MySQL Dependency Extraction

Tabs vs. Spaces

Varsha Ragavendran
Nisha Sharma
Davood Anbarnam
Ayman Abualsunun
Anton Sitkovets
Glib Sitiugin
Kevin Arindaeng

Overview

- Introduction
- Compare dependency extraction techniques for the following:
  - Understand
  - srcML
  - Our Program
- Quantitative & Qualitative Analysis for the following:
  - Understand vs. srcML
  - srcML vs. Our Program
  - Understand vs. Our Program
- Potential Risks and Limitations
- Lessons Learned
- Conclusion

Introduction

- Dependency extraction techniques help to:
  - Analyze large open source code architecture to optimize software design
  - Easily identify dependencies within the code
  - Understand complex code bases with poor documentation

Dependency extraction using Understand

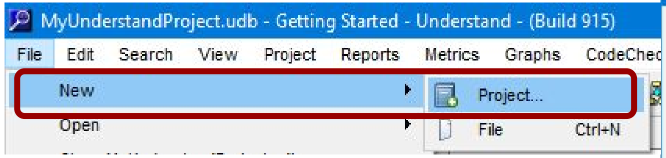
Understand is a static analysis tool focused on:

- source code comprehension, metrics, and standard testing.

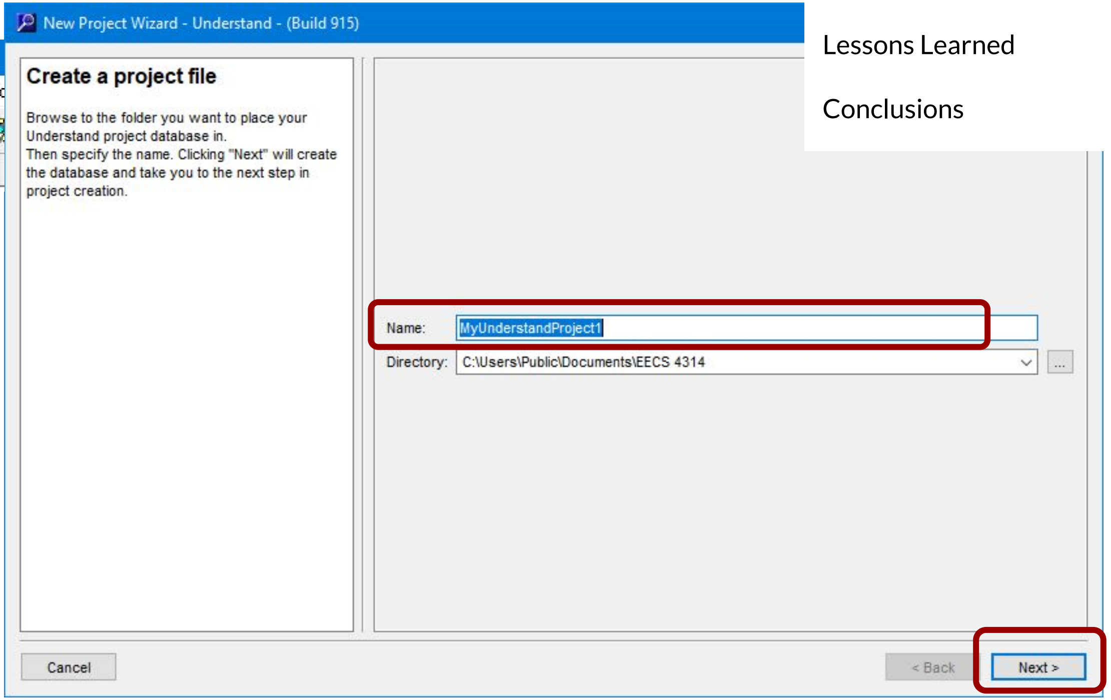
A file can depend on another through:

- Import
- Inheritance
- Implementation
- Method calls and object initializations
- @ Java annotations

Dependency extraction using Understand



Create a new project
And Name it



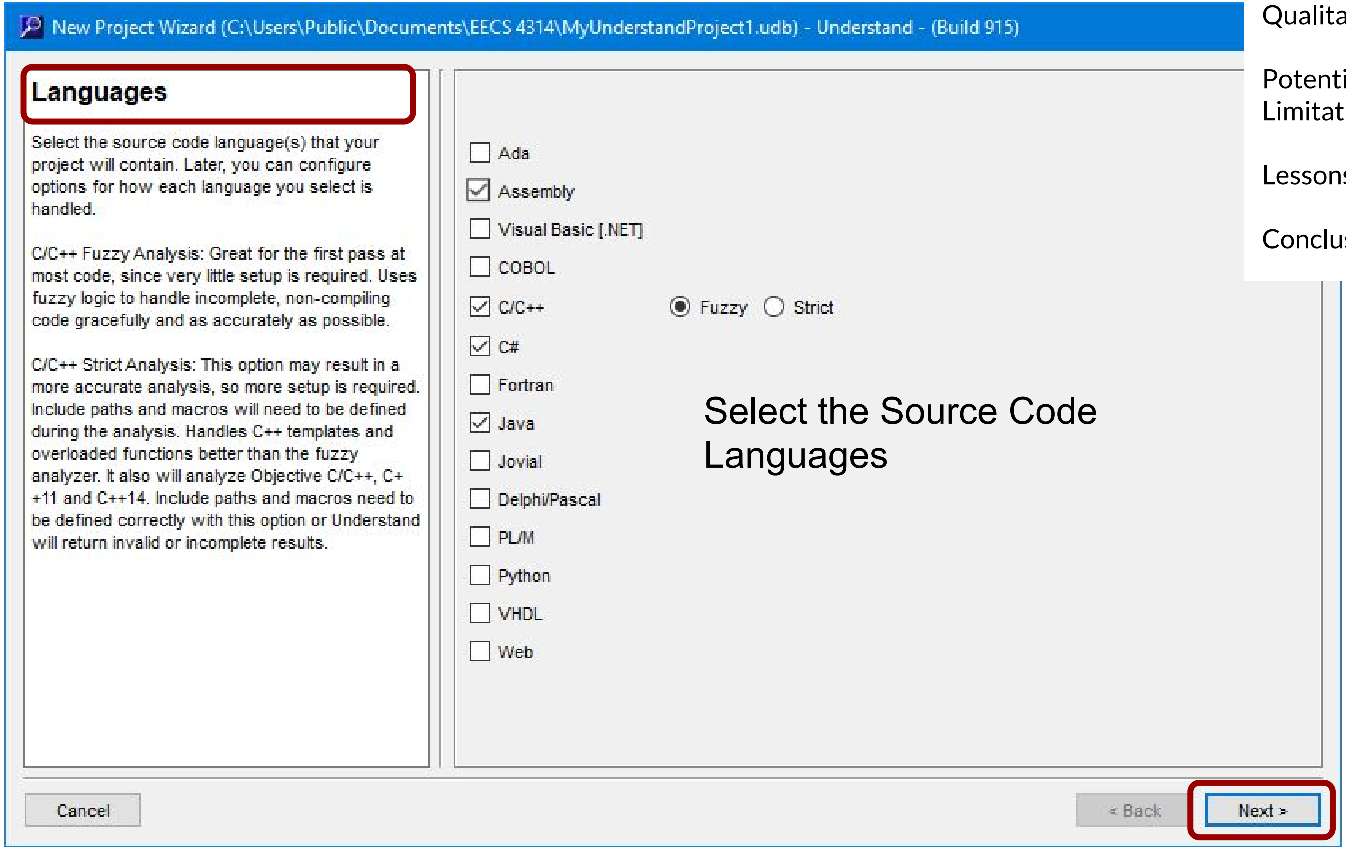
Create a project file

Browse to the folder you want to place your Understand project database in.
Then specify the name. Clicking "Next" will create the database and take you to the next step in project creation.

