C++ ecosystem: the renaissance edition

Anastasia Kazakova

@anastasiak2512
anastasia.kazakova@jetbrains.com

C++ on Sea Online 2020

Intro

- From C++ developer to C++ and .NET product marketing manager
- OKTET Labs, Microsoft Research, Yota / RooX, JetBrains
- St. Petersburg C++ User Group
  - https://www.meetup.com/St-Petersburg-CPP-User-Group/
- C++ Tools PMM

Agenda

1. C++ today
2. Ecosystem researches
3. C++ in 2020
4. Top usages areas
5. C++ standardization
6. To wrap up

Agenda

1. C++ today
2. Ecosystem researches
3. C++ in 2020
4. Top usages areas
5. C++ standardization
6. To wrap up

C++ today (yesterday)

```
#define X(a) myVal_##a,
enum myShinyEnum {
#include "xmacro.txt"
};
#undef X

void handle_value(myShinyEnum en) {
    switch (en) {
        case myVal_a:break;
        case myVal_b:break;
        case myVal_c:break;
        case myVal_d:break;
    }
}
```

```
//xmacro.txt

X(a)
X(b)
X(c)
X(d)
```

C++ today

```
template <class T> int foo(T & t)   { return 1; }
template <class T> int foo(T && t)  { return 2; }
int foo(signed char t)              { return 3; }
int foo(unsigned char t)            { return 4; }
int foo(int)                        { return 5; }

int main() {
    auto C = 'C';
    std::cout << foo(C) << foo('C')
              << foo('C++') << foo("C++");
}
```

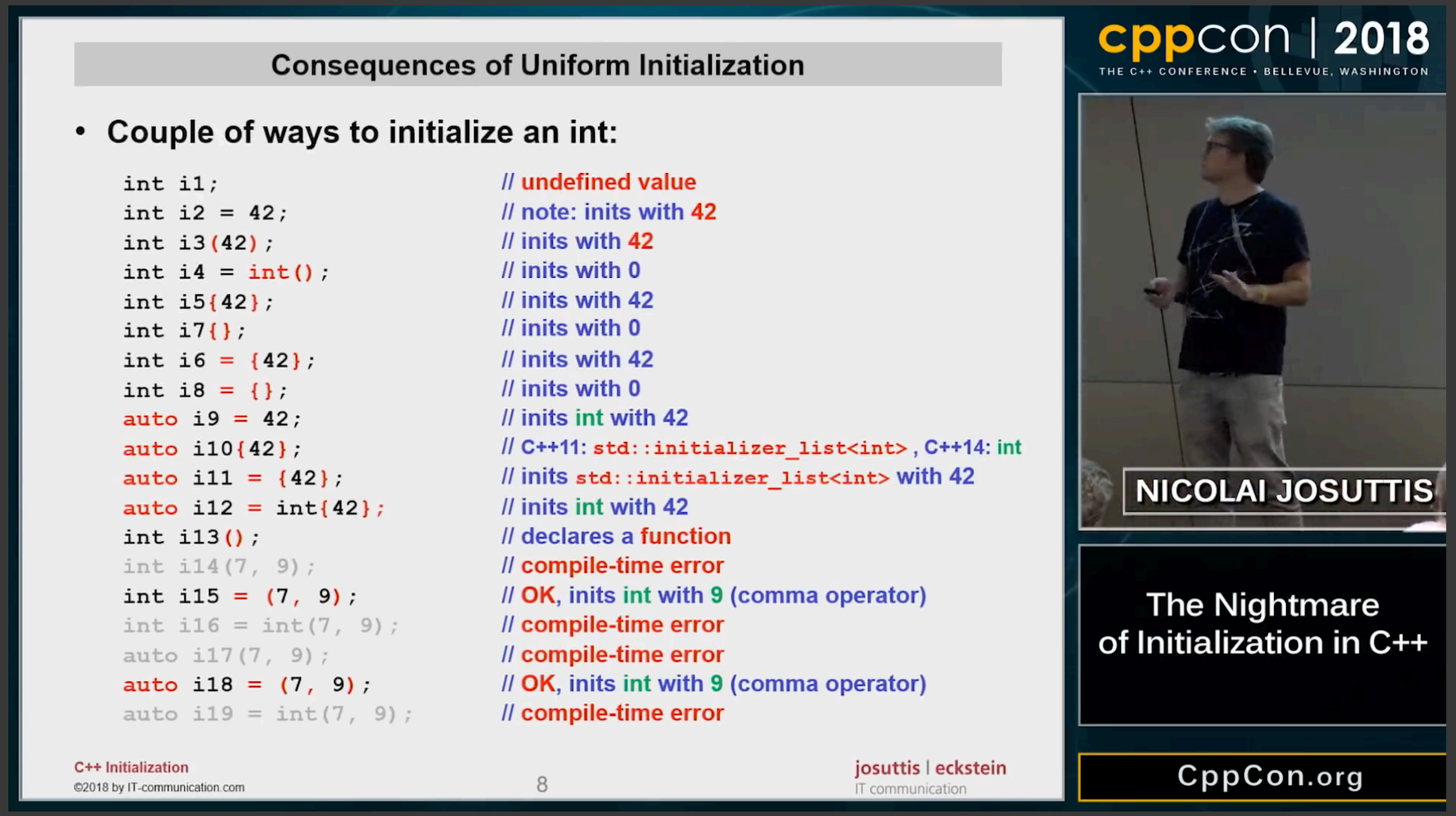
C++ today

```cpp
std::ostream& operator<<(std::ostream& out, const Fraction& f) {
    return out << f.num() << '/' << f.den();
}

void fraction_sample() {
    Fraction f1(3, 8), f2(1, 2);

    std::cout << f1 << " * " << f2 << " = " << f1 * f2 << '\n';
}
```

C++ today

```
template<class T, int ... X>
T pi(T(X...));

int main() {
    return pi<int, 42>;
}
```

C++ today


Consequences of Uniform Initialization
• Couple of ways to initialize an int:
int i1; // undefined value
int i2 = 42; // note: inits with 42
int i3(42); // inits with 42
int i4 = int(); // inits with 0
int i5{42}; // inits with 42
int i7{}; // inits with 0
int i6 = {42}; // inits with 42
int i8 = {}; // inits with 0
auto i9 = 42; // inits int with 42
auto i10{42}; // C++11: std::initializer_list<int> , C++14: int
auto i11 = {42}; // inits std::initializer_list<int> with 42
auto i12 = int{42}; // inits int with 42
int i13(); // declares a function
int i14(7, 9); // compile-time error
int i15 = (7, 9); // OK, inits int with 9 (comma operator)
int i16 = int(7, 9); // compile-time error
auto i17(7, 9); // compile-time error
auto i18 = (7, 9); // OK, inits int with 9 (comma operator)
auto i19 = int(7, 9); // compile-time error
C++ Initialization
©2018 by IT-communication.com
8
josuttis | eckstein
IT communication
cppcon | 2018
THE C++ CONFERENCE • BELLEVUE, WASHINGTON
NICOLAI JOSUTTIS
The Nightmare of Initialization in C++
CppCon.org

C++ today

“Problem is, just because the “features” are there, some people will use them. If you’re coding alone, all is peachy. But working in a team?
10 ways of doing 1 thing != good language.”
(Twitter, @ArenMook, 24 Dec 2018)

C++ today

“With a sufficient number of uses of an API, it does not matter what you promise in the contract: all observable behaviours of your system will be depended on by somebody.”

(Hyrums Law,

Software Engineering at Google,

by Titus Winter, Tom Manshrek, Hyrum Wright)

C++ today

“C++ developers must suffer!”

(ACCU 2018, JetBrains booth)

“Cognitive overhead”

(comments to Sean Parent

"Modern" C++ Ruminations)

“Need naive simple and debuggable code”

(comments to @aras_p tweet)

“Towards a more powerful

and simpler C++”

(Herb Sutter)

Agenda

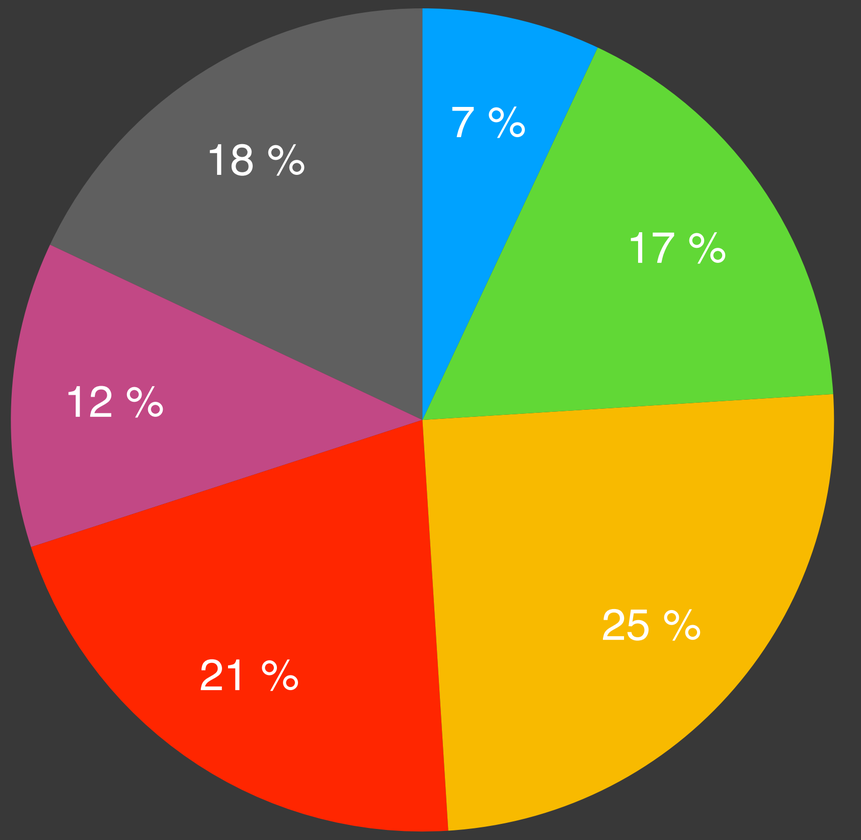
1. C++ today
2. Ecosystem researches
3. C++ in 2020
4. Top usages areas
5. C++ standardization
6. To wrap up

