Flink Dependency Extraction

Team Debeggars

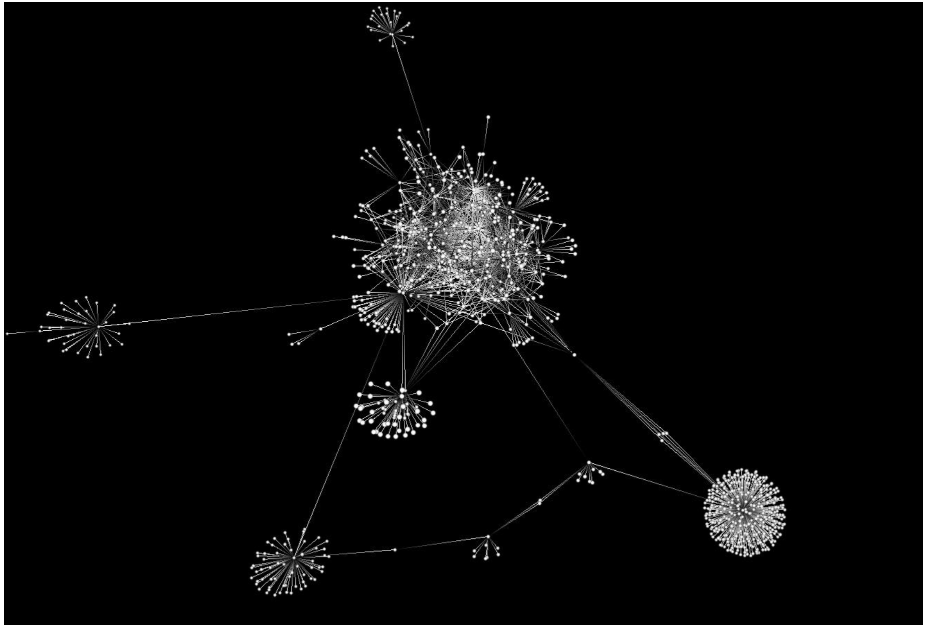
Overview

- Extraction Techniques: Understand, JDEPS, srcML
- Quantitative Comparison Process and Results
- Difference Analysis
- Limitations
- Learned Lessons

Alternatives

POM-Parsings

Jarviz (ASM opcode analysis)



jarviz

Jarviz

Method 1: Understand

- Dependency calculation:
- Finding all the entities (File, Class or Architecture level).
- Getting references for each entity.
- Retrieve the group for the referenced entity.
- Establish dependency based on the reference.
- Done through Language-specific Parser Analysis.