Name: MyUnderstandProject1

Directory: C:\Users\Public\Documents\EECS 4314

Dependency extraction using Understand



New Project Wizard (C:\Users\Public\Documents\EECS 4314\MyUnderstandProject1.udb) - Understand - (Build 915)

Languages

Select the source code language(s) that your project will contain. Later, you can configure options for how each language you select is handled.

C/C++ Fuzzy Analysis: Great for the first pass at most code, since very little setup is required. Uses fuzzy logic to handle incomplete, non-compiling code gracefully and as accurately as possible.

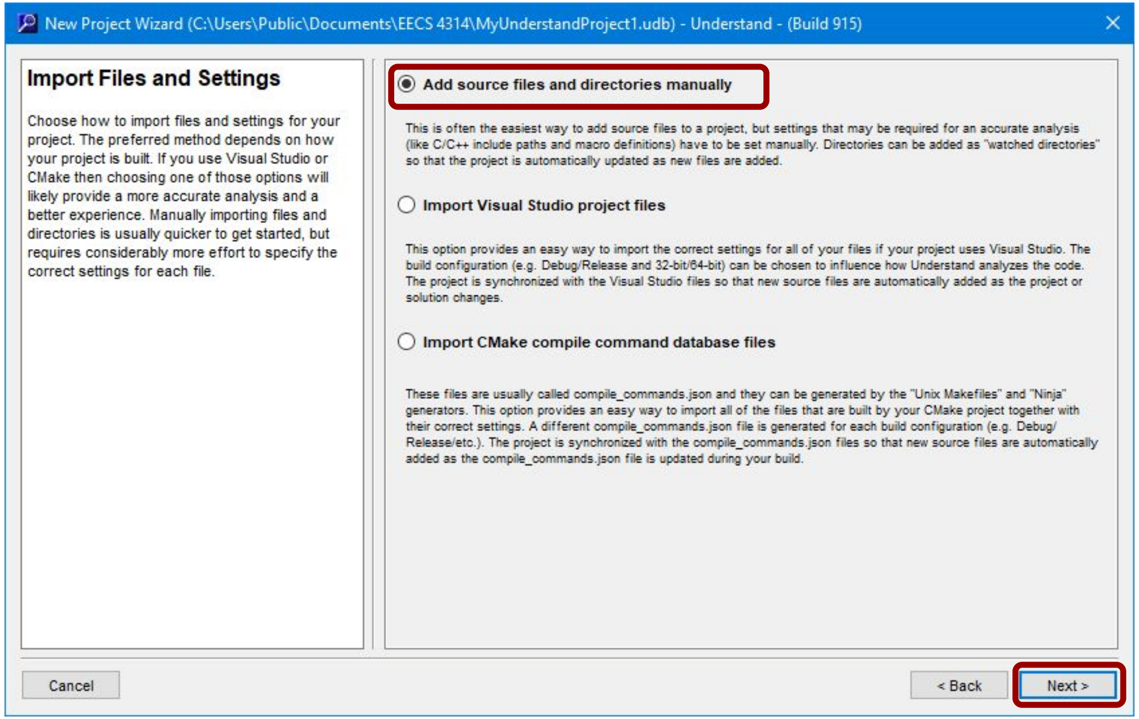
C/C++ Strict Analysis: This option may result in a more accurate analysis, so more setup is required. Include paths and macros will need to be defined during the analysis. Handles C++ templates and overloaded functions better than the fuzzy analyzer. It also will analyze Objective C/C++, C++11 and C++14. Include paths and macros need to be defined correctly with this option or Understand will return invalid or incomplete results.

- ☐ Ada
- ☑ Assembly
- ☐ Visual Basic [.NET]
- ☐ COBOL
- ☑ C/C++ — [✓ Fuzzy] Strict
- ☑ C#
- ☐ Fortran
- ☑ Java
- ☐ Jovial
- ☐ Delphi/Pascal
- ☐ PL/M
- ☐ Python
- ☐ VHDL
- ☐ Web

Select the Source Code Languages

Dependency extraction using Understand

Import Files


New Project Wizard (C:\Users\Public\Documents\EECS 4314\MyUnderstandProject1.udb) - Understand - (Build 915)

Import Files and Settings

Choose how to import files and settings for your project. The preferred method depends on how your project is built. If you use Visual Studio or CMake then choosing one of those options will likely provide a more accurate analysis and a better experience. Manually importing files and directories is usually quicker to get started, but requires considerably more effort to specify the correct settings for each file.

[✓ **Add source files and directories manually**]

This is often the easiest way to add source files to a project, but settings that may be required for an accurate analysis (like C/C++ include paths and macro definitions) have to be set manually. Directories can be added as "watched directories" so that the project is automatically updated as new files are added.

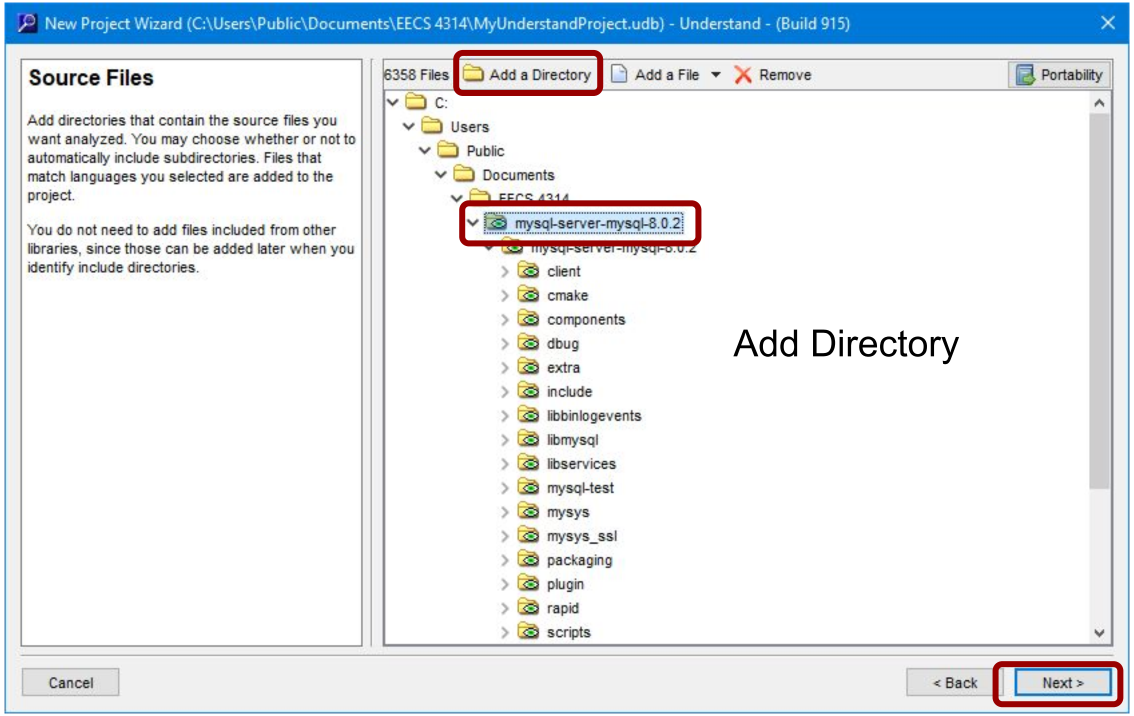
Import Visual Studio project files

This option provides an easy way to import the correct settings for all of your files if your project uses Visual Studio. The build configuration (e.g. Debug/Release and 32-bit/64-bit) can be chosen to influence how Understand analyzes the code. The project is synchronized with the Visual Studio files so that new source files are automatically added as the project or solution changes.

