Graph-Based Analysis of the Maven Central Ecosystem (Open-Source Java Libraries)

***Abstract*—Dependency analysis within large software ecosystems like Maven Central is critical for understanding the structural health and stability of software development. As software grows more reliant on external libraries, developers face challenges in ensuring timely updates, assessing vulnerabilities, and mitigating the risks associated with unmaintained dependencies. The Goblin framework provides a powerful toolset for analyzing these issues at an ecosystem level, offering insights that are crucial for decision-making in both individual projects and broader industry practices. This research aims to uncover trends in ecosystem evolution, identify key risk areas, and propose strategies for improving the resilience and sustainability of software projects.**

I. Introduction

In the ever-changing world of software development, scalable, secure, and maintainable systems depend on efficient dependency management. Package managers like Maven Central enable the development of modern software ecosystems, which are based on complicated networks of interdependencies that connect projects to a tangled hierarchy of transitive and direct dependencies. These links speed up development and encourage code reuse, but they also present problems including dependency sprawl, security flaws, and the possibility that unmaintained libraries will compromise the stability of software.

The Maven Theory, which encourages modular design and the thoughtful use of dependencies to improve software efficiency while reducing redundancy, is incorporated into this study. This methodology has influenced software development procedures over time, directing the transition from isolated libraries to networked, dependency-rich ecosystems. To improve dependency management techniques, a thorough examination of these ecosystems' structural dynamics and evolutionary patterns is necessary as they become larger and more complex. The main objective of the current research is to examine the growth trends, complexity, and maintenance of the dependency graph to understand the evolution of the Maven Central ecosystem. With millions of artifacts and their corresponding dependencies, Maven Central is one of the biggest and most popular repositories for Java libraries. The ecosystem has developed into a very complex web of transitive and direct dependencies over time. For developers, this growing complexity presents many difficulties, such as efficiently managing dependencies, guaranteeing timely updates, and reducing the risks associated with vulnerable or unmaintained packages.

Determining how the ecosystem evolved is a major goal of this analysis. Specifically, it targets the frequency and rhythm of library releases, the incidence of unmaintained libraries, and whether libraries currently depend on more dependencies than they did in the past. These elements are essential for comprehending how software development practices—like agile methodologies—interact and affect release cycles and ecosystem dependency management. Additionally, the study seeks to clarify the difficulties in managing unmaintained dependencies, which are essential to preserving software projects' stability, security, and sustainability. To this end, we answer the following research questions (RQs):

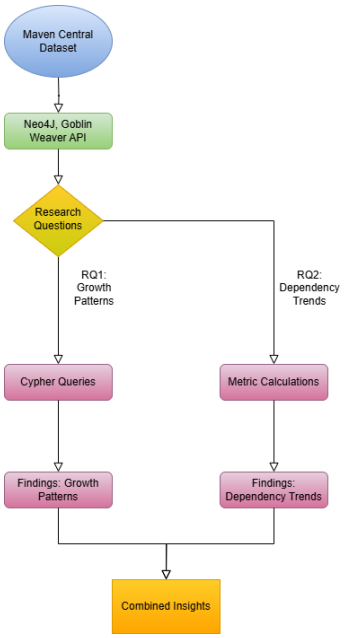

Fig. 1. Approach Overview Caption

- **RQ1: What are the patterns in the growth of the Maven Central graph across different time periods?** This question aims to uncover long-term trends in library dependency usage and management practices.
- **RQ2: Do libraries tend to use more dependencies than in the past?** Our goal is to analyze data spanning from 2005 to 2024 to provide insights into how the Maven Central ecosystem has evolved.

II. Study Design

The dataset utilized in this research is derived from the Maven Central ecosystem and provides a comprehensive view of its dependency graph. It comprises 15,117,217 nodes, which include:

– 658,078 libraries representing distinct software artifacts, and

– 14,459,139 releases corresponding to the various versions of these libraries.