Ecosystem researches

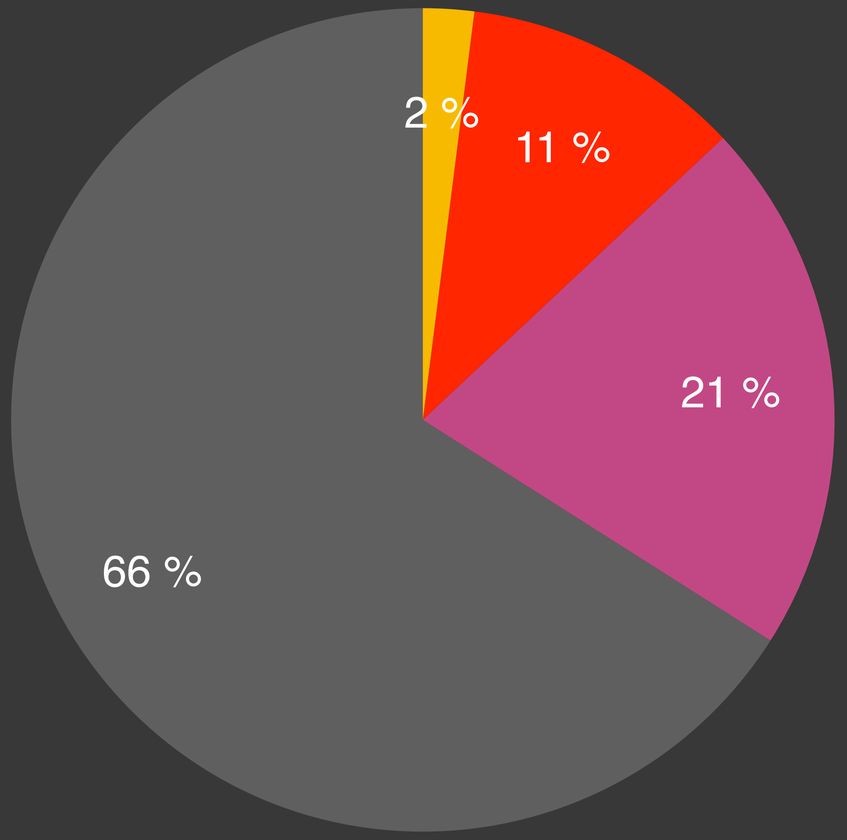
- C++ Foundation “Lite” Survey
  - 1000+
- JetBrains Developers Ecosystem
  - 1800+ / 19000+

Ecosystem researches

Dev Eco audience 2020

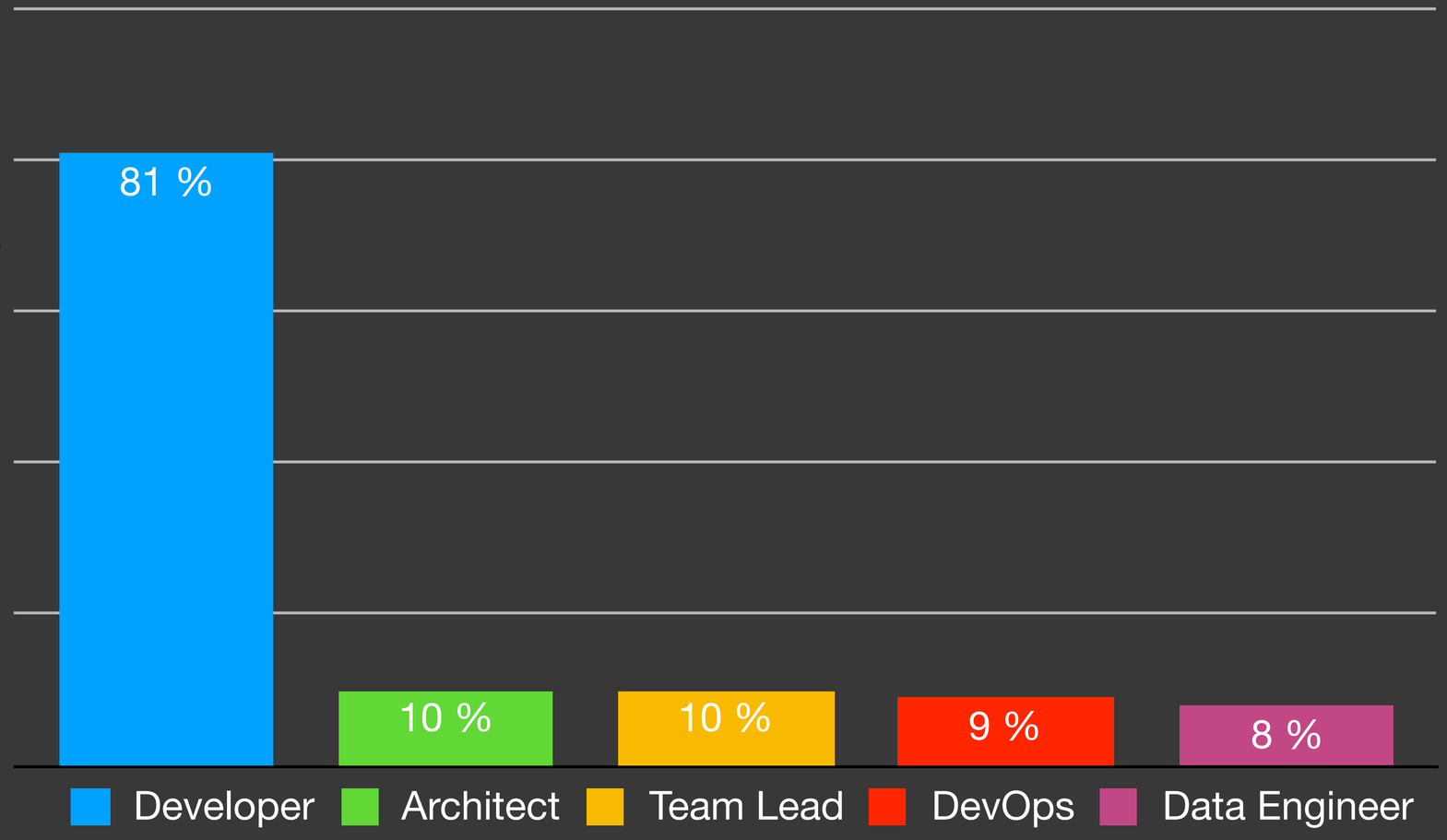


C++ Foundation Lite audience 2020



Ecosystem researches

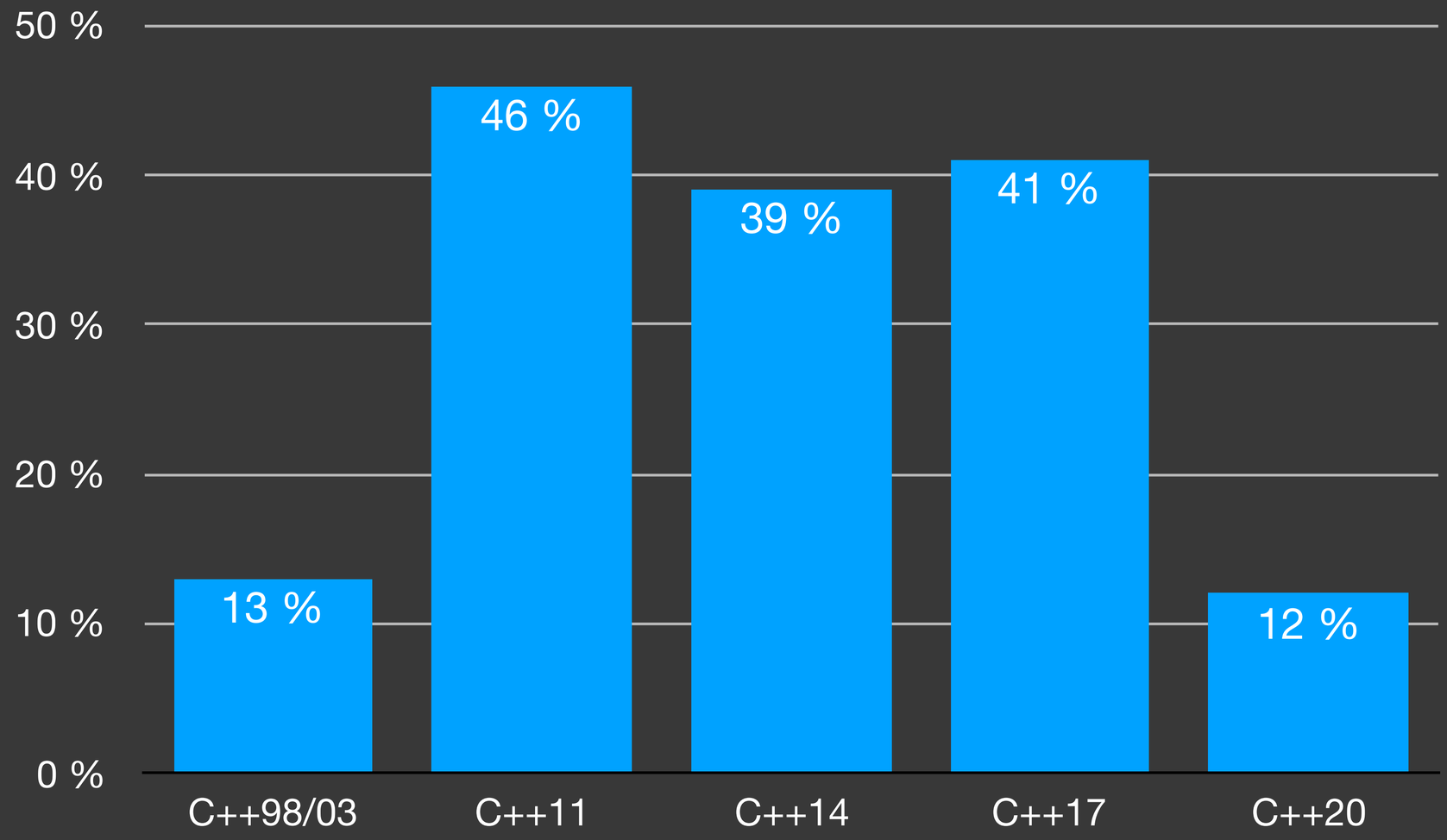
- Fully employed - 42%
- Student - 30%
- Working students - 16%



