Dependency Extraction

pink fluffy unicorns

EECS 4314
Advanced Software Engineering

March 22, 2023

OUR TEAM

Benedict Miguel
Jiachen Wang
Keshav Gainda
Ketan Bhandari
Mohaimen Hassan

Pegah Fallah
Shami Sharma
Sidharth Sudarsan
Suryam Thaker
Xiaochuan Wang

AGENDA

01 **Dependency Extraction Techniques**

Explanation of techniques and diagrams

02 **Quantitative Comparison**

Process and results

03 **Qualitative Comparison**

Process and results

04 **Rationales behind the differences**

05 **Risks and Limitations**

06 **Lessons Learned**

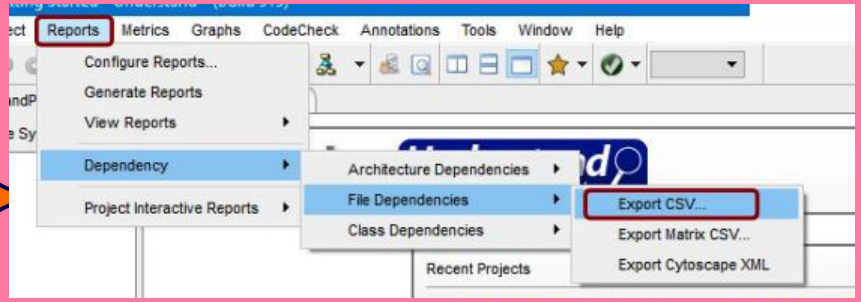
Dependency Extraction

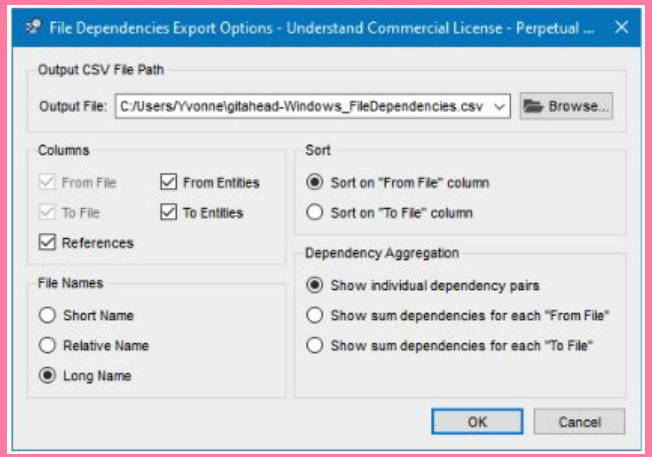
> Understand

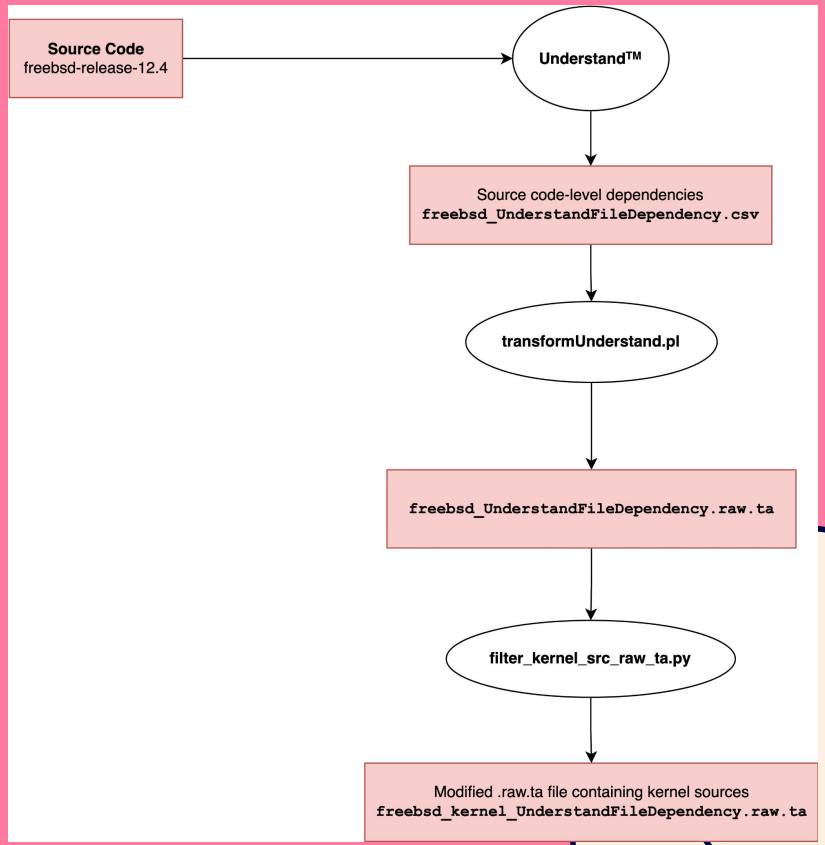
- Static Code Analysis Tool
- Source code-level dependencies:
  - Extracts functions calls and variable access relations (e.g. function y calls function x)
  - Control flow and data flow dependencies
- Dependency through:
  - import/include
  - Inheritance
  - Implementation
  - Method calls/Object initialization
  - Annotations

