CONCRETE SYNTAX PATTERNS

Piërre van de Laar | TMC, HTC 96 Eindhoven

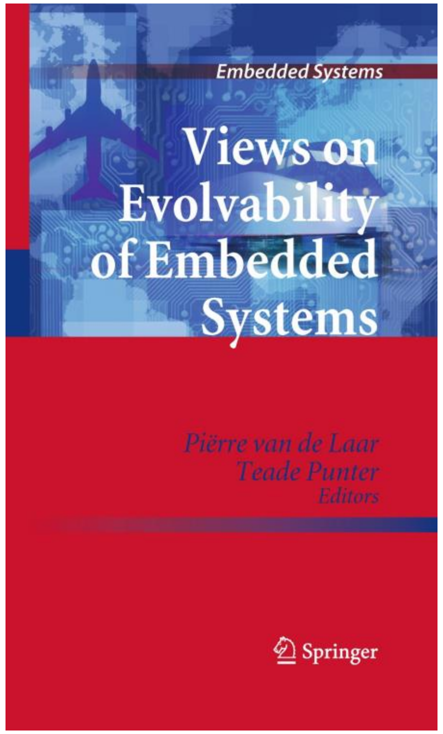
Powered by industry, academia and TNO

ESI

AGENDA

PIËRRE VAN DE LAAR

- Industrial innovator researching evolving product families



- Passionate about architecture, design, and code quality
- Wants to help the young software community to move from green field to brown field development

ESI AT A GLANCE

SYNOPSIS

- Foundation ESI started in 2002
- ESI acquired by TNO per January 2013
- ~60 staff members many with extensive industrial experience
- 8 Part-time professors
- Working at industry locations
- From embedded systems innovation to embedding innovation

FOCUS

Managing complexity of high-tech systems

through

- system architecting
- system reasoning and
- model-driven engineering

delivering

- methodologies validated in cutting-edge industrial practice

PARTNER BOARD

ASML
Canon CANON PRODUCTION PRINTING
itec equipment + automation tech
THALES Building a future we can all trust
PHILIPS
VANDERLANDE
UNIVERSITY OF AMSTERDAM
TU Delft
UNIVERSITY OF TWENTE.
TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY
Radboud University
TNO

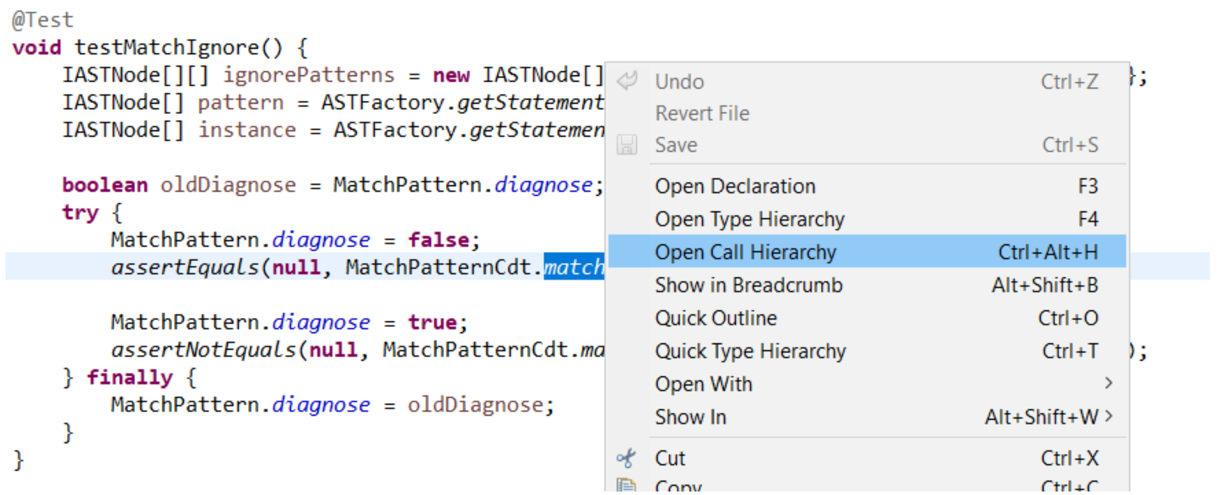
Capgemini engineering
ICT GROUP Strypes

AGENDA

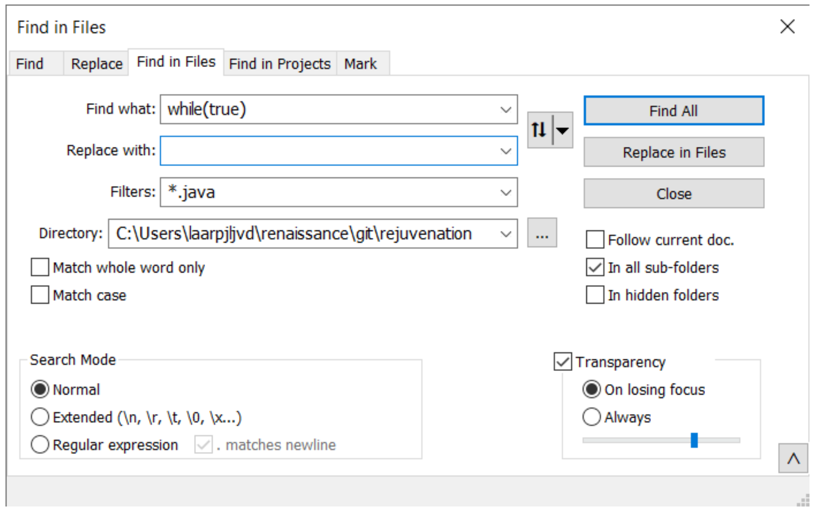
ANALYSIS OF CODE

Including

- Read code
- Search for piece of code
- Data flow
- Call graph
- Inheritance tree



```
@Test
void testMatchIgnore() {
    IASTNode[][] ignorePatterns = new IASTNode[]                                    };
    IASTNode[] pattern = ASTFactory.getStatement
    IASTNode[] instance = ASTFactory.getStatemen

    boolean oldDiagnose = MatchPattern.diagnose;
    try {
        MatchPattern.diagnose = false;
        assertEquals(null, MatchPatternCdt.match

        MatchPattern.diagnose = true;
        assertNotEquals(null, MatchPatternCdt.ma                                    );
    } finally {
        MatchPattern.diagnose = oldDiagnose;
    }
}
```



Find in Files

Find what: while(true)

Replace with:

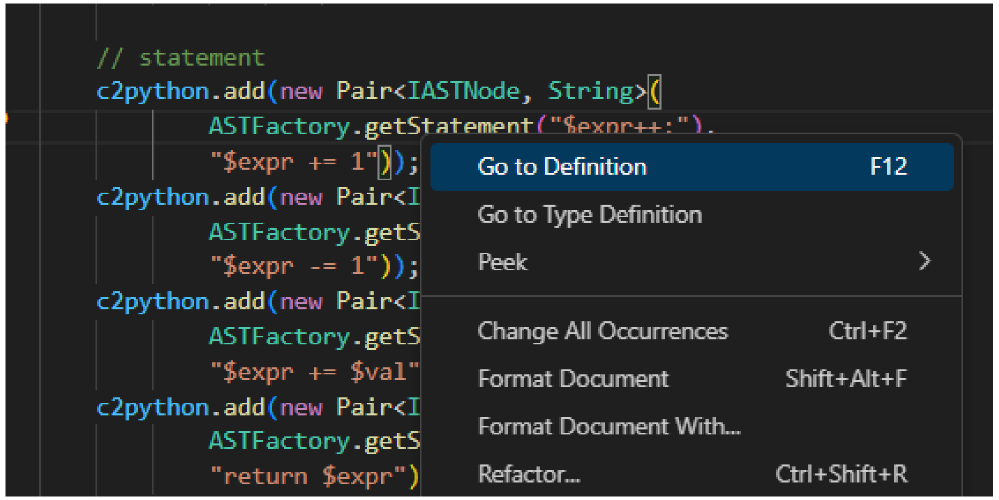
Filters: *.java

Directory: C:\Users\laarpjljvd\renaissance\git\rejuvenation

☐ Match whole word only
☐ Match case
☐ Follow current doc.
☑ In all sub-folders
☐ In hidden folders

Search Mode: [● Normal] ○ Extended (\n, \r, \t, \0, \x...) ○ Regular expression ☐ . matches newline