Agenda

1. C++ today
2. Ecosystem researches
3. C++ in 2020
4. Top usages areas
5. C++ standardization
6. To wrap up

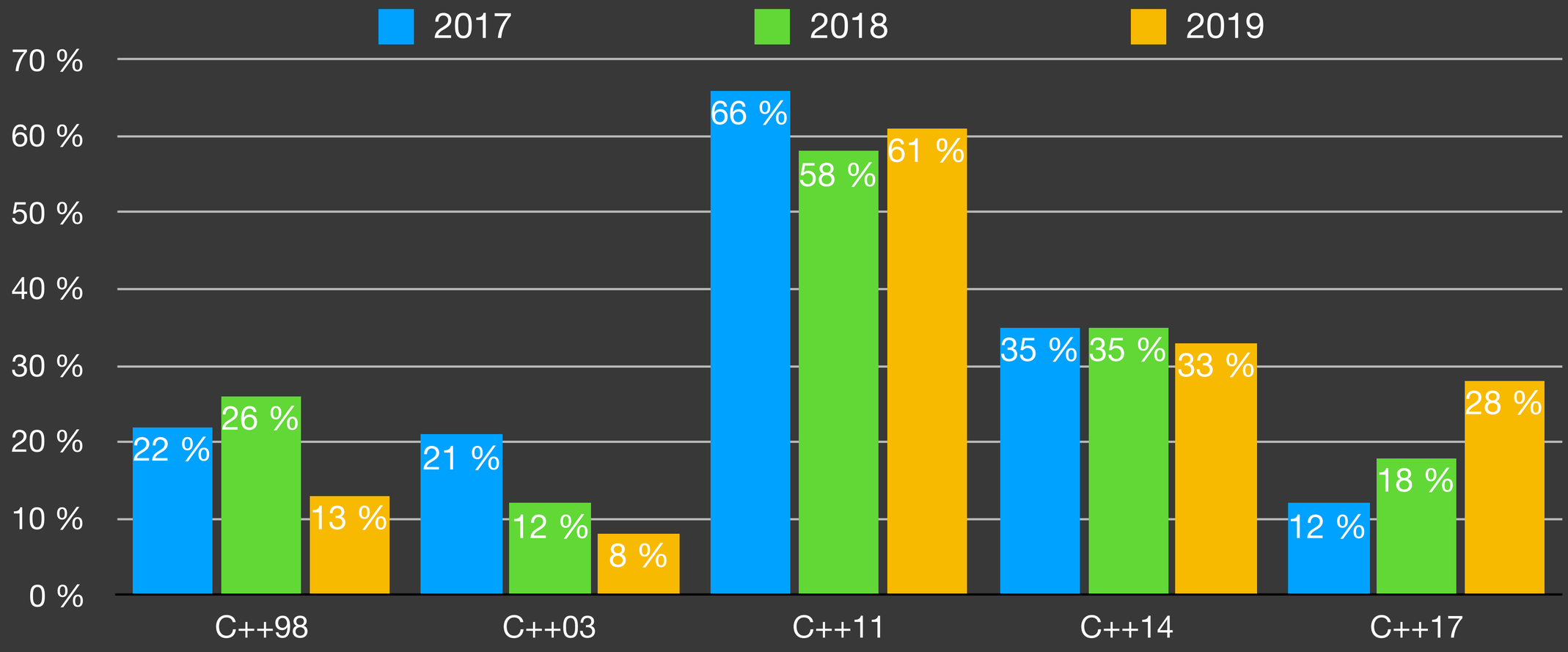
C++ in 2020



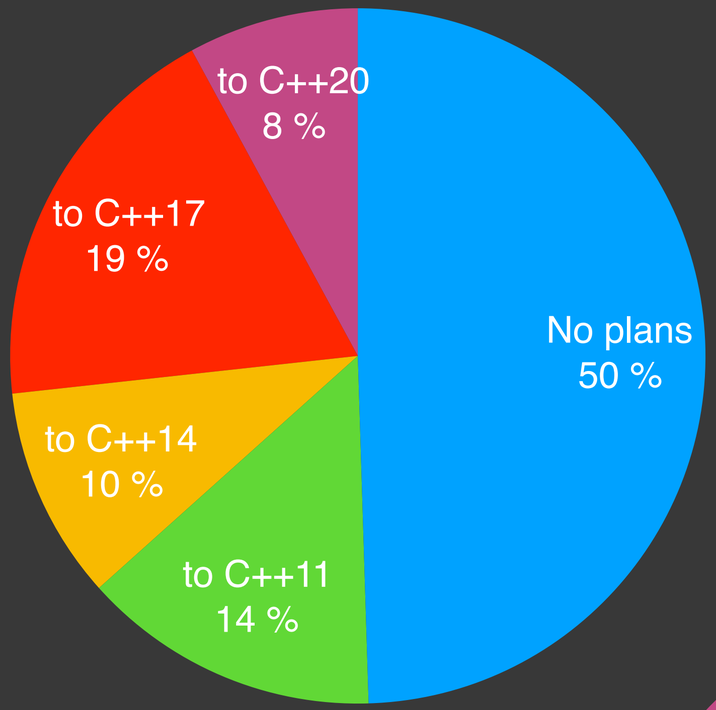
C++ in 2020

C++ Standards	Features	YES: PRETTY MUCH ALL	PARTIAL: LIMITED FEATURES/	NO: NOT ALLOWED
C++98/03	(e.g., exceptions, templates, RTTI)	82 %	13 %	5 %
C++11	(e.g., auto, move semantics, =delete/ =default, shared_ptr, lambdas)	89 %	7 %	3 %
C++14	(e.g., generic lambdas, auto return types, general constexpr functions)	74 %	15 %	11 %
C++17	(e.g., if constexpr, if/switch scoped variables, structured bindings, string_view, optional/any/variant, Parallel STL)	54 %	21 %	25 %
C++20	(e.g., concepts, coroutines, modules)	22 %	17 %	61 %

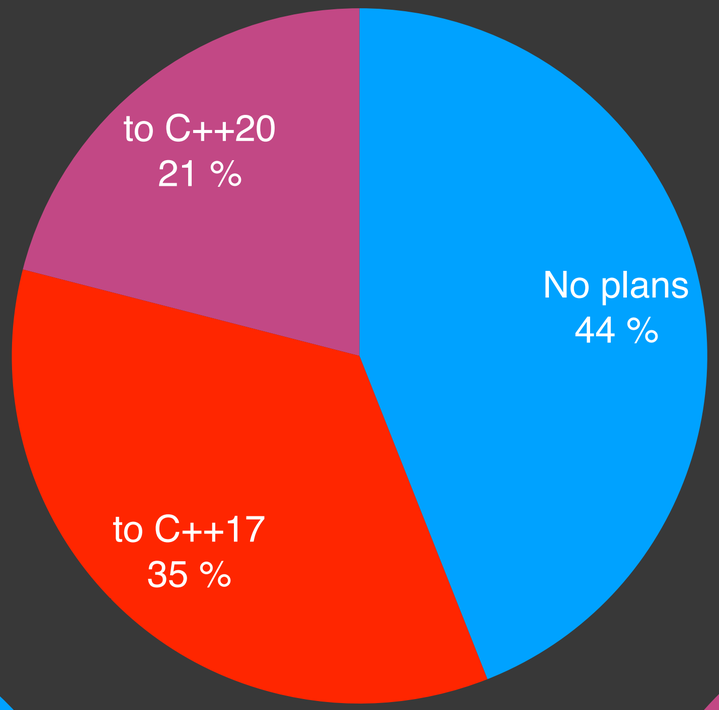
C++ in 2020


2017
2018
2019
70 %
60 %
50 %
40 %
30 %
20 %
10 %
0 %
22 %
26 %
13 %
21 %
12 %
8 %
66 %
58 %
61 %
35 %
35 %
33 %
12 %
18 %
28 %
C++98
C++03
C++11
C++14
C++17

