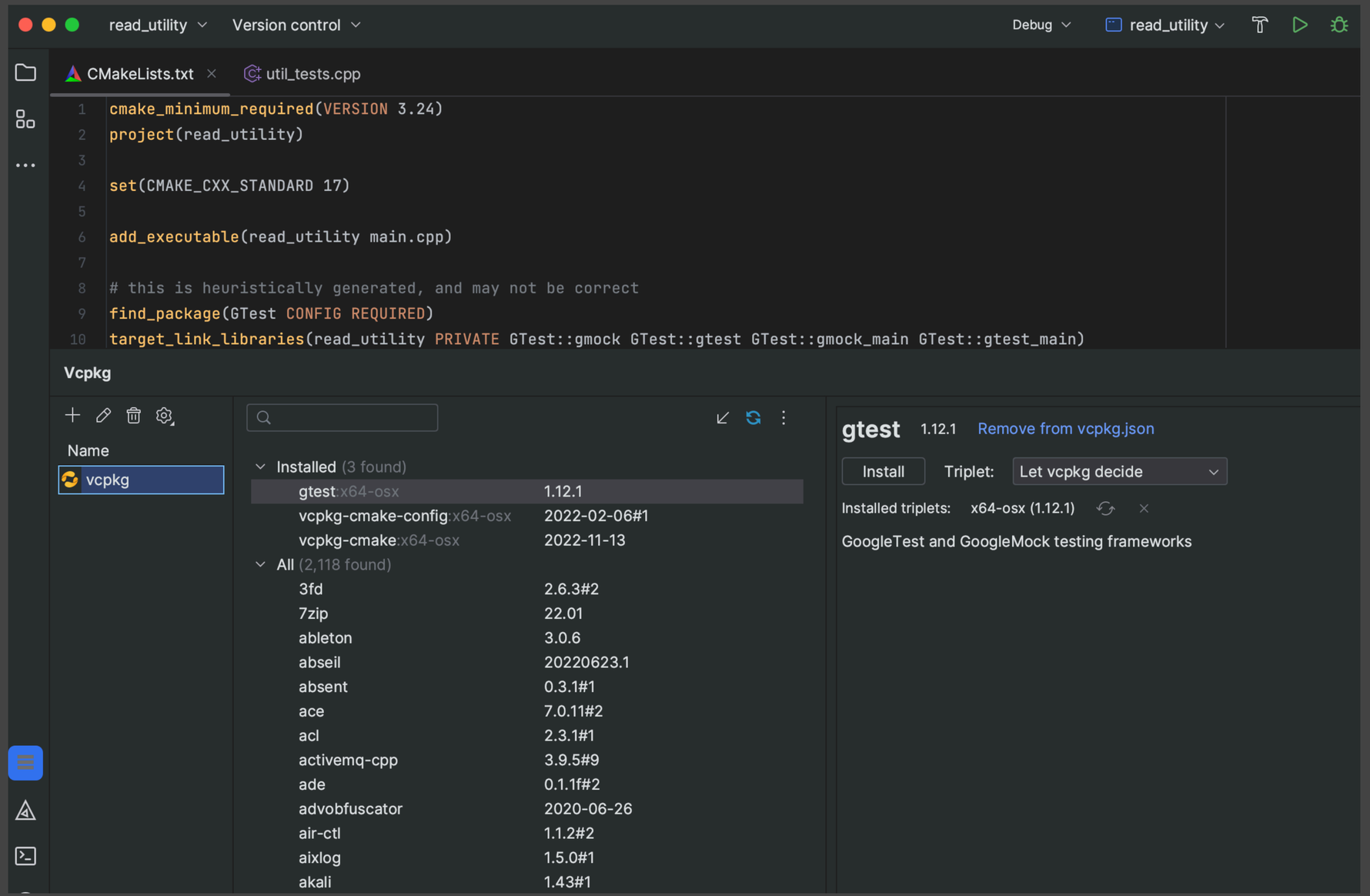
CMake and C++20 modules

4. CMake 3.26: support for modules was added into file-api

Managing dependencies in C++



Managing dependencies in C++