☑ Transparency: [● On losing focus] ○ Always



```
// statement
c2python.add(new Pair<IASTNode, String>(
        ASTFactory.getStatement("$expr++;"),
        "$expr += 1"));
c2python.add(new Pair<I
        ASTFactory.getS
        "$expr -= 1"));
c2python.add(new Pair<I
        ASTFactory.getS
        "$expr += $val"
c2python.add(new Pair<I
        ASTFactory.getS
        "return $expr")
```

1

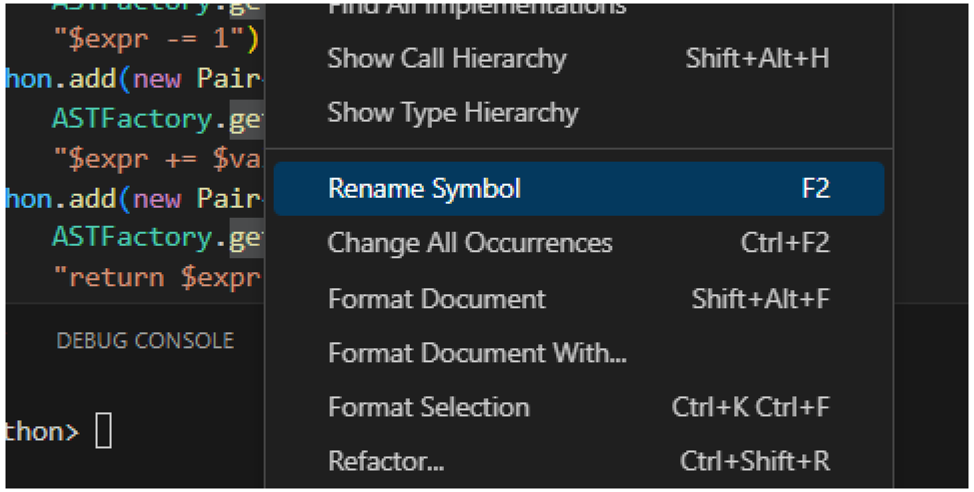
WHO HAS NEVER ANALYZED CODE?

Please raise hand

CHANGE CODE

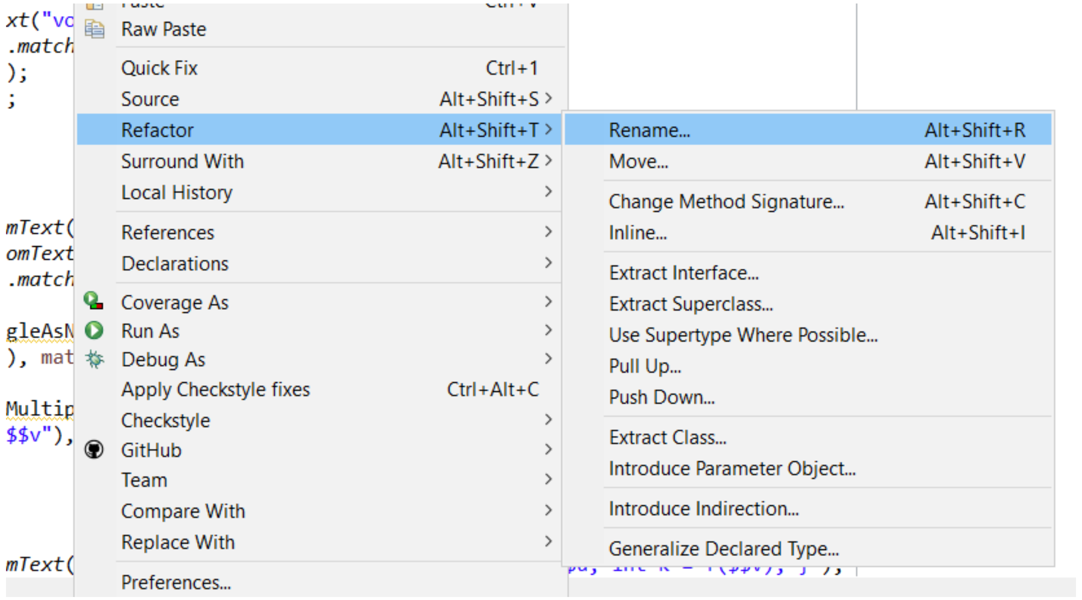
Including

- Rename variable or function
- Solve a bug
- Handle missed corner case
- Improve structure
- Improve quality
- Prepare for new feature
- Add new feature



```
    "$expr -= 1")
hon.add(new Pair
    ASTFactory.ge
    "$expr += $va
hon.add(new Pair
    ASTFactory.ge
    "return $expr
```

Show Call Hierarchy Shift+Alt+H
Show Type Hierarchy
Rename Symbol F2
Change All Occurrences Ctrl+F2
Format Document Shift+Alt+F
Format Document With...
Format Selection Ctrl+K Ctrl+F
Refactor... Ctrl+Shift+R
DEBUG CONSOLE
thon>



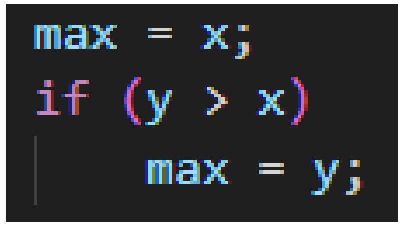
```
xt("vo
.match
);
;

mText(
omText
.match

gleAsN
), mat

Multip
$$v"),

mText(
```

2

WHO HAS NEVER CHANGED CODE?

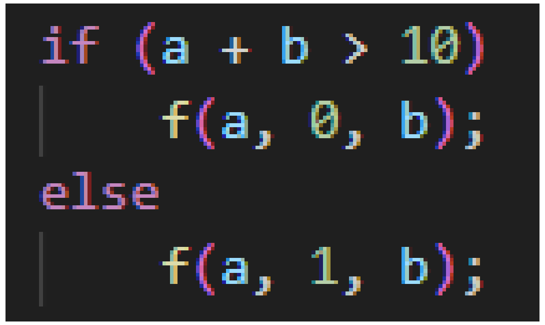
Please raise hand

UNDERSTAND CODE



```
max = x;
if (y > x)
    max = y;
```

The variable max is set to maximum value of the variables x and y



```
if (a + b > 10)
    f(a, 0, b);
else
    f(a, 1, b);
```

The function f is called

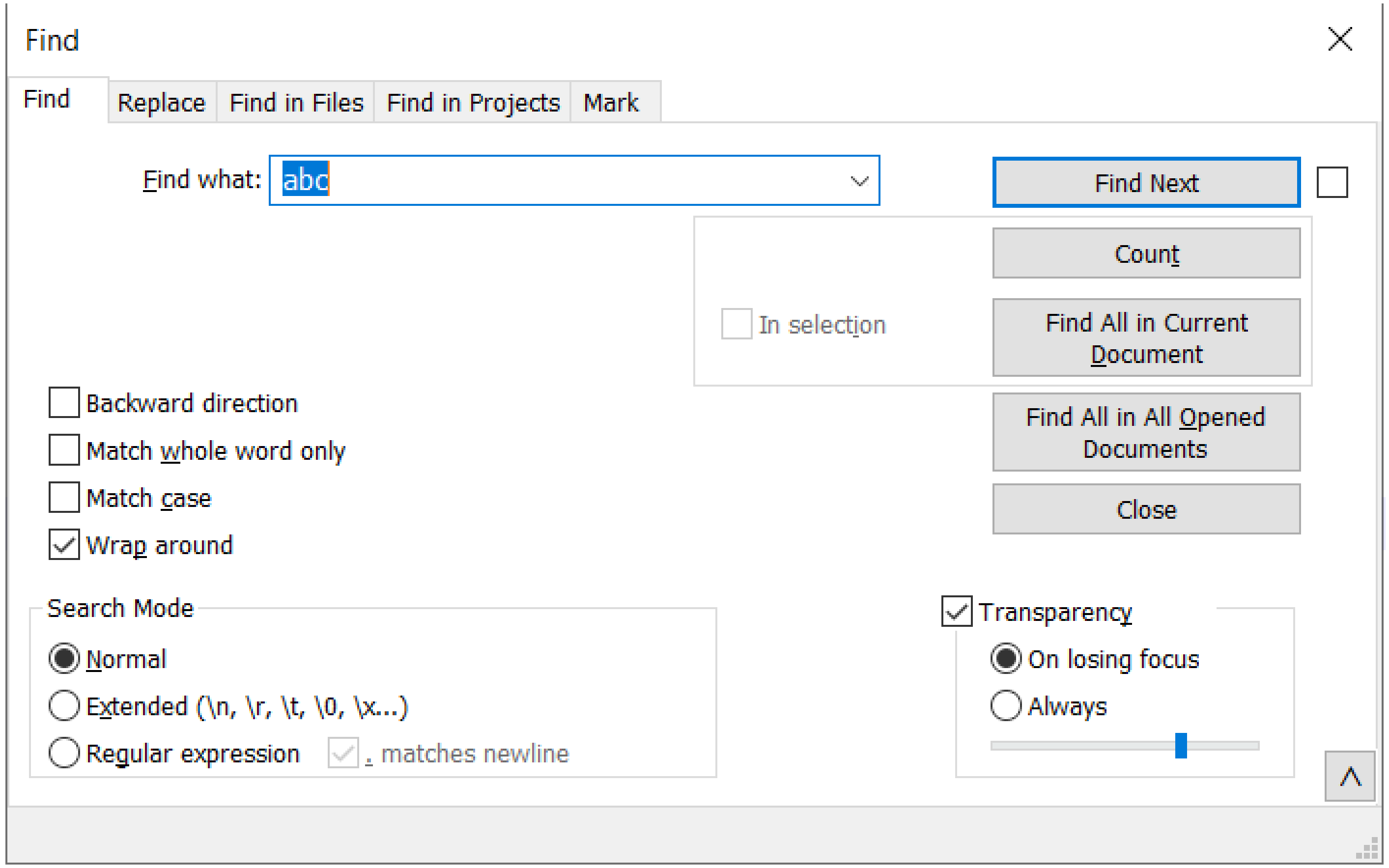
- The first argument is a
- The third argument is b
- The second argument is
  0 when a plus b is larger than 10 and
  1 otherwise