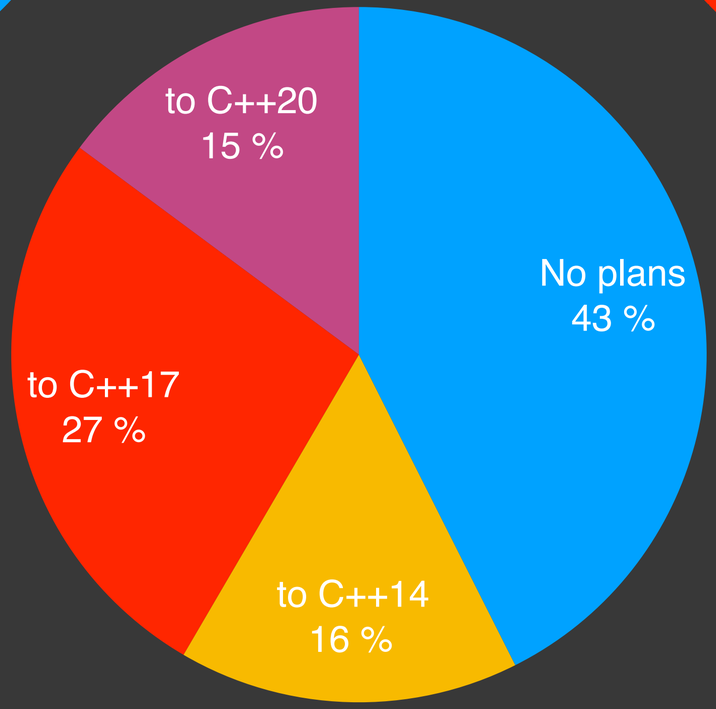
From C++98/03



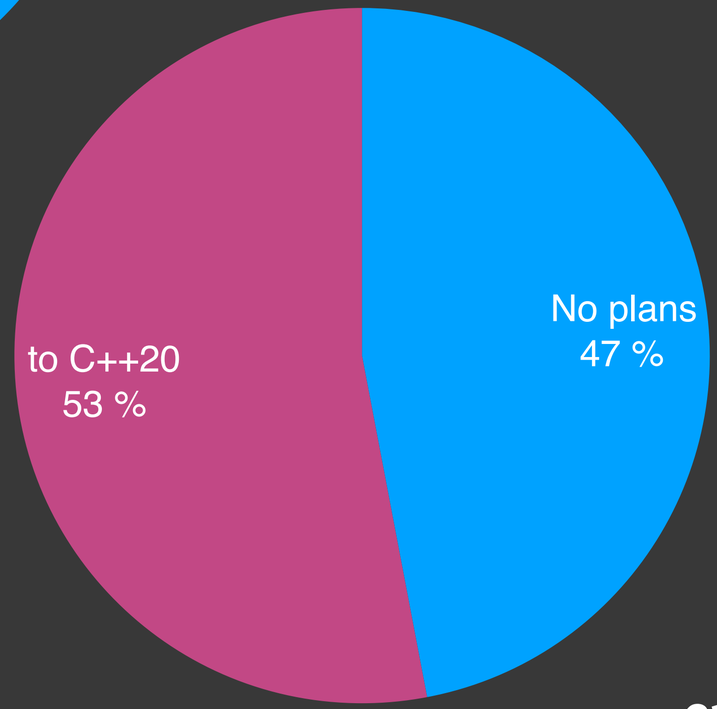
From C++14



From C++11



From C++17



C++ in 2020

“Moving existing code to the latest language standard”
– major pain point for 7%
(C++ Foundation “Lite” 2020)

C++ in 2020

1. “Managing libraries my application depends on”

– major pain point for 47%

2. “Build times”

– major pain point for 42%

3. “Setting up a continuous integration pipeline from scratch (automated builds, tests, …) ”

– major pain point for 32%

C++ in 2020

- Risks when moving to a newer standard:
  - Libraries, SDKs, etc.
  - Compiler support is missing.
  - Other tooling support is missing (IDE, static analyzer, etc.)
  - Things compile but don’t link.
  - Things compile but work differently.
  - Developers expertise is lower.

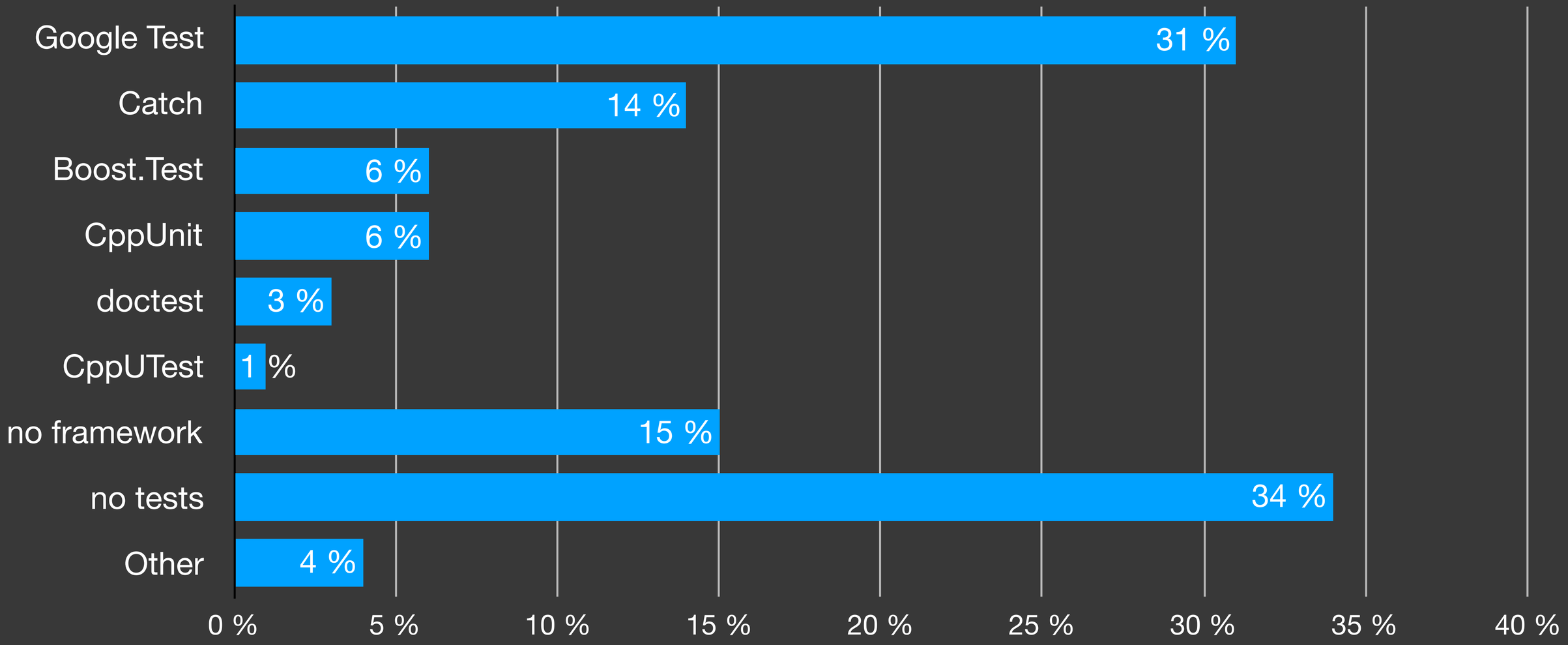
C++ in 2020: Unit Testing

- 101 “reasons” not to write unit tests
  - This is GUI code
  - This depends on disk / network / etc.
  - I already know the code is correct.
  - I need to ship this thing.

[CppCon 2016] David Sankel

“Building Software Capital: How to write the highest quality code and why"