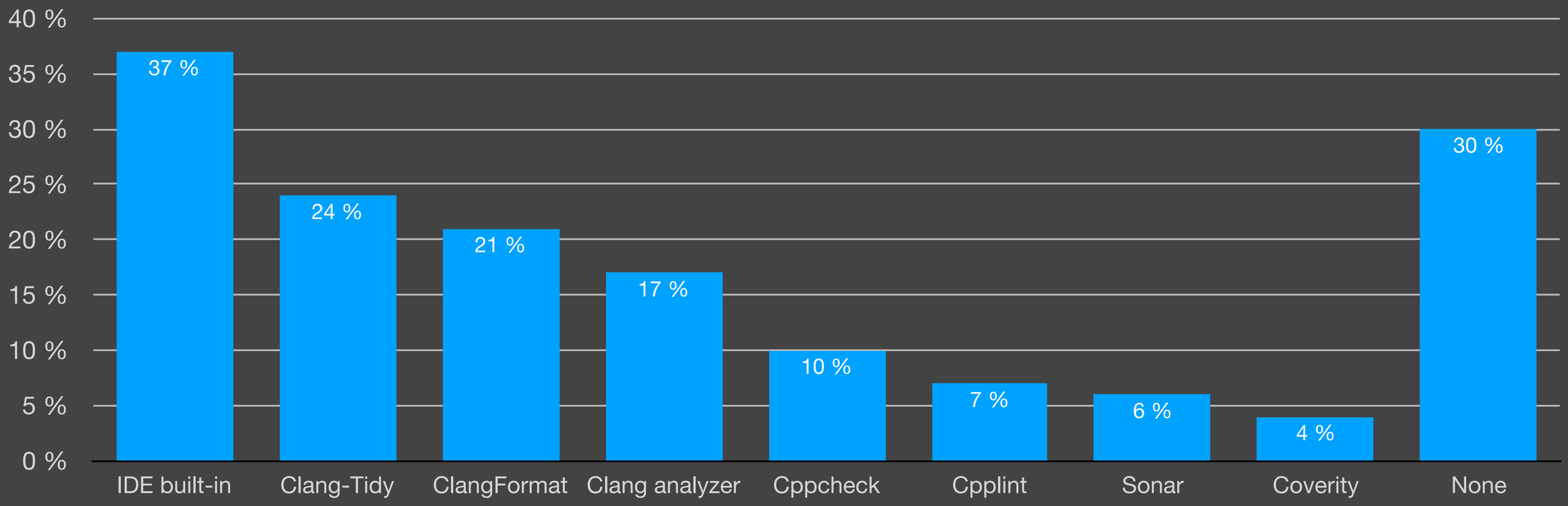


ClangFormat

-

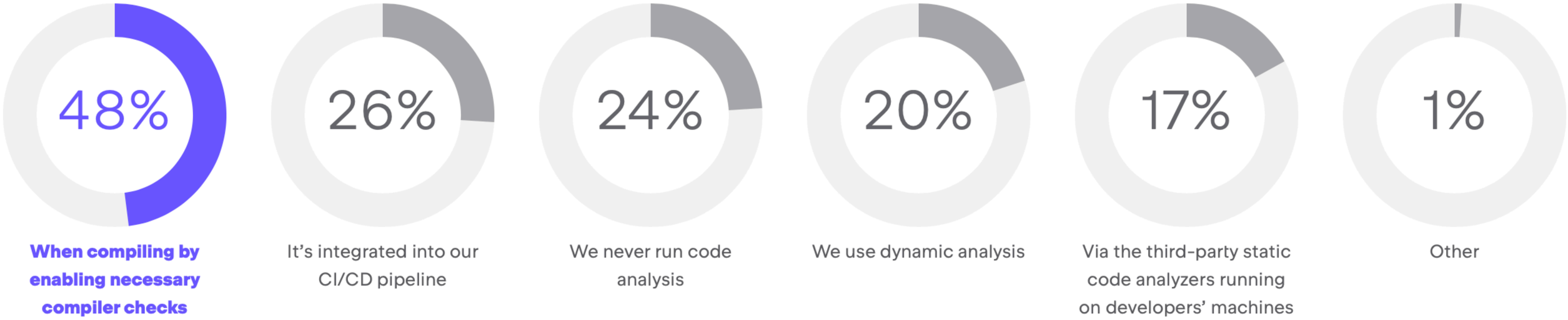
1. Standard tool
2. Fuzzy parser
3. Breaking compatibility between versions

