When Not to Comment

Questions and Tradeoffs with API Documentation for C++ Projects

Andrew Head, Caitlin Sadowski, Emerson Murphy-Hill, Andrea Knight
Google, UC Berkeley, NC State University

Developers Use APIs

```
std::string s =
    absl::FormatTime(
        "My flight lands in Göteborg on %Y-%m-%d at %H:%M",
        landing, timezone);
```

A programmer calls the function **`FormatTime`** from the C++ **`absl`** API.

Programmers use APIs all the time to save time, improve code consistency, etc.

Writing API Documentation

```
std::string FormatTime(const std::string& format, ...);
```

To help developers use APIs, tech writers and maintainers decide when and how to describe:

Behavior `// Formats the given `absl::Time`...`

Usage `// std::string f = absl::FormatTime("%H:%M:%S", ...`

... and best practices, special cases, design rationale, etc.

A Dilemma with Designing Docs

Is the documentation answering the right questions?

... We don't know... How can we know?

What methods can we use to collect developer questions about API documentation?

Our Research Questions

Q1. Are C++ API header comments answering developers' questions?

Q2. Why might answers be missing from the headers?

Q3. When does it matter that comments are missing?

Findings

- **Unanswered API Questions**. 9 types of questions about low-level usage, high-level usage, and implementation.
- **Why comments are missing?** Resistance to update comments for abandoned or young projects, or concerns about bloat and confusion.
- **When comments matter?** If answers can't be recovered from code, and if developers trust comments.

Challenges to Finding API Questions

Bug reports? Infrequently submitted for docs.
Survey? Developers forget their questions.
Observation? Time-consuming.

When to Prompt API Clients for Questions

When to Prompt API Clients for Questions

A header file (`time.h`)

```
// FormatTime
//
// Formats the given `absl::Time`...
// provided format std::string. U...
// the following extensions:
//
std::string FormatTime(
    const std::string& format, ...);
```

When to Prompt API Clients for Questions

A header file (`time.h`)

```
// FormatTime
//
// Formats the given `absl::Time`...
// provided format std::string. U...
// the following extensions:
//
std::string FormatTime(
    const std::string& format, ...);
```

Method signature

When to Prompt API Clients for Questions

A header file (`time.h`)

```
// FormatTime
//
// Formats the given `absl::Time`...
// provided format std::string. U...
// the following extensions:
//
std::string FormatTime(
    const std::string& format, ...);
```

Comments, low-level usage documentation

When to Prompt API Clients for Questions

A header file (`time.h`)

```
// FormatTime
//
// Formats the given `absl::Time`...
// provided format std::string. U...
// the following extensions:
//
std::string FormatTime(
    const std::string& format, ...);
```

An implementation file (`time.cc`)

```
std::string FormatTime(const std::string
  if (t == absl::InfiniteFuture()) retur
  if (t == absl::InfinitePast()) return
  const auto parts = Split(t);
  return cctz::detail::format(format, pa
                              cctz::time
}
```

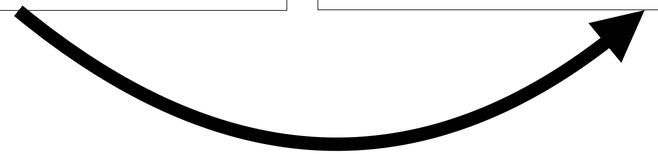
When to Prompt API Clients for Questions

A header file (`time.h`)

```
// FormatTime
//
// Formats the given `absl::Time`...
// provided format std::string. U...
// the following extensions:
//
std::string FormatTime(
    const std::string& format, ...);
```

An implementation file (`time.cc`)

```
std::string FormatTime(const std::string
  if (t == absl::InfiniteFuture()) retur
  if (t == absl::InfinitePast()) return
  const auto parts = Split(t);
  return cctz::detail::format(format, pa
                              cctz::time
}
```



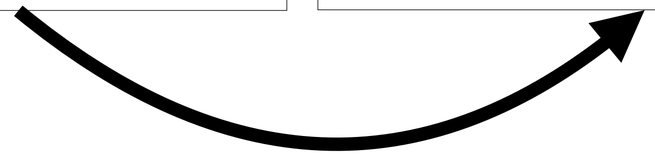
When to Prompt API Clients for Questions

A header file (`time.h`)

```
// FormatTime
//
// Formats the given `absl::Time`...
// provided format std::string. U...
// the following extensions:
//
std::string FormatTime(
    const std::string& format, ...);
```

An implementation file (`time.cc`)

```
std::string FormatTime(const std::string
  if (t == absl::InfiniteFuture()) retur
  if (t == absl::InfinitePast()) return
  const auto parts = Split(t);
  return cctz::detail::format(format, pa
                              cctz::time
}
```

