ANALYZING THE CRITICALITY OF APACHE PACKAGES THROUGH A TEMPORAL DEPENDENCY GRAPH

AUTHOR:
Denis Corlade (D.Corlade@student.tudelft.com)
SUPERVISORS:
Georgios Gousios, Diomidis Spinellis

TU Delft

1 INTRODUCTION & BACKGROUND

- Libraries speed up development [1]
- Adding libraries can lead to vulnerability threats through direct or transitive dependencies:
  - Apache Log4J
  - Equifax
- Current research focuses only on current releases of dependencies, missing the past [2]
- Introducing the time component to create a scalable temporal dependency graph

2 RESEARCH QUESTIONS

RQ1 - What would a graph data structure for package dependencies that contain a time component look like?

RQ2 - Does the introduction of time increase precision?

RQ3 - What are the most widely used Java packages?

MAIN RQ - What are the most widely used Java packages at a given time?

3 METHODOLOGY

- Design the temporal dependency package network [3](Fig 1)
- Gather the data from the Maven Central repository
- Implement the graph data structure and feed the data into it
  - Packages are nodes and dependencies are edges
- Resolve dependencies constraints from Maven (Fig 2)
- Allow querying on time intervals
- Analyze the data with various algorithms at different points in time

Fig 1-Graph Structure

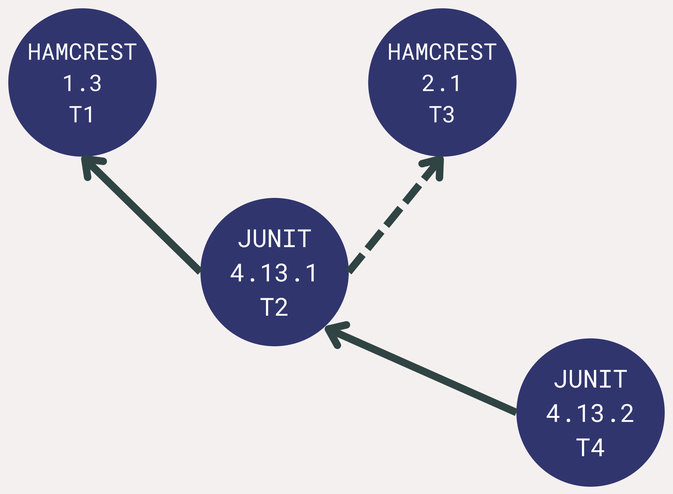
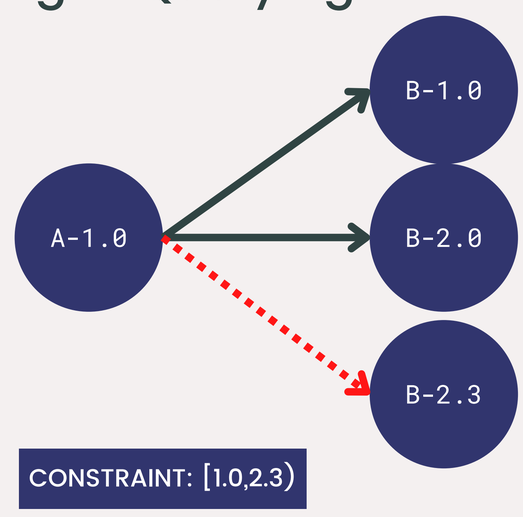


Fig 2-Querying



4 RESULTS

Nodes - 40K
Edges - 1.5M
The graph is constructed using as little information in the nodes as possible to improve efficiency.

The graph structure maintains corectness:

Accuracy was tested by comparing the list of dependencies shown by using the algorithm implement versus the list of dependencies shown by using the repository manager.