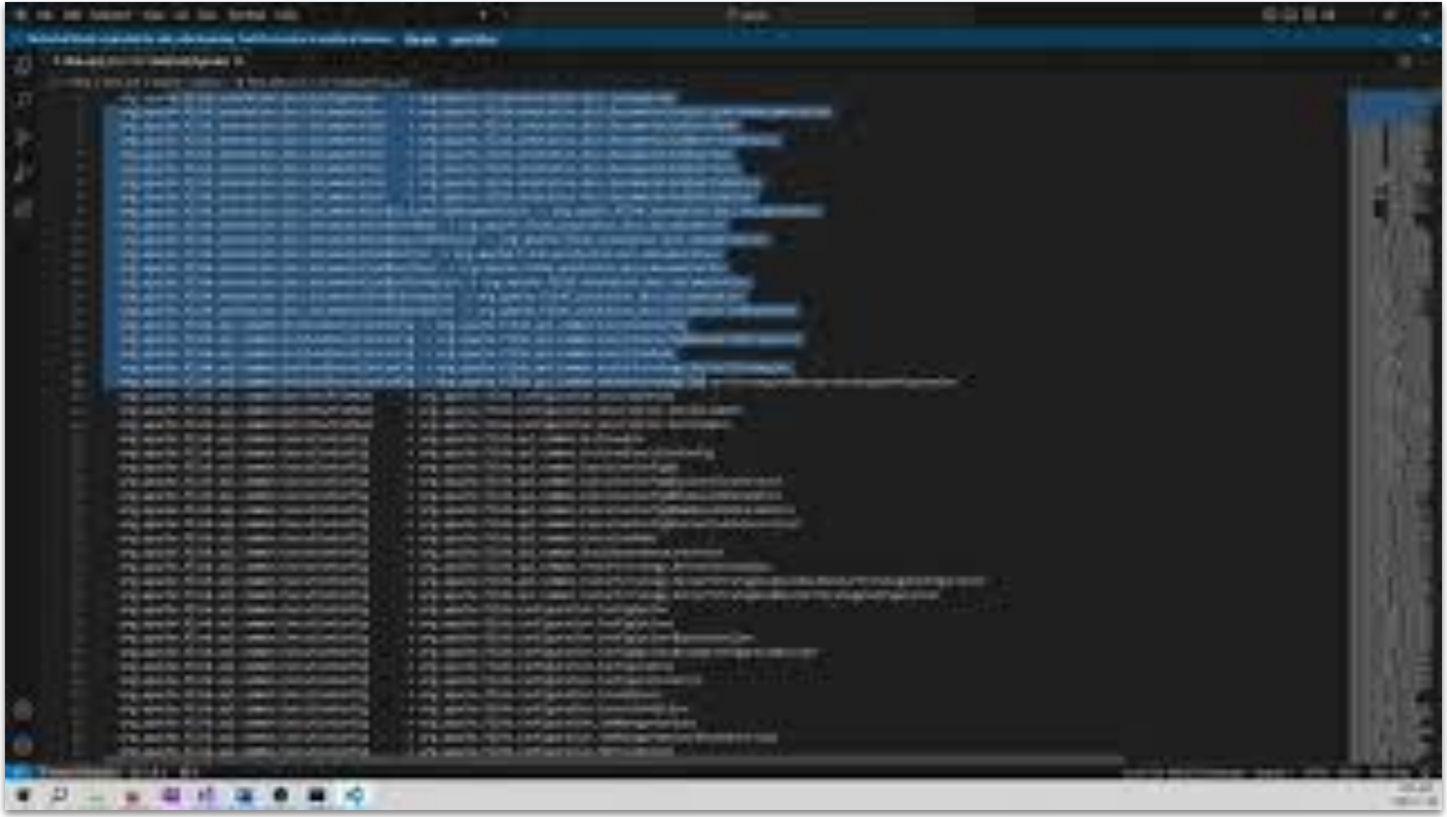
Method 2: jdeps

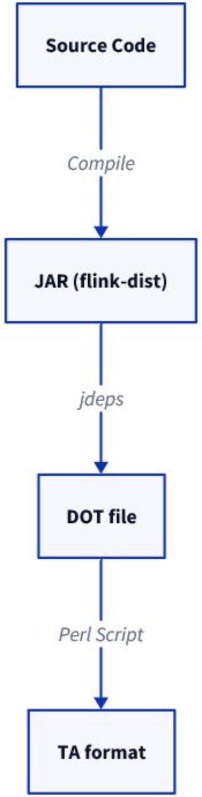
- Command-line tool, dependency analysis.
- Processes bytecodes (JAR files).
- Compiles statically declared dependencies.
- Can produce outputs in various formats including: txt, dot, tgf etc.

Pros: Easy to use, Accessible documentation, Fast.

Cons: Minimal functionalities.

Extraction Process (jdeps)





Source Code

Compile

JAR (flink-dist)

jdeps

DOT file

Perl Script

TA format

Method 3 - srcML

- Converts source code to XML
- Compatible with C, C++, C#, and Java
- Bidirectional transformation - can convert XML output back to source code

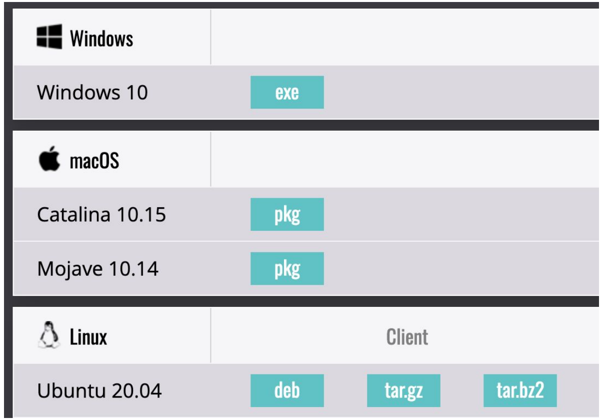
srcML pros/cons

Pros

- Documentation exists
- Standard output format (XML)
- Easy to use (once you deal with old dependency issues)