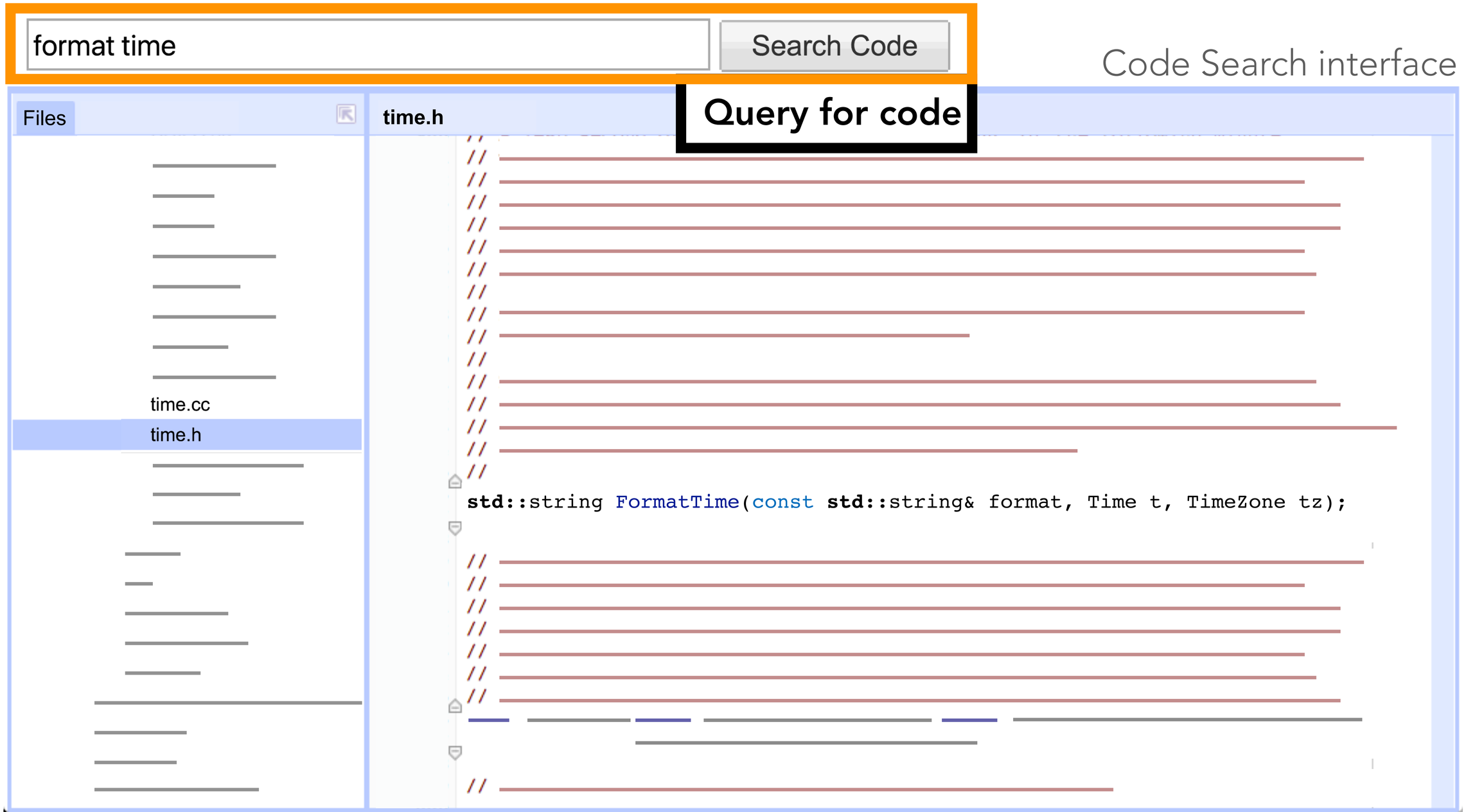


This transition sometimes indicates an API question.

