# An Empirical Study on the Evolution and Maintenance of the Maven Central Dependency Graph

Nirajan Bhattarai[1], Riley Stratton[2], Minhaz Zibran[2*]
[1]*Department of Biomedical and Pharmaceutical Sciences*, *Idaho State University*, Pocatello, USA
nirajanbhattarai@isu.edu
[2]*Department of Computer Science*, *Idaho State University*, Pocatello, USA
rileystratton@isu.edu, [*]minhazzibran@isu.edu

***Abstract*—Frequent new releases are expected to address security and compatibility issues, but may result in security breakages in return, and increase maintenance cost and system complexity. The primary aim of this paper is to conduct an empirical study on the ecosystem-wide evolution of the Maven Central Dependency Graph and the maintenance status of its artifacts, with the ultimate goal of educating the developer community about the evolution and maintenance levels within the Maven Central ecosystem. We examined the growth patterns of library releases over time, focusing on frequency, rhythm, and speed, to understand the influence of project management methodologies such as Agile, security vulnerabilities, and open-source contributions on the evolution of the Maven Central ecosystem. Similarly, we also attempted to measure the maintenance level of artifacts in the ecosystem based on various factors such as popularity, missed releases, and whether vulnerabilities have been addressed or not. Rapid growth between 2004 and 2005 aligned with emergence with agile methodologies, followed by a gradual increase up to 2021, after which it started plateauing or stabilizing, which may be attributed to the emergence of new technologies like AI. In contrast, the growth of vulnerabilities across all severity types experienced a rapid decline after 2021, possibly indicating an increased emphasis on security and improved library maintenance within the community. We found that 5.21% and 94.74% of all libraries are fully and partially maintained, respectively. The study suggests the need for further in-depth research on the most prevalent vulnerabilities in the Maven Central ecosystem, and a well-validated metric system for assessment of combined vulnerability and maintenance level in the Maven Central Dependency Graph.**

***Index Terms*—Maven Central, Dependency Graph, Evolution, Maintenance,Release, Vulnerability, Dependency**

## I. Introduction

Dependencies are third-party software components introduced into the software development framework to reduce costs and accelerate the speed of software development. As dependencies are an indispensable component of the software development process, frequent new releases are expected to address security and compatibility issues, ensuring safe and reliable use (1). However, maintenance effort on these dependency libraries incurs costs and increases complexity (2). Raemaekers et al. highlighted that there will always be tension between version updates intended for mitigating security risks and the security breakages that may result from such updates (3).The study on the evolution of dependency can be approached from two perspectives: ego-centric, focusing on particular dependency libraries; and/or ecosystem-wide, having global insight on dependency ecosystem management and pervasion (4), (2).To assist with these kinds of studies, several repositories and resources are available, such as Maven Central, which maintains over 2.8 million unique artifact repositories for JVM-based artifacts, along with metadata detailing versions, dependencies, and upload dates in a dependency graph. This makes it well-suited for both ego-centric and system-wide big data evolution studies (5). Similarly, various validated metrics have been proposed and incorporated in the Maven Central Database for researchers, eliminating the need to develop and validate their own quantification metrics, which would be particularly challenging for a dataset of this size. Some of these proposed metrics are popularity (6), calculated based on the number of dependents a particular version of a library has in a year, and the freshness (1), which is the computation of the number of releases and the duration elapsed between a particular release and its most recent update.

The main aim of this paper is to analyze the trend in software releases, their associated vulnerabilities, and their impact on popularity and vulnerability maintenance in the Maven Central ecosystem. The trend was analyzed in terms of the rhythm and speed of library releases and the prevalence of vulnerabilities of different severities on a yearly basis. Also, the impact of release dynamics, measured by the speed of library releases, the average gap between consecutive releases, and the number of releases, on the net change in vulnerability and popularity was assessed. The overarching goal is to inform the developer's community about the trend in library releases and their vulnerability and the impact of various release patterns on maintaining vulnerability and popularity.

## II. Method

### A. Terminologies

**Artifact and its release:** The term artifact is used to refer to a library, which is a collection of pre-written code, functions, and routines that developers can use to perform common tasks.Libraries are designed to be reusable and modular(**? ?** ).