3

WHO COULD NOT UNDERSTAND CODE?

Please raise hand

FIND



Find

Find | Replace | Find in Files | Find in Projects | Mark

Find what: abc

- [ ] In selection
- [ ] Backward direction
- [ ] Match whole word only
- [ ] Match case
- [✓] Wrap around

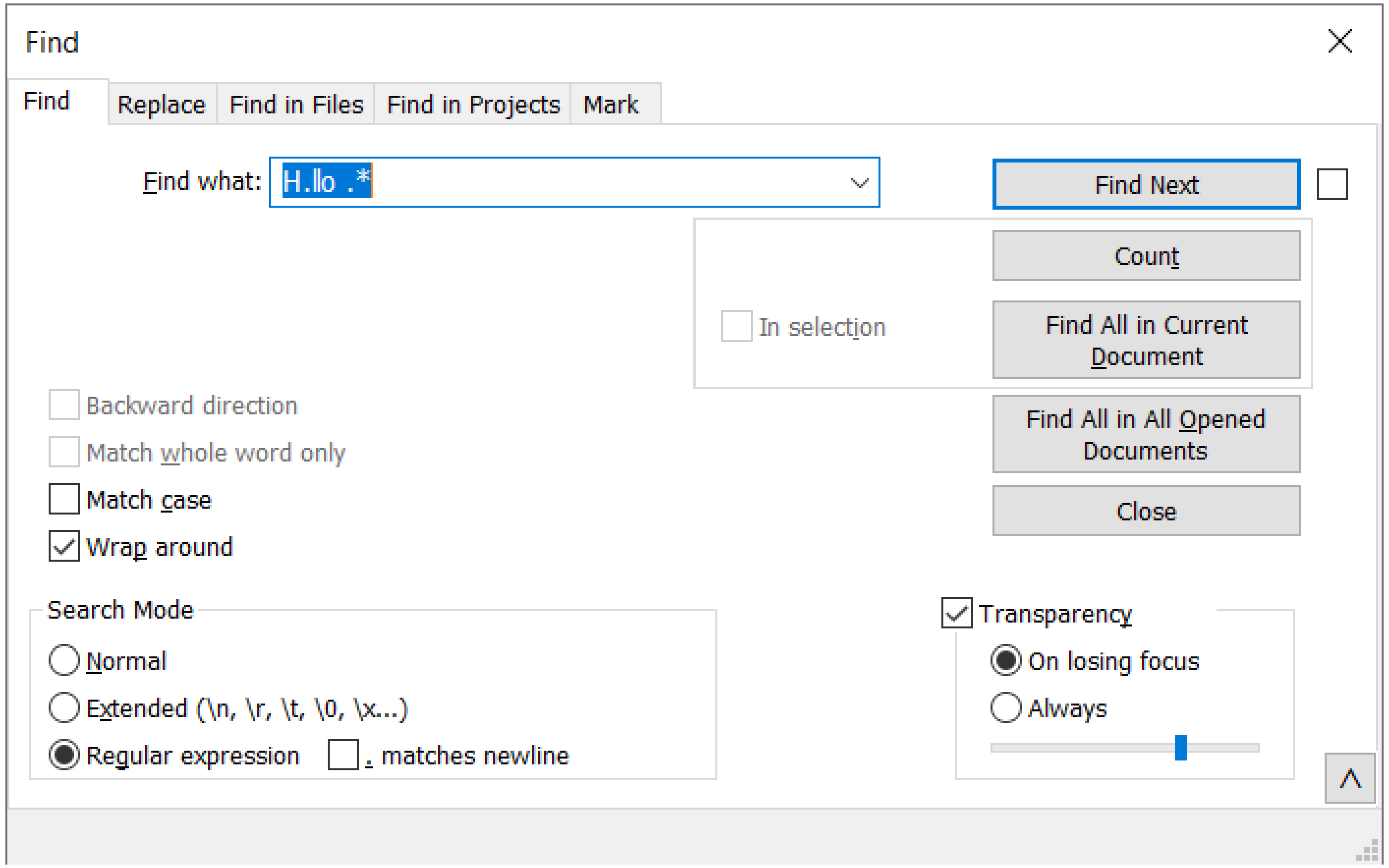
Search Mode
- [● Normal]
- Extended (\n, \r, \t, \0, \x...)
- Regular expression
- [✓] . matches newline

- [✓] Transparency
- [● On losing focus]
- Always

Find Next
Count
Find All in Current Document
Find All in All Opened Documents
Close

abc	✓	Identical
def	✗	
ABC	✓	Equivalent
DEF	✗	

FIND



Find what: H.llo .*

Search Mode: Regular expression

Match case [ ] Wrap around [✓]

Find
Find | Replace | Find in Files | Find in Projects | Mark
Find Next
Count
In selection
Find All in Current Document
Find All in All Opened Documents
Close
Backward direction
Match whole word only
Normal
Extended (\n, \r, \t, \0, \x...)
. matches newline
Transparency
On losing focus
Always

Hello World!	✓
Hoollo Eindhoven	✗
Hallo 040coders	✓

4

WHO COULD NOT PREDICT FIND RESULTS?

Please raise hand

{Ø4ØCODERS.nl}

Tech Talks and Videos. Food and Drinks.

Every 3rd Thursday of the Month,
Somewhere in the 040 Area.

Software developers only. No recruiters or other tie-wearing persons allowed.

AGENDA

CODE IS NOT TEXT

- Find

```
ready(); set(); go();
```

- Match

```
ready(); set(); go();
```

✓

```
ready( ); set( ); go( );
```

✗

```
ready();
set();
go();
```

✗

```
ready();    // first signal - prepare action
set();      /* second signal - acquire resources */
go();       // final signal - take action
```