Prompting for API Questions in Code Search

Code Search interface

Prompting for API Questions in Code Search



Prompting for API Questions in Code Search

Prompting for API Questions in Code Search

Prompting for API Questions in Code Search

Prompting for API Questions in Code Search

format time

Search Code

Code Search interface

Files

time.h

time.cc

time.h

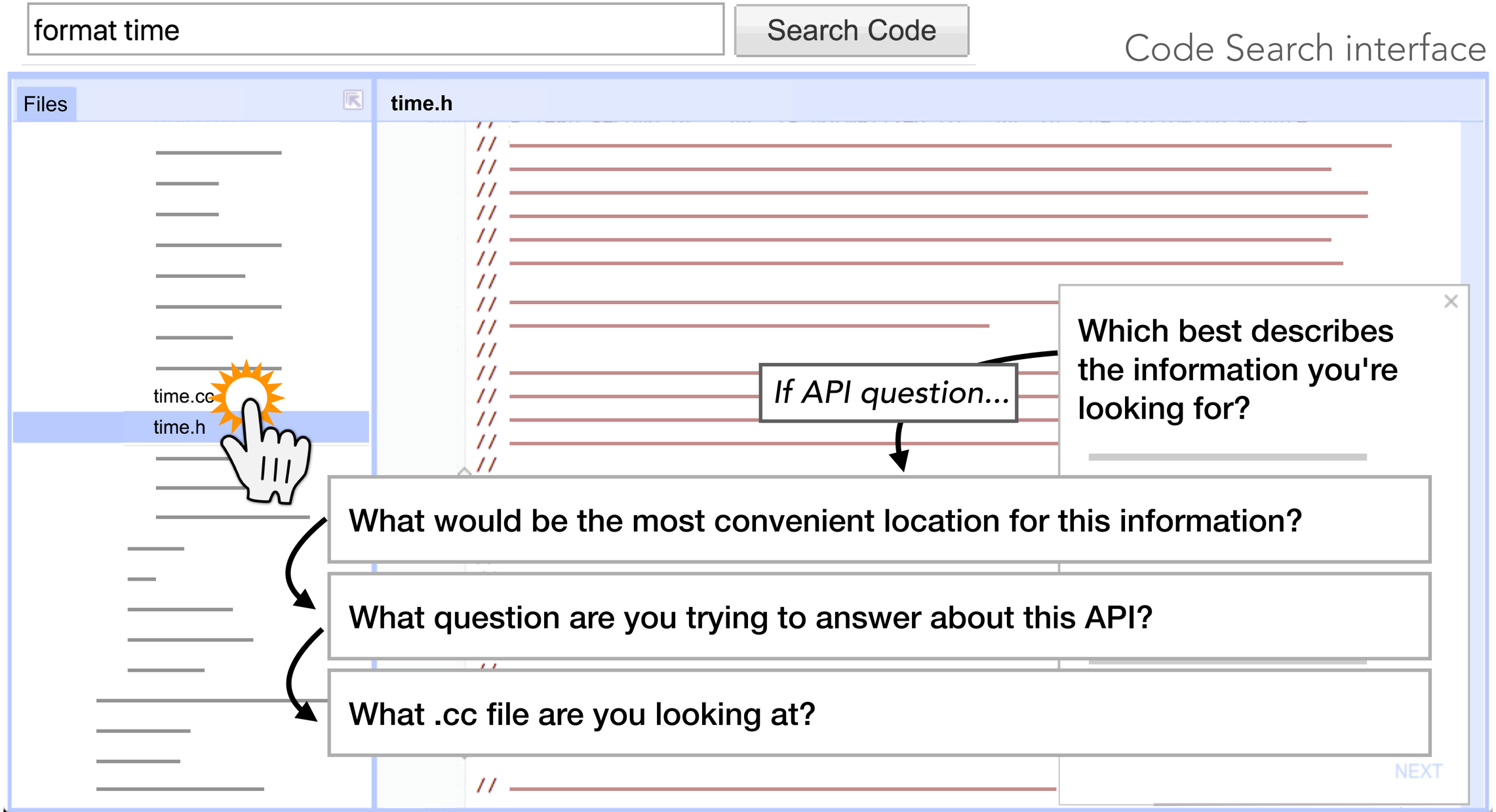
```
//
//
//
//
//
//
//
//
//
//
//
//
//
//
std::string FormatTime(const std::string& form
//
//
//
//
//
//
//
//
//
//
```

After a header-to-implementation transition, Code Search asked if a searcher had API questions.

Which best describes the information you're looking for?

NEXT

Prompting for API Questions in Code Search



Benefits and Limitations of "Header-to-Implementation" Detection

+ Timely: Captures ephemeral questions.
+ Scalable: Deployable within search infrastructure, and can be run on search logs.
- Imperfect: *Needs developer input* to confirm the transition was about API usage.
- Incomplete: Currently only covers low-level documentation in header files.

Mixed Methods Study Design

Monitor Search Behavior

Time	Path
8:00:30	time/clock.h
12:00:00	time/time.h
12:00:10	time/time.cc

Survey in Code Search

1,147 respondents

60 API usage questions
(full C++ code base)

Interview Searchers

What were you looking for? How?

18 searchers

Interview Maintainers

Should your docs answer this question?

8 maintainers

Qualitative Analysis

- API Questions: Card-sorting (2 authors)
- Interviews: Verbatim transcription, open and axial coding of themes
  (1 author, checked by another author)

