Parsing Documentation Comments in Clang

Dmitri Gribenko ⟨gribozavr@gmail.com⟩
High Performance Computing Center at NTUU «KPI»

LLVM Developers' Meeting
November 8, 2012

Outline

- Motivation
- Implementation details
- User-visible features in Clang and libclang

Clang is (was) ignoring documentation comments

- Comments tend to bitrot.
- Syntax errors in markup are not caught.
- Documentation extraction tools don't really understand C++.
- Tools based on Clang libraries will also ignore comments:
  - IDE source editor tools;
  - refactoring tools.

Documentation support: goals

- Help to keep comments up to date and syntactically correct.
- Enhance Clang-based tools with comments support:
  - IDE source editor tools;
  - refactoring tools.
- (Maybe) Build a better documentation extraction tool.

Documentation comments

$\approx$

Doxygen

Implementation

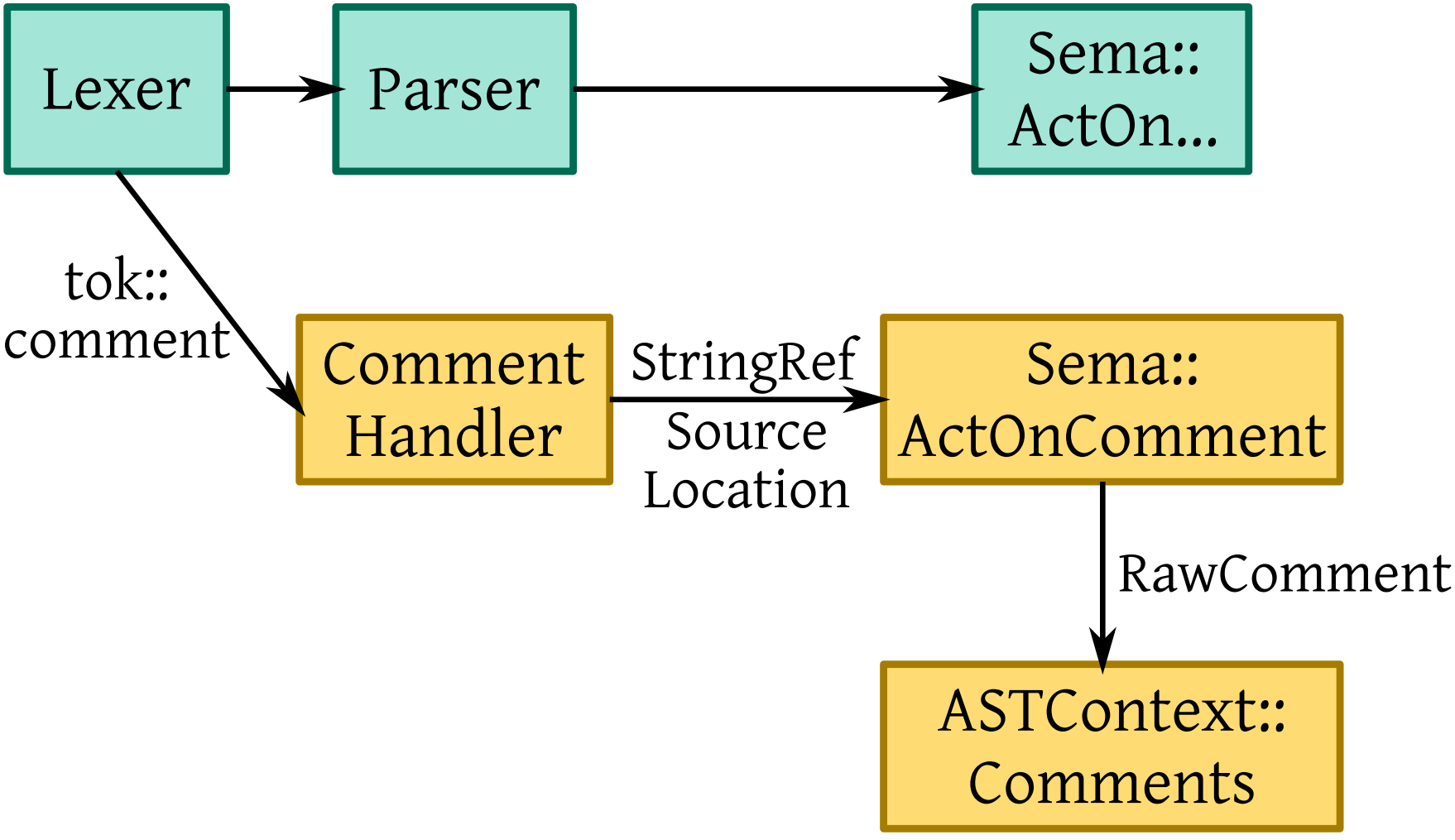
- Finding comments.
- Attaching comments to declarations.
- Comment parsing.
- Comment AST.