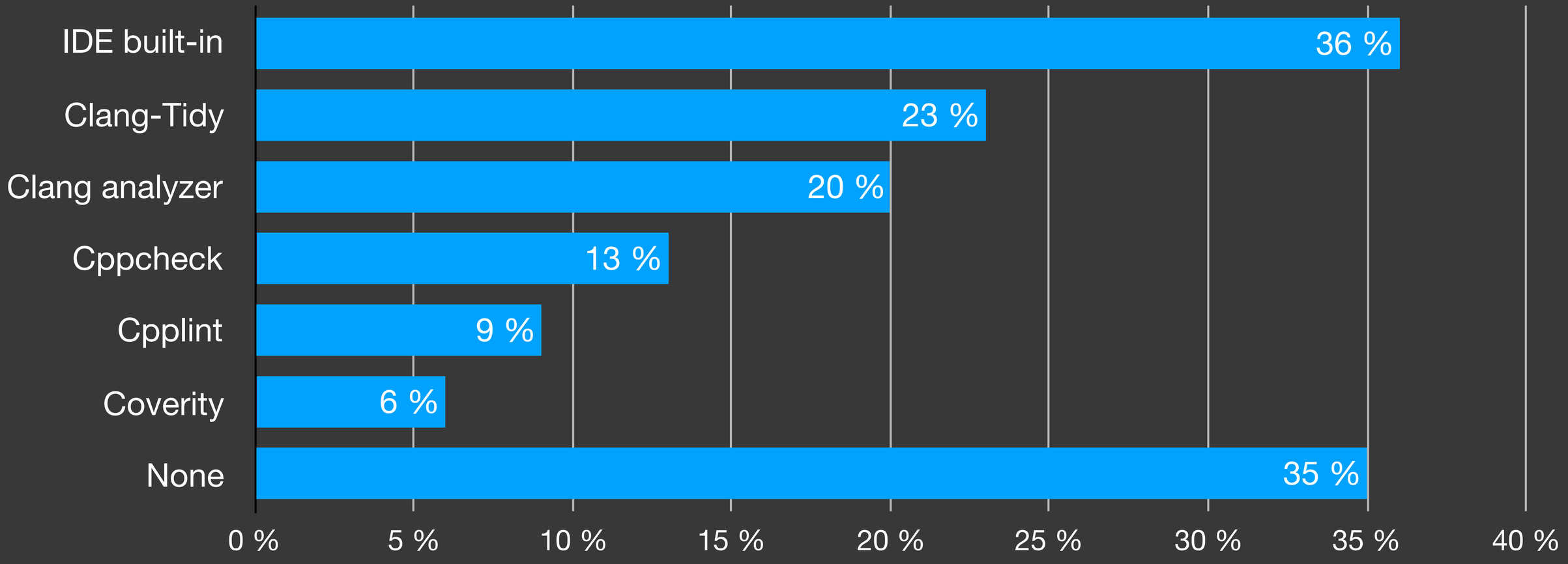
C++ in 2020: Unit Testing



C++ in 2020: Static Analysis

- Adoption static analysis into your workflow:
  - I don’t run it
  - I run it manually from time to time
  - I run it weekly (Friday evenings) on the CI server
  - I keep it always-on in my IDE

C++ in 2020: Static Analysis



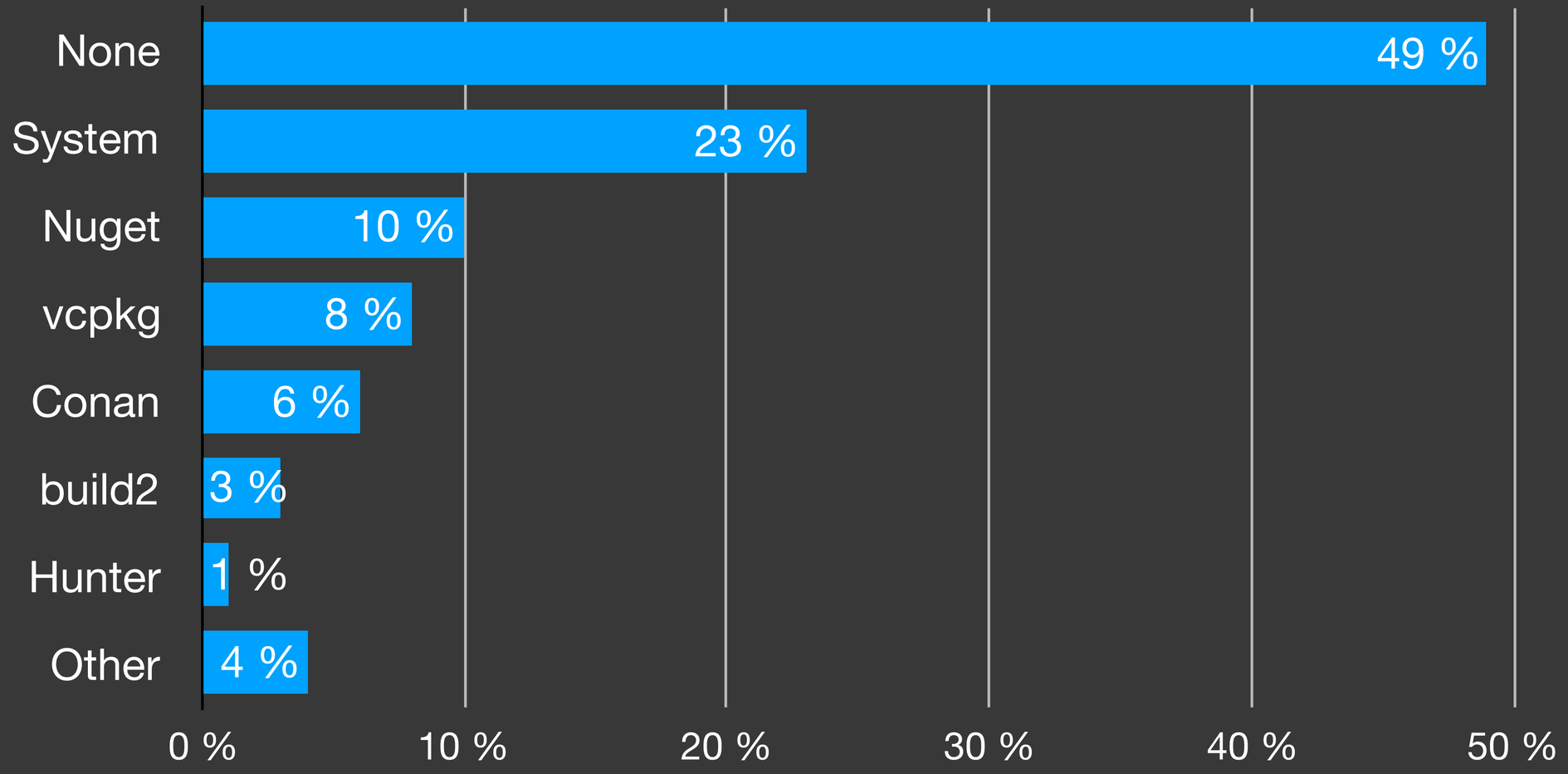
C++ in 2020

1. “Managing libraries my application depends on”
– major pain point for 47%

2. “Build times”
– major pain point for 42%

3. “Setting up a continuous integration pipeline from scratch (automated builds, tests, …) ”
– major pain point for 32%

C++ in 2020: dependency management



C++ in 2020: dependency management

1. “The library source code is part of my build” – 67%

2. “I compile the libraries separately using their instructions” – 58%

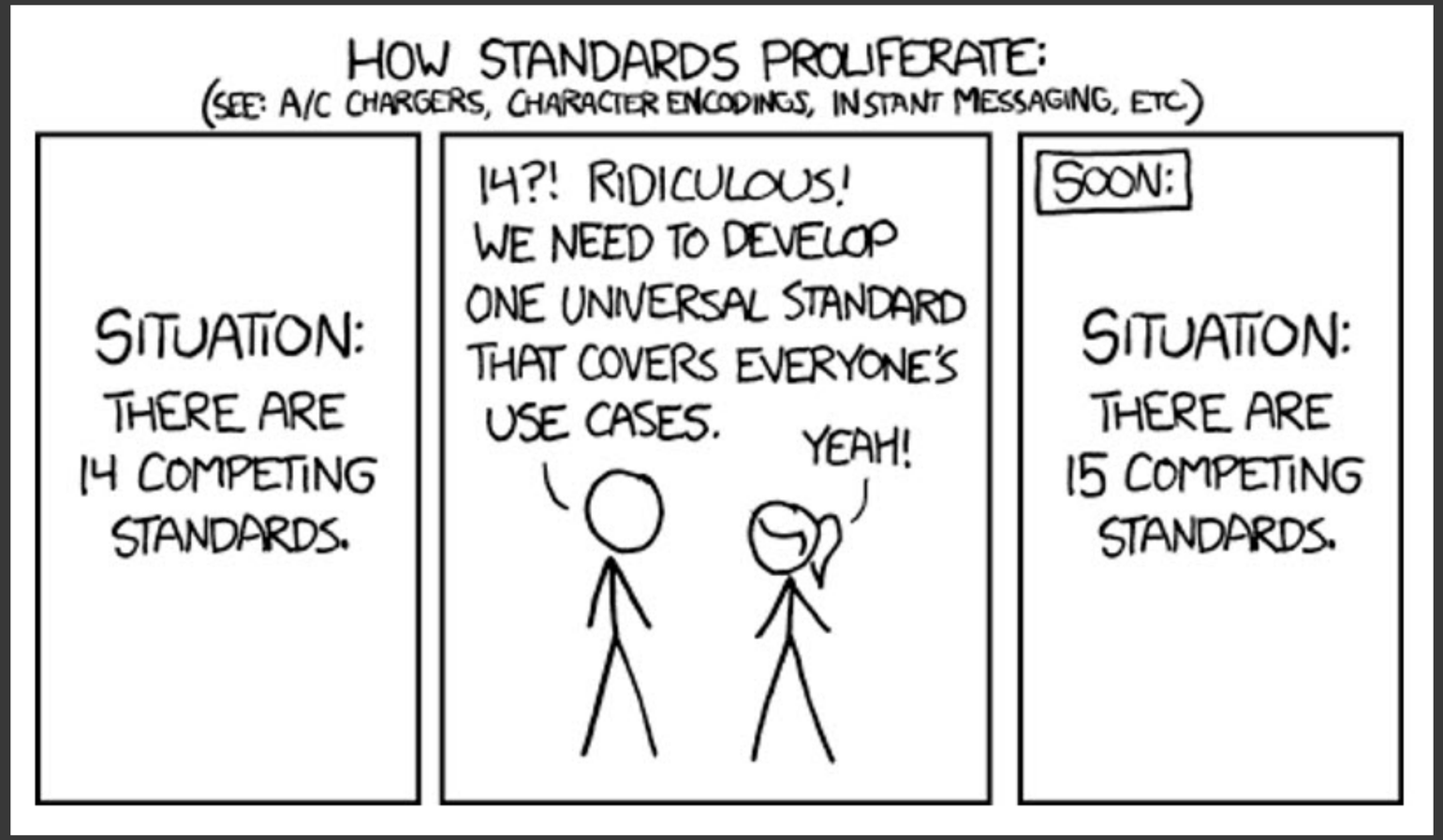
3. “System package managers (e.g., apt, brew, …)” – 39%

