Flink Dependency Extraction

Team Debeggars

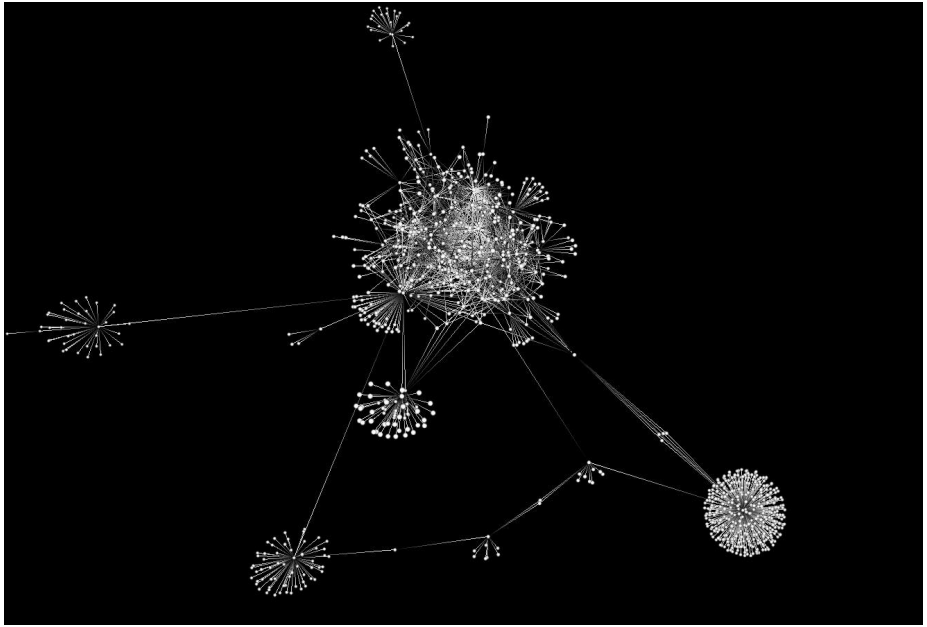
Overview

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

Alternatives

POM-Parsings

Jarviz (ASM opcode analysis)



Jarviz Build failing

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.