Attaching comments to AST nodes

- Teach Clang parser about `tok::comment`:
  - would complicate the parser;
  - would add lots of checks for `tok::comment`;
  - no easy way to turn off.

- Attach comments after the AST is built.

Finding raw comments



Documentation comments

```
/// comment
void f();

/** comment */
void g();

/*! comment */
void h();
```

```
int a; ///< comment

int b; /**< comment */

int c; /*!< comment */
```

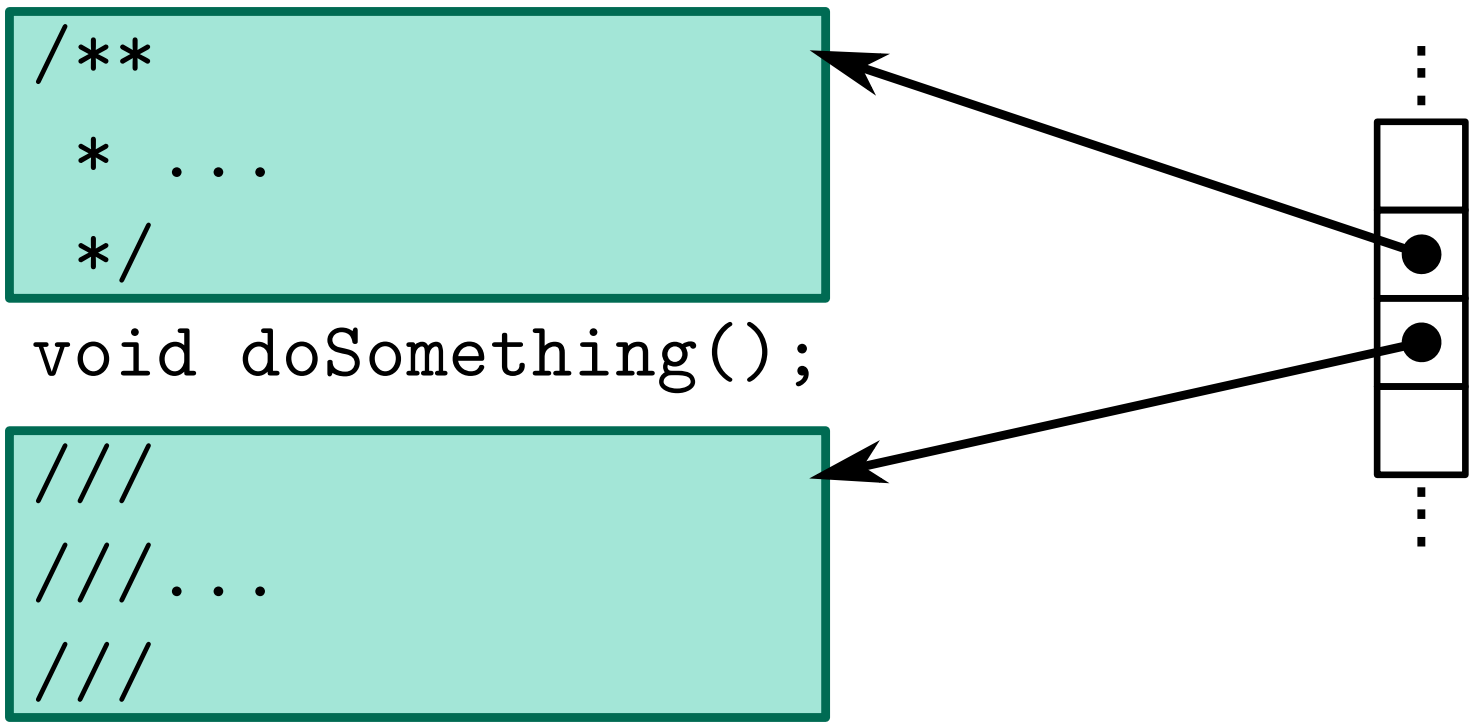
"Almost documentation" comments

```
struct a {
  int x; //< comment
  int y; /*< comment */
};
```