**Dependency:** Dependency, in software development, refers to third-party components utilized to build cost- and time-effective software systems, either by an organization or an individual (**?** ).

**Freshness:** The freshness of a software release can be assessed with two measures: (1) `numberMissedRelease`, which counts the number of newer releases available between a given release and the latest one; (2) `outdatedTimeInMs`, which represents the duration between a given release and the latest one in milliseconds (**? ?** ).

**Popularity:** The popularity of a software release is measured as the count of the number of software versions that depend on it within one year, and before the current dependency graph's date (**?** ).

**Common Vulnerabilities and Exposure (CVE):** CVE is a list of publicly disclosed software vulnerabilities. Each vulnerability is assigned a unique CVE ID, such as CVE-2024-11111. These CVE IDs are used to recognize unique vulnerabilities and coordinate bug fixes (**?** ).

### *B. Research Questions*

**Q1. What are the trends in rhythm, speed, and vulnerability of library releases in the Maven dependency graph?** An analysis of trends in rhythm, speed, and vulnerability provides insight into the evolution of the Maven dependency graph ecosystem and may indicate the effects of various events and developments, such as the effect of the advent of different project methodologies like Agile or changes in developer behavior over time.

**Q2.How does release behavior affect changes in the vulnerability and popularity of libraries in the Maven Central ecosystem?** A change in vulnerability may indicate effective maintenance of a software library, based on the assumption that, for instance, a decrease in vulnerability reflects efforts to mitigate security flaws and shortcomings. As a result, popularity increases, and vice versa. We assume that these factors are directly associated with release behavior.

### *C. Data Acquisition and Pre-processing*

For data acquisition, we selected `with_metrics_goblin_maven_30_08_24.dump`, which is a Neo4j graph database. The database is provided as part of the 2025 Mining Software Repositories (MSR) Mining Challenges (4). It includes artifact, release, and dependency nodes, along with various attributes and metrics. The database was queried using Cypher to extract all relevant information, including metrics associated with artifact releases.

Each release node has a `Timestamp` attribute, representing the exact moment the release occurred, stored in milliseconds since the Unix epoch (January 1, 1970). The `pd.to_datetime()` function was applied to convert the Unix timestamp (in milliseconds) into a readable datetime format. Finally, the `.dt.year` accessor extracts the year from the converted datetime. Metrics related to release nodes—popularity, CVE, `outdatedTimeInMs`, and `numberMissedRelease` (freshness)—were also extracted. Additional metrics include `CVE_count` for each severity level (low, medium, high, critical, and none), as well as a composite vulnerability score.

*1) CVE count and Composite Vulnerability Score:* CVE is presented in JSON format for each release, with key "cve", which represents array of CVE identifiers. The JSON format is first cleaned, and subjected to count, which is measured as the number of the elements in the array. Similarly, the count of each severity level, where were 'LOW', 'MEDIUM', 'HIGH', 'CRITICAL', and 'NONE', within the CVE for each release was extracted. Since each release may have multiple CVEs with different severity levels, we calculated the total vulnerability contribution score based on the severity level of vulnerabilities along with their count in respective severity. The total vulnerability contribution for each release is computed by multiplying the weight assigned to each severity level by its count and then summing them up, as follows: The composite vulnerability is calculated as follows:
Total vulnerability contribution =Count('LOW') X 3.0 + Count('MEDIUM') X 5.5 + Count('HIGH') X 8.0 + Count('CRITICAL') X 9.5 + Count ('None') X 0.
For the release having no vulnerabilities, total vulnerability contribution was assigned as zero. The choice of weights is based on the vulnerability severity scale (7).

*2) Gap between Consecutive Releases:* The **gap between consecutive releases**(in days) of an artifact is the set of time gaps (in days) between consecutive releases.

The time gap between consecutive releases is:

$$\Delta r_i = r_{i+1} - r_i, \quad \forall i \in \{1, \ldots, n-1\} \tag{1}$$

Thus, the release rhythm is:

$$R_{|A,Y|} = \{\Delta r_1, \ldots, \Delta r_{n-1}\} \tag{2}$$

- $A$ be an artifact.
- $Y$ be a given year.
- $R_Y = \{r_1, r_2, \ldots, r_n\}$ be the ordered set of release dates in year $Y$.

