Dependency Families in the Maven Ecosystem

An Analysis of Software Dependency Graphs

EEMCS, Delft University of Technology, The Netherlands
Author: Wojciech Graj <w.graj@student.tudelft.nl>
Supervisors: Sebastian Proksch, Cathrine Paulsen
https://resolver.tudelft.nl/uuid:67d109a8-e7b6-45eb-995c-f2cf8851ac69

1. Introduction

- Apache Maven is a build automation tool used for Java
- What is a "dependency family"?
  - Dependencies from the same organization, designed to be used together

    For example:
  - com.fasterxml.jackson>{jackson-bom, jackson-datatype-guava, ...}
  - org.apache.lucene>{lucene-core, lucene-queryparser, lucene-queries, ...}
- Why is this worth researching?
  - Determine best practices for maintainers based on existing conventions
  - Identify patterns downstream users can expect
  - Justify development of new functionality in Maven to make them easier to use or maintain

Research questions:

1) How can we detect which dependencies belong to the same dependency family?
2) What are some common patterns among dependency families?
   2 a) How are dependency family sizes distributed, and how much of the Maven ecosystem to they account for?
   2 b) How is the frequency of use of individual dependencies in any family distributed?
   2 c) How often are versions out-of-sync and how frequently are releases without code changes published to keep their versions in sync?

2. Dataset

- Dataset: Central Maven Index
  - 688 201 Artifacts
  - 15 985 044 POM files
  - 16 297 705 Releases
  - 151 010 512 Dependency usages

3. Dependency Family Detection

- Construct graph of dependencies
- Edge weights: linear combination of pairwise co-use of dependencies, and existence of parent-child relation in POM
  - Higher $\alpha \rightarrow$ More influence from parent-child relation
- Evaluation criteria: Jaccard index comparing similarity to manually-identified families (J∈[0,1], higher is better)

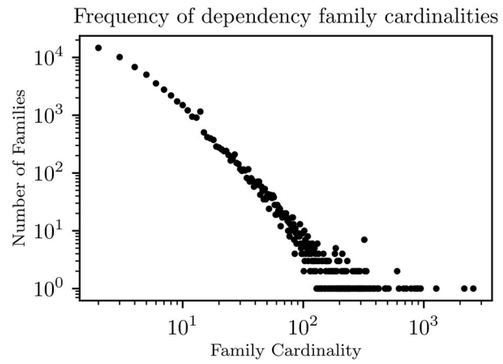
Algorithm	Res	α	J
Connected Components	-	1.0	0.236
Leiden	0.06	0.0	0.432
Louvain	**0.003**	**0.96**	**0.575**

- Difficult to get better results without more data
  - Very difficult to calculate similarities of package names
- Detected families are satisfactory

4. Family-Based Insights

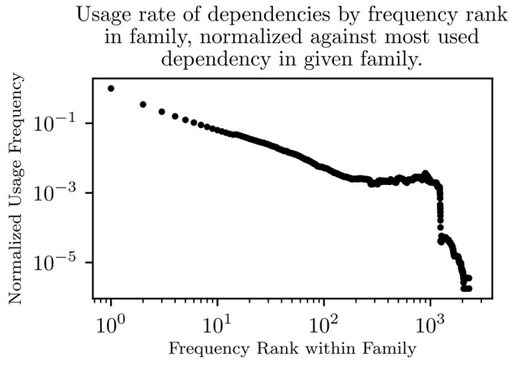
4.1. Cardinalities and Pervasiveness

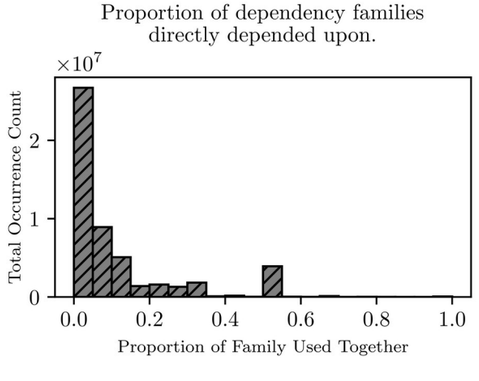
- 76.0% of all artifacts belong to a dependency family
- Most families very small
  - 0.570% have a cardinality > 100



4.2. Usage Rates

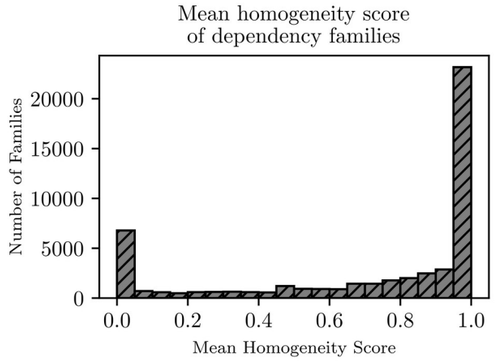
- Most families have a few main dependencies and many peripheral ones
- Only small subset of families is typically used





4.3. Version Homogeneity and Empty Releases

- 55.3% of all dependency families consistently have same version number
- Releases with no code changes are used to keep versions in sync
- Many releases (50.0%) have no code changes but produce different source JAR because reproducible builds are not default



5. Future Work & Recommendations

- Investigate impact of storing binary diffs between releases, as many are very similar
- Enable reproducible builds by default
- Examine trends in dependency usage in downstream non-dependency software
- Examine other non-Java dependency management systems

TU Delft