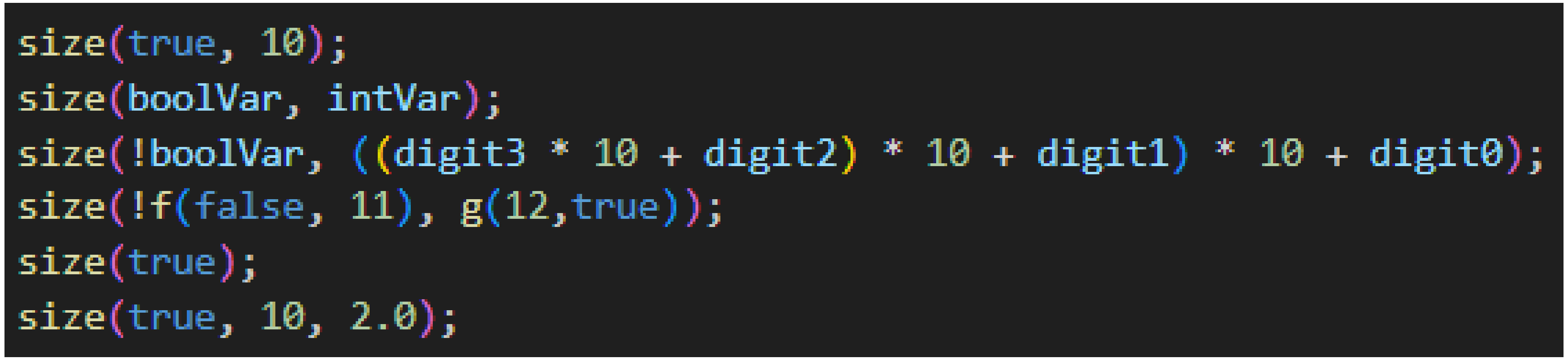
✗

- Text-based search is sensitive to white spaces and comments

REGULAR EXPRESSIONS CANNOT HANDLE CODE

- Code is data with clear syntax and semantics

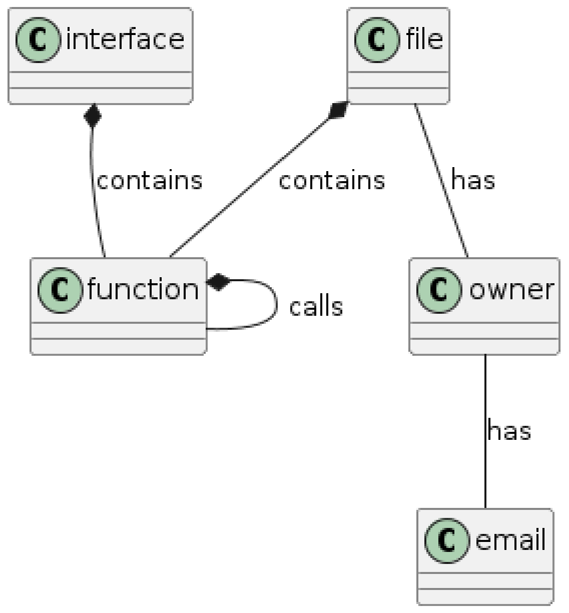
- Find size function with exactly two arguments



```
size(true, 10);                                                              ✓
size(boolVar, intVar);                                                       ✓
size(!boolVar, ((digit3 * 10 + digit2) * 10 + digit1) * 10 + digit0);        ✓
size(!f(false, 11), g(12,true));                                             ✓
size(true);                                                                  ✗
size(true, 10, 2.0);                                                         ✗
```

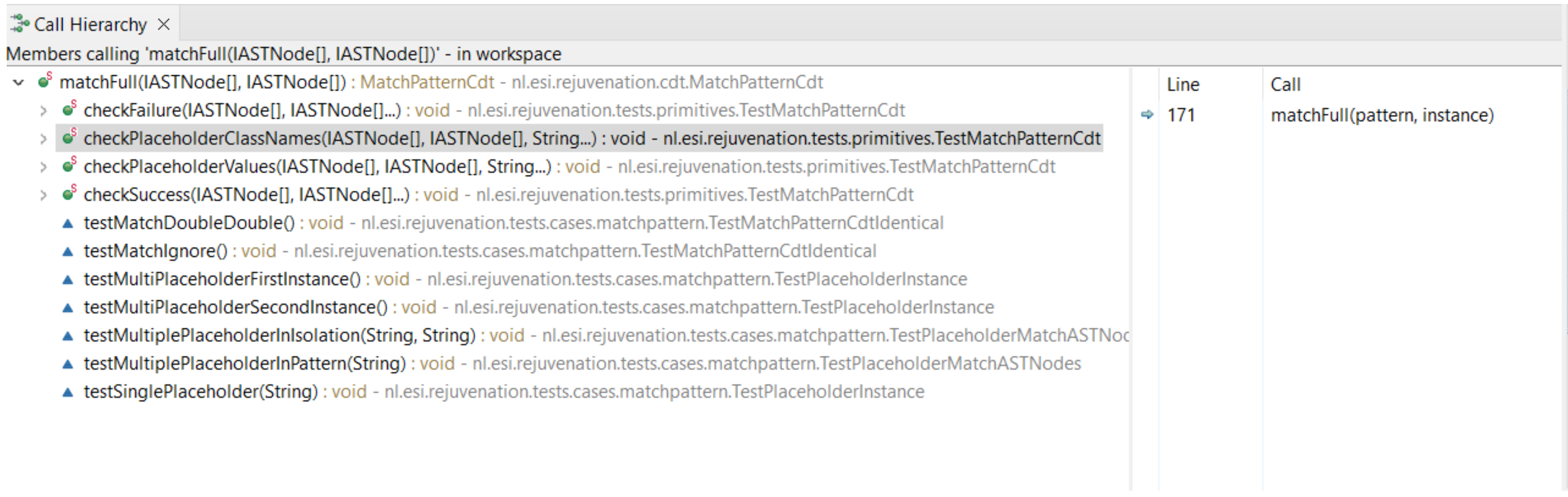
- Regular Expressions cannot handle arbitrary levels of nested brackets and expressions

TOOLS CANNOT BE INTEGRATED OR EXTENDED



Contact all stakeholders by email before changing an interface

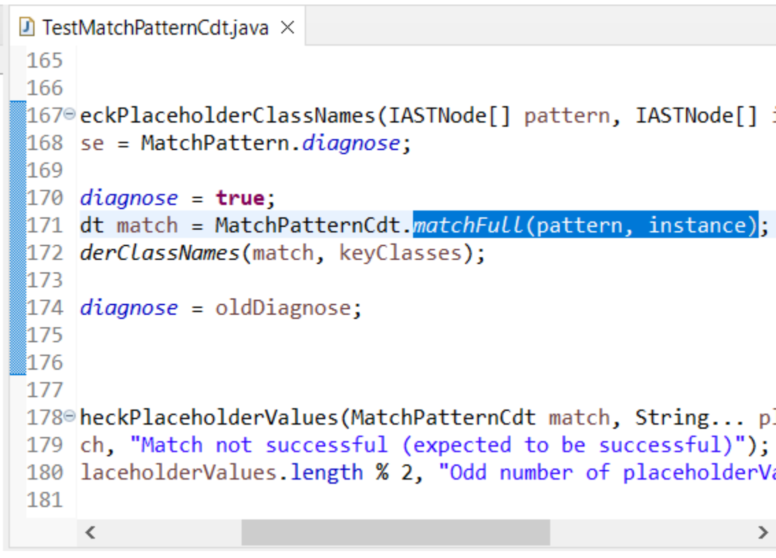
- Software developers can develop software tools
- Database contains relation between code owners and files
- IDE offers call graph of a single function


Call Hierarchy

Members calling 'matchFull(IASTNode[], IASTNode[])' - in workspace

```
matchFull(IASTNode[], IASTNode[]) : MatchPatternCdt - nl.esi.rejuvenation.cdt.MatchPatternCdt
  checkFailure(IASTNode[], IASTNode[]...) : void - nl.esi.rejuvenation.tests.primitives.TestMatchPatternCdt
  checkPlaceholderClassNames(IASTNode[], IASTNode[], String...) : void - nl.esi.rejuvenation.tests.primitives.TestMatchPatternCdt
  checkPlaceholderValues(IASTNode[], IASTNode[], String...) : void - nl.esi.rejuvenation.tests.primitives.TestMatchPatternCdt
  checkSuccess(IASTNode[], IASTNode[]...) : void - nl.esi.rejuvenation.tests.primitives.TestMatchPatternCdt
  testMatchDoubleDouble() : void - nl.esi.rejuvenation.tests.cases.matchpattern.TestMatchPatternCdtIdentical
  testMatchIgnore() : void - nl.esi.rejuvenation.tests.cases.matchpattern.TestMatchPatternCdtIdentical
  testMultiPlaceholderFirstInstance() : void - nl.esi.rejuvenation.tests.cases.matchpattern.TestPlaceholderInstance
  testMultiPlaceholderSecondInstance() : void - nl.esi.rejuvenation.tests.cases.matchpattern.TestPlaceholderInstance
  testMultiplePlaceholderInIsolation(String, String) : void - nl.esi.rejuvenation.tests.cases.matchpattern.TestPlaceholderMatchASTNoc
  testMultiplePlaceholderInPattern(String) : void - nl.esi.rejuvenation.tests.cases.matchpattern.TestPlaceholderMatchASTNodes
  testSinglePlaceholder(String) : void - nl.esi.rejuvenation.tests.cases.matchpattern.TestPlaceholderInstance
```

Line	Call
171	matchFull(pattern, instance)


TestMatchPatternCdt.java

```
165
166
167 eckPlaceholderClassNames(IASTNode[] pattern, IASTNode[]
168 se = MatchPattern.diagnose;
169
170 diagnose = true;
171 dt match = MatchPatternCdt.matchFull(pattern, instance);
172 derClassNames(match, keyClasses);
173
174 diagnose = oldDiagnose;
175
176
177
178 heckPlaceholderValues(MatchPatternCdt match, String... p
179 ch, "Match not successful (expected to be successful)");
180 laceholderValues.length % 2, "Odd number of placeholderVa
181
```