Code Analysis: What?



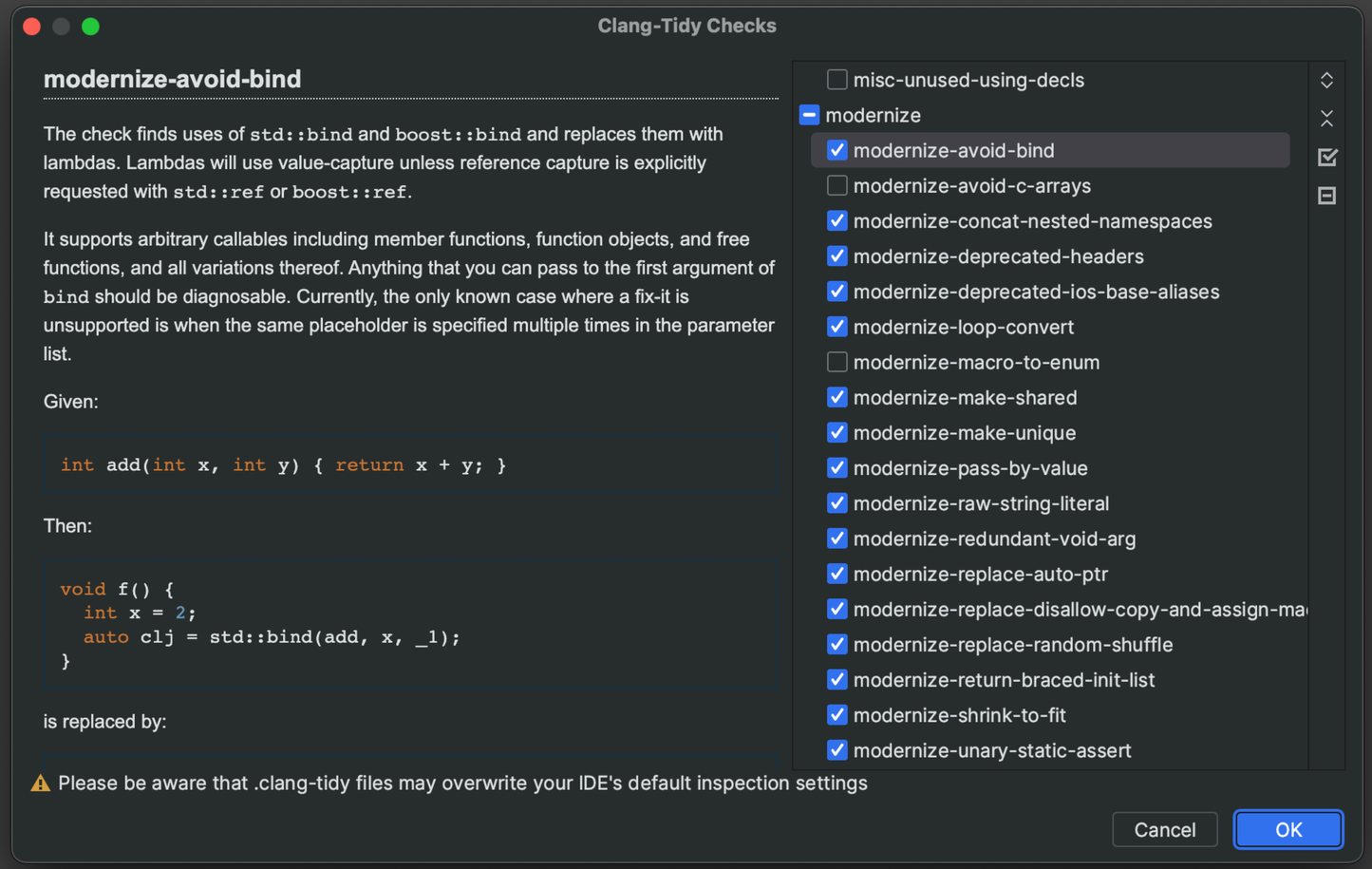
Code Analysis: When?