Import CMake compile command database files

These files are usually called compile_commands.json and they can be generated by the "Unix Makefiles" and "Ninja" generators. This option provides an easy way to import all of the files that are built by your CMake project together with their correct settings. A different compile_commands.json file is generated for each build configuration (e.g. Debug/ Release/etc.). The project is synchronized with the compile_commands.json files so that new source files are automatically added as the compile_commands.json file is updated during your build.

Dependency extraction using Understand

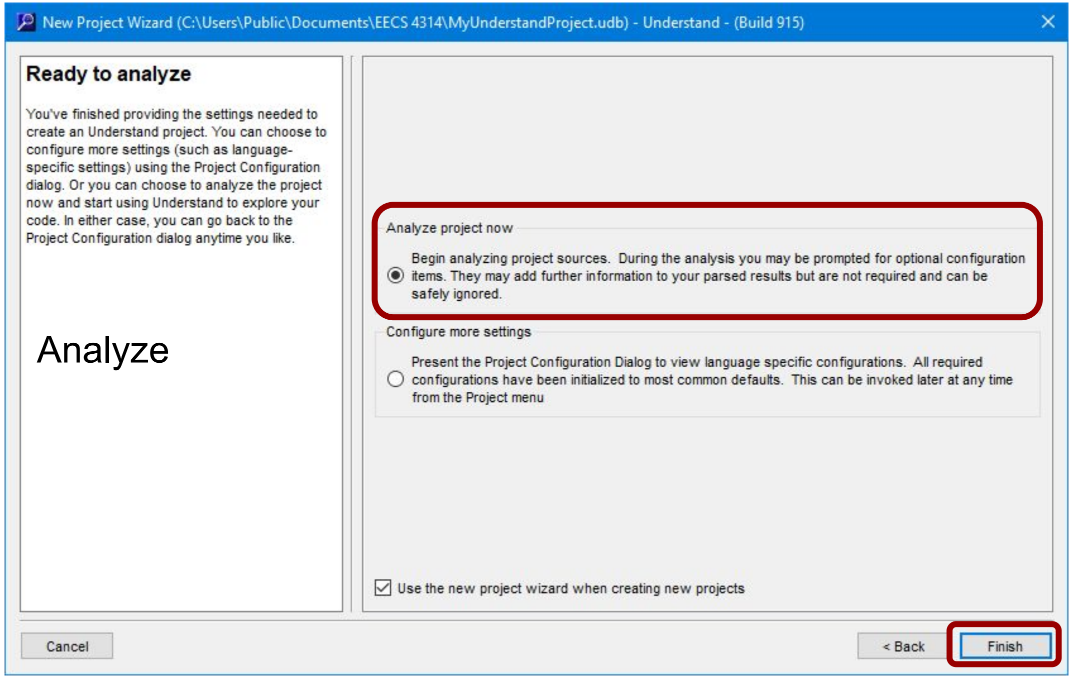

New Project Wizard (C:\Users\Public\Documents\EECS 4314\MyUnderstandProject.udb) - Understand - (Build 915)

Source Files

Add directories that contain the source files you want analyzed. You may choose whether or not to automatically include subdirectories. Files that match languages you selected are added to the project.

You do not need to add files included from other libraries, since those can be added later when you identify include directories.

Dependency extraction using Understand



New Project Wizard (C:\Users\Public\Documents\EECS 4314\MyUnderstandProject.udb) - Understand - (Build 915)

Ready to analyze

You've finished providing the settings needed to create an Understand project. You can choose to configure more settings (such as language-specific settings) using the Project Configuration dialog. Or you can choose to analyze the project now and start using Understand to explore your code. In either case, you can go back to the Project Configuration dialog anytime you like.

Analyze

Analyze project now

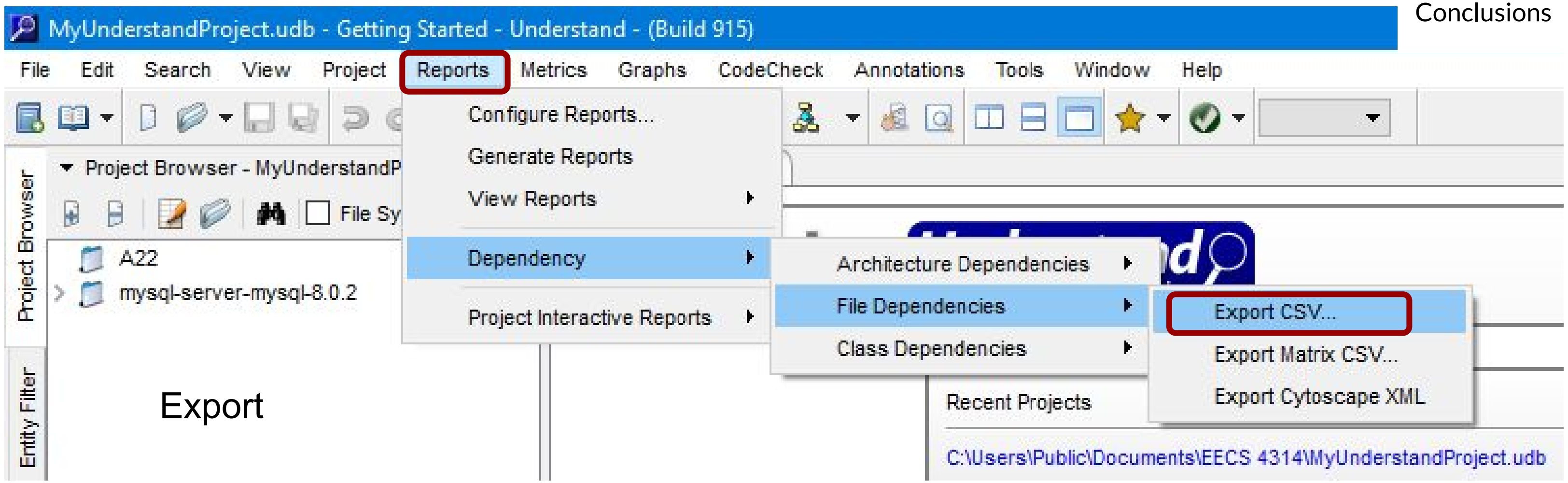
[✓] Begin analyzing project sources. During the analysis you may be prompted for optional configuration items. They may add further information to your parsed results but are not required and can be safely ignored.

Configure more settings

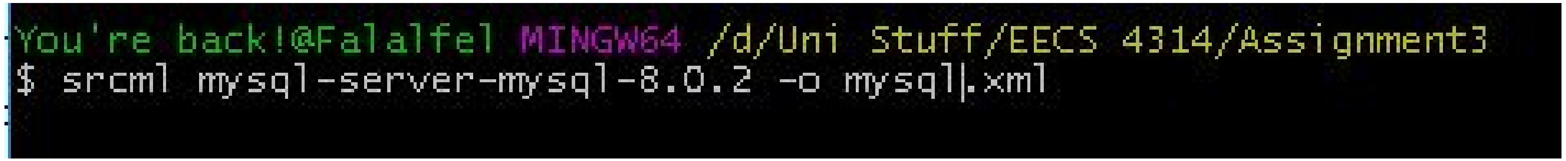
Present the Project Configuration Dialog to view language specific configurations. All required configurations have been initialized to most common defaults. This can be invoked later at any time from the Project menu