4. “I download prebuilt libraries from the Internet” – 33%

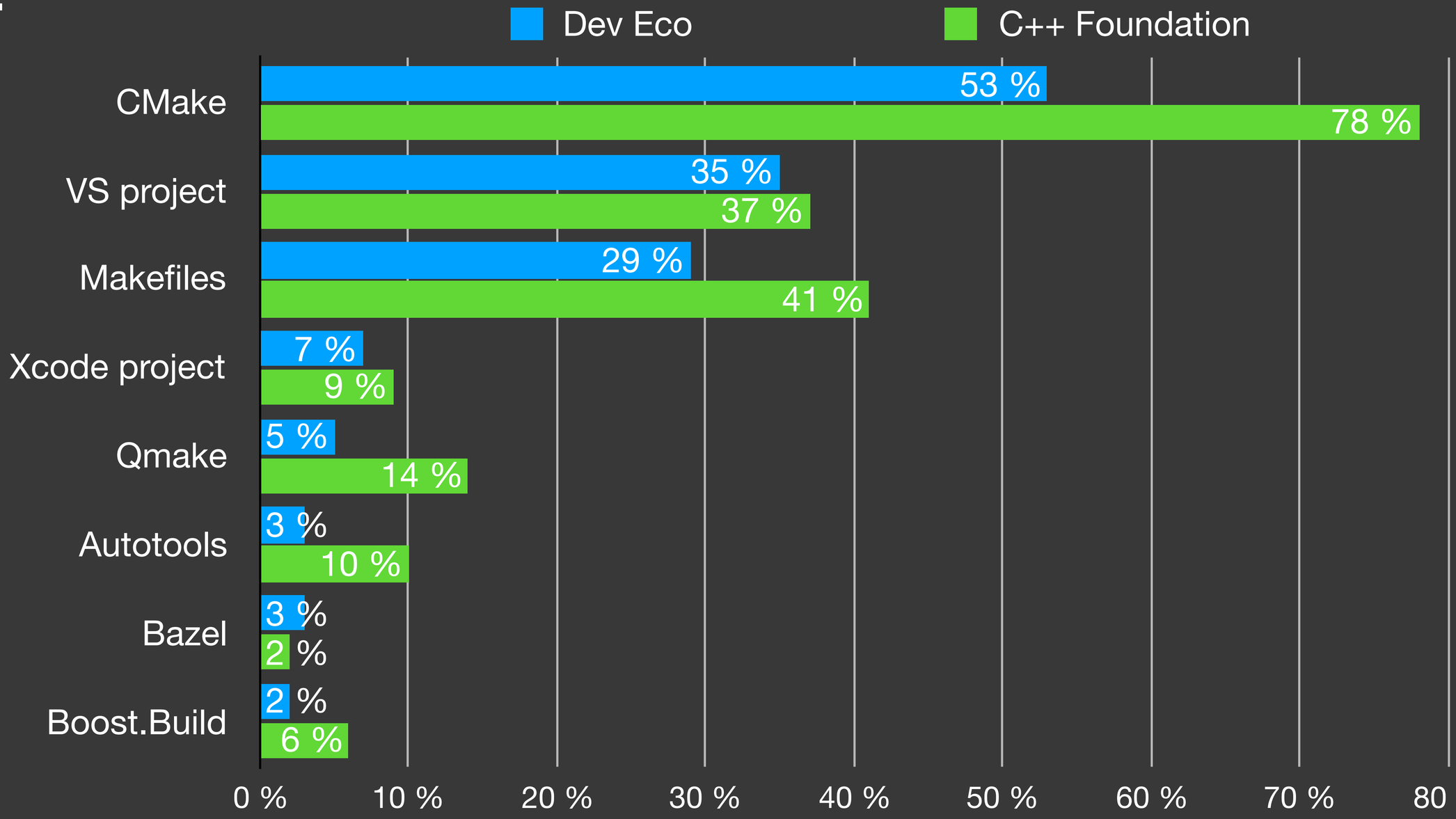
5. Conan, vcpkg – 14% each

(C++ Foundation “Lite” 2020)

C++ in 2020: project models



C++ in 2020: project models



C++ in 2020: project models

“Managing CMake projects”
– major pain point for 27%

“Managing Makefiles ”
– major pain point for 22%

“Managing MSBuild projects”
– major pain point for 19%

Agenda

1. C++ today
2. Ecosystem researches
3. C++ in 2020
4. Top usages areas
5. C++ standardization
6. To wrap up

Top C++ area

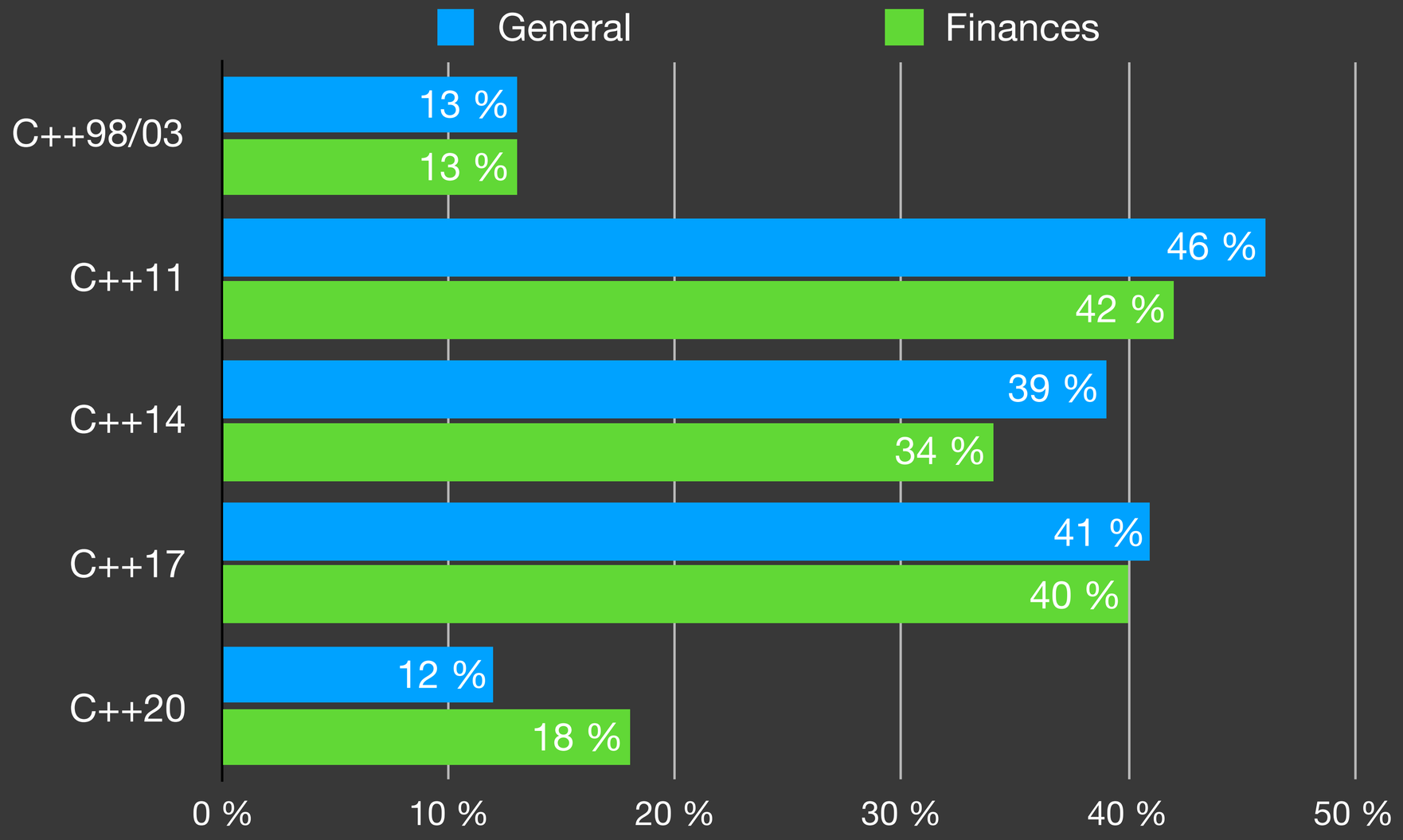
- Finances / Banking / Trading (HFT)
- Embedded systems
- Game Dev
- Machine Learning, Networking, AI, ...