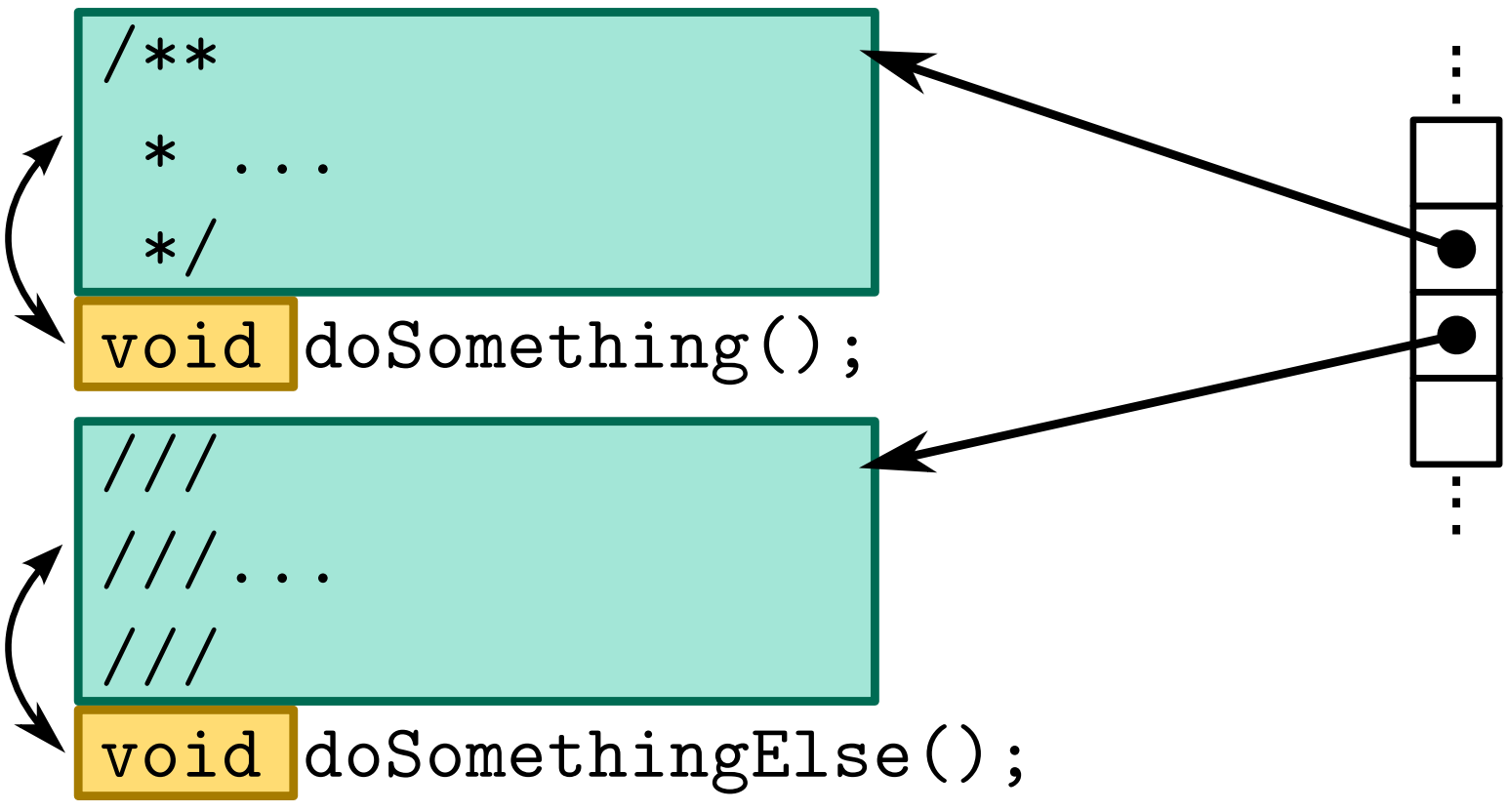
"Almost documentation" comments

```
struct a {
  int x; ///< comment
  int y; /**< comment */
};

example.c:8:10: warning: not a Doxygen trailing comment
  int x; //< comment
         ^~~
         ///<
```