How do you or your team run code analysis?



Clang-Tidy

1. Baseline
2. Modernize checks, C++ Core Guidelines checks, Google checks, etc.
3. Custom checks



Data Flow Analysis

–

Dangling pointer, lifetimes, index out of bounds,

unreachable code, endless loops, etc.

```
7   #include "hu_stuff.h"
8   extern  patch_t *hu_font[HU
9
```

f_finale.c 3 warnings

Unreachable code

Index may have a value of '63' which

Index may have a value of '63' which

```
545
546 #define P2TEXT \
547 "Even the deadly Arch-Vile labyrinth could\n"\
548 "not stop you, and you've gotten to the\n"\
549 "prototype Accelerator which is soon\n"\
550 "efficiently and permanently deactivated.\n"\
551 "\n"\
552 "You're good at that kind of thing."
553
554
555 // after map 20
```

If we add an apostroph (`) here, the app will crash

Data Flow Analysis
–

Function scope → TU → CTU

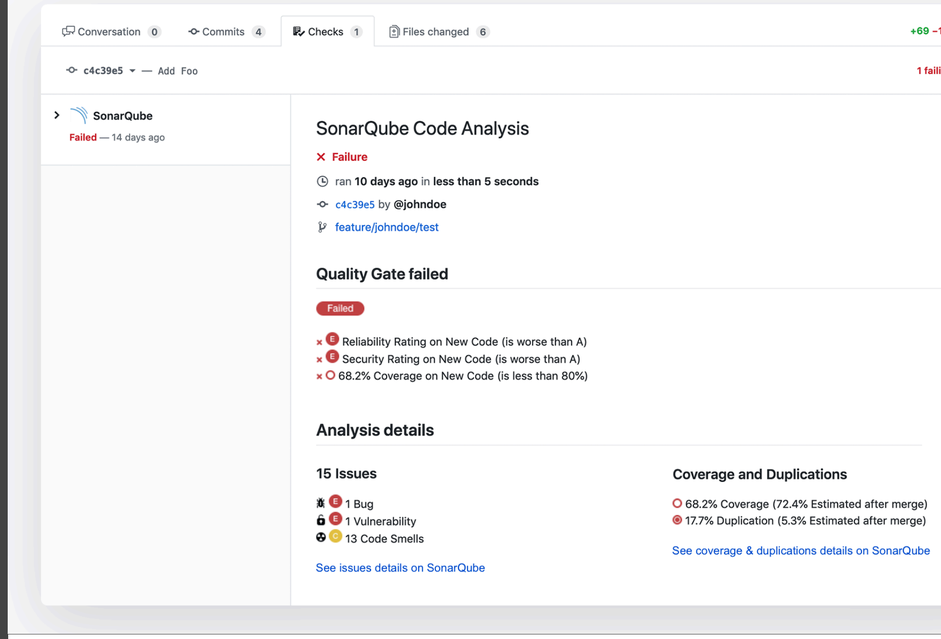
DSL analysis

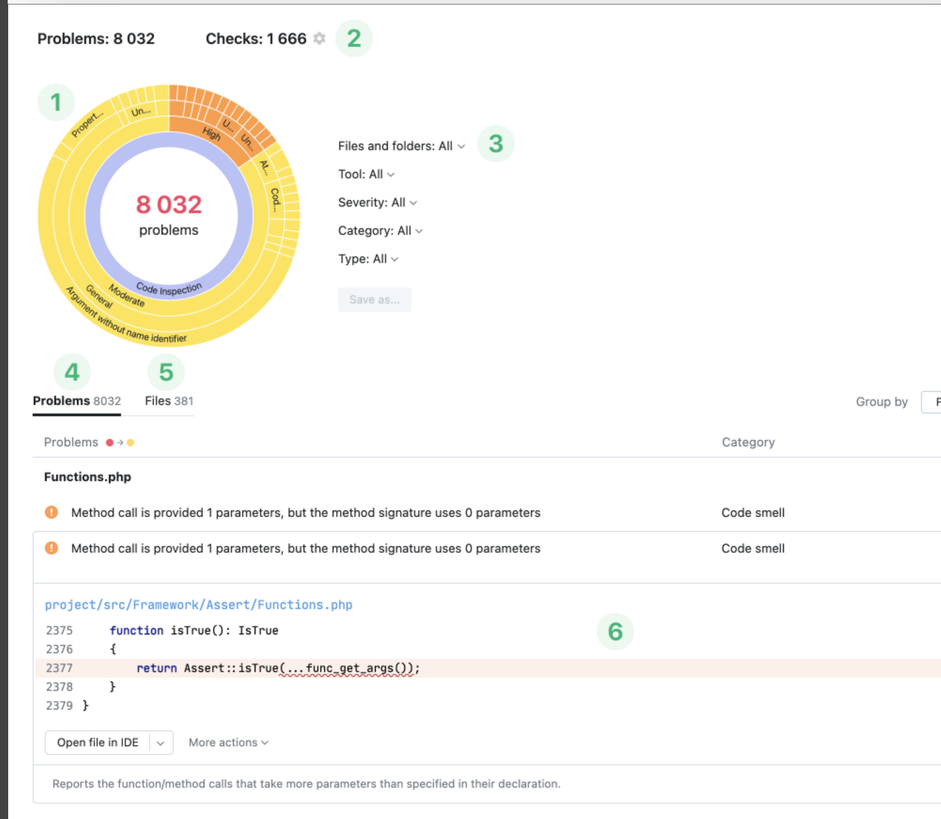
1. Clazy for Qt
2. Unreal Header Tool for UE
3. Embedded checkers

Code Analysis on CI

—

1. Code reviews & pull requests
2. Target/platform-specific checks
3. Long/resource-greedy checks
4. Health check, timeline
5. Management tool
6. Good for open-source projects
7. SonarSource tools, JetBrains Qodana