LEARNING CURVE FOR TOOLS

- Powerful tools, like linter and compiler, parse code

- Development and maintenance of a parser is huge effort
  - Industrial quality C++ compiler at least 2 years
  - CDT parser of Eclipse will not support C++ 20 and beyond

- Parser represents code as Abstract Syntax Tree (AST)

- Abstract Syntax Tree not developed for analysis and change but for high performance

ABSTRACT SYNTAX TREE

Hello World

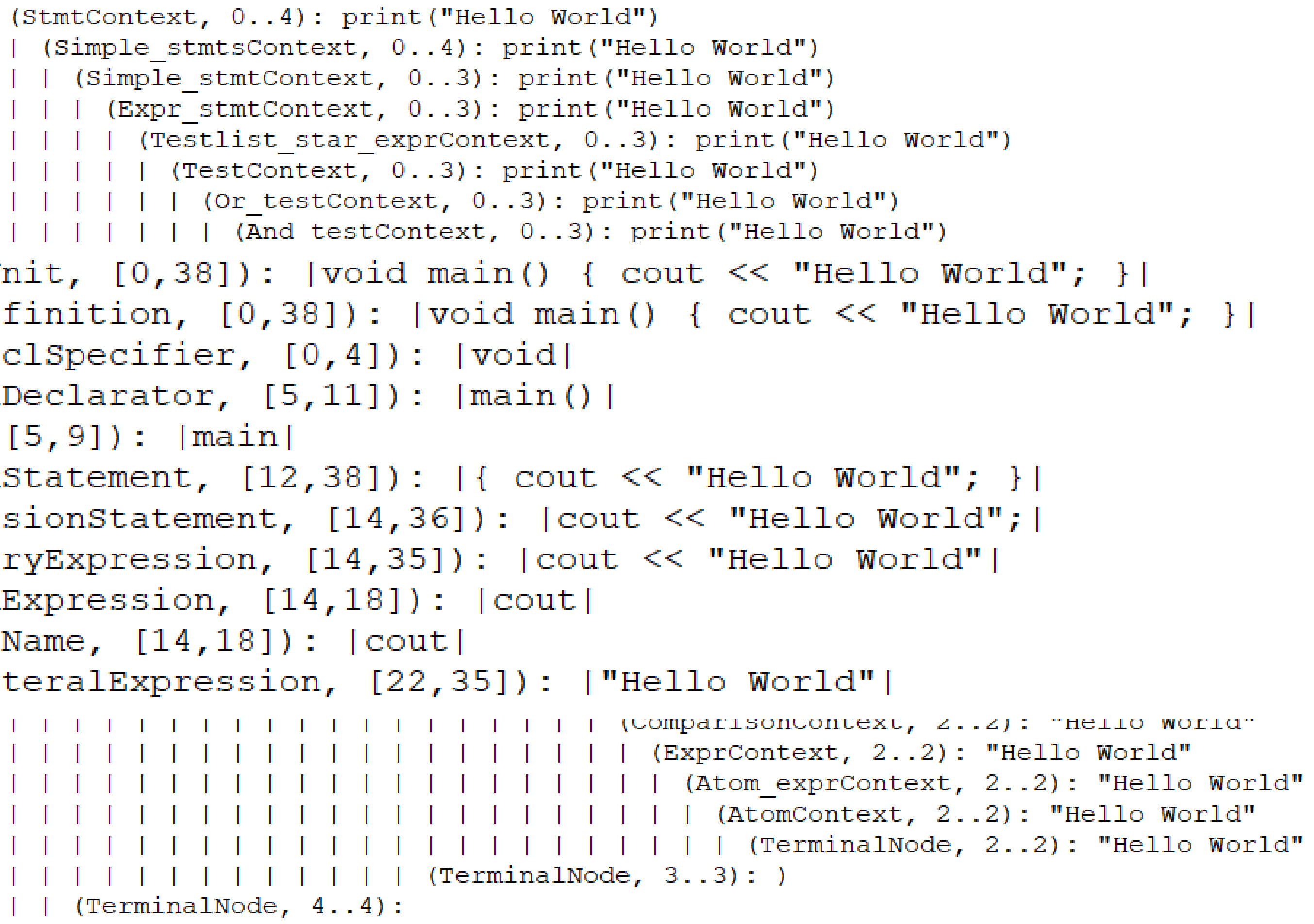
in Python (ANTLR grammar)

in C++ (CDT)

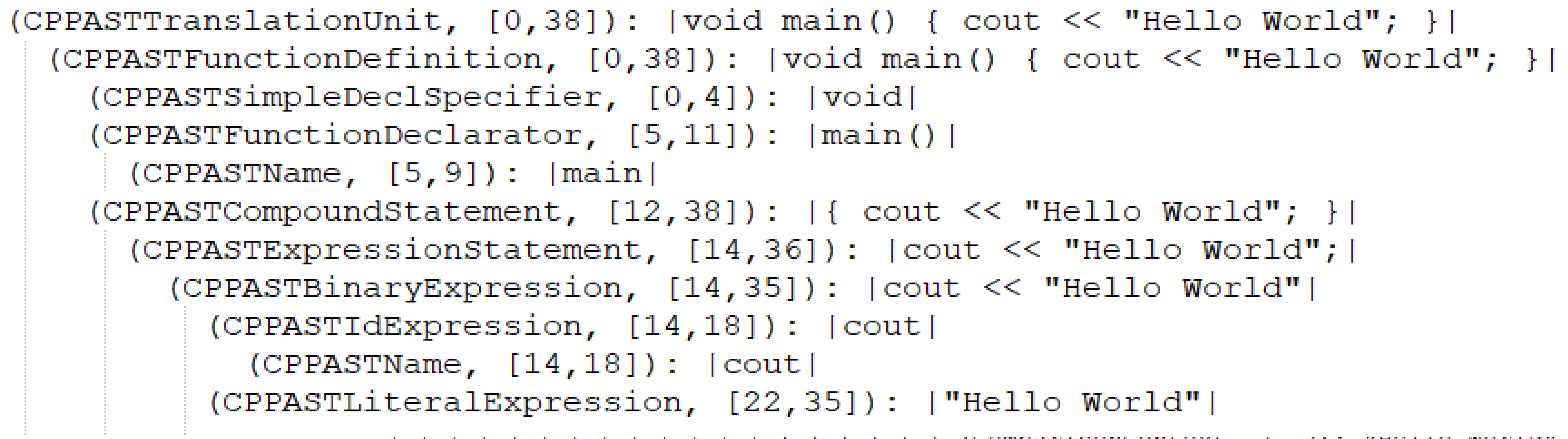
Complicated!

Too steep learning curve

Especially for occasional usage



```
(StmtContext, 0..4): print("Hello World")
| (Simple_stmtsContext, 0..4): print("Hello World")
| | (Simple_stmtContext, 0..3): print("Hello World")
| | | (Expr_stmtContext, 0..3): print("Hello World")
| | | | (Testlist_star_exprContext, 0..3): print("Hello World")
| | | | | (TestContext, 0..3): print("Hello World")
| | | | | | (Or_testContext, 0..3): print("Hello World")
| | | | | | | (And_testContext, 0..3): print("Hello World")
| | | | | | | | | | | | | | | | | | | (ComparisonContext, 2..2): "Hello World"
| | | | | | | | | | | | | | | | | | | | (ExprContext, 2..2): "Hello World"
| | | | | | | | | | | | | | | | | | | | | (Atom_exprContext, 2..2): "Hello World"
| | | | | | | | | | | | | | | | | | | | | | (AtomContext, 2..2): "Hello World"
| | | | | | | | | | | | | | | | | | | | | | | (TerminalNode, 2..2): "Hello World"
| | | | | | | | | | | | | (TerminalNode, 3..3): )
| | (TerminalNode, 4..4):
```



```
(CPPASTTranslationUnit, [0,38]): |void main() { cout << "Hello World"; }|
  (CPPASTFunctionDefinition, [0,38]): |void main() { cout << "Hello World"; }|
    (CPPASTSimpleDeclSpecifier, [0,4]): |void|
    (CPPASTFunctionDeclarator, [5,11]): |main()|
      (CPPASTName, [5,9]): |main|
    (CPPASTCompoundStatement, [12,38]): |{ cout << "Hello World"; }|
      (CPPASTExpressionStatement, [14,36]): |cout << "Hello World";|
        (CPPASTBinaryExpression, [14,35]): |cout << "Hello World"|
          (CPPASTIdExpression, [14,18]): |cout|
            (CPPASTName, [14,18]): |cout|
          (CPPASTLiteralExpression, [22,35]): |"Hello World"|
```