Cons

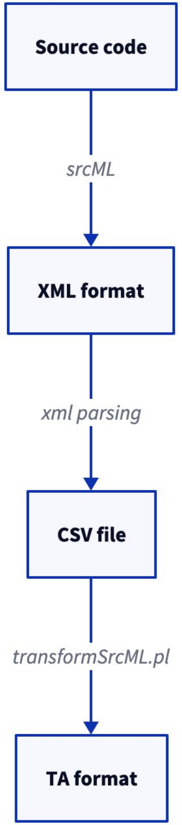
- Doesn’t seem to be actively maintained
  - Last commit in 2021
  - Downloads are all for old OS versions
- The output by itself isn’t very helpful for analysis



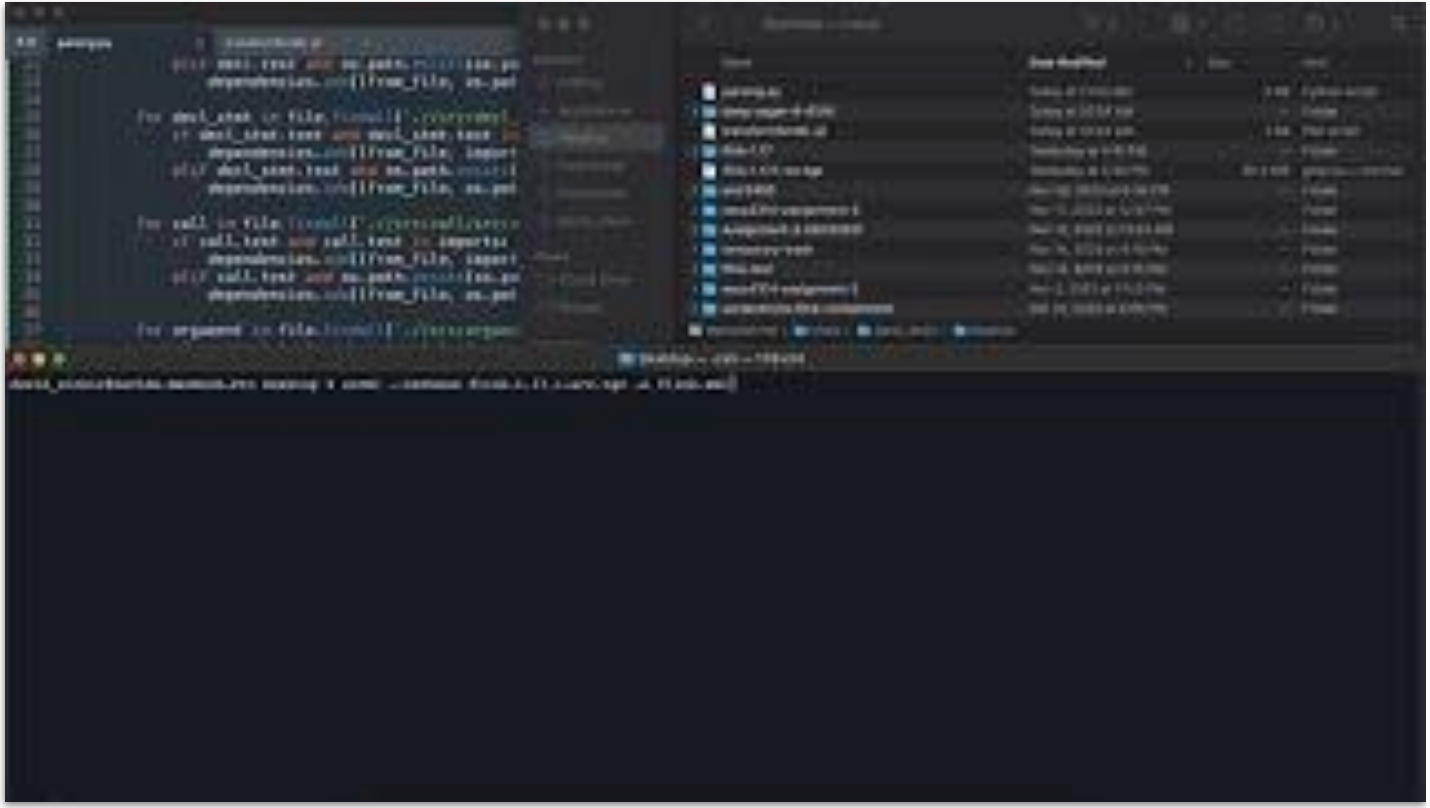
Windows			
Windows 10	exe		
macOS			
Catalina 10.15	pkg		
Mojave 10.14	pkg		
Linux	Client		
Ubuntu 20.04	deb	tar.gz	tar.bz2