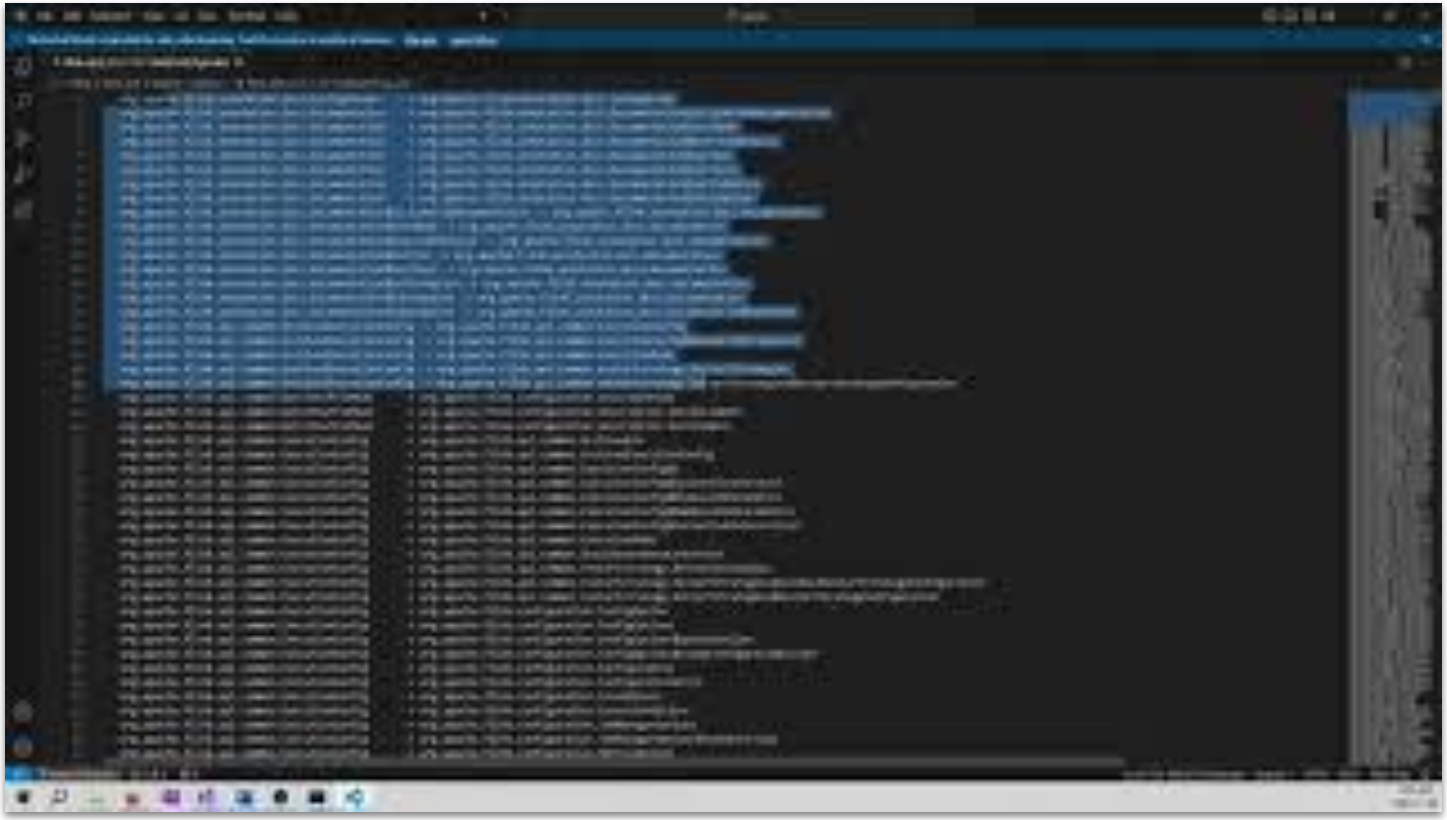
Method 2: jdeps

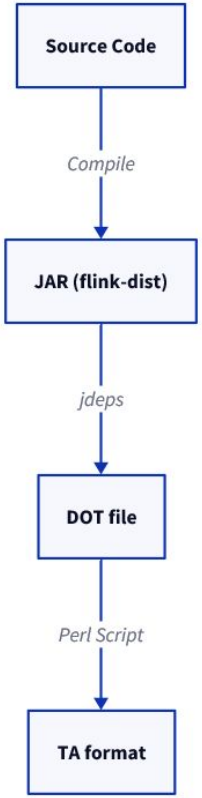
- Command-line tool, class dependency analysis.
- Processes bytecodes (JAR files).
- Compiles statically declared dependencies (Import statements)
- Can produce outputs in various formats including: txt, dot, tgf etc.

Pros: Easy to use, Accessible documentation, Fast.

Cons: Minimal functionalities, fully reliant on “Import”.

Extraction Process (jdeps)





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)