*3) Release Speed:* The speed of release of an artifact can be defined as the number of releases of an artifact, $x_a$, within a specific year, $y_b$, per total days from its first release to the last release in that year (2).

$$\text{Speed}_{|x_a, y_b|} = \frac{N_{a,b}}{\Delta T_{a,b}} \tag{3}$$

where $\Delta T_{a,b}$ represents the time interval (measured in days) between the first release and the last release of an artifact, $N_{a,b}$ represents the number of releases of artifact $x_a$ within a specific year $y_b$, and $N_{a,b-1}$ represents the number of releases in the previous year.

### *D. Change in vulnerability and Popularity of libraries*

The net change in an artifact's popularity and vulnerability was computed by summing the changes in the popularity score and the composite vulnerability score across consecutive releases of an artifact, sorted by release date. Libraries were then categorized into groups: no net change, net increase, and net

decrease in popularity and vulnerability. The ”no net change” category was further divided into two subcategories: artifacts with a zero score across releases and those with a constant fixed score(no change), resulting in four groups. Additionally, the number of releases, average consecutive release gap in days, release speed(release per day), and frequency of artifacts for each group were determined for each group for association.

## III. Results

### A. Trend in speed, rhythm, and vulnerability of library release in Maven Central Ecosystem

The analysis of ecosystem vulnerability for each year, representing the total number of vulnerabilities based on different severities, exhibited four distinct phases: a stable phase (Phase I: until 2010), rapid growth (Phase II: 2011–2016), a plateau (Phase III: 2017–2020), and a decline (Phase IV: 2021 and beyond)(Fig.1).

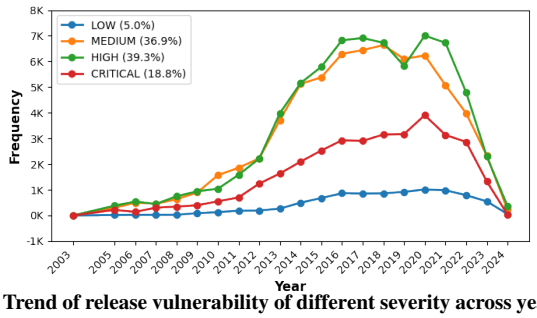


Fig. 1. **Trend of release vulnerability of different severity across years.** The CVE label for each release can have none, one, or multiple associated CVEs. The CVEs associated with each release were separated, and the severities of the individual CVEs were identified and counted. Any CVE label in an incorrect format was excluded. The line graph illustrates the trend in the count of different CVE severity levels, with each dot representing the number of cases of a specific severity type found in that particular year within the ecosystem.

A similar evolution pattern was observed when the rhythm (days difference between consecutive releases of each artifact) (Fig. 2) and speed (number of releases per day for each artifact) (Fig. 3) of library releases were projected in a box plot over the years. We observed two waves, the first in Phase I and the second in Phase II. In Phase III, days difference of library release of the system becomes shorter and consistent and Phase IV, it becomes more shorter and consistent. We also observed a gradual increase in speed when examining the median release speed across years (Fig. 4). The distribution of speed remained less variable and consistent in Phase I and Phase II. In Phase III, the distribution of release speed across libraries became increasingly wider, while in Phase IV, it stabilized and became more consistent. These observations are based upon the distribution and median value represented in a box plot.

### B. Impact of Release Behavior on Changes in Vulnerability and Popularity of a Library

We observed that release speed was significantly lower, while the average total number of releases was higher for libraries with a net decrease in vulnerability and a net increase

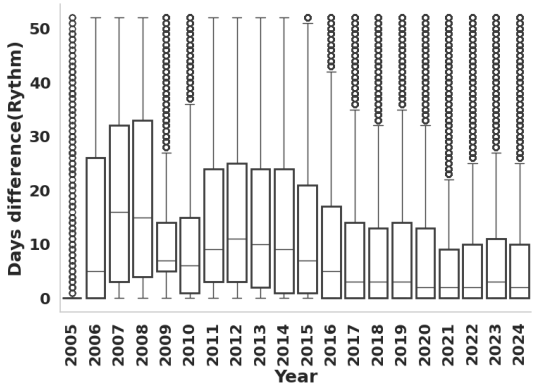


Fig. 2. **Distribution of rhythms of library release in the different years.** The X-axis represents different years, and the Y-axis represents the days' difference between consecutive releases of an artifact. The box represents the distribution of the time gap between consecutive library release of artifacts(rhythm) within a year.