Dependency Extraction

> Understand

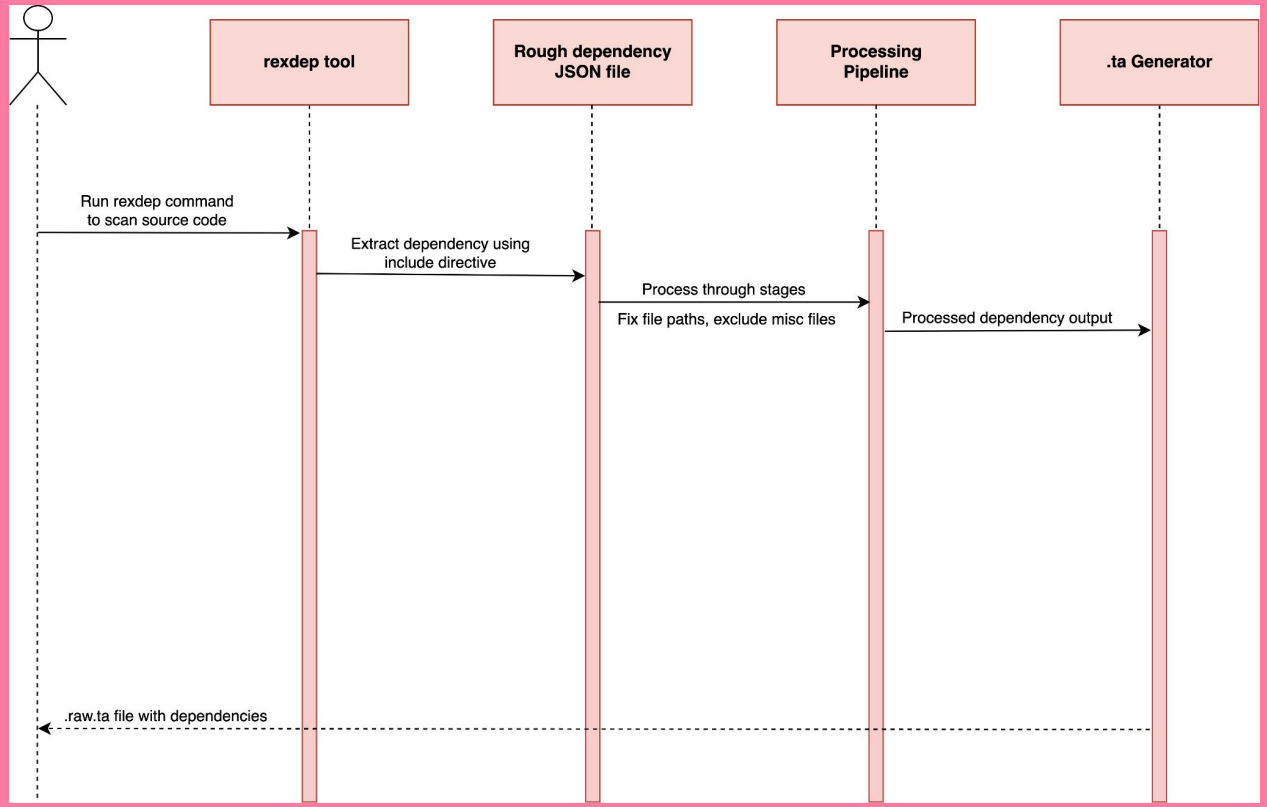






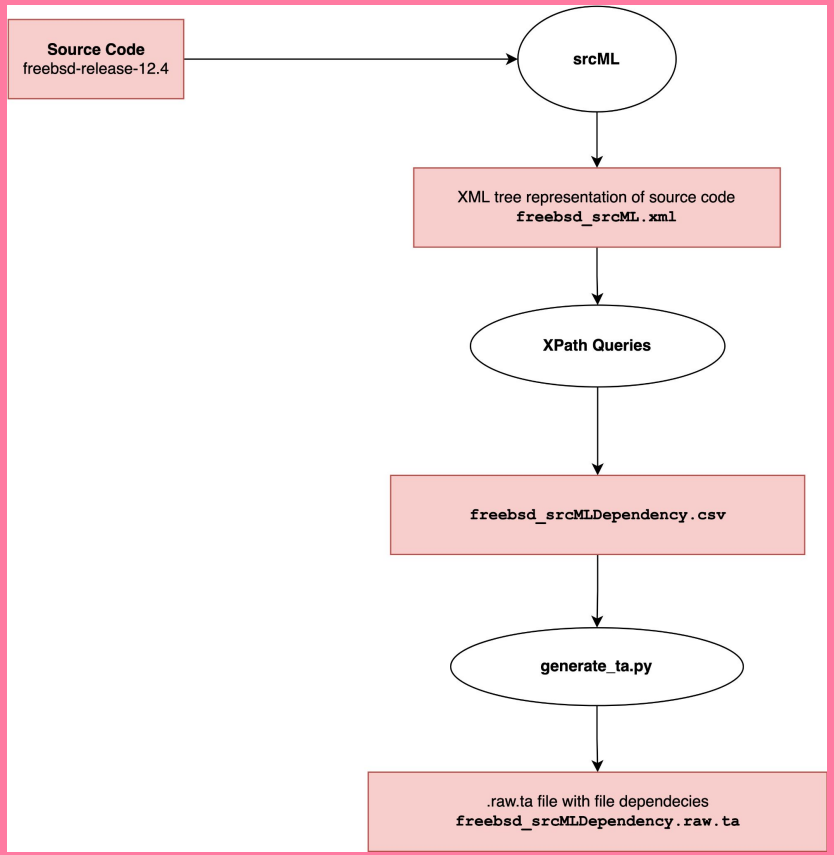
Dependency Extraction

> #include directive



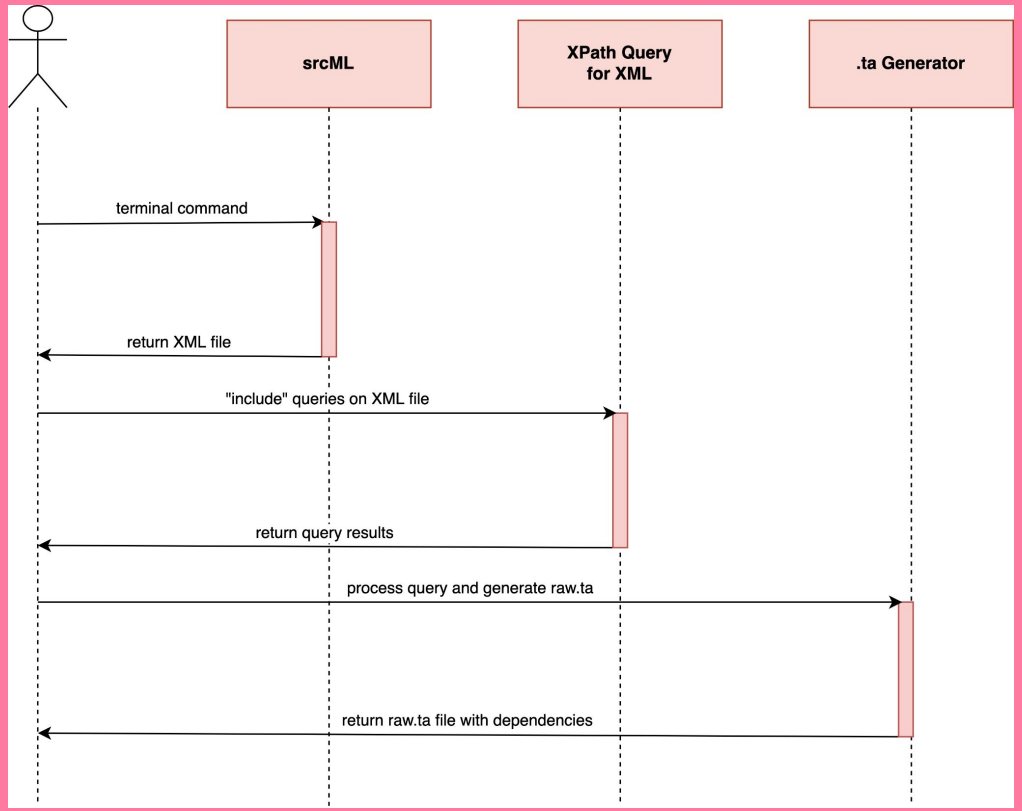
Dependency Extraction

> srcML



Dependency Extraction

> srcML



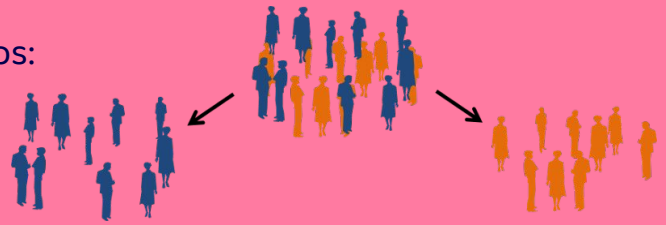
COMPARISON PROCESS