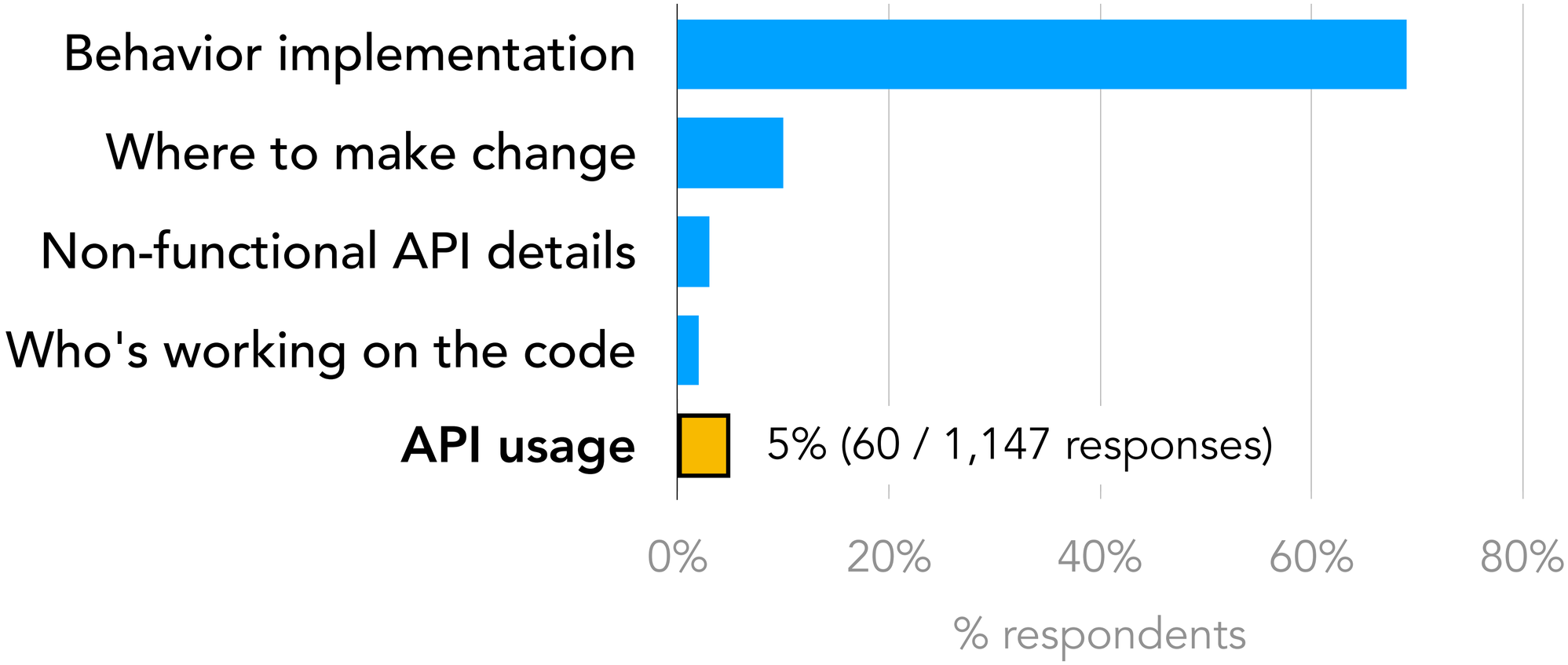
Closer Look at Results

Q1. Are C++ API header comments answering developers' questions?

Q2. Why might answers be missing from the headers?

Q3. When does it matter that comments are missing?

Why Developers Visited Implementation



Sample: Collected API Questions

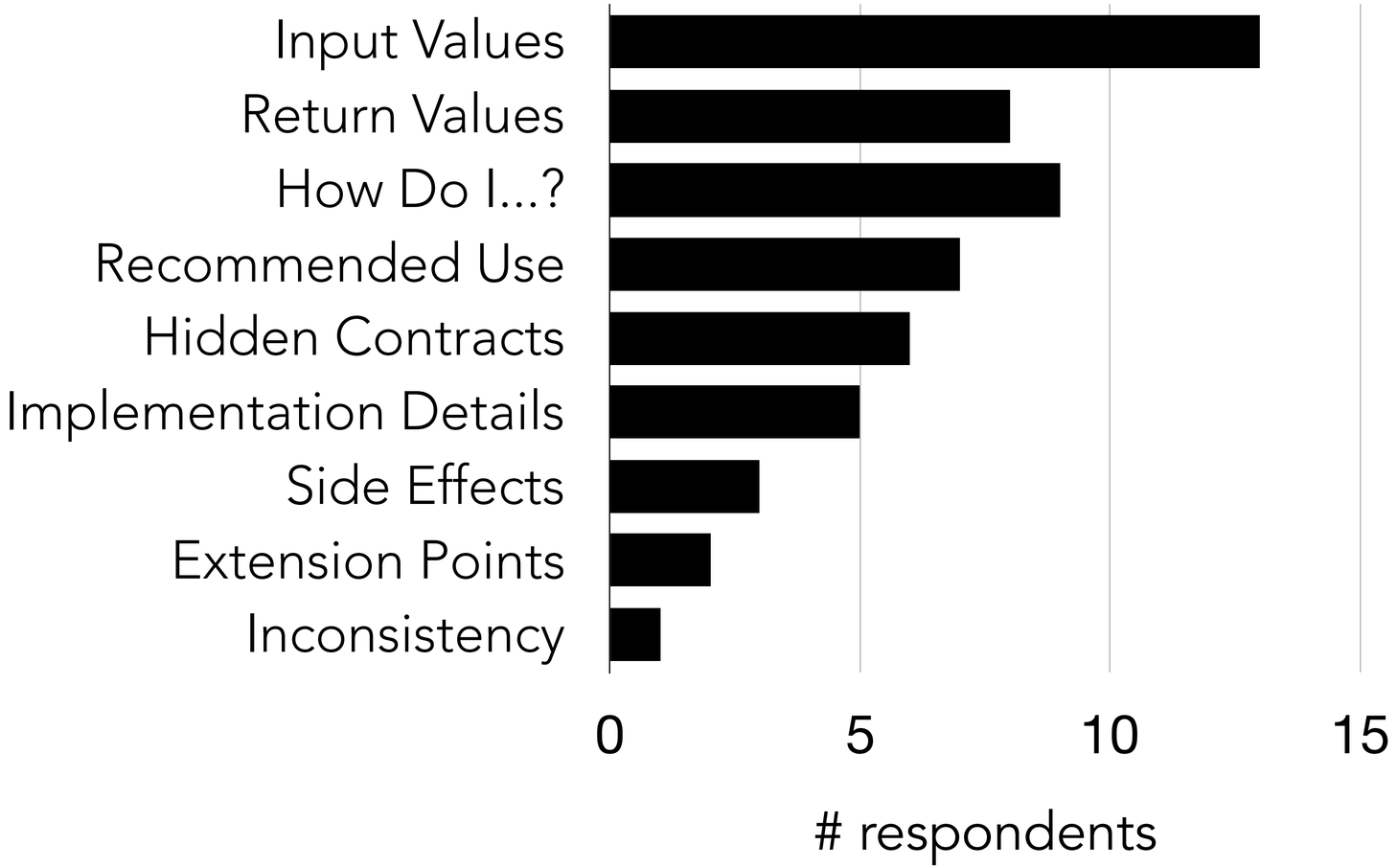
*"**What does the return value mean** and how can this method fail"*