LIMITATIONS OF CURRENT TOOLS

- Inappropriate tools
  - Code isn’t text, regular expressions cannot handle code
- Tools are hard to integrate and extend
  - Lack of API
- Too steep learning curve
  - AST is complicated

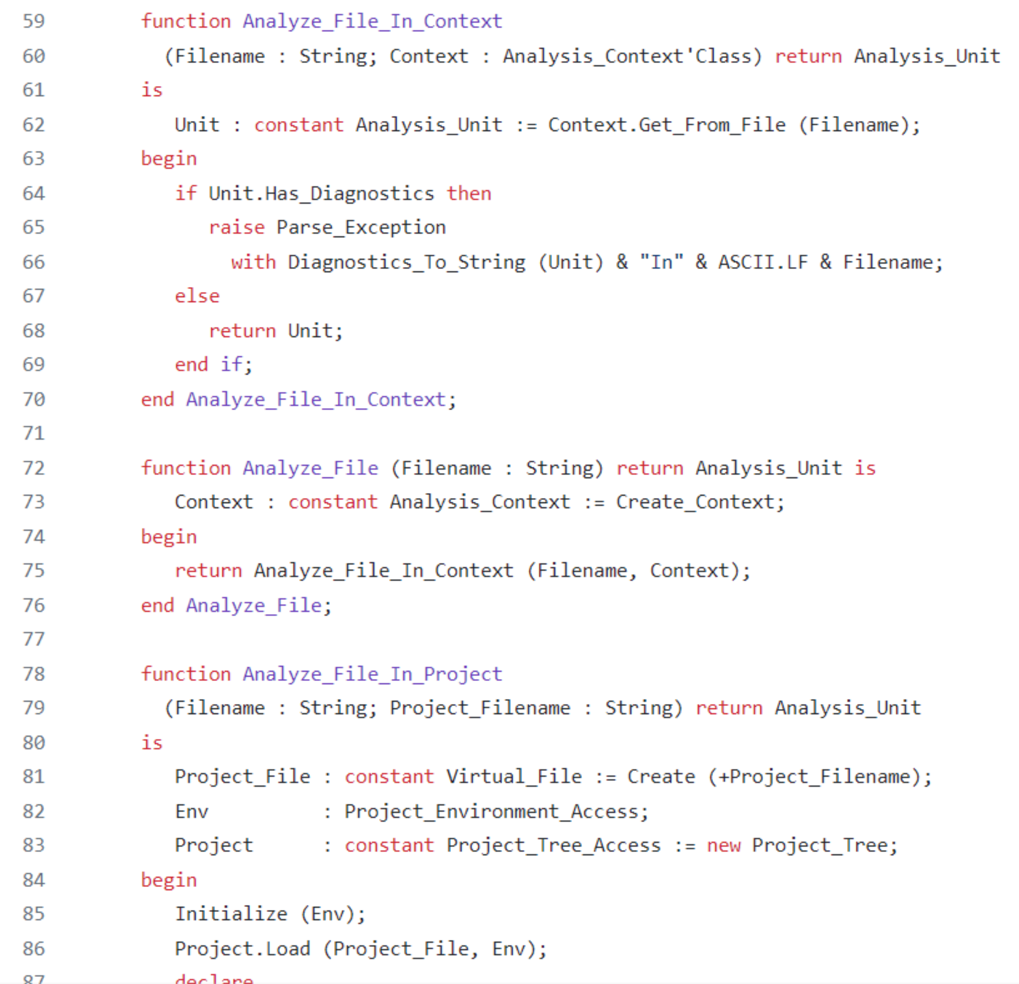
AGENDA

ABSTRACT VS CONCRETE SYNTAX

- All coders can read code

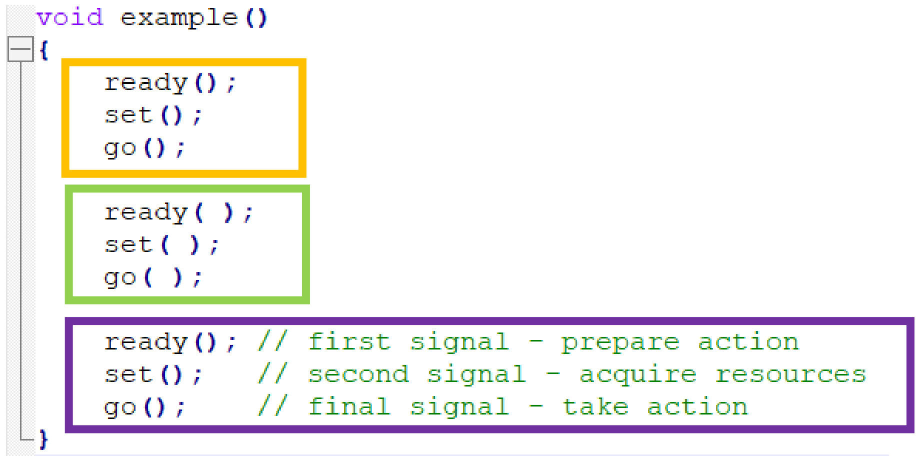
- All coders know the concrete syntax

- Not all coders know the abstract syntax



```
function Analyze_File_In_Context
  (Filename : String; Context : Analysis_Context'Class) return Analysis_Unit
is
   Unit : constant Analysis_Unit := Context.Get_From_File (Filename);
begin
   if Unit.Has_Diagnostics then
      raise Parse_Exception
        with Diagnostics_To_String (Unit) & "In" & ASCII.LF & Filename;
   else
      return Unit;
   end if;
end Analyze_File_In_Context;

function Analyze_File (Filename : String) return Analysis_Unit is
   Context : constant Analysis_Context := Create_Context;
begin
   return Analyze_File_In_Context (Filename, Context);
end Analyze_File;

function Analyze_File_In_Project
  (Filename : String; Project_Filename : String) return Analysis_Unit
is
   Project_File : constant Virtual_File := Create (+Project_Filename);
   Env          : Project_Environment_Access;
   Project      : constant Project_Tree_Access := new Project_Tree;
begin
   Initialize (Env);
   Project.Load (Project_File, Env);
   declare
```

EQUIVALENT SUBTREES



```
void example()
{
    ready();
    set();
    go();

    ready( );
    set( );
    go( );

    ready(); // first signal - prepare action
    set();   // second signal - acquire resources
    go();    // final signal - take action
}
```

```
(CPPASTExpressionStatement, [21,29])
  (CPPASTFunctionCallExpression, [21,28])
    (CPPASTIdExpression, [21,26])
      (CPPASTName, [21,26]): |ready|
(CPPASTExpressionStatement, [34,40])
  (CPPASTFunctionCallExpression, [34,39])
    (CPPASTIdExpression, [34,37])
      (CPPASTName, [34,37]): |set|
(CPPASTExpressionStatement, [45,50])
  (CPPASTFunctionCallExpression, [45,49])
    (CPPASTIdExpression, [45,47])
      (CPPASTName, [45,47]): |go|
```

