Dependency Update Adoption Patterns in the Maven Software Ecosystem

Baltasar Berretta, Augustus Thomas, Heather Guarnera

Department of Mathematical and Computational Sciences, The College of Wooster, USA

1

Context

Regular dependency updates protect dependent software components from upstream bugs, security vulnerabilities, and poor code quality. Measures of dependency updates across software ecosystems involve two key dimensions:

- the time span during which a release is being newly adopted (**adoption lifespan**), and
- the extent of adoption across the ecosystem (**adoption reach**).

We examine correlations between adoption patterns in the Maven software ecosystem and two factors:

- the magnitude of code modifications (**semantic change,** which is the extent of modifications affecting the meaning or behavior of the code) in an upstream dependency, and
- the relative **maintenance rate** of upstream packages.

2

Research Questions

RQ1: How does semantic change size correlate with adoption patterns?

- Correlations between:
  - Magnitude of semantic change size (major, minor, patch), and
  - Number of dependents and adoption lifespan

RQ2: How does maintenance activity correlate with adoption patterns?

- Maintenance rate: custom metric that quantifies the number of releases relative to a release's adoption lifespan

$$maintence\ rate = \frac{\#\ of\ releases}{adoption\ lifespan\ (years)}$$

3

Insights

	Semantic Change Size			Maintenance Rate		
	Major C.X.X	**Minor** X.C.X	**Patch** X.X.C	**High** $x > 0.1$	**Medium** $0.01 \leq x \leq 0.1$	**Low** $x < 0.01$
Number of packages	3,526,059 (84%)	587,676 (14%)	83,954 (2%)	1,830,042 (44%)	181,600 (4%)	2,186,047 (52%)
Average dependents	17.65	8.09	4.21	8.58	19.73	22.05
Average adoption lifespan (in days)	34.52	25.78	19.73	0.36	39.88	59.84

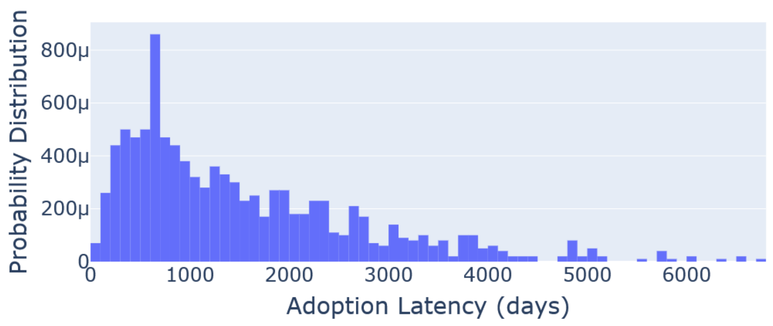
Major releases are most common, most depended, adopted for longest

Most releases are distributed less frequently than once every 10 adoption lifespans

4

Maven Trends

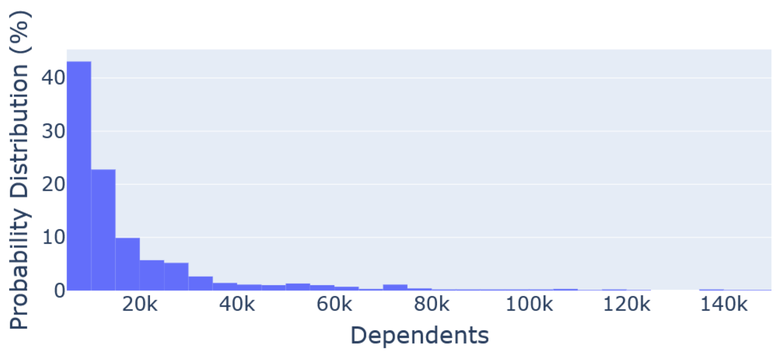
Using the Goblin Weaver framework, we find adoption latency in the Maven ecosystem follows a log-normal distribution while adoption reach exhibits an exponential decay distribution.



Adoption Latencies
Time between first and last adoption of a package

Modal adoption latency ~ 2 years

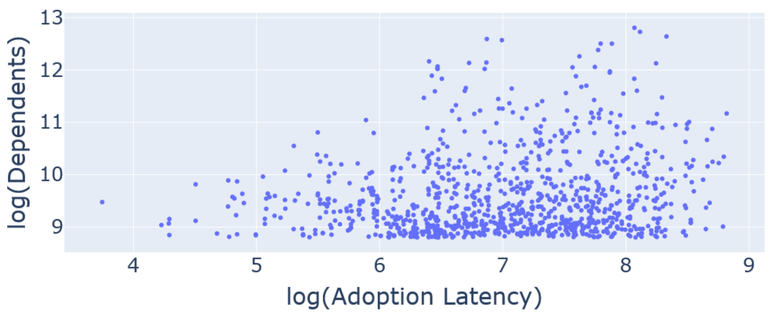
Figure 1: Probability distribution of adoption lifespan across a sample of Maven packages (N=1,000)



Number of Dependents
Total number of releases that are dependent on a project

Number of dependents is exponentially distributed

Figure 2: Probability distribution of number of dependents across a sample of Maven packages (N=1,000)



Dependent-Latency Correlation
Number of dependents as a function of adoption latency

Adoption latency is loosely correlated with number of dependents

Figure 4: Logarithm of dependent number as a function of the logarithm of adoption lifespan

5

Threats to Validity

- Dataset sampling: due to infrastructure & query-time constraints for "Maven trends" we only analyzed the 1,000 packages with the highest number of dependents
- Semantic versioning compliance the change caused by a release may not be represented according to the versioning convention (i.e. some "patches" may change APIs causing longer adoption times)

6

Conclusion & Future Work

- Maintainers of highly-depended packages should expect longer adoption periods for their major releases
- Consumers of highly maintained components should integrate automated dependency update tools
- Minor and patches are adopted at the same rate
  - Tendencies to adopt minor changes?
- Larger semantic version changes correlate to longer adoption lifespans
  - Measuring semantic change size by adoption latency

7

References

D. Jaime, J. E. Haddad, and P. Poizat, "Goblin: A framework for enriching and querying the Maven Central Dependency Graph," in Proceedings of the 21st International Conference on Mining Software Repositories, ser. MSR '24. Association for Computing Machinery, 2024, pp. 37–41. [Online]. Available: https://dl.acm.org/doi/10.1145/3643991.3644879

A. Tsakpinis, "Analyzing maintenance activities of software libraries," in Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering, ser. EASE '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 313–318. [Online]. Available: https://doi.org/10.1145/3593434.3593474

H. He, R. He, H. Gu, and M. Zhou, "A large-scale empirical study on Java library migrations: prevalence, trends, and rationales," in Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ser. ESEC/FSE 2021. New York, NY, USA: Association for Computing Machinery, 2021, pp. 478–490. : https://doi.org/10.1145/3468264.3468571

Poster Number 12

MSR 2025

Acknowledgements

Thank you to the Wooster Department of Mathematical and Computational Sciences, and the STEM Student Success Initiative at Wooster for supporting our participation in this conference. Thank you to our third author for graciously encouraging, advising, proof-reading, guiding and otherwise helping us with this project; we would not have our first conference paper without you!

THE COLLEGE OF WOOSTER