*"**What method to use** to convert the current timestamp into a string"*

*"**How this API passes data** to TensorFlow session run calls in C"*

... and 50+ others

Types of API Questions

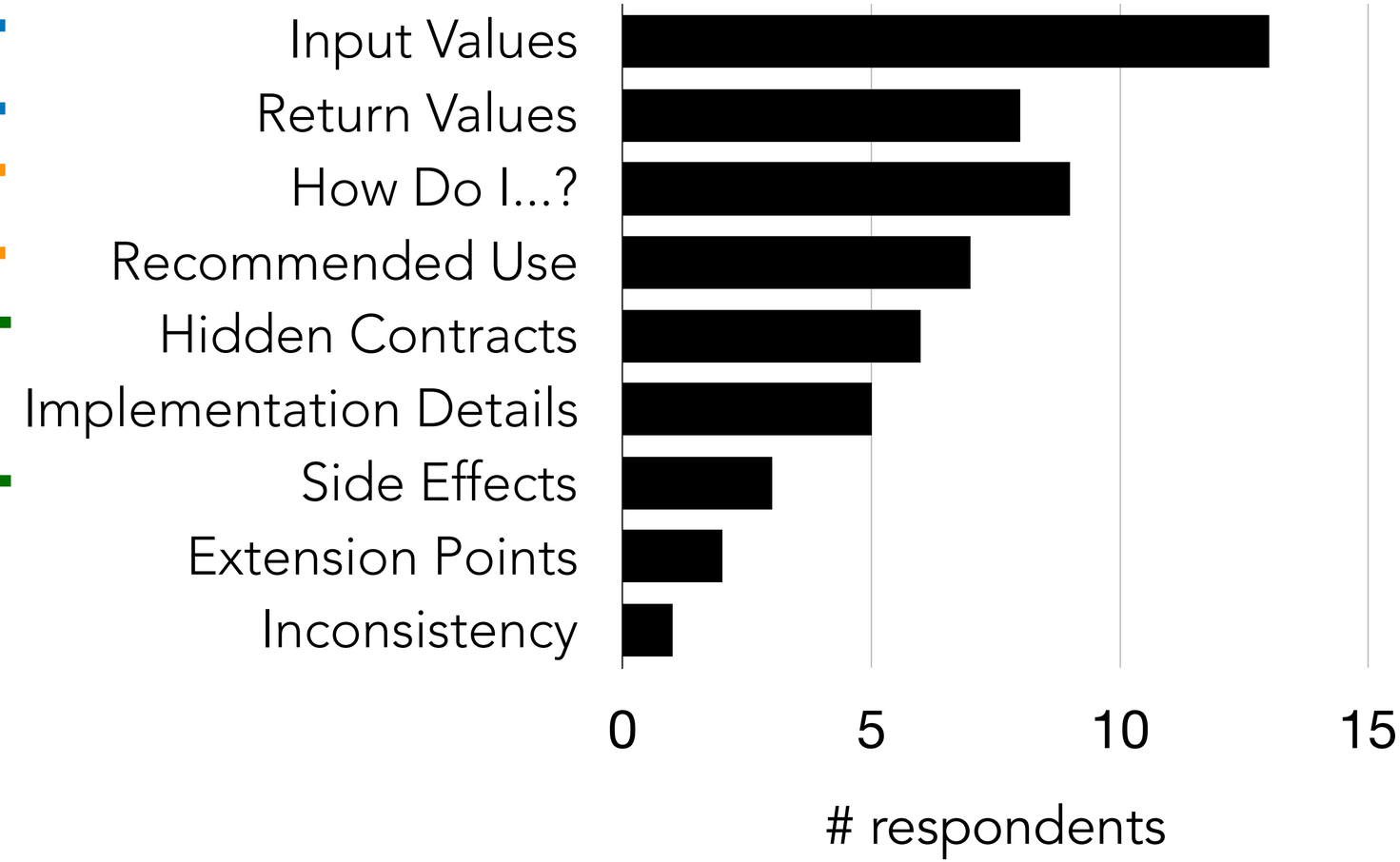


Types of API Questions

Low-Level Usage {

High-Level Usage {

Implementation {

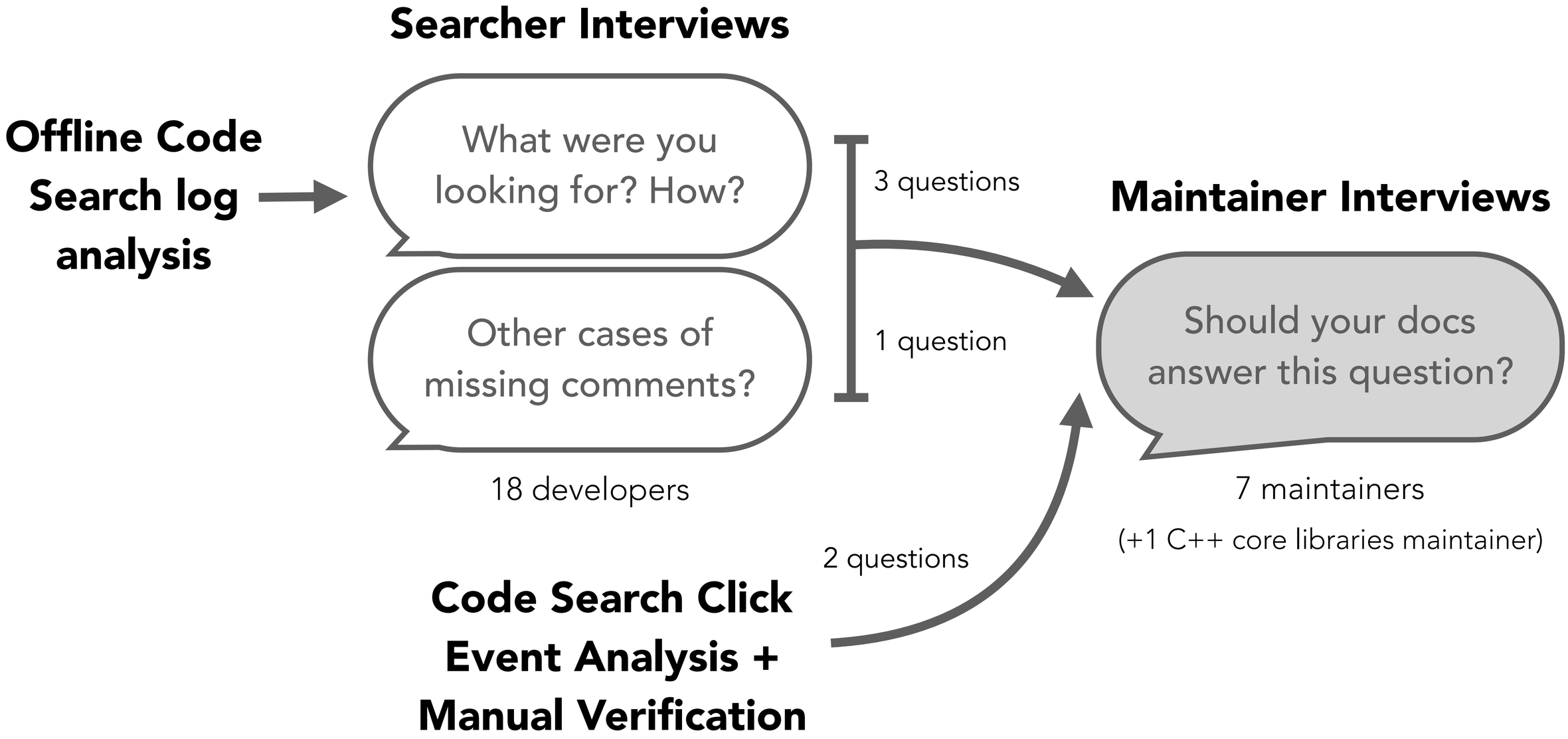


Q1. Do the header comments answer the right questions?

Clearly not always. We collected 60 cases where developers opened implementation code to check on API usage.

Writers should **consider at least 9 types of questions.** Most of these aren't reported in past studies.

Q2. Why are comments missing?



Reason 1: Not the Right Time

Reason 1: Not the Right Time

Too late. "**It’s unlikely this will ever get changed again**... ostensibly it’s my team that’s responsible for it, but... if you didn’t schedule this meeting I would have forgotten this file existed."

Reason 1: Not the Right Time

Too late. "**It's unlikely this will ever get changed again**... ostensibly it's my team that's responsible for it, but... if you didn't schedule this meeting I would have forgotten this file existed."

Too early. Reluctance to invest in comments when the current focus was evolving and fixing the code.

Reason 2: Minimal Explanations

Reason 2: Minimal Explanations

Avoiding bloat. "How often do you want to go into details, which can be easily too much?"