The graph further contains 134,119,545 edges, categorized as:

– 119,660,406 dependency edges, which define the relationships between libraries and their dependencies, and
– 14,459,139 versioning edges, which link libraries to their respective releases.

This dataset serves as a foundation for studying ecosystem evolution, enabling insights into the structural and temporal dynamics of dependencies within Maven Central.

A. Central Dependency Graph Database for Maven

Ecosystem analysis is made easier by the Maven Central dependency graph's structure, which represents libraries and their relationships in a thorough manner. Two node types and two edge types make up the graph, and each has a distinct function in modeling the ecosystem.

1) Node Types

1) Artifact Nodes (Library-Level):
   - Maven ID (`group.artifact`, or `g.a`)
   - Boolean `found`: Indicates whether the artifact is available in the Maven Central repository. Even if not found, these nodes provide dependency information for artifacts outside the ecosystem (e.g., `compile`, `test`, etc.)
2) Release Nodes (Version-Level):
   - Maven ID (`group.artifact.version`, or `g.a.v`)
   - Release timestamp: Specifies the date and time of the release.
   - Version information: Tracks the specific version of the library.

2) Edge Type

1) Dependency Edges:
   - Source: Release nodes.
   - Target: Artifact nodes.
   - Attributes:
     – Target version: Specifies a version or version range of the dependency.
     – Scope: Defines the purpose of the dependency (e.g., `compile`, `test`, etc.).
2) Versioning Edges:
   - Source: Artifact nodes.
   - Target: Release nodes.
   - Type: `relationship_AR`, representing the link between a library and its individual releases.

B. Weaver API for Dependency Graph Analysis

Maven Central's dependency graph can be queried and enhanced with the Weaver API, a RESTful service that provides flexibility for specialized analytical requirements. Because of its extensibility, researchers can manage and add their own metrics or derived data.

1) Features

- Permanent Added Values - To keep computed metrics (such as "CVE," "Freshness," "Popularity," and "Speed"), Weaver uses dynamic AddedValue nodes. To maintain consistency, these nodes are updated or recalculated if the underlying datasets change.
- Extensibility - Researchers can establish novel metrics by:
  – Adding something to the AddedValueEnum list.
  – Implementing the necessary logic and extending *AbstractAddedValue.*
  – Using the *AggregateValue* interface to create aggregated metrics.
- Dynamic API Management - Endpoints make it possible to add, edit, or remove metrics and automate cleanup following database changes to preserve the accuracy of the data.

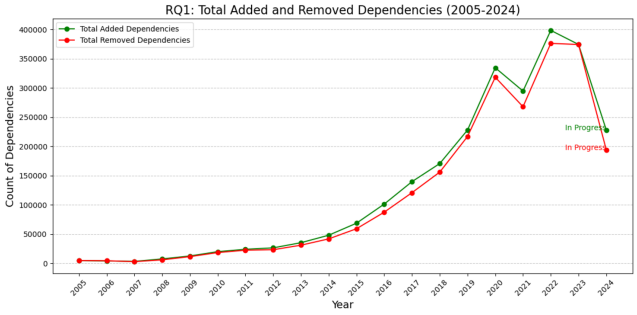

Fig. 2. Growth Patterns Over Time

C. Significance for Ecosystem Research

This dataset creates a strong framework for examining the development of software ecosystems, especially when paired with the Weaver API's capabilities. Researchers can investigate important topics including dependence growth patterns, maintenance procedures, and ecosystem resilience by utilizing graph-theoretical models and the flexibility of the Weaver API. In addition to facilitating in-depth empirical research, this method offers practical advice on enhancing dependency management and maintaining the integrity of software development ecosystems. This framework improves the Maven Central dependency graph's resilience and streamlines analysis for sophisticated ecosystem study.