[✓] Use the new project wizard when creating new projects

Dependency extraction using Understand

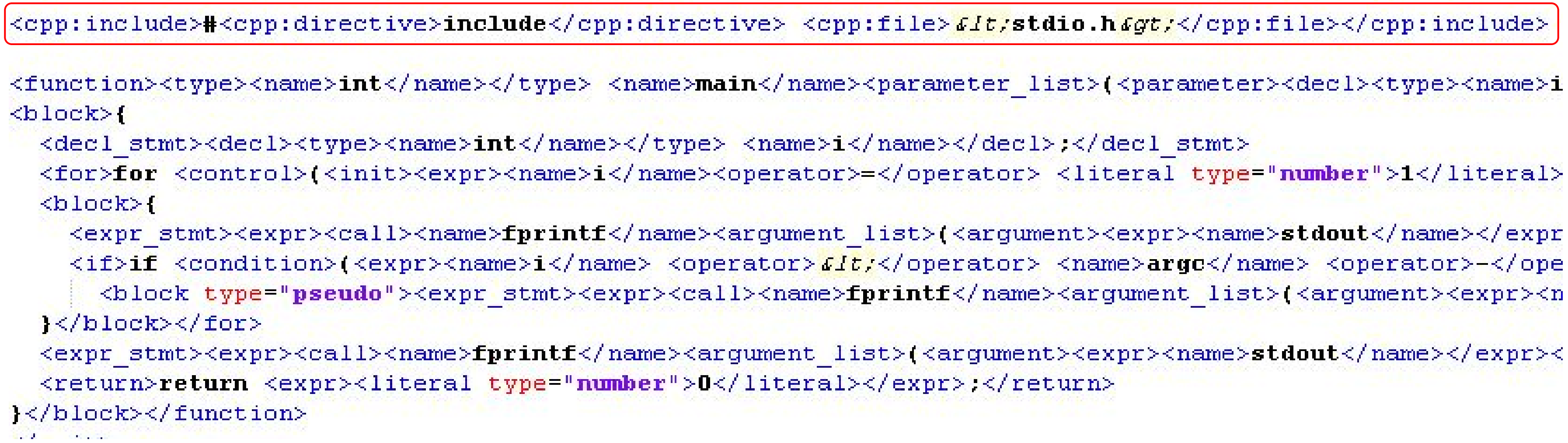


Dependency extraction using srcML



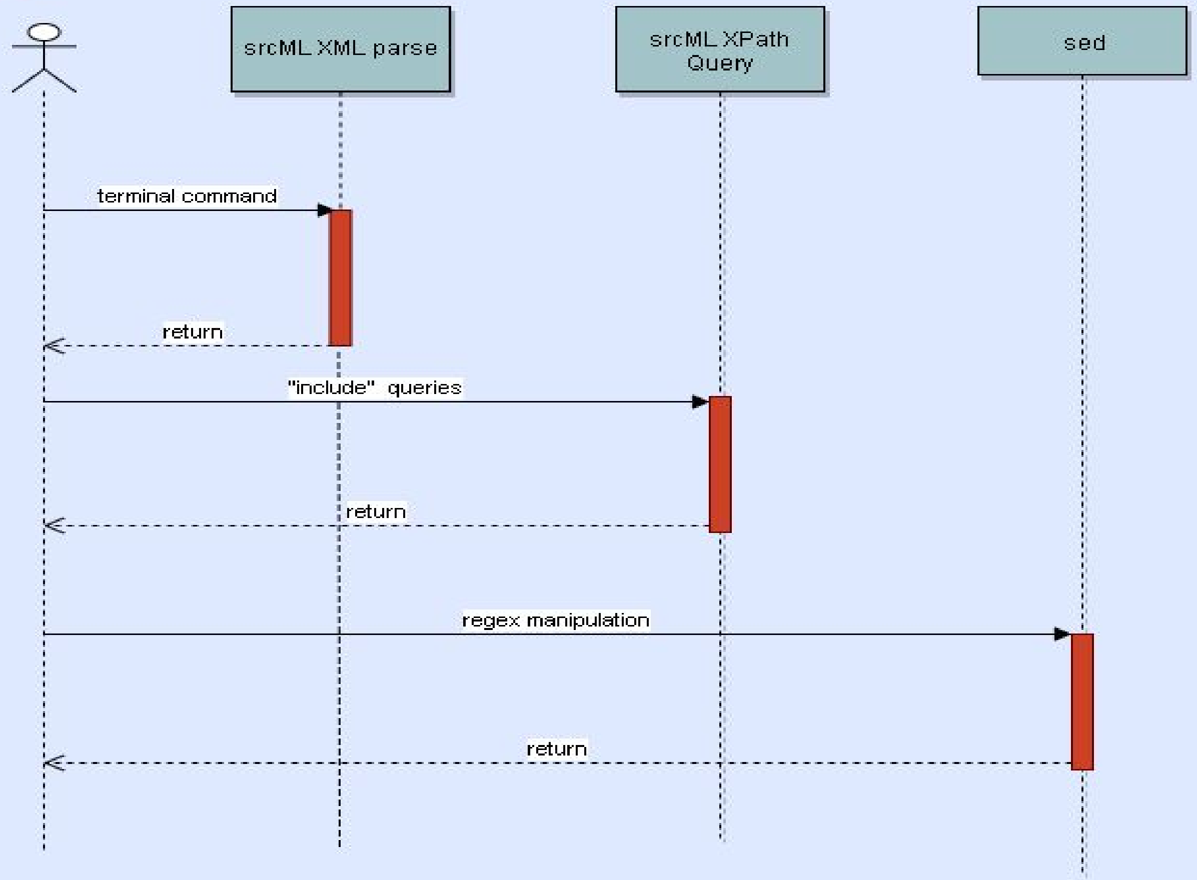
```
You're back!@Falalfel MINGW64 /d/Uni Stuff/EECS 4314/Assignment3
$ srcml mysql-server-mysql-8.0.2 -o mysql.xml
```

Dependency extraction using srcML



```
<cpp:include>#<cpp:directive>include</cpp:directive> <cpp:file>&lt;stdio.h&gt;</cpp:file></cpp:include>

<function><type><name>int</name></type> <name>main</name><parameter_list>(<parameter><decl><type><name>i
<block>{
  <decl_stmt><decl><type><name>int</name></type> <name>i</name></decl>;</decl_stmt>
  <for>for <control>(<init><expr><name>i</name><operator>=</operator> <literal type="number">1</literal>
  <block>{
    <expr_stmt><expr><call><name>fprintf</name><argument_list>(<argument><expr><name>stdout</name></expr
    <if>if <condition>(<expr><name>i</name> <operator>&lt;</operator> <name>argc</name> <operator>-</ope
      <block type="pseudo"><expr_stmt><expr><call><name>fprintf</name><argument_list>(<argument><expr><n
  }</block></for>
  <expr_stmt><expr><call><name>fprintf</name><argument_list>(<argument><expr><name>stdout</name></expr><
  <return>return <expr><literal type="number">0</literal></expr>;</return>
}</block></function>
```