Reason 2: Minimal Explanations

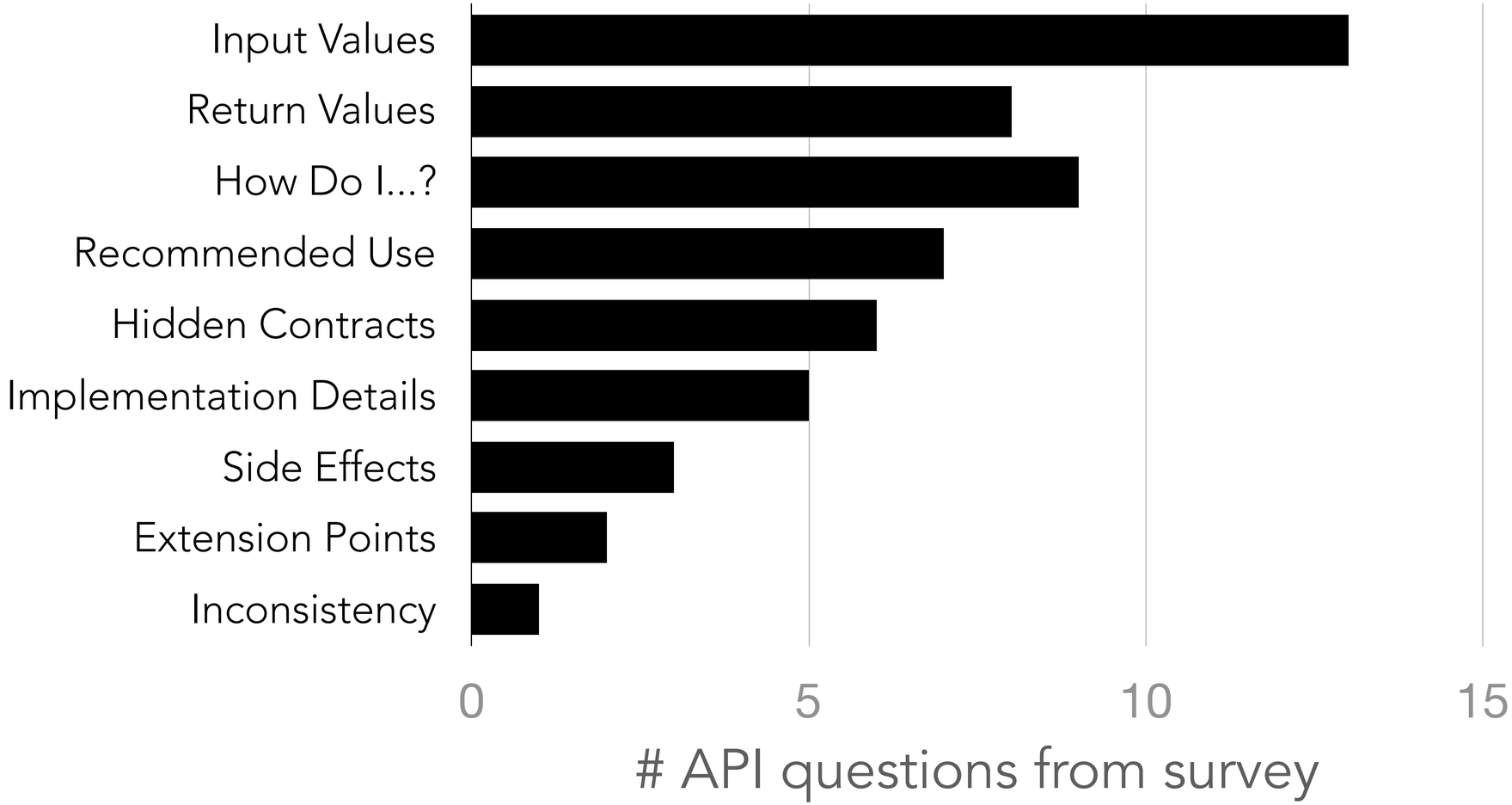
Avoiding bloat. "How often do you want to go into details, which can be easily too much?"

Avoiding misunderstanding. "...if you say something is slow, you'll get people writing alternatives first of all, or not using it... "

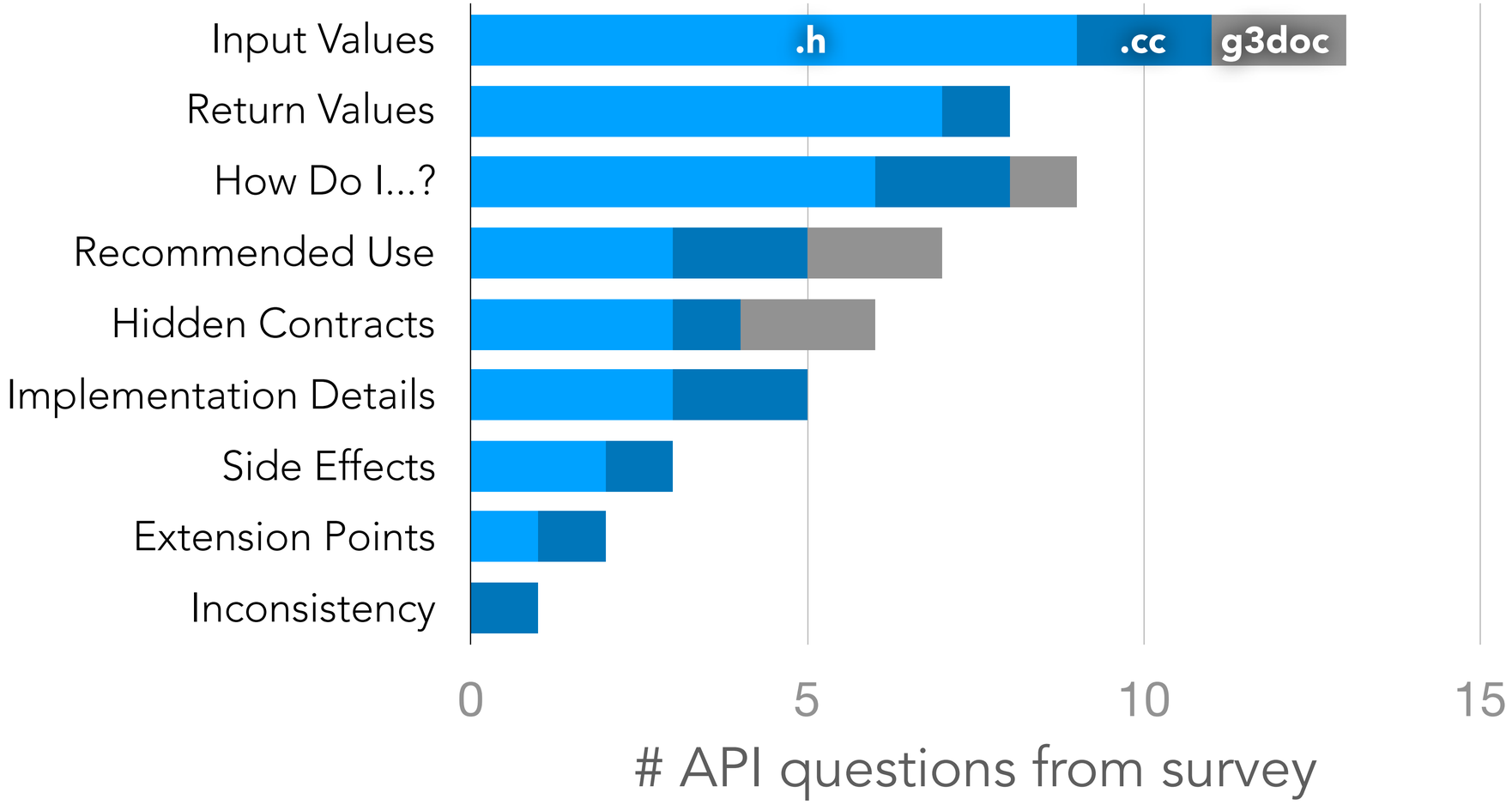
Q2. Why might answers be missing from the headers?