- **Measurement**: Directly count the similarity and uniqueness for each pair of methods and then print the result
  - **Instances** & **Dependencies**
    - **Common** in both
    - **Unique** in method 1
    - **Unique** in method 2

```
Common instances: 12347
Unique instances in file 1: 221
Unique instances in file 2: 81
Common dependencies:76093
Unique dependencies in file 1:22834
Unique dependencies in file 2:25129
```

COMPARISON PROCESS

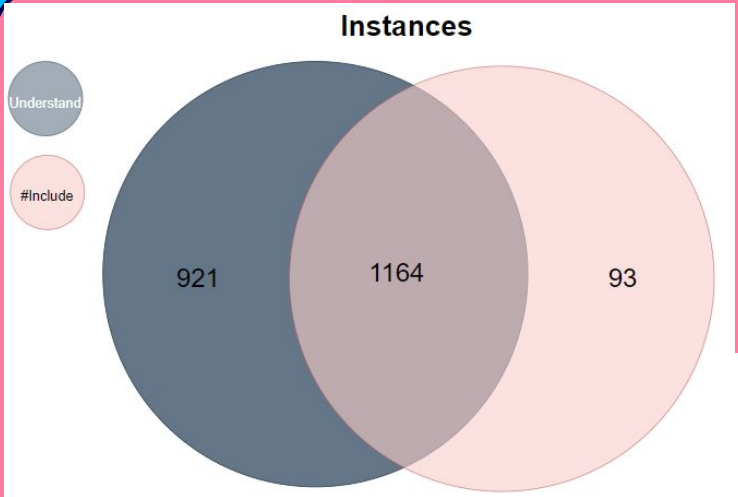
- **Stratification:** Sampling Method
  - Process of dividing members of the population into homogeneous subgroups before sampling
  - We have our data divided into three subgroups:
    - Common
    - Unique to A
    - Unique to B