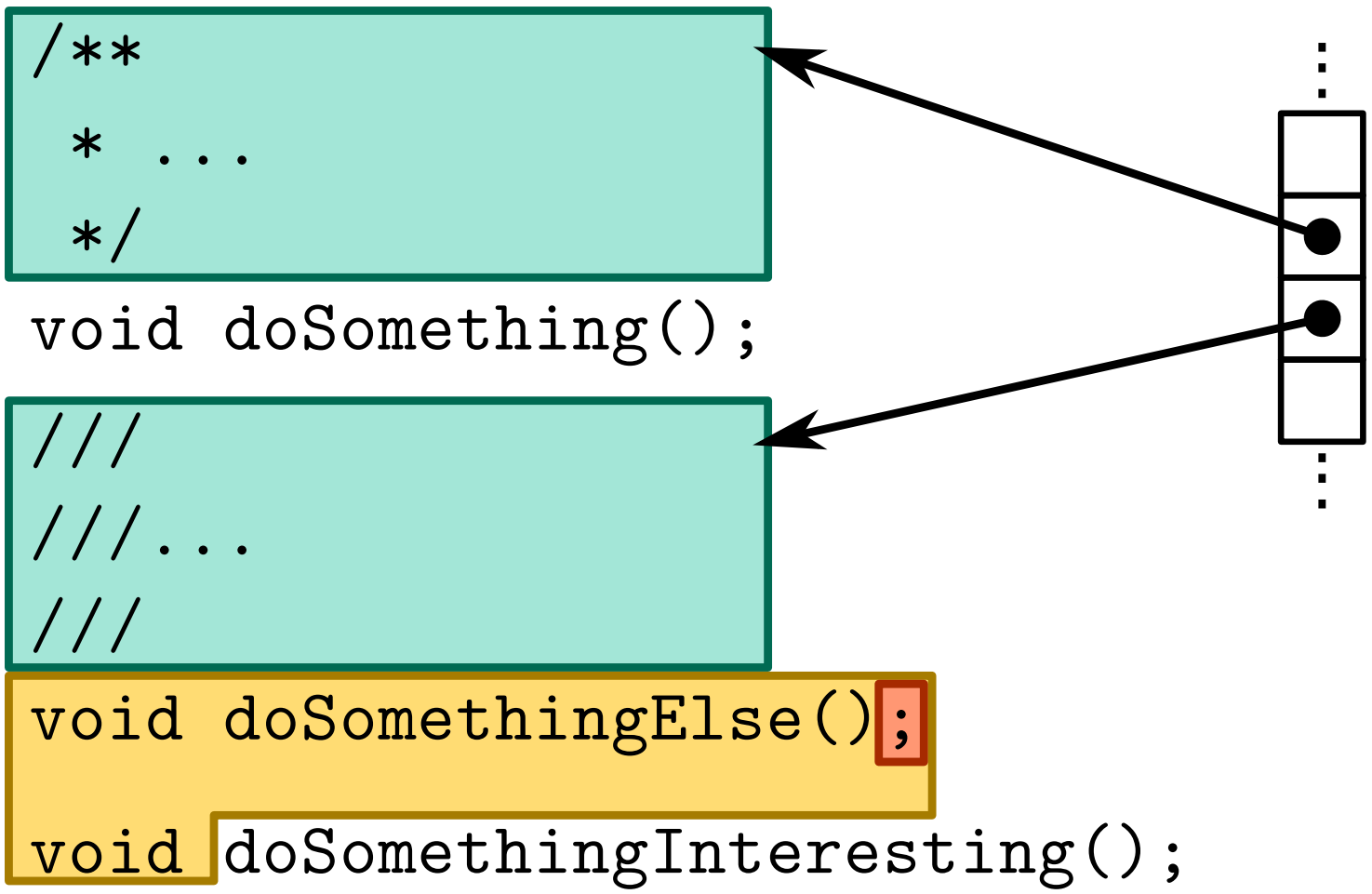
Attaching comments to AST nodes



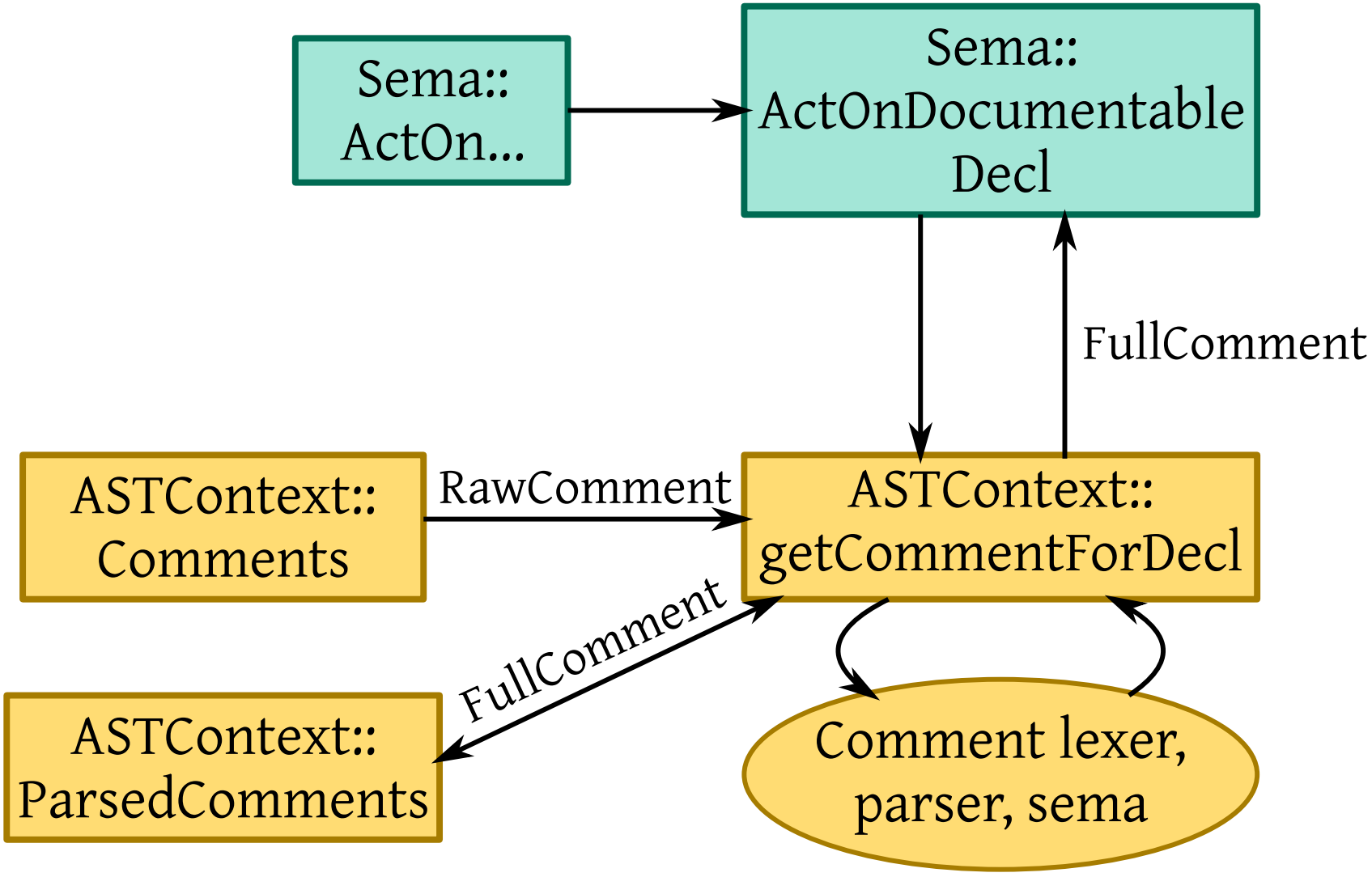
Attaching comments to AST nodes



Attaching comments to AST nodes



Attaching comments, eager mode



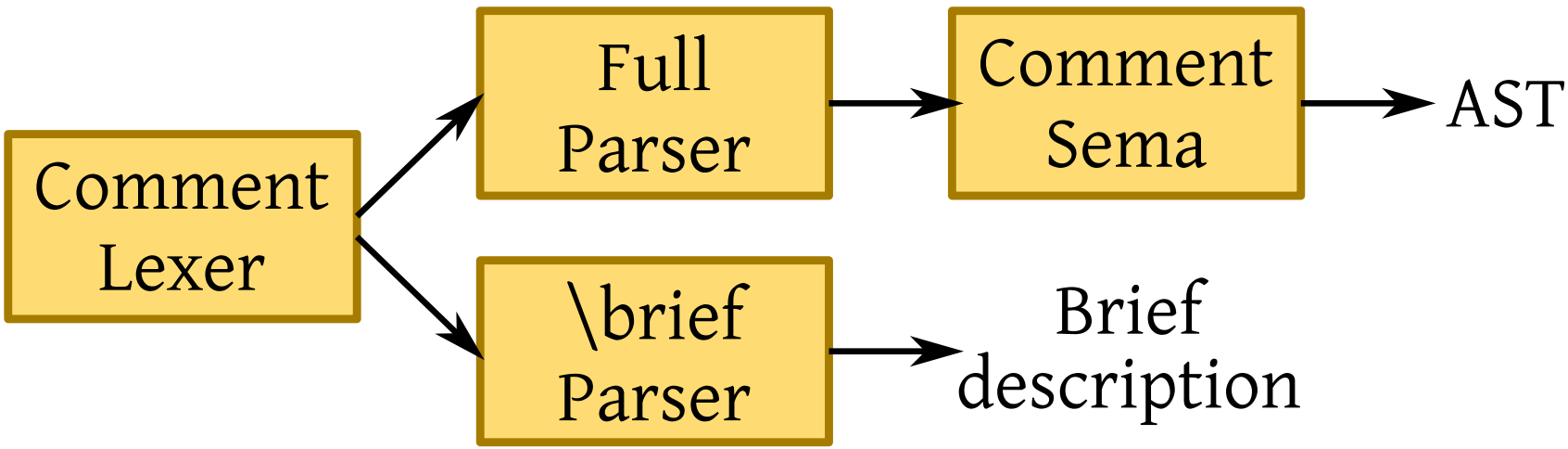