The project may be **abandoned**, **too young**, or maintainers believe answers could add **bloat** or **confusion**.

Survey Respondents Preferred Answers in Headers



Survey Respondents Preferred Answers in Headers



In **61.7%** of cases, it would have been most convenient to find an answer in a header—even for some high-level usage questions and implementation questions.

When Comments Could Help Interviewees

- **To avoid involved code inspection.** 2 / 6 interviewees with API questions searched through multiple files, one of whom gave up.
- **To understand recommended use.** Protototyping "messy code" by looking at an API's implementation, then "making it clean" by looking in its comments.

When Comments Wouldn't Have Helped

Interviewees often **didn't trust comments**, and sometimes skipped them, or disregarded them after reading them.

- "... I have stopped reading comments, because the comments are just lies."
- One implementation visit because "it was actually documented properly, but I didn’t believe it."

Trust Depends on Project

“... there’s those sorts of [general utilities], and those tend to be very well documented. And then there’s the team-specific internal code, which is all very horribly documented.”

Q3. When does it matter that comments are missing?

When answers **can't easily be recovered from code**, and **when developers trust comments** (which isn't always).

Takeaways

- **Methods.** Piloted a technique that collects API questions professional software developers ask.
- **API Questions.** Revealed 9 types of questions developers asked about APIs when opening implementation code.
- **Stakes.** Helped document costs, benefits, and factors influencing whether maintainers will and should update documentation comments.

Looking Forward

- Maintainers should refer to the questions developers ask about APIs when writing documentation.
- Stakeholders should consider relative gain and barriers to updates when choosing where to answer API questions.
- Others should extend our methods to gain insight into questions developers ask during their work, and design tools and artifacts that provide the answers.

A Time to Prompt API Users

A header file (`time.h`)

```
// FormatTime
//
// Formats the given `absl::Time`...
// provided format std::string. U...
// the following extensions:
//
std::string FormatTime(
    const std::string& format, ...);
```

An implementation file (`time.cc`)

```
std::string FormatTime(const std::strin
  if (t == absl::InfiniteFuture()) retu
  if (t == absl::InfinitePast()) return
  const auto parts = Split(t);
  return cctz::detail::format(format, p
                              cctz::tim
}
```

- Function signature (no definitions)
- Method-level comments

This transition sometimes indicates an API question.

x

Mixed Methods Study Design

Survey in Code Search

60 API usage questions
(full C++ code base)

1,147 respondents
60 with API questions

Monitor Search Logs

Time	Path
8:00:30	time/clock.h
12:00:00	time/time.h
12:00:10	time/time.cc

Interview Searchers

What were you looking for? How?

18 searchers

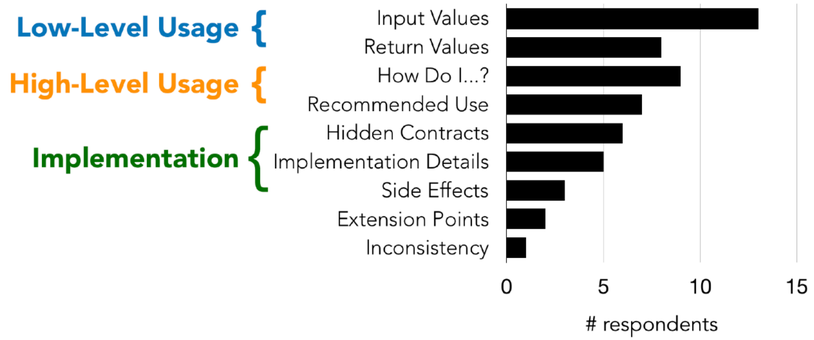
Interview Maintainers

Should your docs answer this question?

8 maintainers

Q1. API questions

Types of API Questions



x

Q3. When does it matter that comments are missing?

When answers **can't easily be recovered from code**, and **when developers trust comments** (which isn't always).

x

Read the paper at **https://tinyurl.com/icse18-comment**