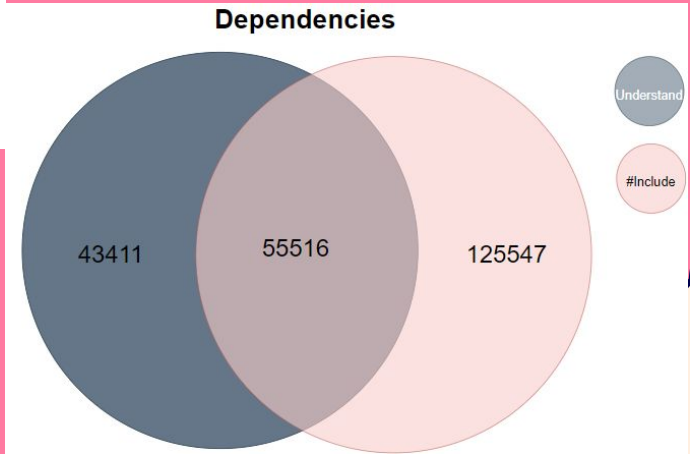


- **Creative Research Systems:** Sampling Calculator
  - Confidence Level - 95%
  - Confidence Interval - 5%

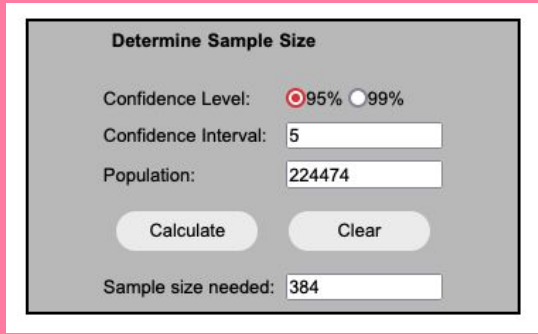
Creative Research Systems
Your Complete Survey Software Solution Since 1982