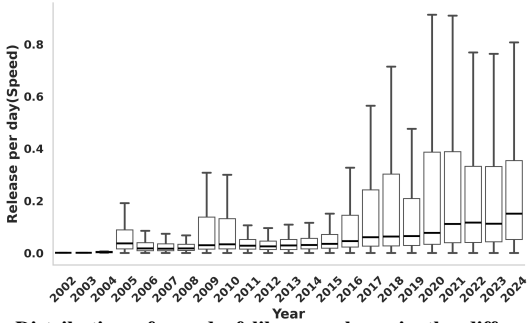


Fig. 3. **Distribution of speed of library release in the different years.** The X-axis represents different years, and the Y-axis represents the number of release per day of an artifact(seed). The box represents the distribution of the speed of a release of artifacts within a year.

in popularity. The average consecutive release gap (in days) was not significantly different across different vulnerability change groups but was slightly higher for libraries with a net decrease in composite vulnerability. In contrast, it was highest for libraries with a net decrease in popularity and lowest for those with constant popularity compared to other groups. Similarly, there was no significant increase in popularity with a net decrease in vulnerability, nor a significant net decrease in vulnerability with a net increase in popularity, compared to interactions between other groups.

### C. Vulnerability Maintenance among libraries

– need correction here

We measured the net change in the vulnerability of libraries across their releases, which is the sum of the composite vulnerability score changes between consecutive releases of an artifact. A net decrease in vulnerability indicates effective maintenance of the library from a vulnerability standpoint, a net decrease indicates a lack of effective maintenance of the library from the security standpoint, net change of zero may indicate either that the libraries had no vulnerabilities across releases or that the vulnerability level remained constant.

We also computed the net change in popularity to explore any potential relationship between the net change in composite

| Change in Composite Vulnerability | # of artifacts | # of releases | Net change in popularity | Avg. consecutive release gap (days) | Releases per day |
|---|---|---|---|---|---|
| No Composite Vulnerability | 492412 | 28.82 ± 94.60<br>(28.56 - 29.09) | 3.94 ± 275.81<br>(3.17 - 4.72) | 90.95 ± 175.50<br>(90.46 - 91.44) | 84762.44 ± 3431258.26<br>(75178.63 - 94346.25) |
| Constant Composite Vulnerability | 356 | 72.96 ± 131.59<br>(59.24 - 86.68) | 497.28 ± 6907.72<br>(-222.74 - 1217.29) | 93.08 ± 128.49<br>(79.69 - 106.48) | 577259.77 ± 5802074.72<br>(-27509.29 - 1182028.84) |
| Net decrease in composite vulnerability | 1001 | 98.52 ± 159.71<br>(88.61 - 108.42) | 132.37 ± 970.30<br>(72.19 - 192.55) | 103.89 ± 210.96<br>(90.80 - 116.97) | 13.17 ± 24.60<br>(11.64 - 14.69) |
| Net increase in composite vulnerability | 20 | 58.95 ± 104.50<br>(10.04 - 107.86) | 329.50 ± 1277.36<br>(-268.32 - 927.32) | 90.44 ± 104.22<br>(41.66 - 139.22) | 6307212.12 ± 28206653.04<br>(-6893907.86 - 19508332.10) |

TABLE I
COMPARISON OF COMPOSITE VULNERABILITY CHANGES IN ARTIFACTS AND THEIR IMPACT ON RELEASES.