GroupID:ArtifactID:Version	Value
junit:junit:4.13.2	1
org.slf4j:slf4j-api:1.7.36	1
org.scala-lang:scala-library:2.13.8	1
com.google.guava:guava:31.1-jre	0.833
org.mockito:mockito-core:4.6.1	1

Fig 3-Accuracy table

The time component increases precision:

Showing all dependencies of "junit" package:

GroupID:ArtifactID (constraint)	Versions
org.hamcrest:hamcrest-core (1.3)	2.1, 2.2
org.hamcrest:hamcrest-library (1.3)	2.1, 2.2
(t) org.hamcrest:hamcrest (2.1)	2.1, 2.2

The above table does not take time into consideration, and in case one is to query for a specific interval, it would not provide the wanted results.

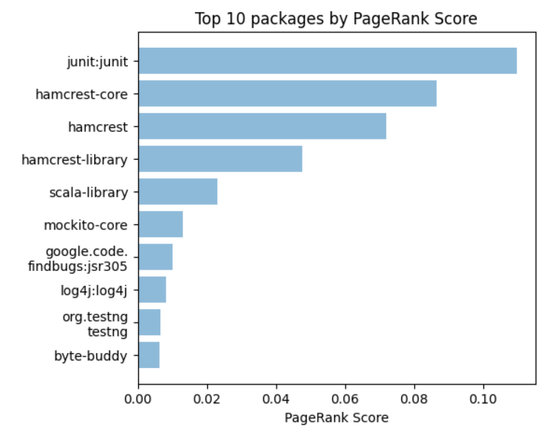
Showing all dependencies of "junit" package after 2019:

GroupID:ArtifactID (constraint)	Versions
org.hamcrest:hamcrest-core (1.3)	2.2
org.hamcrest:hamcrest-library (1.3)	2.2
(t) org.hamcrest:hamcrest (2.1)	2.2

Fig 4-Dependencies on time intervals

What is the most used software?

Analysis of most used packages calculated with the PageRank algorithm.
Plotting the rankings over time shows interesting trends



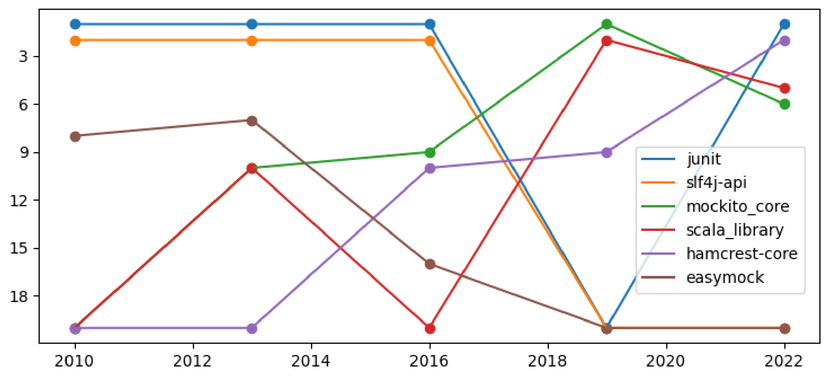


Fig 5&6-Analysis of the most used packages currently and over time

5 CONCLUSION

The one major contribution is adding the possibility of querying on different time ranges in a package dependency network, more precisely, a TDPN.

Moreover, it provides an insight into the ecosystem of Maven, its dimensions, and how vulnerability can be observed as a potential side-effect of showing software usage over time.

6 FUTURE WORK

- A larger dataset could be downloaded.
- Implement the graph in a different language from Golang to improve efficiency
- Comparison between the graph implemented in this paper and other previous work.
- Analyze the packages per different scope (test, provided etc.)
- A more in depth validation of the work and correctness.

RELATED LITERATURE
[1]Parastoo Mohagheghi and Reidar Conradi. Quality, productivity and economic benefits of software reuse: a review of industrial studies. Empirical Software Engineering, 12(5):471–516, Oct 2007.
[2]César Soto-Valero, Amine Benelallam, Nicolas Harrand, Olivier Barais, and Benoit Baudry. The emergence of software diversity in maven central. In Proceedings of the 16th International Conference on Mining Software Repositories, MSR '19, page 333–343. IEEE Press, 2019.
[3]Riivo Kikas, Georgios Gousios, Marlon Dumas, and Dietmar Pfahl. Structure and evolution of package dependency networks. In 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), pages 102–112, 2017.