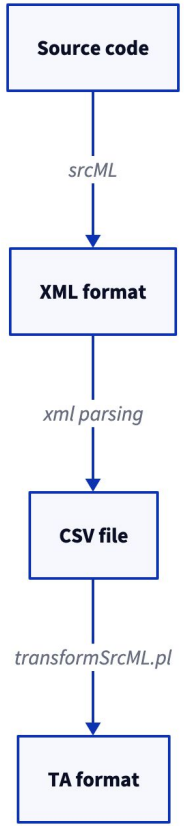
Cons

- 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

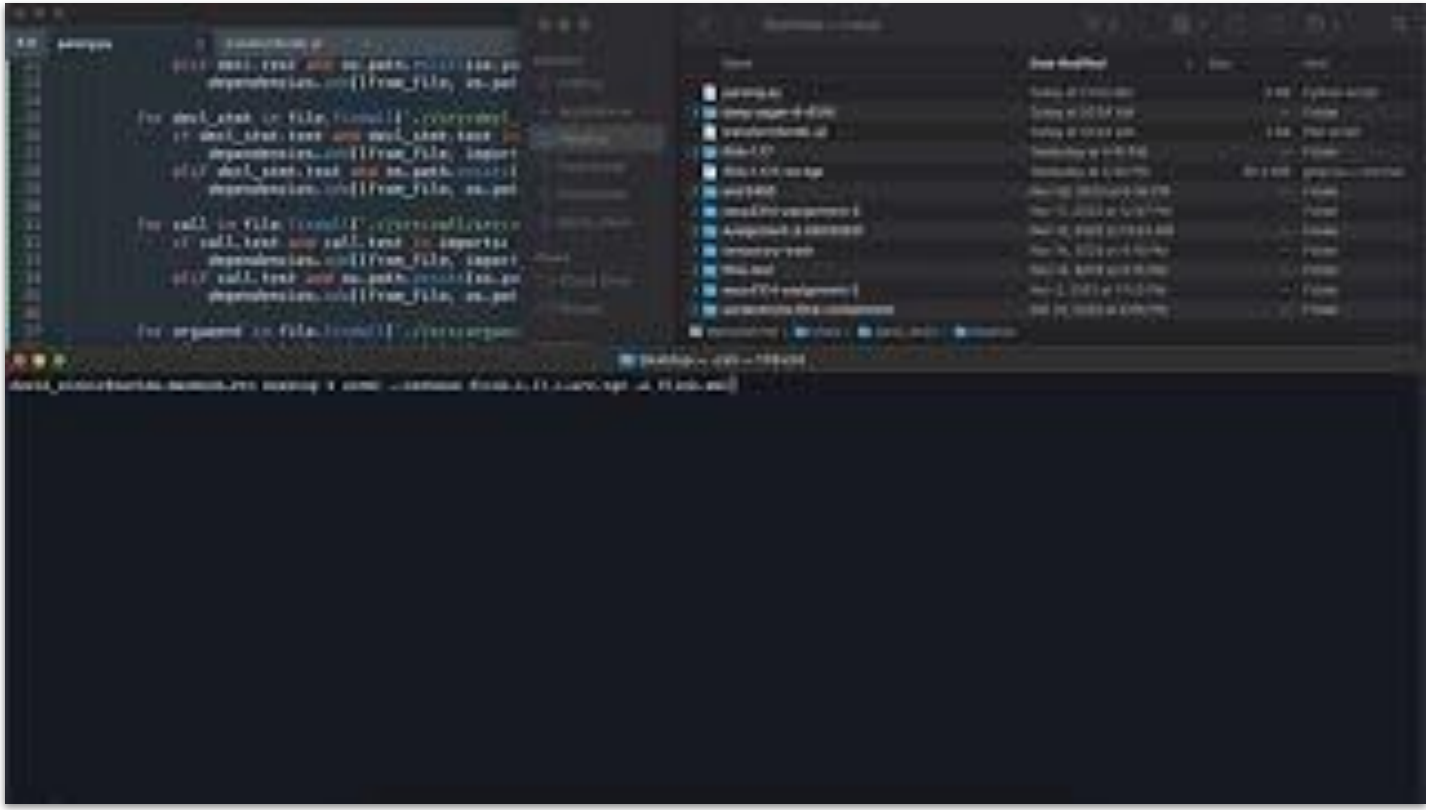
Windows			
Windows 10	exe		
macOS			
Catalina 10.15	pkg		
Mojave 10.14	pkg		
Linux	Client		
Ubuntu 20.04	deb	tar.gz	tar.bz2