Understand vs #include





Understand vs #include



Total Population: **43,411 + 125,547 + 55,516 = 224,474**

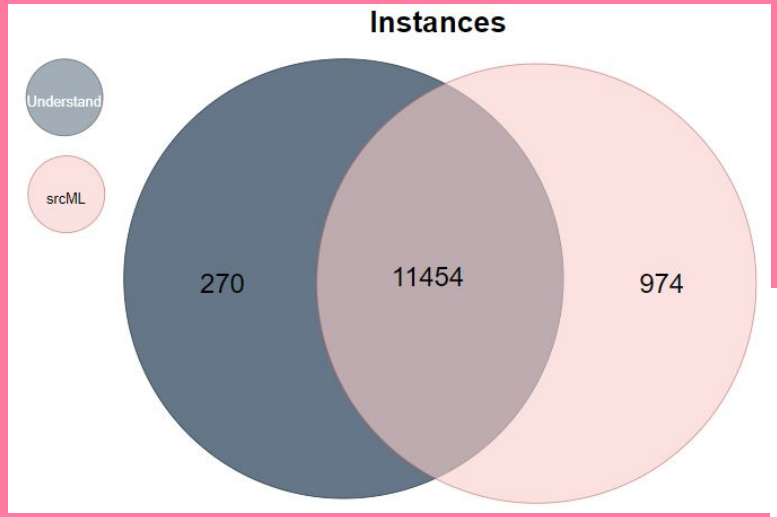
Sample size needed: **384**

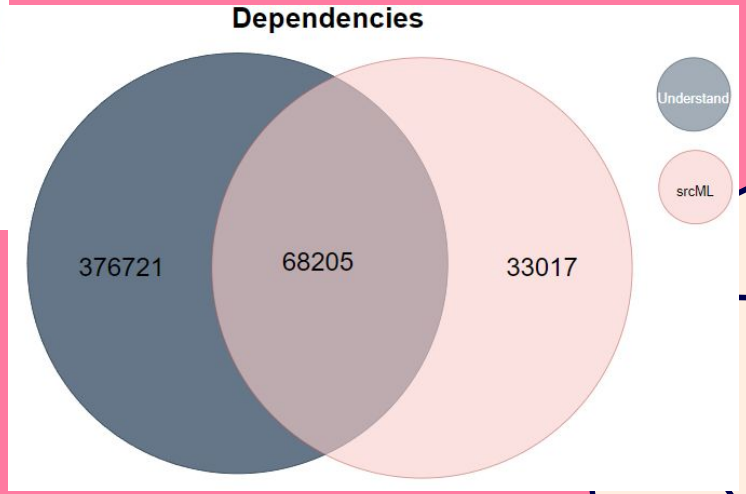
Subgroup	**Equivalent Sample Size** (Subpopulation / Total Population)	**Percentage of Total Sample Size**
Common Dependencies	(55,516/224,474) * 384 = 95 cases	(95/384) * 100 = ~24.7%
Unique to Understand	(43,411/224,474) * 384 = 74 cases	(74/384) * 100 = ~19.3%
Unique to #include	(125,547/224,474) * 384 = 215 cases	(215/384) * 100 = ~56%

Understand vs #include

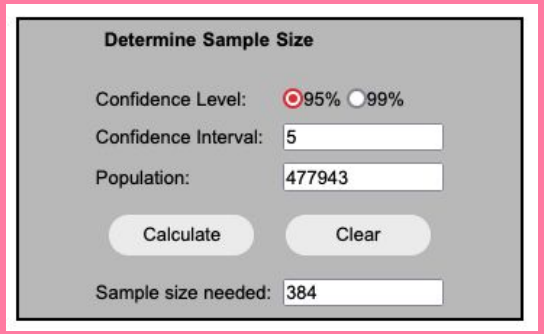
- **Granularity of Scanning**
  - Understand: Comprehensively scans and analyzes source code at a lower level
  - include: Derive dependencies purely based on *#include* directives
- **File types**
  - Understand: includes ***.c, * .h** and ***.m** files
  - include: includes all extensions mentioned in *#include :* ***.c, *.h, *.m, *.S, *.inc, *.dts**
- **Architecture dependent code**
  - Understand: Accurately extracts relations for architecture-dependent code
  - include: *#include* uses an expression such as *"machine/../xxx"* and not the exact names of the directories.

Understand vs srcML





Understand vs srcML



Total Population: **376,721 + 68,205 + 33,017 = 477,943**

Sample size needed: **384**

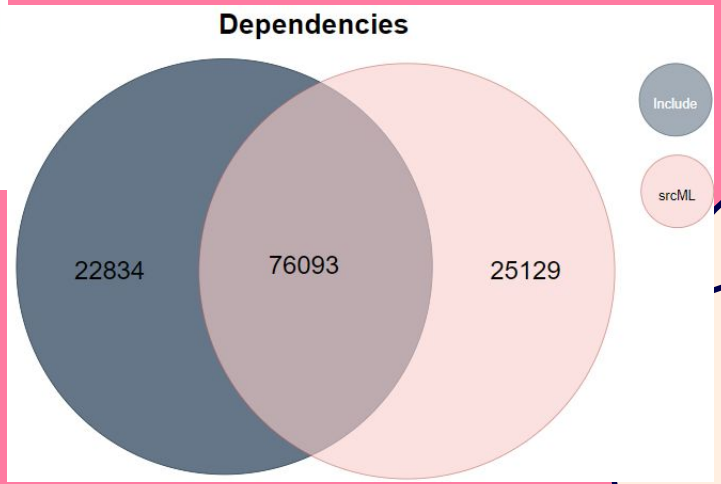
Subgroup	**Equivalent Sample Size** (Subpopulation / Total Population)	**Percentage of Total Sample Size**
Common Dependencies	(68,205/477,943) * 384 = 55 cases	(55/384) * 100 = ~14.3%
Unique to Understand	(376,721/477,943) * 384 = 303 cases	(303/384) * 100 = ~78.8%
Unique to srcML	(33,017/477,943) * 384 = 27 cases	(215/384) * 100 = ~7%