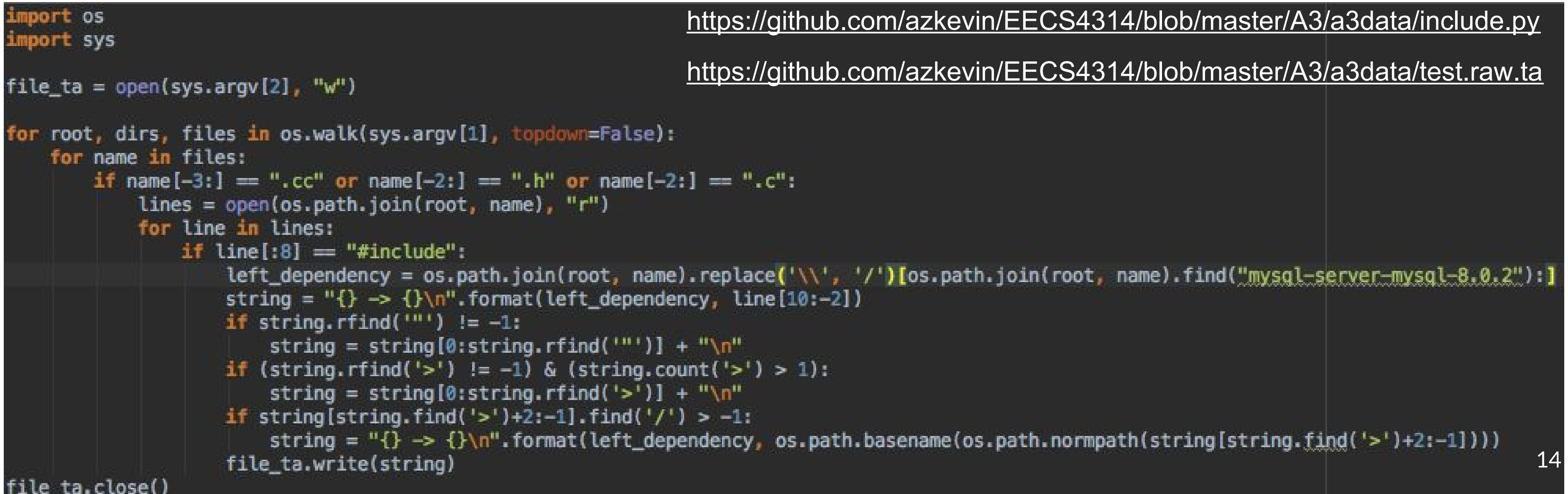
srcML Extraction



Dependency Extraction using Program

Wrote a Python script that walks through the MySQL source code folder.

Checks every “.cc”, “.c” and “.h” file. Parses each line of the file and writes the dependency between the current file and the files being included using “#include” to a raw.ta file.