Comments from "related" declarations

```
/// \brief Does foo.
/// \param a blah.
void f(int a);

void f(int b) { // picks up the comment above.
}
```

Comments from "related" declarations

```
struct A {
  /// \brief Does foo.
  virtual void f();
};

struct B : public A {
  virtual void f(); // picks up A::f()'s comment.
};
```

Comment parsing



Comment AST

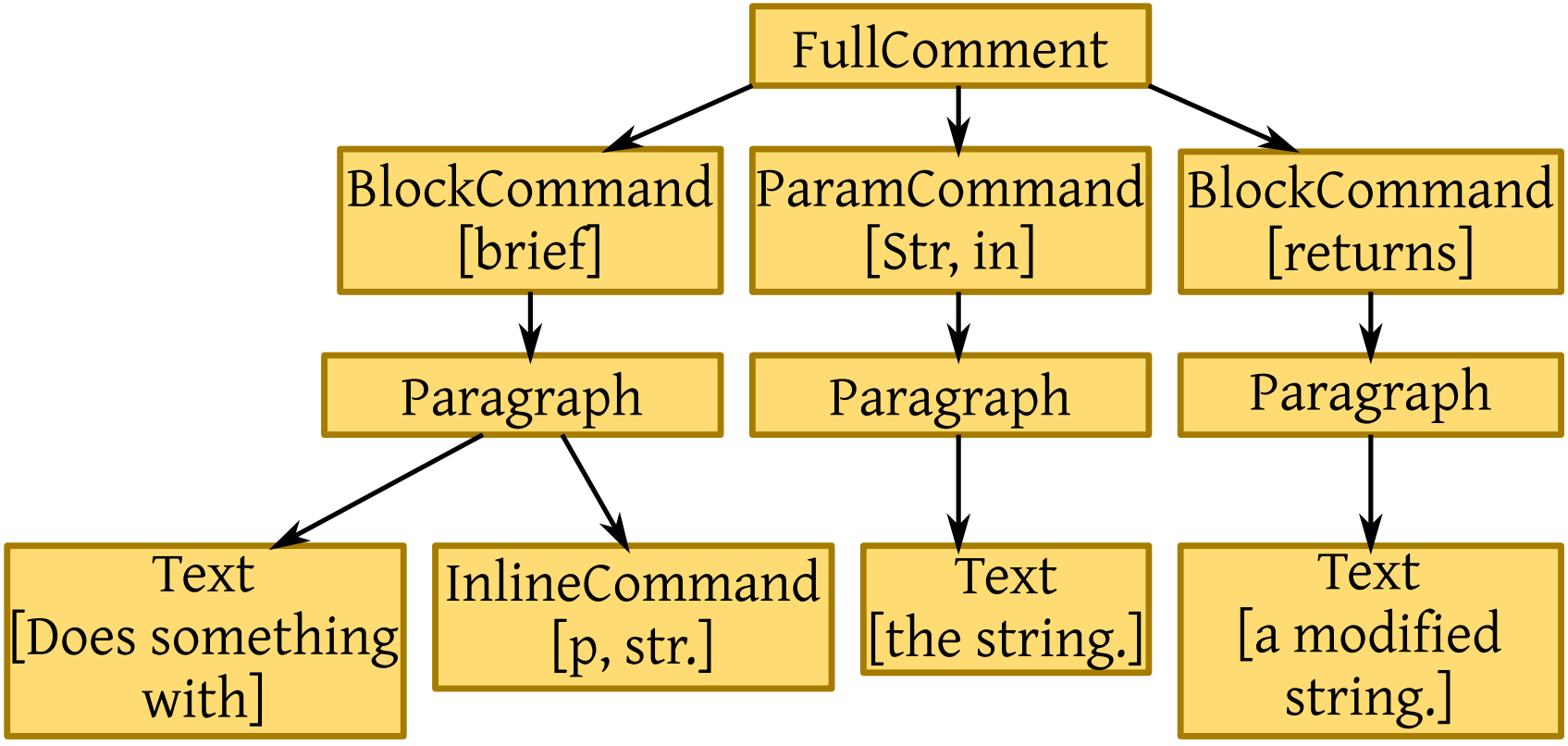
- Root `FullComment` node has children which are block content.
- `ParagraphComment` nodes have children which are inline content.