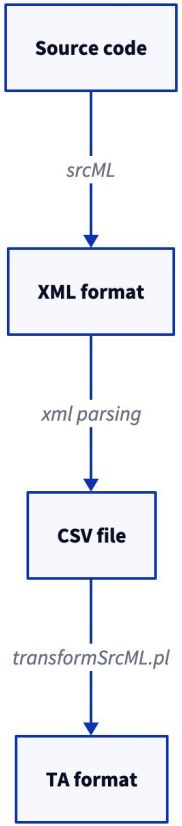
Extraction process using srcML

- 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



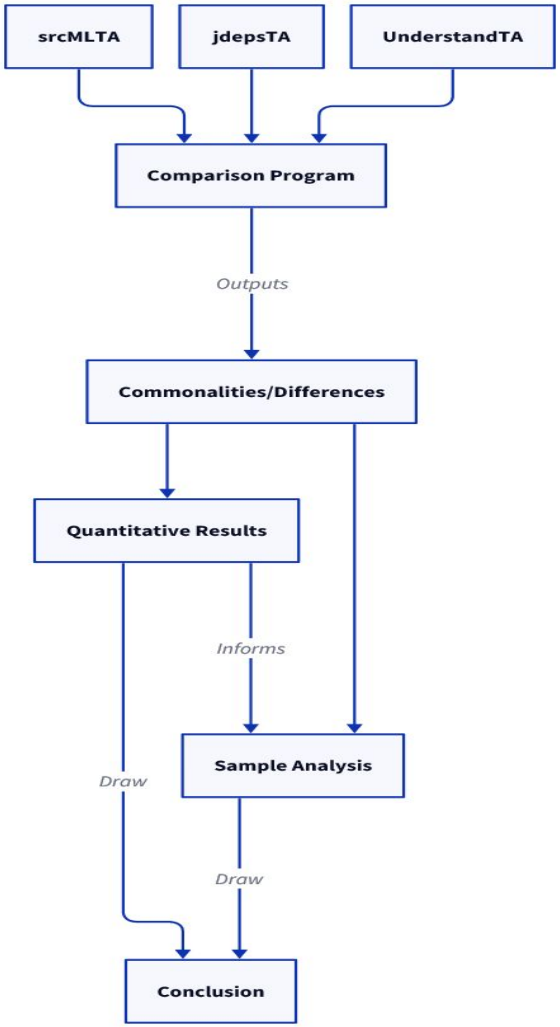
Extraction process





Quantitative Comparison Process

```
var totalKeys = understand.Keys.Union(jdep.Keys).Union(srcML.Keys).ToList();
string qResult = "Total entity count: " + totalKeys.Count() + "\r\n";
var commonKeys = understand.Keys.Intersect(jdep.Keys).Intersect(srcML.Keys).ToList();
qResult += "Common entity count: " + commonKeys.Count() + "\r\n";

var uniqueUKeys = understand.Keys.Except(jdep.Keys).Except(srcML.Keys).ToList();
var uniqueJKeys = jdep.Keys.Except(understand.Keys).Except(srcML.Keys).ToList();
var uniqueSKeys = srcML.Keys.Except(understand.Keys).Except(jdep.Keys).ToList();
qResult += "Unique Understand Entity count: " + uniqueUKeys.Count() + "\r\n";
qResult += "Unique jdep Entity count: " + uniqueJKeys.Count() + "\r\n";
qResult += "Unique srcML Entity count: " + uniqueSKeys.Count() + "\r\n";

var UandJ = understand.Keys.Intersect(jdep.Keys).ToList();
qResult += "U/J Intersect: " + UandJ.Count() + "\r\n";
var UandS = understand.Keys.Intersect(srcML.Keys).ToList();
qResult += "U/S Intersect:" + UandS.Count() + "\r\n";
var SandJ = jdep.Keys.Intersect(srcML.Keys).ToList();
qResult += "S/J Intersect: " + SandJ.Count() + "\r\n\r\n\r\n";

int uTotalLink = understand.SelectMany(kvp => kvp.Value).Count();
int jTotalLink = jdep.SelectMany(kvp => kvp.Value).Count();
int sTotalLink = srcML.SelectMany(kvp => kvp.Value).Count();
var uLinks = new Dictionary<string, List<string>>();
int ulinkCount, jlinkCount, slinkCount;
ulinkCount = jlinkCount = slinkCount = 0;
var jLinks = new Dictionary<string, List<string>>();
var sLinks = new Dictionary<string, List<string>>();
int totalLinks = 0;
foreach (var key in totalKeys)
{
    var value1 = understand.ContainsKey(key) ? understand[key] : new List<string>();
    var value2 = jdep.ContainsKey(key) ? jdep[key] : new List<string>();
    var value3 = srcML.ContainsKey(key) ? srcML[key] : new List<string>();
    var count = value1.Union(value2).Union(value3).Count();
    totalLinks += count;

    var uLink = value1.Except(value2).Except(value3).ToList();
    var jLink = value2.Except(value1).Except(value3).ToList();
    var sLink = value3.Except(value1).Except(value2).ToList();

    if (uLink.Count > 0)
    {
        uLinks.Add(key, value1.Except(value2).Except(value3).ToList());
        ulinkCount += uLink.Count;
    }

    if (jLink.Count > 0)
    {
        jLinks.Add(key, value2.Except(value1).Except(value3).ToList());
        jlinkCount += jLink.Count;
    }

    if (sLink.Count > 0)
    {
        sLinks.Add(key, value3.Except(value1).Except(value2).ToList());
        slinkCount += sLink.Count;
    }
}
qResult += "Total unique dependencies Count: " + totalLinks + "\r\n";

qResult += "Understand dependencies Count: " + uTotalLink +"\r\n";
qResult += "jdep dependencies Count: " + jTotalLink + "\r\n";
qResult += "srcML dependencies Count: " + sTotalLink + "\r\n";
```