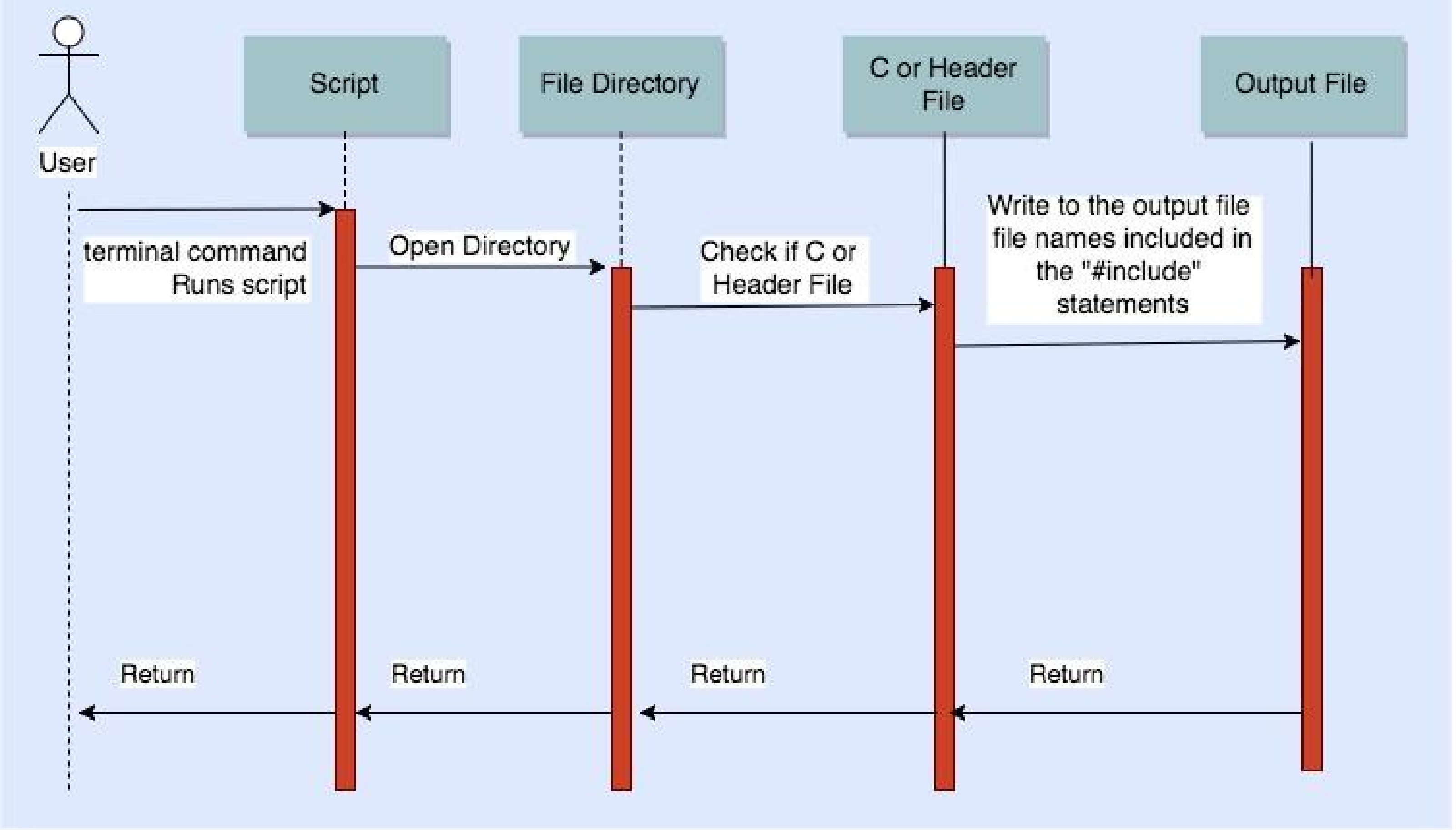


https://github.com/azkevin/EECS4314/blob/master/A3/a3data/include.py

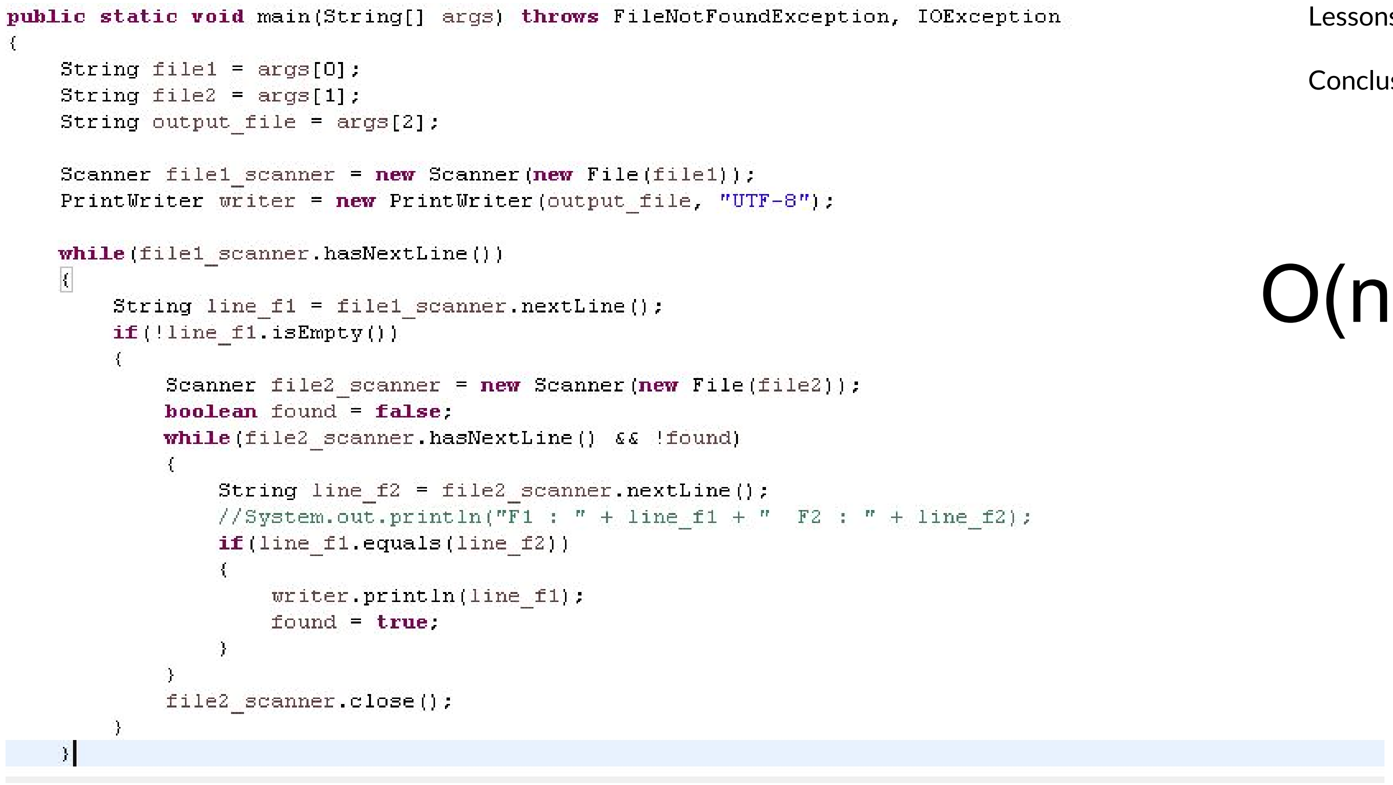
https://github.com/azkevin/EECS4314/blob/master/A3/a3data/test.raw.ta

```
import os
import sys

file_ta = open(sys.argv[2], "w")

for root, dirs, files in os.walk(sys.argv[1], topdown=False):
    for name in files:
        if name[-3:] == ".cc" or name[-2:] == ".h" or name[-2:] == ".c":
            lines = open(os.path.join(root, name), "r")
            for line in lines:
                if line[:8] == "#include":
                    left_dependency = os.path.join(root, name).replace('\\', '/')[os.path.join(root, name).find("mysql-server-mysql-8.0.2"):]
                    string = "{} -> {}\n".format(left_dependency, line[10:-2])
                    if string.rfind('"') != -1:
                        string = string[0:string.rfind('"')] + "\n"
                    if (string.rfind('>') != -1) & (string.count('>') > 1):
                        string = string[0:string.rfind('>')] + "\n"
                    if string[string.find('>')+2:-1].find('/') > -1:
                        string = "{} -> {}\n".format(left_dependency, os.path.basename(os.path.normpath(string[string.find('>')+2:-1])))
                    file_ta.write(string)
file_ta.close()
```

Find Dependencies using Include Directives in Source Code


User
Script
File Directory
C or Header File
Output File
terminal command Runs script
Open Directory
Check if C or Header File
Write to the output file file names included in the "#include" statements
Return
Return
Return
Return

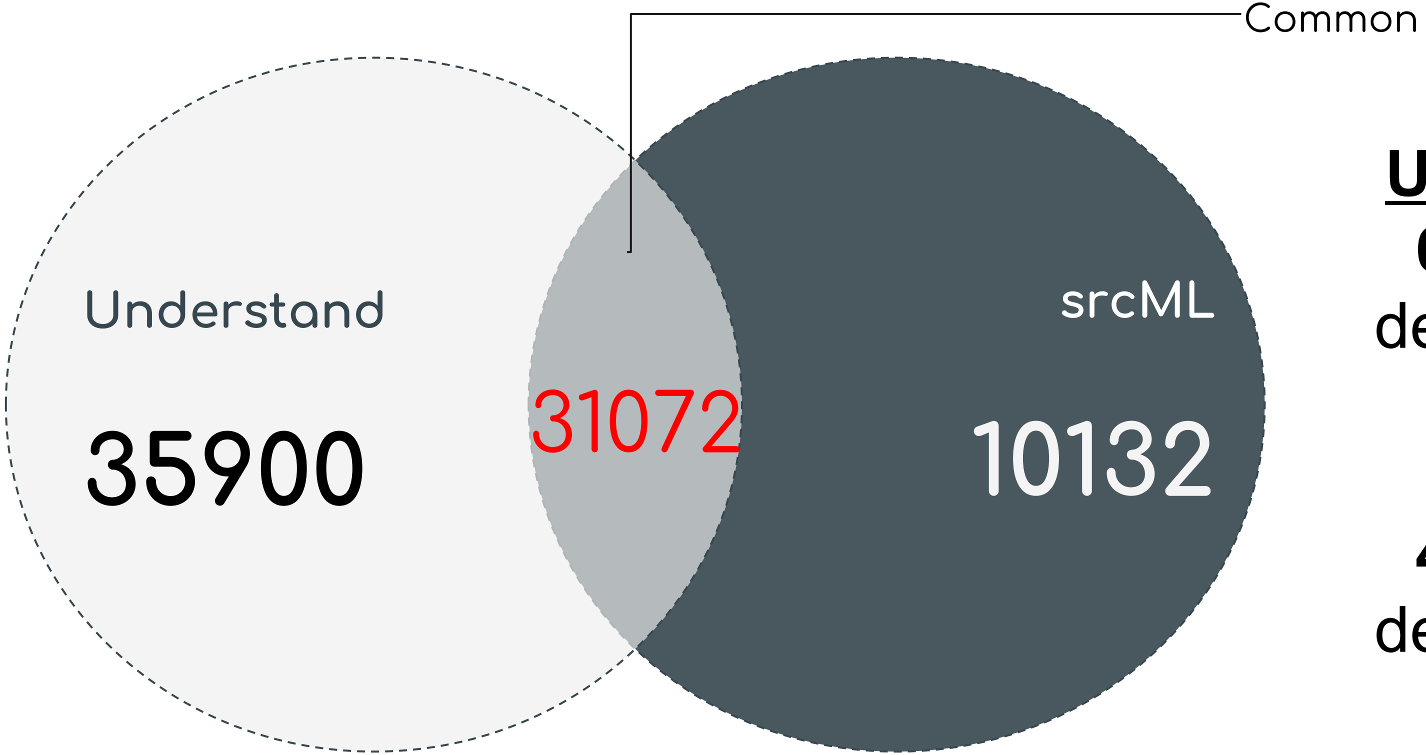
Finding Common Dependencies



```
public static void main(String[] args) throws FileNotFoundException, IOException
{
    String file1 = args[0];
    String file2 = args[1];
    String output_file = args[2];

    Scanner file1_scanner = new Scanner(new File(file1));
    PrintWriter writer = new PrintWriter(output_file, "UTF-8");

    while(file1_scanner.hasNextLine())
    {
        String line_f1 = file1_scanner.nextLine();
        if(!line_f1.isEmpty())
        {
            Scanner file2_scanner = new Scanner(new File(file2));
            boolean found = false;
            while(file2_scanner.hasNextLine() && !found)
            {
                String line_f2 = file2_scanner.nextLine();
                //System.out.println("F1 : " + line_f1 + "  F2 : " + line_f2);
                if(line_f1.equals(line_f2))
                {
                    writer.println(line_f1);
                    found = true;
                }
            }
            file2_scanner.close();
        }
    }
```

$O(n^2)$

Quantitative analysis: Understand vs srcML



Understand:
66972 total dependencies

srcML:
41204 total dependencies

Qualitative analysis: Understand vs srcML

Used sampling calculator with the following data:

1) Understand: Confidence level of 95%, Confidence interval of +/- 6.92%, total population: 76,474. Sampling size: 200

Using stratified sampling method:

a) Overlap: 31,072/76,474 * 200 ~ 81 cases.
b) Understand: 35,900/76,474 * 200 ~ 93 cases.
c) srcML: 10,132/76,474* 200 ~ 26 cases.