Extraction process using srcML

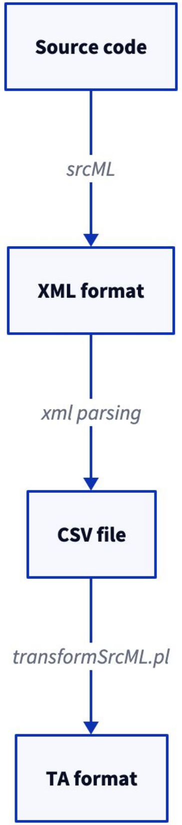
- Download and install srcML[1]
- Run srcML on the flink zip
  - srcml --verbose flink-1.17.1-src.tgz -o flink.xml
- Run a Python script on XML data to produce CSV
  - xml.etree.ElementTree module to parse through XML[2]
  - script specifies dependency extraction logic
  - csv module to return two columns: ‘From File’ and ‘To File’
- Run transformSrcML.pl to produce raw.ta file
  - modified version of original transformUnderstand.pl



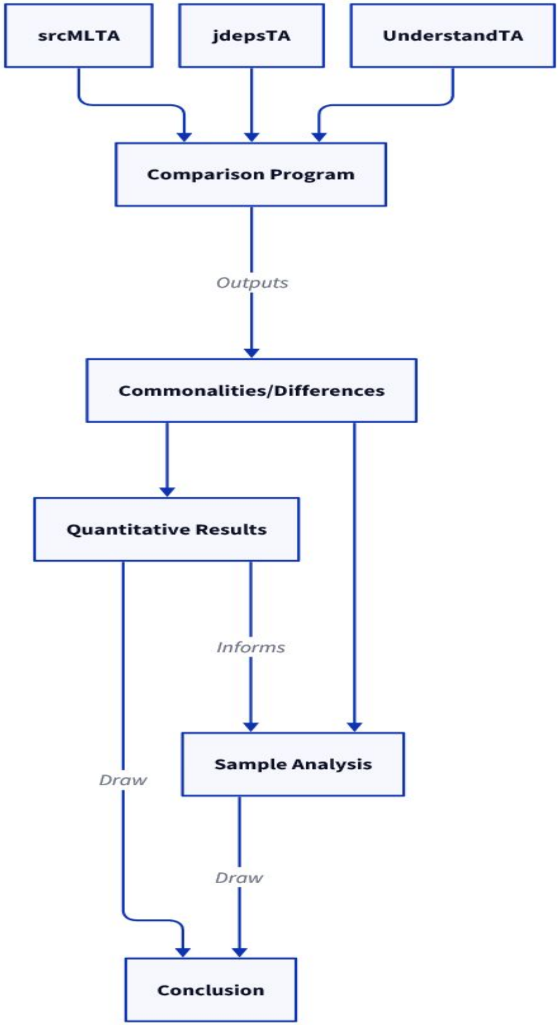
Extraction process



```
            elif decl.text and os.path.[illegible](os.pa
                dependencies.add((from_file, os.pat

        for decl_stmt in file.findall('.//src:decl_
            if decl_stmt.text and decl_stmt.text in
                dependencies.add((from_file, import
            elif decl_stmt.text and os.path.[illegible](
                dependencies.add((from_file, os.pat

        for call in file.findall('.//src:call/src:n
            if call.text and call.text in importsu
                dependencies.add((from_file, import
            elif call.text and os.path.[illegible](os.pa
                dependencies.add((from_file, os.pat

        for argument in file.findall('.//src:argum
```