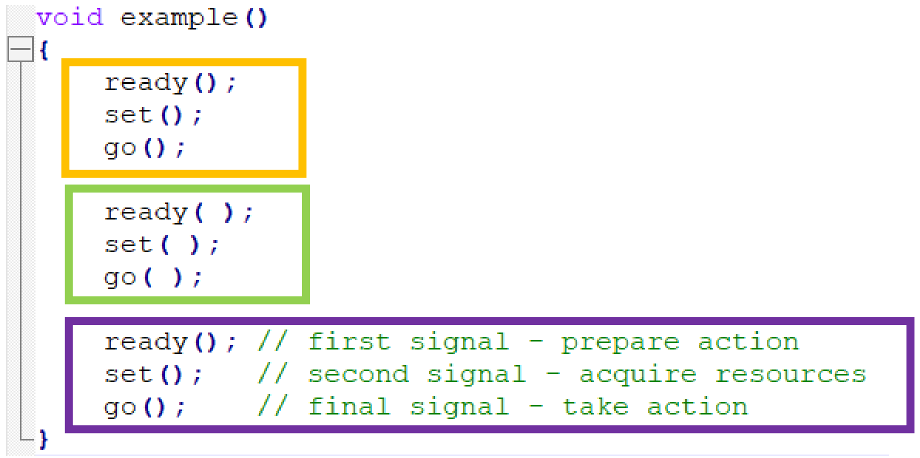
```
(CPPASTExpressionStatement, [56,65])
  (CPPASTFunctionCallExpression, [56,64])
    (CPPASTIdExpression, [56,61])
      (CPPASTName, [56,61]): |ready|
(CPPASTExpressionStatement, [70,77])
  (CPPASTFunctionCallExpression, [70,76])
    (CPPASTIdExpression, [70,73])
      (CPPASTName, [70,73]): |set|
(CPPASTExpressionStatement, [82,88])
  (CPPASTFunctionCallExpression, [82,87])
    (CPPASTIdExpression, [82,84])
      (CPPASTName, [82,84]): |go|
```

```
(CPPASTExpressionStatement, [94,102])
  (CPPASTFunctionCallExpression, [94,101])
    (CPPASTIdExpression, [94,99])
      (CPPASTName, [94,99]): |ready|
(CPPASTExpressionStatement, [140,146])
  (CPPASTFunctionCallExpression, [140,145])
    (CPPASTIdExpression, [140,143])
      (CPPASTName, [140,143]): |set|
(CPPASTExpressionStatement, [190,195])
  (CPPASTFunctionCallExpression, [190,194])
    (CPPASTIdExpression, [190,192])
      (CPPASTName, [190,192]): |go|
```

ESI
Powered by industry, academia and TNO

Powered by industry, academia and TNO

EQUIVALENT SUBTREES



```
void example()
{
    ready();
    set();
    go();

    ready( );
    set( );
    go( );

    ready(); // first signal - prepare action
    set();   // second signal - acquire resources
    go();    // final signal - take action
}
```

```
(CPPASTExpressionStatement, [      ])
  (CPPASTFunctionCallExpression, [      ])
    (CPPASTIdExpression, [      ])
      (CPPASTName, [      ]): |ready|
(CPPASTExpressionStatement, [      ])
  (CPPASTFunctionCallExpression, [      ])
    (CPPASTIdExpression, [      ])
      (CPPASTName, [      ]): |set|
(CPPASTExpressionStatement, [      ])
  (CPPASTFunctionCallExpression, [      ])
    (CPPASTIdExpression, [      ])
      (CPPASTName, [      ]): |go|
```

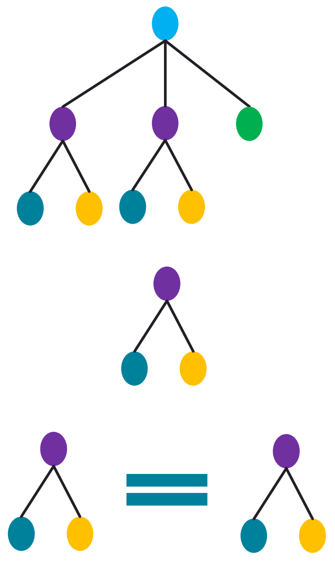
```
(CPPASTExpressionStatement, [      ])
  (CPPASTFunctionCallExpression, [      ])
    (CPPASTIdExpression, [      ])
      (CPPASTName, [      ]): |ready|
(CPPASTExpressionStatement, [      ])
  (CPPASTFunctionCallExpression, [      ])
    (CPPASTIdExpression, [      ])
      (CPPASTName, [      ]): |set|
(CPPASTExpressionStatement, [      ])
  (CPPASTFunctionCallExpression, [      ])
    (CPPASTIdExpression, [      ])
      (CPPASTName, [      ]): |go|
```

```
(CPPASTExpressionStatement, [       ])
  (CPPASTFunctionCallExpression, [       ])
    (CPPASTIdExpression, [    ])
      (CPPASTName, [      ]): |ready|
(CPPASTExpressionStatement, [        ])
  (CPPASTFunctionCallExpression, [        ])
    (CPPASTIdExpression, [        ])
      (CPPASTName, [        ]): |set|
(CPPASTExpressionStatement, [        ])
  (CPPASTFunctionCallExpression, [        ])
    (CPPASTIdExpression, [        ])
      (CPPASTName, [        ]): |go|
```

Powered by industry, academia and TNO

HOW TO ANALYZE AND CHANGE CODE?

- Find instances of a pattern within the code
- Parser represents code as Abstract Syntax Tree
- An instance is a subtree: a piece of that Abstract Syntax Tree
- All instances of a pattern have **equivalent** subtrees
- Use standard tree matching
- Yet, do not expose the AST to the user!

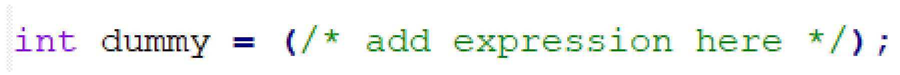


HOW TO GET THE AST OF A PATTERN?

- Without exposing the AST to the user
- Use the parser!
- Limited kinds of patterns
  - Statement(s)
  - Declaration(s)
  - Expression
- Parsers are not designed for concrete syntax patterns, yet!

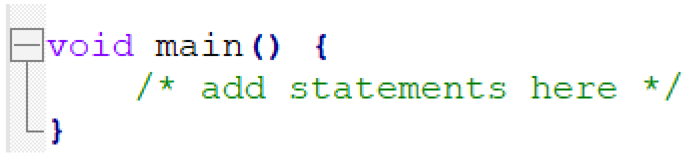
HOW TO EASILY GET THE SUBTREE OF A PATTERN?

- Make small program around pattern for parser



```
int dummy = (/* add expression here */);
```

```
(CPPASTTranslationUnit, [0,18]): |int dummy = (...);|
(CPPASTSimpleDeclaration, [0,18]): |int dummy = (...);|
(CPPASTSimpleDeclSpecifier, [0,3]): |int|
(CPPASTDeclarator, [4,17]): |dummy = (...)|
(CPPASTName, [4,9]): |dummy|
(CPPASTEqualsInitializer, [10,17]): |= (...)|
(CPPASTUnaryExpression, [12,17]): |(...)|
```



```
void main() {
    /* add statements here */
}
```

```
(CPPASTTranslationUnit, [0,22]): |void main() { ... }|
(CPPASTFunctionDefinition, [0,22]): |void main() { ... }|
(CPPASTSimpleDeclSpecifier, [0,4]): |void|
(CPPASTFunctionDeclarator, [5,11]): |main()|
(CPPASTName, [5,9]): |main|
(CPPASTCompoundStatement, [12,22]): |{ ... }|
```

- Extract relevant subtree from AST for pattern

EXTRA INGREDIENT: PLACEHOLDERS

- Match any AST node
  - Single statement, single expression, function name, …
  - Comparable to **.** the wildcard of regular expressions
  - **$name** in C++, **$S_name** in Ada, …

- Match list of AST Nodes
  - List of arguments, list of parameters, list of initial values, list of enumeration values, list of statements, …
  - Comparable to **.*** wildcard with Kleene star of regular expressions
  - **$$name** in C++, **$M_name** in Ada, …

REJUVENATION LIBRARY

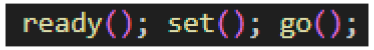
- Enable developer to focus on analysis and change
  - Steps on complete code base
    - Gather information, combine knowledge, simplify, …
  - Actions within step
    - Find, apply, replace, filter, …

- Fluent interface supports developer
  - Works on code, yet hides AST representation
    - Uses Concrete Syntax Patterns
  - Extendable
    - Integrates in any program
  - Ensures the same code is analyzed as is built
    - Same include paths, same defines
  - Ensures changes are formatted
    - Same pretty printer, same configuration settings
  - Ensures high performance
    - Parallelizes analysis and change

AGENDA

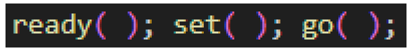
C++ CODE EXAMPLE

- What C++ code matches this C++ pattern `Pattern.statements("ready(); set(); go();")` ?