Code Analysis++

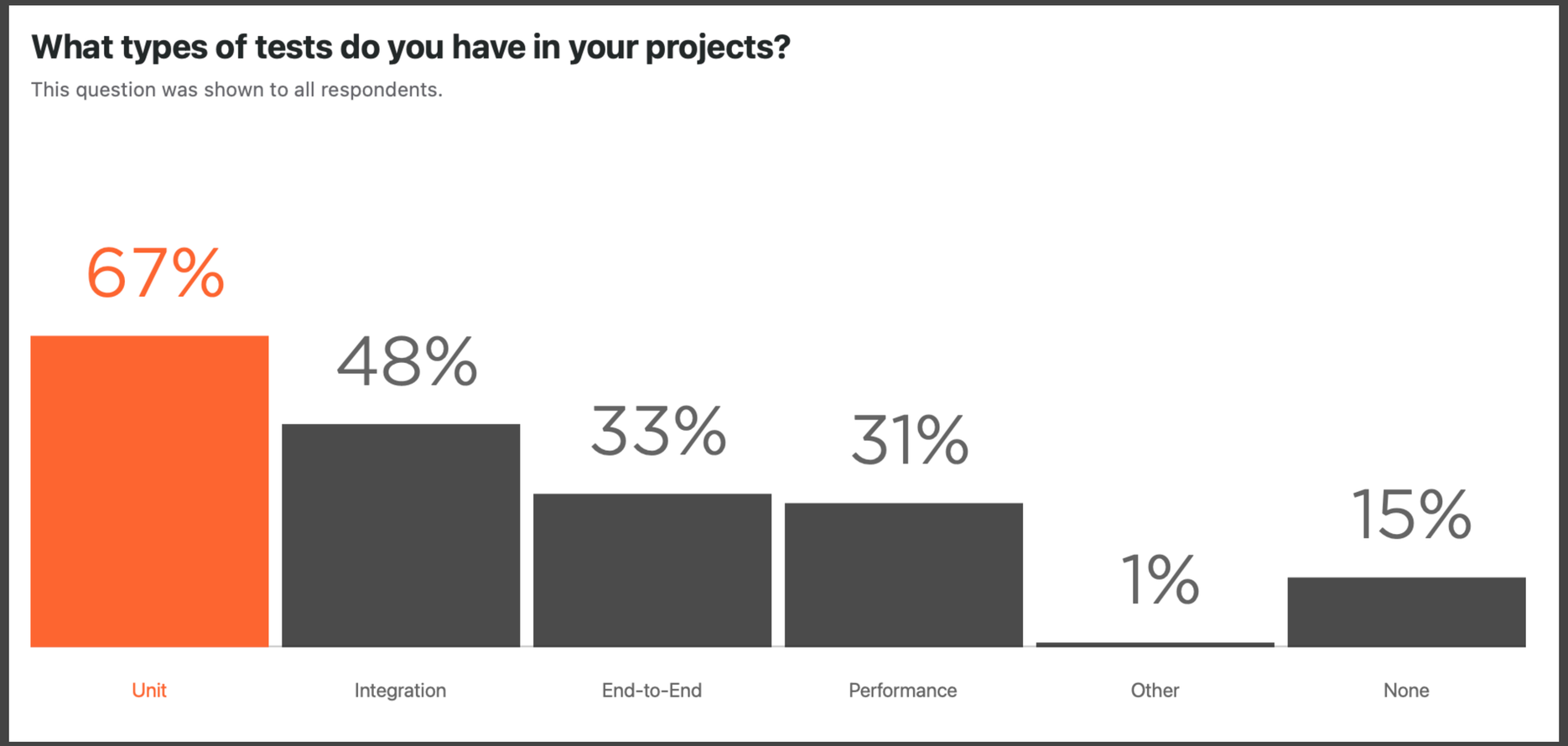
–

Find my talk from C++Now 2019 or NDC TechTown 2022

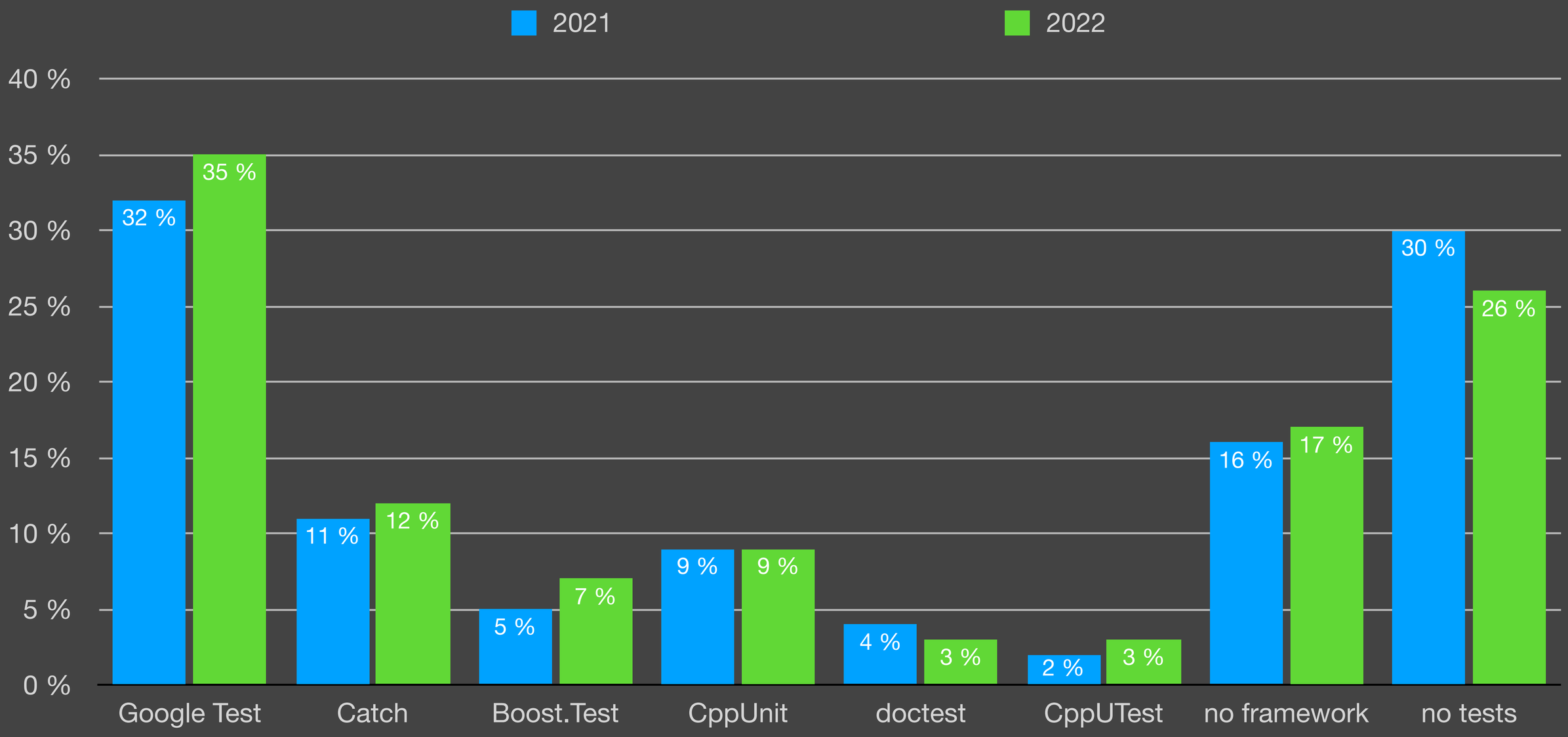
Unit Testing

–

75% of all respondents say testing plays an integral role in their development.



Unit Testing



Code Coverage
—

1. Line coverage
2. Statement coverage
3. Brach coverage

Code Coverage

1. llvm-com https://llvm.org/docs/CommandGuide/llvm-cov.html
   - Clang
   - -fprofile-instr-generate -fcoverage-mapping
2. gcov https://gcc.gnu.org/onlinedocs/gcc/Gcov.html#Gcov
   - GCC
   - -fprofile-arcs -ftest-coverage or an equivalent —coverage
   - llvm-cov gcov with Clang since v4.2

TL;DR

1. Do we have a standard C++ toolset? Not as Swift or Rust.
2. Do we have commonly used tools for various use cases? Yes.
3. Do we miss some tools? Yes.
4. Can we unify more? Yes.

References

1. [2023 Annual C++ Developer Survey, C++ Foundation](https://isocpp.org/files/papers/CppDevSurvey-2023-summary.pdf)
2. [JetBrains Developer Ecosystem 2022](https://www.jetbrains.com/lp/devecosystem-2022/cpp/)
3. [ReSharper C++ Syntax Style](https://blog.jetbrains.com/rscpp/2021/03/30/resharper-cpp-2021-1-syntax-style/)
4. [C++ Market Infographic, 2015](https://blog.jetbrains.com/clion/2015/07/infographics-cpp-facts-before-clion/)
5. [What Belongs In The C++ Standard Library? Bryce Adelstein Lelbach](https://www.youtube.com/watch?v=OgM0MYb4DqE)
6. [Code Analysis++, C++Now 2019](https://www.youtube.com/watch?v=qUmG61aQyQE)