Quantitative Comparison Process



Quantitative Comparison Results - INSTANCE

Totals Count:15912

Understand: 12977 (82%) Jdeps: 4523 (28%) srcML: 14453 (91%)

Common: 4133 (26%)

Understand - unique: 1069 (7%)

Jdeps - unique: 39 (0.002%)

srcML - unique: 2896 (18%)

Understand/Jdeps: 4484 (28%)

Understand/srcML: 11557 (73%)

srcML/Jdeps: 4133 (26%)

Sample Insight (Entities)

(95% CL/ 5% CI)
Common: 4133 (26%) S:352: Covers most of the top level subsystems, interfaces.

Jdeps - unique: 39 (0.002%): Majority “NOT FOUND”, some top-level abstractions(requests, message handlers.)

Understand/srcML:
Sampling Difference
1420 (U Except S) S:303 : .py/ .ts classes, test-related files.
2896 (S Except U) S:339 : Util classes for SQL, Schemas, Class level (Not file level) entities including defined Data Types.

Quantitative Comparison Results - Dependencies

Totals: 135780
Common: 15231 (11%)
Understand: 120013 (88%)   Jdeps: 44080 (32%)   srcML: 67784 (49%)

Understand - unique: 50134 over 10498 entities

Jdeps - unique: 3327 over 1685 entities

srcML - unique: 12852 over 4704 entities

Sample Insight (Dependencies)

Understand - unique: 50134, S:381 : .py file related dependencies, test dependencies, high number of dependencies per entity. abstract interfaces such as Internal.java, tuple.java, types.java etc. ~10% FP

Jdeps - unique: 3327, S: 344 : transitive (high-degree) dependencies, and inaccurate dependencies. ~70% FP

srcML - unique: 12852, S: 373: similarly to entity extraction, because the method is class-level, Util classes/ functions, Data Types, Schemas, a good portion within the same package. ~8% FP

Precision/Recall

Understand: Precision= 100351 / 120013 = 84%
Recall = 79%

Jdeps: Precision = 17231/44080 = 39%
Recall = 15%

srcML: Precision = 66155/67784 = 97%
Recall = 50%

Limitations

- Human judgement/manual processing involved when it comes to sample analysis and inference. Judgements were made based on previous knowledge of Apache Flink architecture.

- Limited understanding of the tools and their limitations.

Technique Summary

Understand: Comparatively comprehensive dependency extraction, high number of dependencies per entity. Requires some post-processing and iterative measures to help understand architecture (could be addressed by learning the GUI).

Jdeps: Simplistic/Barebone dependency extraction that presents a decent picture of core subsystems in this case. Prone to mistakes/bugs. Unable to identify non-java entities.

srcML: Class-level extractions identifying high number of entities. Heavily influenced by parsing script logic.

Lessons Learned

- Multiple techniques and the pros and cons associated with each technique when going about dependency extraction

- Some tools were difficult to use and resulted in issues (Jarviz) — alternative had to be used

- Results of assessment match many of the limitations for each technique (ie; Jdeps is more prone to mistakes/bugs)

Conclusion / Main points reiterated

Method-Specific Insights:

- Understand: Focused on dependency calculation through language-specific analysis.
- Jdeps: Utilized for command-line class dependency analysis, mainly processing bytecodes.
- srcML: Aimed at converting source code to XML, compatible with multiple programming languages.

Main points reiteration/Insights:

- Extraction techniques overview: Discussed JDEPS, srcML, and Understand as methods for dependency extraction.
- Showcased a numerical comparison of the effectiveness of different methods.
- Highlighted the percentage of dependencies and entities identified by each technique.
- Pros and Cons of each method: Discussed the advantages, like ease of use and availability of documentation.
- Explained the process of using Python scripts to extract and analyze dependencies from source code.

References

[1] https://www.srcml.org/#download

[2] https://docs.python.org/3/library/xml.etree.elementtree.html