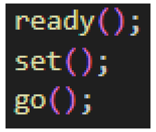
```
ready(); set(); go();
```

✓



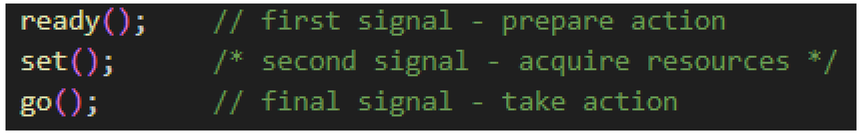
```
ready( ); set( ); go( );
```

✓



```
ready();
set();
go();
```

✓



```
ready();    // first signal - prepare action
set();      /* second signal - acquire resources */
go();       // final signal - take action
```

✓

Powered by industry, academia and TNO

PLACEHOLDER FOR SINGLE NODE

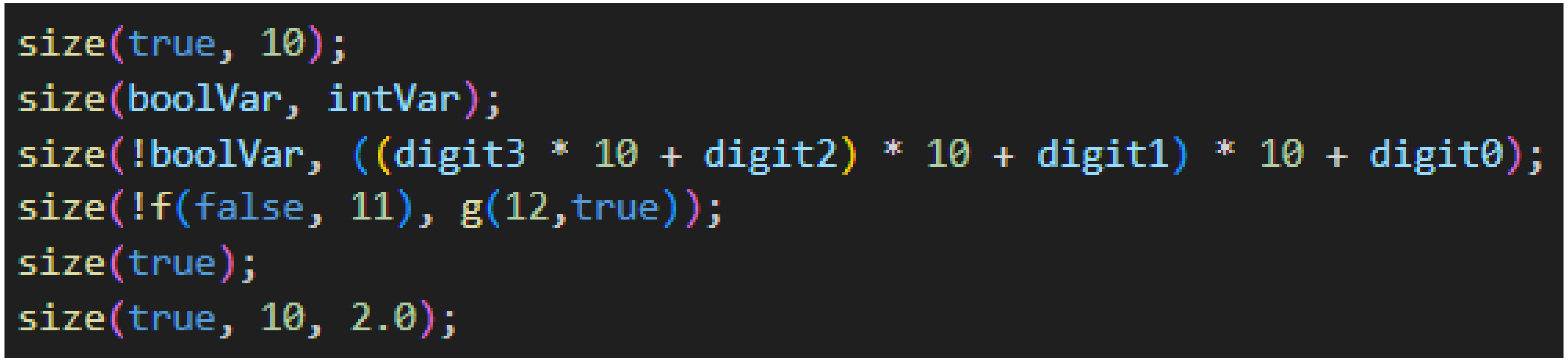
- What code matches this pattern `Pattern.declaration("int $name;")` ?

✓	`int a;`	✓
✗	`int anotherValue = 0;`	✗
✗	`bool value;`	✓
✓	`int b_c;`	✓
✓	`int mySpecialVariable;`	✓

- What code matches this pattern `Pattern.declaration("$type $name;")` ?

PLACEHOLDER FOR SINGLE NODE

- What code matches this pattern `Pattern.statement("size($arg1, $arg2);")`?



`size(true, 10);`	✓
`size(boolVar, intVar);`	✓
`size(!boolVar, ((digit3 * 10 + digit2) * 10 + digit1) * 10 + digit0);`	✓
`size(!f(false, 11), g(12,true));`	✓
`size(true);`	✗
`size(true, 10, 2.0);`	✗

PLACEHOLDER FOR LIST OF NODES

- What code matches this pattern `Pattern.statement("size($$args);")` ?

```
✓ size(true, 10);
✓ size(boolVar, intVar);
✓ size(!boolVar, ((digit3 * 10 + digit2) * 10 + digit1) * 10 + digit0);
✓ size(!f(false, 11), g(12,true));
✓ size();
✓ size(true);
✓ size(true, 10, 2.0);
```

MULTIPLE OCCURRENCES OF PLACEHOLDERS

- What code matches this pattern `Pattern.statement("if($cond) $stmt; else $stmt;")` ?
- Reoccurrence of a placeholder adds a constraint
  - All occurrences of a placeholder must be equivalent.

```
if (cond) return 0; else return 0;
```

✓

```
if (cond) return false; else return 0;
```

✗

```
if (x > 3) {
    x++;  // increment x
} else {
    x++;  /* increase x */
}
```

✓

AGENDA

FIND & APPLY

- Find `Pattern.expression("$x.size() == 0")`
- Apply `match -> count++`
- Apply `match -> System.out.println(match.getNodes()[0].getFileLocation())`
- Apply `match -> ASTShower.showNode(match.getSingleAsNode("$x"))`

FILTER

- Find, filter, and apply

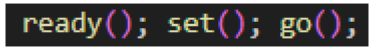
- Use filter for additional checks
  - Anything is possible
  - Typically, check on placeholders

```
match -> match.getSingleAsString("$f").startsWith("PackagePrefix")
```

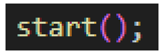
AGENDA

FIND & REPLACE

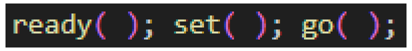
- Find `Pattern.statements("ready(); set(); go();")` and replace with `"start();"`



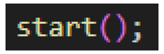
```
ready(); set(); go();
```



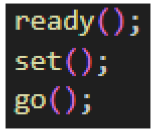
```
start();
```



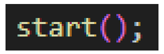
```
ready( ); set( ); go( );
```



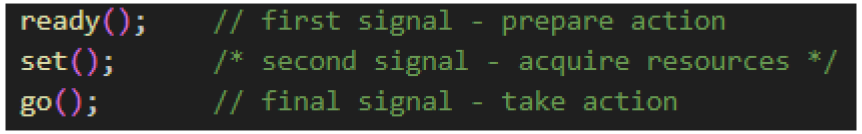
```
start();
```



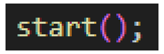
```
ready();
set();
go();
```



```
start();
```



```
ready();    // first signal - prepare action
set();      /* second signal - acquire resources */
go();       // final signal - take action
```



```
start();
```

✱ depends on configuration

FIND & REPLACE

Back reference

- Find `Pattern.expression("$x * $x")` and replace with `"power($x, 2)"`

```
var * var
```

```
(x+y) * (x+y)
```

```
power(var, 2)
```

✶

```
power((x+y), 2)
```

✶Need for post-processing

FILTER (1/2)

- Find, filter, and replace
- Use filter to prevent incorrect changes from happening

TEST YOUR CHANGES! (1/3)

- Is finding the pattern

```
Pattern.statement("if ($cond) { $f($$before, $tValue, $$after); } else { $f($$before, $fValue, $$after); }")
```

  and replacing by

```
"$f($$before, $cond ? $tValue : $fValue, $$after);"
```

  correct?

- No – One reason

  Execution order is changed

  `$cond` is no longer executed first

  and this might result in different behavior

```
if (i++ == 3) {
    f(i, 0, i);
} else {
    f(i, 1, i);
}
```

```
f(i, i++ == 3 ? 0 : 1, i);
```

same variable

- Hopefully at least one test case will fail!

TEST YOUR CHANGES! (2/3)

- Is finding the pattern

```
Pattern.statement("if ($cond) { $f($$before, $tValue, $$after); } else { $f($$before, $fValue, $$after); }")
```

  and replacing by

```
"bool condValue = $cond; $f($$before, condValue ? $tValue : $fValue, $$after);"
```

  correct?

- No – One reason

  Introduction of variable shadows

  variable with same name when present

```
if (i == 3) {
    g (condValue, 0);
} else {
    g (condValue, 1);
}
```

```
bool condValue = i == 3;
g (condValue, condValue ? 0 : 1);
```

- Compiler will warn for hiding names by shadowing variables

TEST YOUR CHANGES! (3/3)

- Both replacements are wrong for overloaded functions

```
if (cond) {
    f("hello");
} else {
    f(1);
}
```

```
f(cond ? "hello" : 1);
```

- Compiler will not warn when the relevant conversion functions exist!

FILTER (2/2)

- Find, filter, and replace
- Use filter to prevent incorrect changes from happening
- Library of standard filter functions
  - Side effect of placeholder?
  - Interaction between placeholders?
  - Hide variable?
  - Same definition?

```
function Has_Side_Effect
  (Match : Match_Pattern; Placeholder_Name : String) return Boolean;
   --  Has Execution of Expression a side effect?
   --  Side effects include:
   --     variables are changed, write to file, write to screen, ...

function Has_Effect_On
  (Match : Match_Pattern; Placeholder_A, Placeholder_B : String)
   return Boolean;
--  Does place_holder effect place_holder B?
--  Effects include
--        * output parameter of a function
--          used in the other placeholder
--        * side effect of a function (i.e. state change)
--          used in the other placeholder

function Are_Independent
  (Match : Match_Pattern; Placeholder_1, Placeholder_2 : String)
   return Boolean;
--  Are the placeholders independent?
--  In other words, can the order of execution of these placeholders
--  be changed without changing the behaviour of the program?
```

AGENDA

SUMMARY

- 040coders analyze and change software
- Existing tooling is limited
  - Tools are inappropriate for code, tools cannot be integrated, tools have steep learning curve
- Rejuvenation library overcomes limitations
  - Concrete Syntax Patterns

MAY THE PARSER BE WITH YOU!

ESI

Powered by industry,
academia and TNO