Understand vs srcML

- **Granularity of Scanning**
  - Understand: Comprehensively scans and analyzes source code at a lower level
  - srcML: Derive dependencies purely based on *#include* directives
- **File types**
  - Understand: includes ***.c, * .h** and ***.m** files
  - srcML: includes all extensions mentioned in *#include :* ***.c, *.h, *.m, *.S, *.inc, *.dts**
- **Architecture dependent code**
  - Understand: Accurately extracts relations for architecture-dependent code
  - srcML: *#include* uses an expression such as *"machine/../xxx"* and not the exact names of the directories.

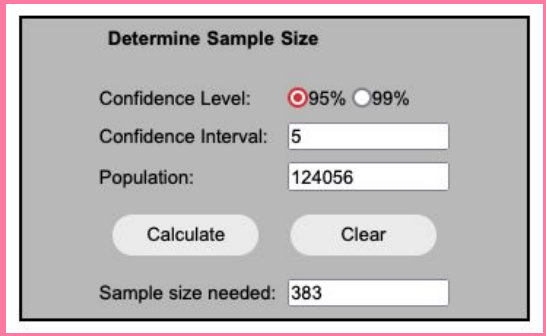
Quantitative Analysis

#include vs srcML



Qualitative Analysis

#include vs srcML



Total Population: **22,834 + 76,093 + 25,129 = 124,056**

Sample size needed: **383**

Subgroup	**Equivalent Sample Size** (Subpopulation / Total Population)	**Percentage of Total Sample Size**
Common Dependencies	(76,093/124,056) * 383 = 236 cases	(236/383) * 100 = ~61.3%
Unique to #include	(22,834/124,056) * 383 = 71 cases	(71/383) * 100 = ~18.4%
Unique to srcML	(25,129/124,056) * 383 = 78 cases	(78/383) * 100 = ~20.3%

#include vs srcML

- Similar techniques: using #include directives as the source of truth for dependencies
- #include solution performed post-processing to fix inconsistencies
- Differences in XML-based querying (XPath) vs Unix-utility based querying (sed, awk, grep)

Limitations and Risks

Understand :
Language-based limitation, understand can only support the C++ java and python, some new language like go can not be analysed by it.
Since we don't know the algorithm that understand uses, there might be accuracy issues which are tough to debug.
We can not analyse the whole system.

#include:
Language-based limitations
Will miss a lot of dependency due to miss the dependencies that have deeper dependencies

SrcML:
Language-based limitations
Generation and querying is time consuming.

Lessons Learned

- The differences between the extraction techniques differ file types of their outputs which in effect their processing time
  - Understand outputs a .csv file. Using Pandas or Excel, manipulating these values are very easy
  - The Include initiative uses .json files, which is relatively fast to process.
  - scrML by-far takes the longest because of XML file type. Using xpath to process large XML files increases the processing time
- There are also differences in the content of their output
  - srcML
    - Can't create cLinks,

      "cLinks file1.c **file2.c**

      because the source code-XML can't represent to connect of **file2.c**
- In ranking the number of dependencies each of technique uses, the ranking is as follows
  - Understand (15.9 mb)
  - Include (11.6 mb)
  - srcML (8.9 mb)

References

1. https://cgit.freebsd.org/src/tree
2. https://docs.freebsd.org/en/books/arch-handbook/book/
3. *Marshall Kirk McKusick, George V. Neville-Neil, Robert N.M. Watson (2015)* The Design and Implementation of the FreeBSD® Operating System, 2 e.d.
4. *William Joy, Robert Fabry, Samuel Leffler, M. Kirk McKusick, Michael Karels,* Berkeley Software Architecture Manual 4.4BSD Edition https://docs.freebsd.org/44doc/psd/05.sysman/paper.html
5. *David Garlan, Mary Shaw* (1994) An Introduction to Software Architecture