| Change in Popularity | # of Artifacts | # of Releases | Net Change in Composite Vulnerability | Avg. Consecutive Release Gap (Days) | Releases per Day |
|---|---|---|---|---|---|
| No Popularity | 389092 | 21.95 ± 68.46<br>(21.74 - 22.17) | -0.01 ± 0.74<br>(-0.01 - -0.01) | 94.36 ± 180.97<br>(93.79 - 94.93) | 280.07 ± 10241.29<br>(247.89 - 312.25) |
| Constant Popularity | 22082 | 31.60 ± 93.67<br>(30.36 - 32.83) | -0.05 ± 1.05<br>(-0.06 - -0.04) | 50.24 ± 91.03<br>(49.04 - 51.44) | 144.46 ± 9623.90<br>(17.52 - 271.41) |
| Net Decrease in Popularity | 3175 | 9.49 ± 22.59<br>(8.70 - 10.27) | -0.06 ± 2.27<br>(-0.14 - 0.02) | 114.26 ± 289.59<br>(104.19 - 124.34) | 136.21 ± 4437.97<br>(-18.22 - 290.64) |
| Net Increase in Popularity | 79440 | 63.55 ± 170.51<br>(62.36 - 64.73) | -0.13 ± 3.08<br>(-0.15 - -0.11) | 84.77 ± 157.81<br>(83.67 - 85.87) | 33.52 ± 3182.03<br>(11.39 - 55.65) |

TABLE II
COMPARISON OF COMPOSITE VULNERABILITY CHANGES IN ARTIFACTS AND THEIR IMPACT ON RELEASES.

vulnerability and the net change in popularity. This allows us to assess whether changes in vulnerability follow a similar trend to changes in popularity, or to identify any significant similarities or differences. The net change in popularity is calculated as the sum of the changes in popularity of the library between consecutive artifact releases. A positive change in popularity may indicate effective library maintenance, while a negative change could suggest various factors such as vulnerability issues, infrequent releases, long gaps between updates, or even may be correlated to overall longevity of the library.

Finally, we characterized the effectively maintained libraries based on the constraints set for vulnerability, popularity, frequency of release, and overall longevity of the library. Effective maintenance of a library is thus defined as a state where there is a positive net change in composite vulnerability (primary), a positive net change in popularity, no more than average of one-month gap in library releases, and at least a five-year lifespan.

## IV. DISCUSSION

In this study, we studied evolution all three components of software eco-system: software libraries, its releases, and dependences of releases across years, with special emphasis on rhythm, speed, maintenance and vulnerabilities. We observed growth over the years in terms of frequency, speed, and vulnerabilities until 2021. A study by Decan et al. also had similar observation, increase in dependency network over time (8). Similarly, observing the rhythm of an artifact's lifecycle provides valuable insights into patterns and release speeds, enabling meaningful comparisons (2) . We analyzed the evolution in relation to the development of various agile methodologies to identify any significant correlations or associations.

After 2021,the growth began to plateau, with a decline in the growth of vulnerabilities. Beyond 2021, the period is marked with emergence of artificial intelligence and secure system design in the software development practices. The integration of artificial intelligence with Agile DevOps has promoted adaptive system design to meet the rapidly changing digital space, automating different operation which would otherwise require human input (9). It might have contributed to the dynamics we observed.

Similarly, the frequency of vulnerabilities across all severity levels began to plateau after 2016 and decreased significantly after 2021. This trend suggests advancements in cybersecurity (10), driven by heightened security concerns and the implementation of stronger measures to address them. Notably, from 2021 onward, there has been a significant rise in vulnerability detection and mitigation techniques, particularly those based on machine learning and data mining for automated detection (11), (12). The trend has manifested in maven central dependency ecosystem as well, it seems. However, the top

five most frequent vulnerabilities in the ecosystem were of high severity, suggesting need for more security vigilance and mitigation measures. In addition, only five percent of the total libraries were full maintained based upon the criteria we set for determining maintenance level which is explained in method section. However, ninety percent of the libraries were only partially maintained, which requires additional effort to keep them up to date and mitigate existing vulnerabilities. This, in turn, would enhance their usability and, as a result, increase their popularity. However, distribution of libraries varies with enforcement of different criteria, which is the limitation of this approach, and the study recommend an in-depth study to determine criteria for measuring software maintenance based upon different metric that can be used to detect a library as well maintained or not.