C++ in Finances

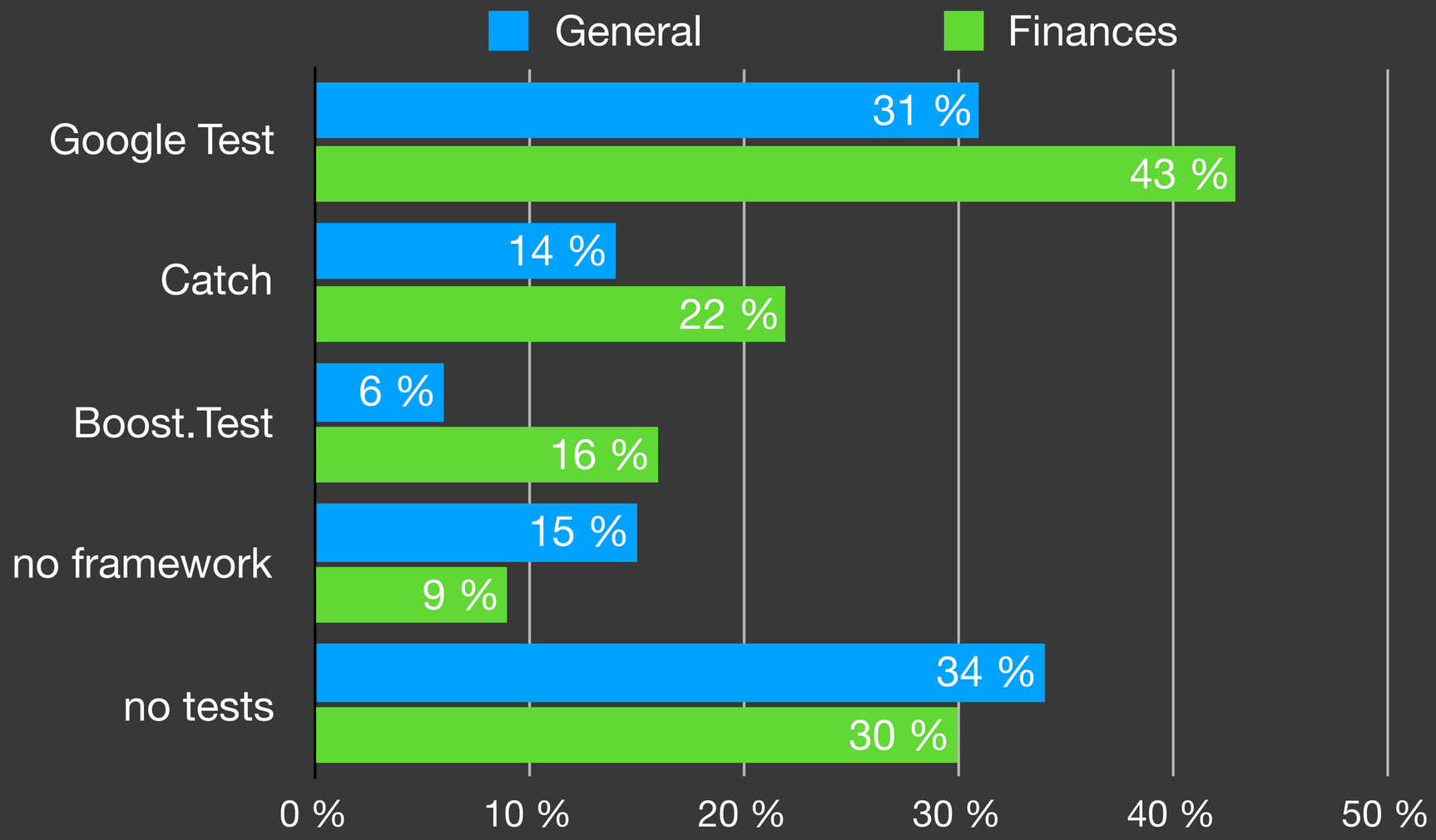
C++ in Finances

- Performance = low latency.
- Know what your compiler is generating.
- Branch prediction is important.
- Pools of the preallocated objects. Reuse instead of reallocate.
- 18% C++20!
- Widely using unit testing with the known frameworks.