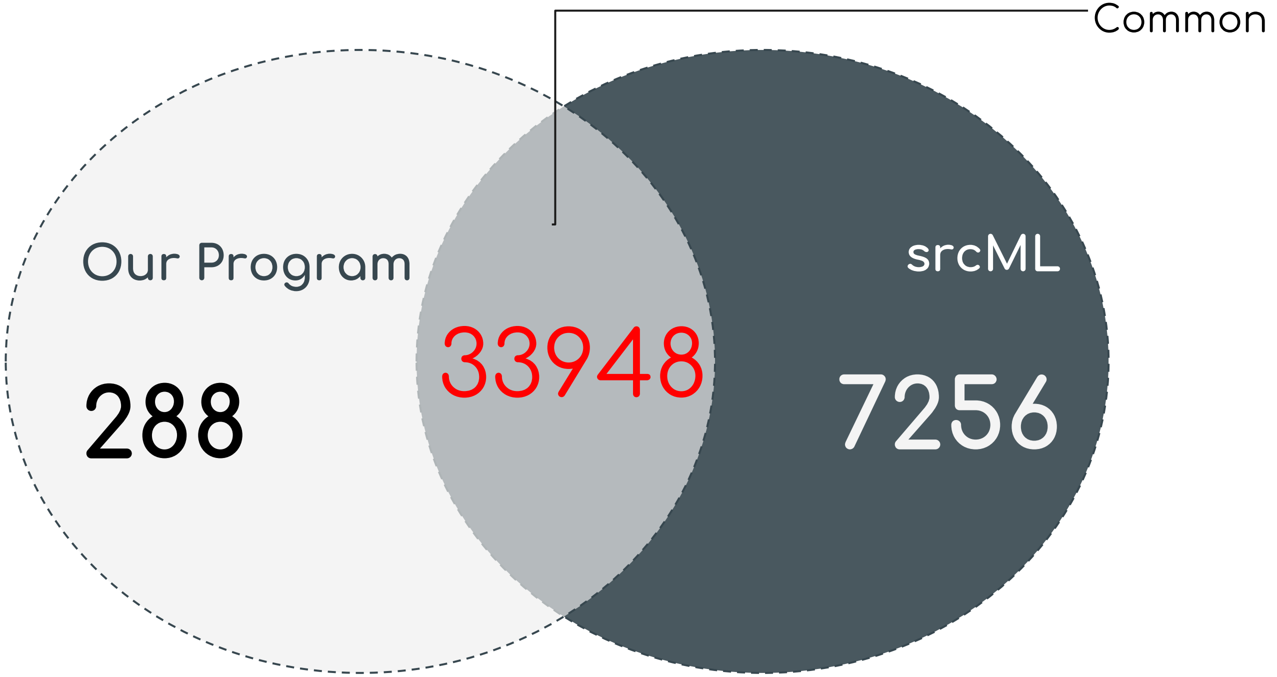
Noticeable differences: Understand vs srcML

The srcML tool only looks for the “Include” in the .c files, sometimes dependencies can be found in a different way such as inheritance.

Understand scans the whole source code directory, including all types of files such as .txt or .yy.

Quantitative Analysis: Our Program vs. srcML

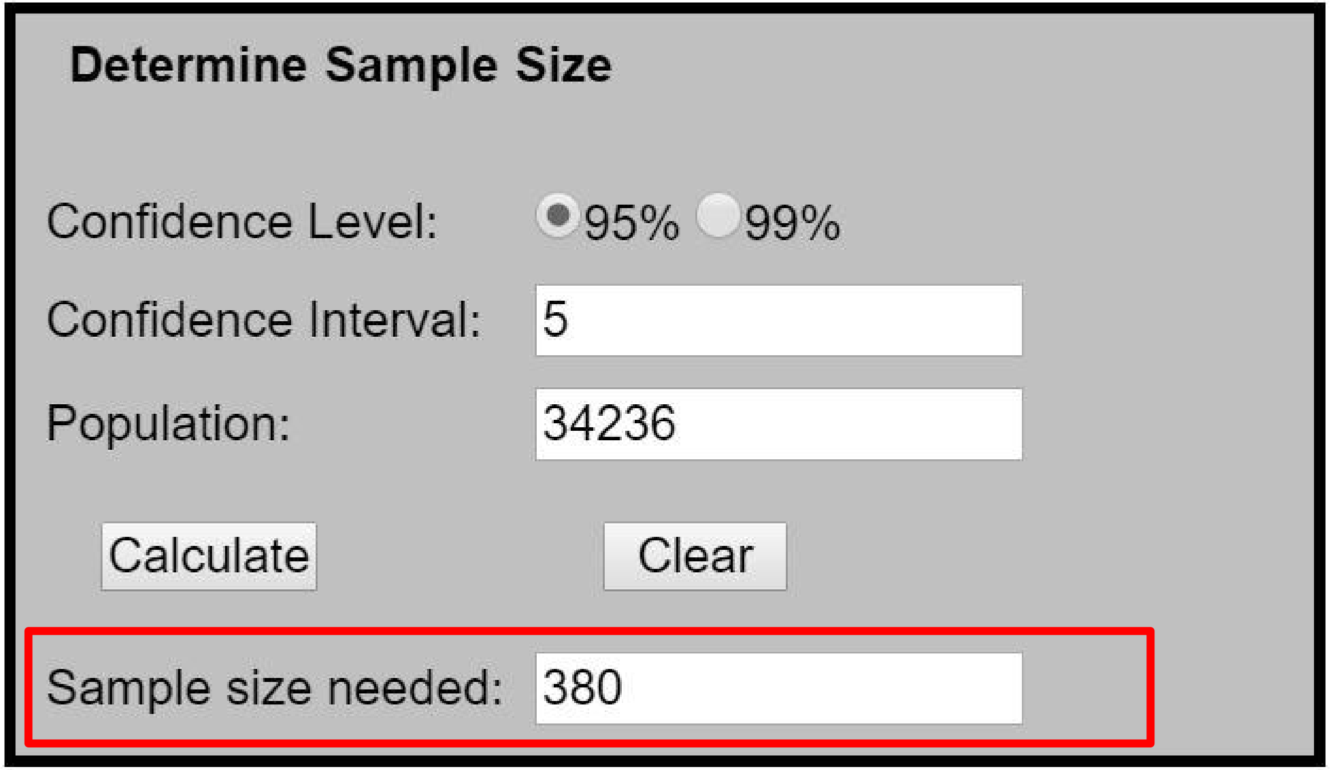
https://github.com/azkevin/EECS4314/blob/master/A3/a3data/srcML_Include_Common