QRESULT.txt | TACompare.csproj | Program.cs

```
Total entity count: 15912
Common entity count: 4133
Unique Understand Entity count: 1069
Unique jdep Entity count: 39
Unique srcML Entity count: 2896
U/J Intersect: 4484
U/S Intersect:11557
S/J Intersect: 4133


Total unique dependencies Count: 135780
Understand dependencies Count: 120013
jdep dependencies Count: 44080
srcML dependencies Count: 67784
Understand uniquely extracted 50134 dependencies over 10498 entities.
jdeps uniquely extracted 3327 dependencies over 1685 entities.
srcML uniquely extracted 12852 dependencies over 4704 entities.
```

SRESULT.txt | QRESULT.txt | TACompare.csproj | Program.cs

```
Common Entities(352): ChangelogStateBackendLocalHandle.java
Value.java
AbstractServerHandler.java
SerializedThrowable.java
PartialDispatcherServices.java
AsynchronousBlockWriterWithCallback.java
DoubleZeroConvergence.java
InputSelectable.java
BinaryUnionNode.java
IdPartitioner.java
BooleanPrimitiveArraySerializer.java
PlanUnwrappingSortedReduceGroupOperator.java
KeyValueStateIterator.java
DefaultExecutionDeploymentReconciler.java
AggregatedTaskManagerMetricsHeaders.java
ResponseBody.java
ContinuousProcessingTimeTrigger.java
PlanLeftUnwrappingCoGroupOperator.java
TaskExecutorToResourceManagerConnection.java
UnregisteredMetricGroups.java
BloomFilter.java
TriggerResponse.java
SlidingEventTimeWindows.java
TaskSlotTableImpl.java
ExecutionDeploymentTrackerDeploymentListenerAdapter.java
InputSelection.java
CollectSinkFunction.java
ChainedStateHandle.java
KubernetesResourceManagerFactory.java
LeaderRetrievalListener.java
SystemResourcesMetricsInitializer.java
LeaderRetriever.java
OperatorCoordinator.java
Tuple21.java
LegacySourceTransformation.java
ChannelStateCheckpointWriter.java
DateSerializer.java
LeaderElectionEventHandler.java
PipelineExecutorServiceLoader.java
AsynchronousFileIOChannel.java
DataSinkNode.java
JobVertexIdPathParameter.java
ListAccumulator.java
JobExecutionResultHandler.java
```

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 - LINKS

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, mainly consisting of abstract interfaces such as Internal.java, tuple.java, types.java etc.

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

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.

U and S Except J - 6462, S: 363: Again, heavily test-related, abstractions and utils.

Precision/Recall

Total Unique Dependencies: 135780
Common Dependencies: 15231 (All relevant instances?)
Understand: Precision~= 15231/ 120013 = 13% (Much higher if the focus is on test file dependencies as well as some abstract interface usage)

Jdeps: Precision ~= 15231/44080 = 35%

srcML: Precision ~= 15231/67784 = 22%

Recall ~= 15231/15231 = 100% for all 3. (More than likely lower, due to relevant dependencies missed by all 3 methods).

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.

- Difficult to quantify performance using classification metrics because it is hard to judge relevance.

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 (many irrelevant to the overall architecture). Heavily influenced by parsing script logic.

Lessons Learned

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

-Each technique provides different types of result (ie; srcML provides results for java code, converting script to XML)

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

-Assessment of quantitative analysis and the results of which match common facts 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 tools like Docker and 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