Comment AST

- Root `FullComment` node has children which are block content.
- `ParagraphComment` nodes have children which are inline content.

- Block content:
  - paragraph;
  - block command (e. g., `\brief` or `\verbatim`).
- Inline content:
  - text;
  - inline command (e. g., `\c`);
  - HTML tag.

Comment AST




```
/// \brief Does something with \p str.
/// \param [in] Str the string.
/// \returns a modified string.
void do_something(const std::string &str);
```

User-visible features

- Diagnostics for comments.
- Enhanced code completion APIs (libclang).
- Export comments as XML (libclang).
- Performance.

Diagnostics

```
/// \brief Does something with \p str.
/// \param [in] Str the string.
/// \returns a modified string.
void do_something(const std::string &str);
```

Diagnostics

```
/// \brief Does something with \p str.
/// \param [in] Str the string.
/// \returns a modified string.
void do_something(const std::string &str);

example.cc:4:17: warning: parameter 'Str' not found
        in the function declaration [-Wdocumentation]
/// \param [in] Str the string.
                ^~~
example.cc:4:17: note: did you mean 'str'?
/// \param [in] Str the string.
                ^~~
                str
```

Diagnostics

```
/// \brief Does something with \p str.
/// \param [in] Str the string.
/// \returns a modified string.
void do_something(const std::string &str);
```

```
example.cc:5:6: warning: '\returns' command used
        in a comment that is attached to a function
        returning void [-Wdocumentation]
/// \returns a modified string.
    ~^~~~~~~~~~~~~~~~~~~~~~~~~~
```

Diagnostics

```
/// \param x value of X coordinate.
/// \param x value of Y coordinate.
void do_something(int x, int y);
```

```
example.cc:2:12: warning: parameter 'x' is already
        documented [-Wdocumentation]
/// \param x value of Y coordinate.
           ^
```

Diagnostics

```
/// \brief Does something.
/// \deprecated
void do_something();
```

Diagnostics

```
/// \brief Does something.
/// \deprecated
void do_something();

example.cc:4:6: warning: declaration is marked with
        '\deprecated' command but does not have
        a deprecation attribute
/// \deprecated
    ~^~~~~~~~~~
example.cc:5:19: note: add a deprecation attribute
        to the declaration to silence this warning
void do_something();
                  ^
                    __attribute__((deprecated))
```

Diagnostics

```
#define MY_DEPRECATED __attribute__((deprecated))
/// \brief Does something.
/// \deprecated
void do_something();

example.cc:4:6: warning: declaration is marked with
        '\deprecated' command but does not have
        a deprecation attribute
/// \deprecated
    ~^~~~~~~~~~
example.cc:5:19: note: add a deprecation attribute
        to the declaration to silence this warning
void do_something();
                  ^
                    MY_DEPRECATED
```

LLVM and Clang are 99% -Wdocumentation clean.

Thanks to James Dennett!

LLVM and Clang are 99% -Wdocumentation clean.

Thanks to James Dennett!

Please configure LLVM and Clang with
`"--with-extra-options=-Wdocumentation -Wno-documentation-deprecated-sync"`
and help to clean up the last bits.