Our Program:
34236 total dependencies

srcML:
41204 total dependencies

Qualitative Analysis: Our Program vs. srcML



Determine Sample Size

Confidence Level:	[✓ 95%] 99%
Confidence Interval:	5
Population:	34236

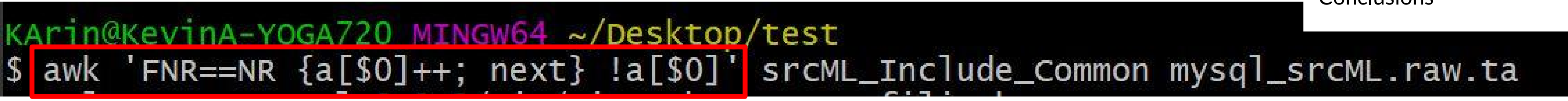
Sample size needed: 380

Qualitative Analysis: Our Program vs. srcML

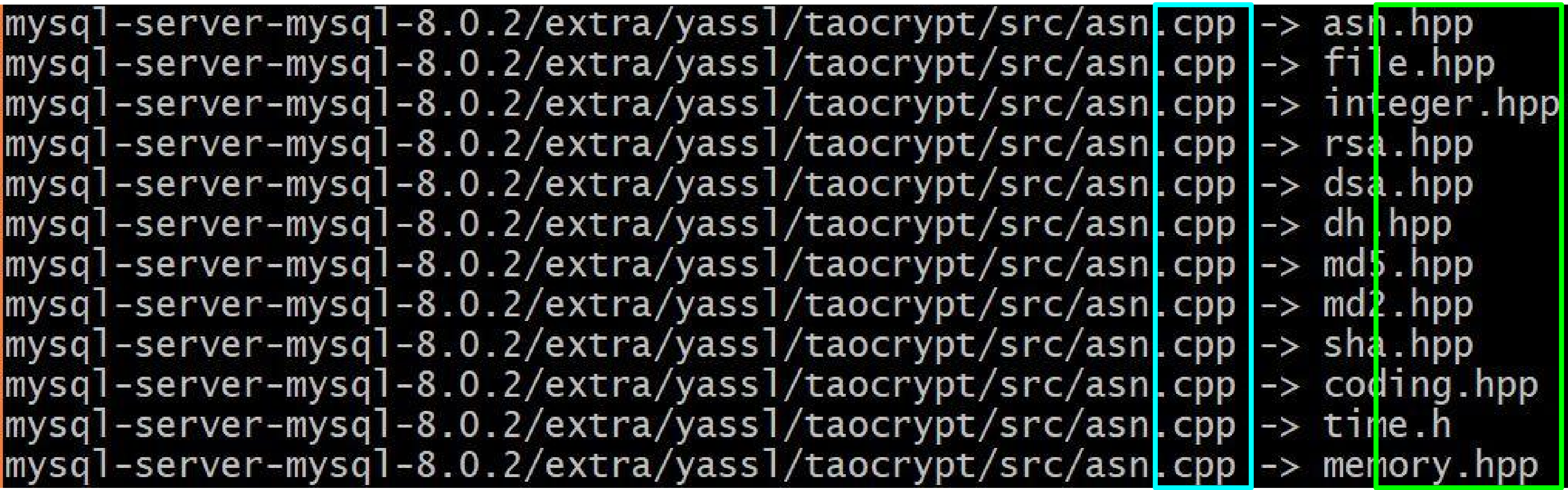
Using stratified sampling method:

Overlap: (33948/41492) * 380 = ~311 cases.	~81.8%
Our Program: (288/41492) * 380 = ~3 cases.	~0.79%
srcML: (7256/41492)* 380 = ~66 cases.	~17.4%

Noticeable Differences: Our Program vs. srcML

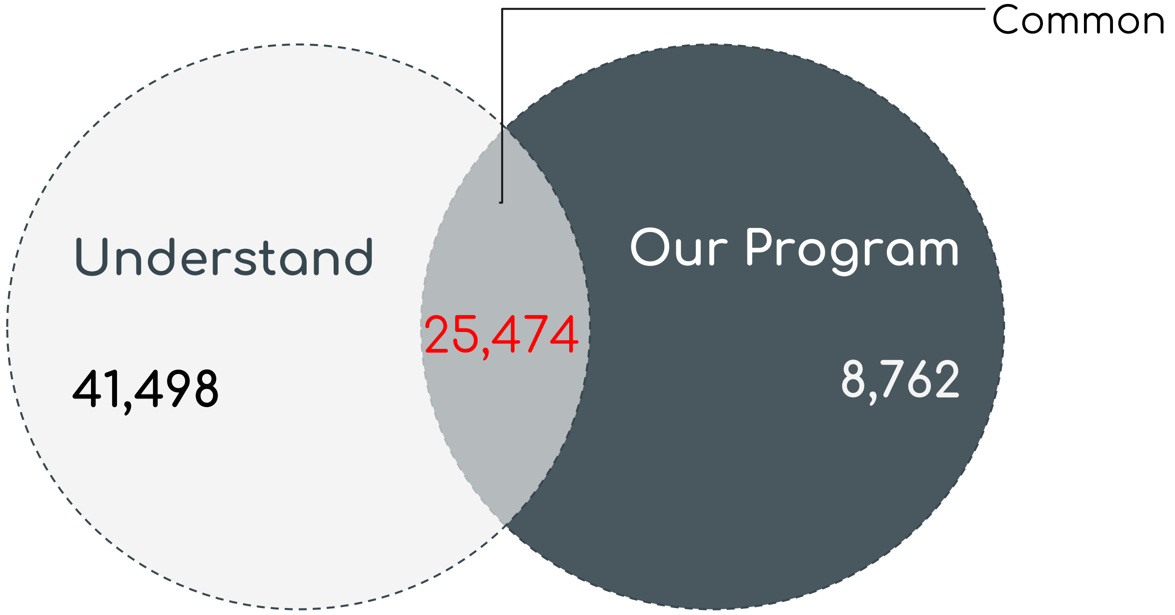


```
KArin@KevinA-YOGA720 MINGW64 ~/Desktop/test
$ awk 'FNR==NR {a[$0]++; next} !a[$0]' srcML_Include_Common mysql_srcML.raw.ta
```



```
mysql-server-mysql-8.0.2/extra/yassl/taocrypt/src/asn.cpp -> asn.hpp
mysql-server-mysql-8.0.2/extra/yassl/taocrypt/src/asn.cpp -> file.hpp
mysql-server-mysql-8.0.2/extra/yassl/taocrypt/src/asn.cpp -> integer.hpp
mysql-server-mysql-8.0.2/extra/yassl/taocrypt/src/asn.cpp -> rsa.hpp
mysql-server-mysql-8.0.2/extra/yassl/taocrypt/src/asn.cpp -> dsa.hpp
mysql-server-mysql-8.0.2/extra/yassl/taocrypt/src/asn.cpp -> dh.hpp
mysql-server-mysql-8.0.2/extra/yassl/taocrypt/src/asn.cpp -> md5.hpp
mysql-server-mysql-8.0.2/extra/yassl/taocrypt/src/asn.cpp -> md2.hpp
mysql-server-mysql-8.0.2/extra/yassl/taocrypt/src/asn.cpp -> sha.hpp
mysql-server-mysql-8.0.2/extra/yassl/taocrypt/src/asn.cpp -> coding.hpp
mysql-server-mysql-8.0.2/extra/yassl/taocrypt/src/asn.cpp -> time.h
mysql-server-mysql-8.0.2/extra/yassl/taocrypt/src/asn.cpp -> memory.hpp
```

Quantitative Analysis: Understand vs. Our Program



<u>Understand:</u>
66972 total dependencies

<u>Our Program:</u>
34236 total dependencies

Qualitative Analysis: Understand vs. Our Program

Used sampling calculator with the following data:

1) Understand: Confidence level of 95%, Confidence interval of +/- 6.92%, total population: 76,474. Sampling size: 200

Using stratified sampling method:

a) Overlap: 25,474/75,729 * 200 ~ 67 cases.
b) Understand: 41498/75,729 * 200 ~ 110 cases.
c) Our Program: 8762/75,729* 200 ~ 23 cases.

Noticeable Differences: Understand vs. Our Program

- Our Program does not detect .java, .cpp and .hpp files for dependency derivation
- This explains why Our program detects much less number of dependencies than Understand
- Our Program only looks for the “Include” in the .c files, sometimes dependencies can be found in a different way such as inheritance

Potential Risks and Limitations

1) srcML puts an extra XML tag before each keyword statement, we have to parse that with a java code to get the “include” statements.
2) srcML skips a significant number of dependencies since it only parse extensions that it supports.
3) Can’t get access to the source code of understand, comparing the tool as a black box is limited.
4) Can’t analyze the whole output, pick a sample.
5) Hidden dependencies that get generated after the build.
6) The Program we developed for method 3 skips certain extensions as well.
7) The program requires Python interpreter to run which is a limitations.

Lessons learned

Conclusion

- Many ways to extract dependencies
- Trade-offs associated with each technique
- Do the dependencies tell the real story?
  - Compare
  - Contrast
  - Conclude

Question Period