III. Experimental Results

A. RQ1: What are the patterns in the growth of the Maven Central graph across different time periods?

By examining its architectural evolution, this research question looks into trends in the Maven Central dependency graph's expansion over time. The goal of the study is to identify patterns in dependence generation, acceptance, and fading over time by looking at changes in the network, such as the addition of new nodes (artifacts) and edges (dependencies). This analysis sheds light on changes in methods of development, the dynamics of library popularity, and the ecosystem's scalability.

Motivation. To address this research question, the study investigates how the development of the ecosystem is impacted by outside variables, such as changing software trends and project management techniques. Understanding these trends provides a thorough understanding of how developers deal with the difficulties of managing dependencies in a vast ecosystem.

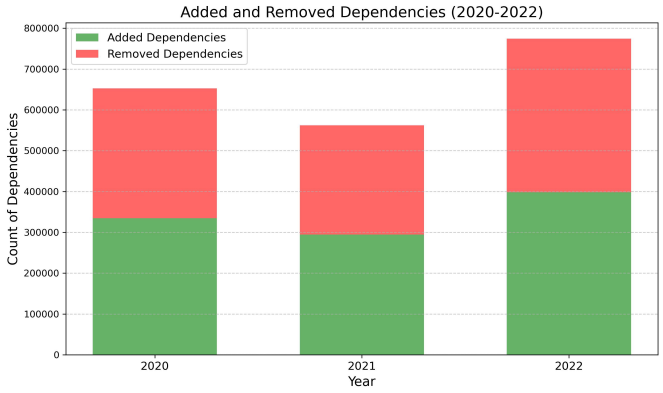

Fig. 3. Impact of Global Disruptions on Dependency Changes (2020–2022) - a

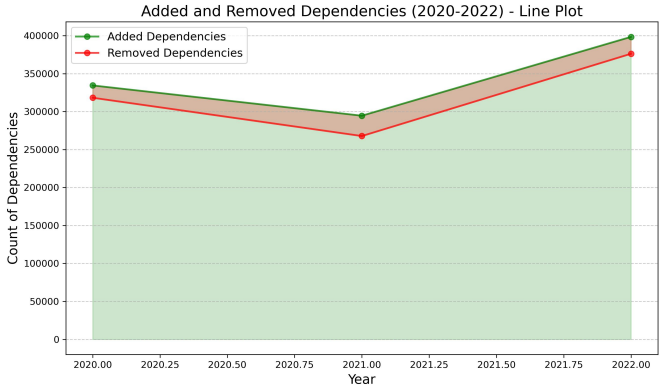

Fig. 4. Impact of Global Disruptions on Dependency Changes (2020–2022) - b

Results. Analyzing the Maven Central ecosystem over the previous 20 years (2005–2024) shows clear trends in dependency utilization and growth that are impacted by changing software development goals and world events. The following are the main conclusions:

- **2021: Significant Decline in Dependency Activity:** A sharp decline in both the addition and removal of dependencies was observed in 2021. Developers and software teams shifted their focus from active development to maintaining existing systems, prioritizing stability over feature development. Budget constraints, resource limitations, and workflow disruptions led to reduced activity in dependency updates. Many teams avoided making disruptive changes and instead concentrated on stabilizing their ecosystems during this uncertain period.

These results demonstrate the influence of outside variables on the development and evolution of the Maven Central graph, including international crises. They emphasize how crucial it is to comprehend dependency patterns in order to overcome obstacles in the upkeep and development of software ecosystems.

1) Key Observations from Figure 2, Figure 3 and Figure 4

1) The number of dependency additions increased steadily and peaked about 2020.
2) Active dependency management techniques were indicated by the identical rising trend seen in removed dependencies.
3) Activity briefly decreased in 2021 before showing indications of resurgence in 2024.

The results of the investigation show that the overall number of dependencies of the Maven Central ecosystem has been steadily increasing over time. This steady expansion is a reflection of the ecosystem's organic development, which is fueled by a growing emphasis on modular designs and reusable components to reduce redundancy and increase development efficiency.