## V. Conclusion

In this study, we present trend and growth of various component of software ecosystem in Maven Central dependency graph. We observed a general trend of growth in frequency, speed, and vulnerabilities until 2021. Beyond 2021, the trend began to plateau, except for the frequency of vulnerabilities of different severity, which started to decrease. This trend may be attributed to rapid advancements in the field of software development, driven by the emergence of new technologies like artificial intelligence, along with increased efforts to improve software security. The fully maintained libraries were just five percent of total libraries, suggest need for more maintained effort among the developer's community. However, the criteria used to determine the maintenance level were not well validated nor derived from standard guidelines, which limits its adoption and implementation. The study suggests the need for further in-depth research on the most prevalent vulnerabilities in the Maven Central ecosystem, as well as the development of metric-based evaluation criteria for the rapid identification of various characteristics of libraries, their releases, and associated dependencies, such as maintenance level, combined vulnerability score, etc.

## VI. Disclaimer Statement

The AI assistant was used to improve the grammar and sentence structure, as well as to derive summaries from the research article. However, AI was not used to generate content or write any part of the paper.

## References


[1] J. Cox, E. Bouwers, M. V. Eekelen, and J. Visser, "Measuring Dependency Freshness in Software Systems," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Florence, Italy: IEEE, May 2015, pp. 109–118. [Online]. Available: http://ieeexplore.ieee.org/document/7202955/

[2] D. Jaime, J. E. Haddad, and P. Poizat, "A Preliminary Study of Rhythm and Speed in the Maven Ecosystem," 2023.

[3] S. Raemaekers, A. Van Deursen, and J. Visser, "The Maven repository dataset of metrics, changes, and dependencies," in *2013 10th Working Conference on Mining Software Repositories (MSR)*. San Francisco, CA, USA: IEEE, May 2013, pp. 221–224. [Online]. Available: http://ieeexplore.ieee.org/document/6624031/

[4] D. Jaime, J. E. Haddad, and P. Poizat, "Navigating and Exploring Software Dependency Graphs using Goblin," 2024.

[5] C. Soto-Valero, A. Benelallam, N. Harrand, O. Barais, and B. Baudry, "The Emergence of Software Diversity in Maven Central," in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. Montreal, QC, Canada: IEEE, May 2019, pp. 333–343. [Online]. Available: https://ieeexplore.ieee.org/document/8816742/

[6] A. Zerouali, T. Mens, G. Robles, and J. M. Gonzalez-Barahona, "On the Diversity of Software Package Popularity Metrics: An Empirical Study of npm," in *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. Hangzhou, China: IEEE, Feb. 2019, pp. 589–593. [Online]. Available: https://ieeexplore.ieee.org/document/8667997/

[7] M. Aggarwal, "A Study of CVSS v4.0: A CVE Scoring System," in *2023 6th International Conference on Contemporary Computing and Informatics (IC3I)*. Gautam Buddha Nagar, India: IEEE, Sep. 2023, pp. 1180–1186.

[8] A. Decan, T. Mens, and P. Grosjean, "An empirical comparison of dependency network evolution in seven software packaging ecosystems," *Empirical Software Engineering*, vol. 24, no. 1, pp. 381–416, Feb. 2019.

[9] Mohan Harish Maturi,, Sai Sravan Meduri,, Hari Gonaygunta,, and Geeta Sandeep Nadella, "A Systematic Literature Review: The Recent Advancements in AI Emerging Technologies and Agile DevOps," *International Meridian Journal*, vol. 2, no. 2, pp. 1–23, 2020.

[10] J. Kaur and K. R. Ramkumar, "The recent trends in cyber security: A review," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 8, pp. 5766–5781, Sep. 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1319157821000203

[11] A. Ghosh, T. O'Connor, and G. McGraw, "An automated approach for identifying potential vulnerabilities in software," in *Proceedings. 1998 IEEE Symposium on Security and Privacy (Cat. No.98CB36186)*. Oakland, CA, USA: IEEE Comput. Soc, 1998, pp. 104–114.

[12] H. Hanif, M. H. N. Md Nasir, M. F. Ab Razak, A. Firdaus, and N. B. Anuar, "The rise of software vulnerability: Taxonomy of software vulnerabilities detection and machine learning approaches," *Journal of Network and Computer Applications*, vol. 179, p. 103009, Apr. 2021.