Enhanced code completion

Code completion provides:

- text or patterns to insert;
- priority;
- availability information;

Enhanced code completion

Code completion provides:

- text or patterns to insert;
- priority;
- availability information;
- *brief documentation*.

```
/// \param x ...
/// \param y ...
/// \brief Does foo.
/// \returns bar.
int foo(int x, int y);
```

```
/// Does foo.
///
/// Sometimes does bar.
int foo(int x, int y);
```

XML export

- A comment can be converted to XML.
- XML schema: `bindings/xml/comment-xml-schema.rng`.
- XML document includes:
  - parsed comment;
  - information about the declaration.

XML export example

```
/// \brief Does quux.
/// \tparam T foo.
/// \param x bar.
template<typename T> void foo(int x, T t);
```

```
<?xml version="1.0"?>
<Function templateKind="template" file="example.cc"
          line="5" column="6">
  <Name>foo</Name>
  <USR>c:@FT@&gt;1#Tfoo#I#t0.0#</USR>
  <Declaration>template &lt;typename T&gt;
               void foo(int x, T t)</Declaration>
  <Abstract><Para> Does quux. </Para></Abstract>
  <TemplateParameters> ...
  <Parameters> ...
```

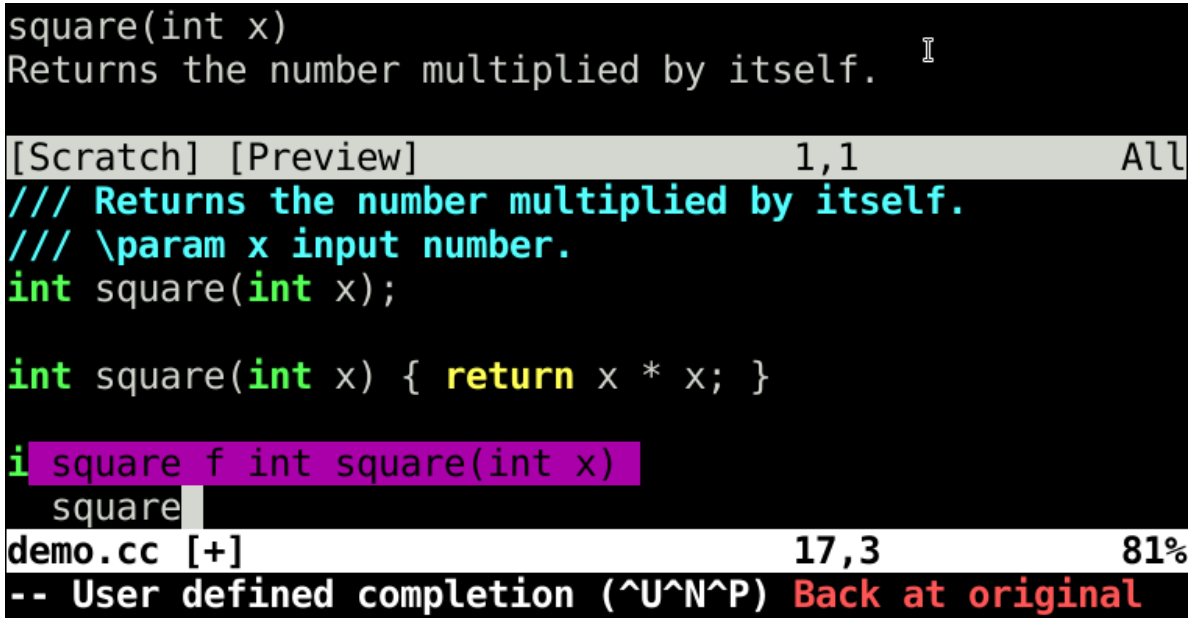
Performance

- Parse comments only if `-Wdocumentation` is passed.
- If you have **lots** (> 10000) of documentation comments, you might see a $< 5\%$ regression in `-fsyntax-only` time.
  - Parsing time is only a small fraction of compile time.
- Comments from system headers are skipped during normal compilation.
  - There is no sense in parsing them for diagnostics.
  - Pass `-fretain-comments-from-system-headers` if you want them back.

Demo

Vim with clang_complete

clang_complete with brief documentation support



https://github.com/gribozavr/clang_complete

Future work

- Resolve parameter names in `\p`.
- Resolve declaration references.
- Attach comments to macros.
- AST matchers?
- Integrate this feature with other IDEs (maybe after clangd?)
- Write a better documentation extraction tool.
    - Fix the declaration pretty-printer.

Summary

Now Clang can:

- parse documentation comments;
- find semantic errors in comments (try `-Wdocumentation`);
- export comments as XML.

This work enables more awesome features in future!

Q&A