C++ in Finances



C++ in Finances

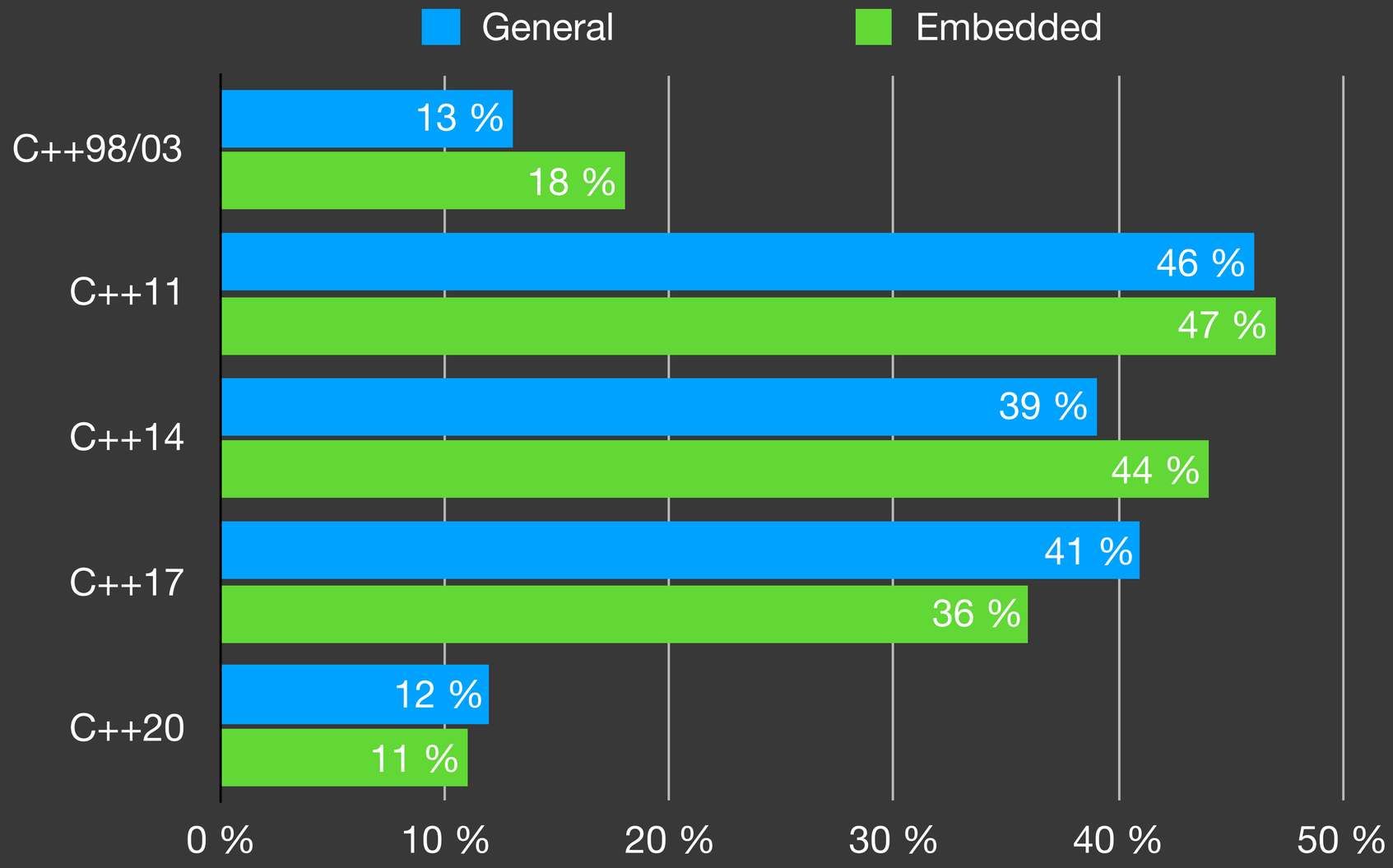


C++ in Embedded

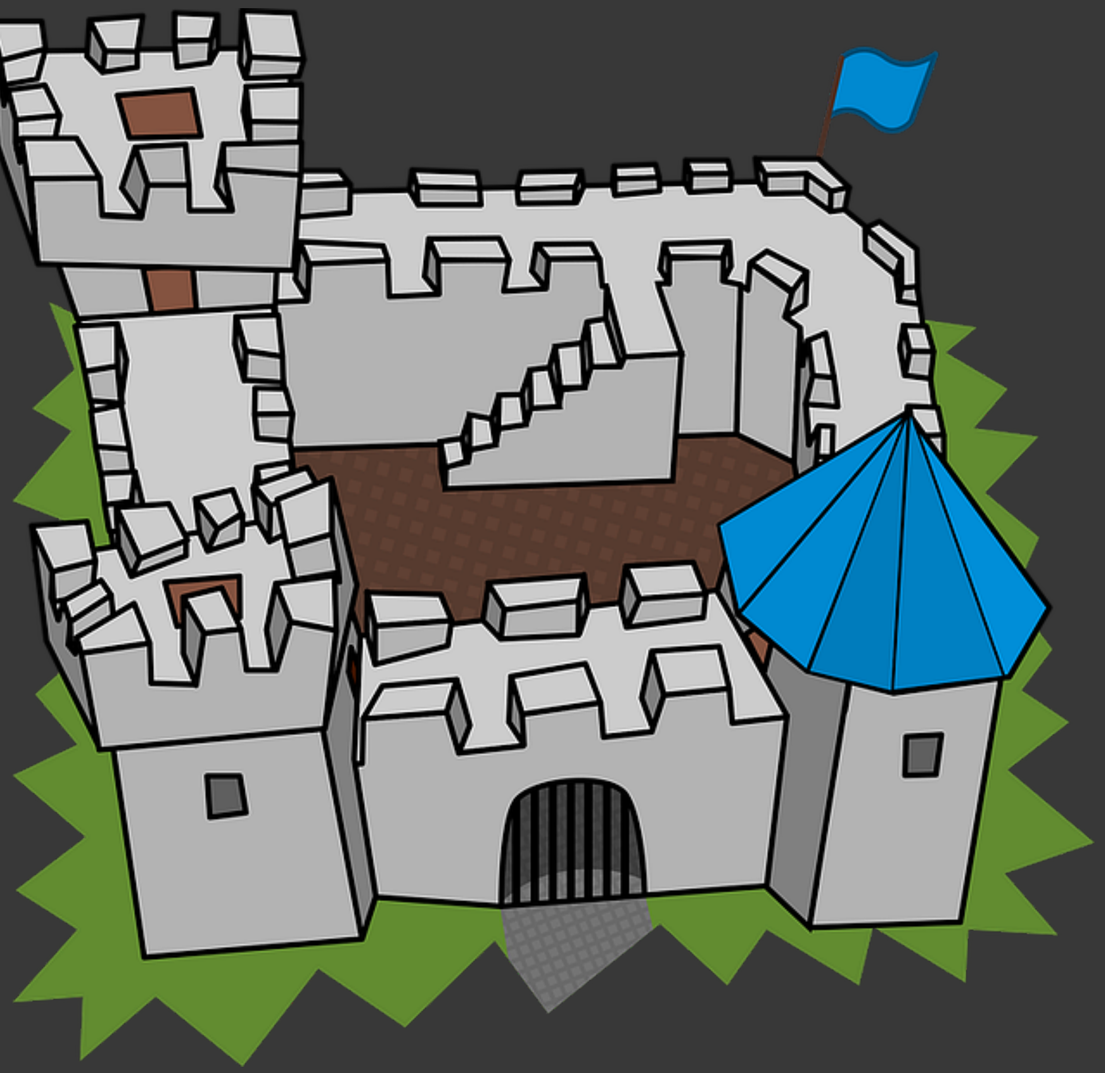
C++ in Embedded

- MCU vendors, SDK, starter kits, custom compilers.
- Testing / Standards compliance / Certification tools.
- C-structs with function pointers. Address arithmetics.
- Special memory structure / data packaging.
- Older C++ standards are more popular.

C++ in Embedded



C++ in Games



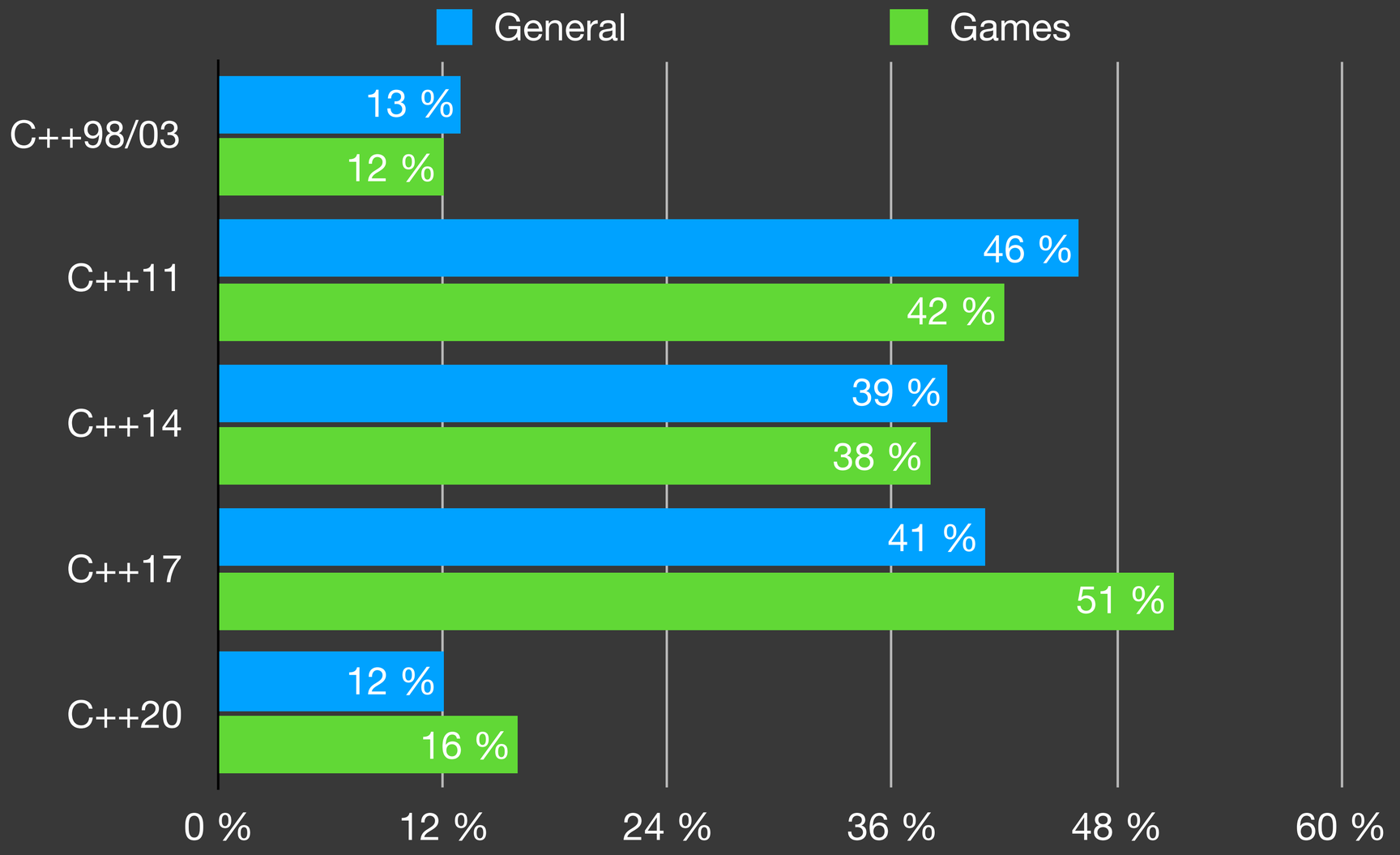
C++ in Games

- Windows platform.
- AAA games are mostly C++, custom engines.
- Rendering is in C.
- SDK for consoles.
- Reflection mechanisms.
- Specific allocators (InplaceArray<ubi32, 8>).
- Specific STL (EASTL, etc.).

C++ in Games

```
UCLASS(config=Game)
class APremiumGameProjectile : public AActor
{
  GENERATED_BODY()

  /** Sphere collision component */
  UPROPERTY(VisibleDefaultsOnly, Category=Projectile)
  class USphereComponent* CollisionComp;

public:
  APremiumGameProjectile();

  /** called when projectile hits something */
  UFUNCTION()
  void OnHit(UPrimitiveComponent* HitComp, AActor* OtherActor,
             UPrimitiveComponent* OtherComp, FVector NormalImpulse,
             const FHitResult& Hit);
   //…
}
```

C++ in Games


General
Games
C++98/03
13 %
12 %
C++11
46 %
42 %
C++14
39 %
38 %
C++17
41 %
51 %
C++20
12 %
16 %
0 %
12 %
24 %
36 %
48 %
60 %

Agenda

1. C++ today
2. Ecosystem researches
3. C++ in 2020
4. Top usages areas
5. C++ standardization
6. To wrap up

C++ standardization in 2020

- Towards a more powerful and simpler C++ © Herb Sutter
  - Static exceptions, Lifetime, Metaclasses
- Post-C++20
  - A modular standard library
  - Coroutine-supporting types in standard library
- Contracts, Reflection, Pattern Matching
- Networking, Audio, etc.

“Overall plan for C++23” (http://wg21.link/P0592)

Agenda

1. C++ today
2. Ecosystem researches
3. C++ in 2020
4. Top usages areas
5. C++ standardization
6. To wrap up

To wrap up

- C++ developers are constantly comparing and learning from Rust, Go, D, and others.
- C++ developers often discuss what they would remove from the language.
- The complexity looks scaring.
- “Thriving in a crowded and changing world: C++ 2006–2020” (https://dl.acm.org/doi/abs/10.1145/3386320)

Thank you!

References

- C++ Foundation Developer “Lite” Survey
  - [2020-4] https://isocpp.org/files/papers/CppDevSurvey-2020-04-summary.pdf
- The State of Developer Ecosystem Survey
  - [2020] https://www.jetbrains.com/lp/devecosystem-2020/cpp/
- Herb Sutter “Thoughts on a more powerful and simpler C++ (5 of N)”
  - [CppCon 2018] https://www.youtube.com/watch?v=80BZxujhY38
- Nicolas Fleury "C++ in Huge AAA Games"
  - [CppCon 2014] https://www.youtube.com/watch?v=qYN6eduU06s
- Scott Wardle “Memory and C++ debugging at Electronic Arts”
  - [CppCon 2015] https://www.youtube.com/watch?v=8KIvWJUYbDA
- EASTL - Electronic Arts Standard Template Library
  - [2007] http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2007/n2271.html
  - [GitHub] https://github.com/electronicarts/EASTL
- Carl Cook “When a Microsecond Is an Eternity: High Performance Trading Systems in C++”
  - [CppCon 2017] https://www.youtube.com/watch?v=NH1Tta7purM