Long-term trends also show that dependency modifications became more frequent as Maven Central developed, highlighting the ecosystem's reliance on reused artifacts and third-party modules. Temporary decreases, as those shown in 2021, highlight the impact of outside variables. These results demonstrate the Maven Central ecosystem's dynamic character and its adaptability to external and technical stresses.

2) Conclusion:

The Maven Central ecosystem's growth patterns show a consistent and long-term expansion interspersed with periods of adaptation and recuperation. Dependency management techniques have changed dramatically over time, mirroring the ecosystem's expanding complexity and the increased focus on efficiency, security, and stability. These patterns demonstrate how the ecosystem is dynamic, adjusting to new developments in technology and outside threats while continuing to grow steadily.

B. RQ2: Do libraries tend to use more dependencies than in the past?

The study intends to identify changes in development processes, such as the use of modular designs and reusable components, by examining patterns in dependency counts across various libraries. Reducing redundancy, increasing efficiency, and simplifying software development are all made possible by dependency management, according to the Maven theory. Knowing the evolution of dependency usage offers important insights into how libraries adjust to shifting development paradigms and increasing ecosystem demands.

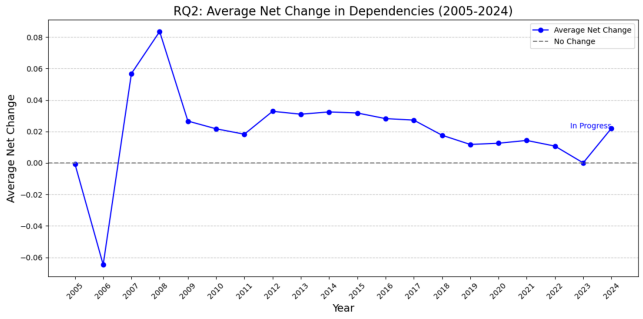

Fig. 5. Dependency Usage Trends

Motivation. Whether libraries in the Maven Central ecosystem have demonstrated a growing propensity to employ more dependencies over time is the subject of this research topic. The possible trade-offs linked to larger dependency consumption are also examined in this analysis, including increased complexity and reliance on external objects. The results advance our knowledge of ecosystem evolution and dependencies in contemporary software development.

Results. Our results show that, for most years, the average net change in dependence counts has been consistently positive, suggesting that libraries are becoming more dependent on external dependencies. This pattern demonstrates how modularization and dependency management techniques have become more widely used as the ecosystem has developed. Dependency trends stabilized between 2010 and 2020, indicating the ecosystem's transition to maturity and equilibrium. The following are the main conclusions:

- **2006: Initial Decline:** The Maven Central ecosystem was still in its infancy in 2006, and many libraries were either stand-alone projects or had few dependencies. Dependency management and modularization were not yet commonplace techniques. A "clean slate" era, in which libraries were mainly independent and superfluous dependencies were eliminated during early development attempts, is reflected in the period's notable negative net change.
- **2007: Steep Increase:** A significant turning point in the ecosystem's history was reached in 2006, when net dependency change sharply increased. This change occurred at the same time that Maven, a dependency management tool, became more widely used, encouraging libraries to include external dependencies and reusable modules in their projects. Software development quickly became modularized at this time, indicating the shift to a networked ecosystem where libraries started using third-party components to improve functionality and expedite development.

These results highlight how Maven Central has revolutionized dependency strategies and the overall development of software ecosystems over time.

1) Key Observations from Fig 2

1) Over the majority of years, the average net change in dependencies has been continuously positive, suggesting a tendency toward a greater dependence on external libraries.
2) The dependency graph's relative stabilization between 2010 and 2020 indicates that the ecosystem matured, slowing growth and stabilizing dependencies.
3) A significant drop in the net change of dependencies in 2022 might be the result of increased focus on dependency cleanliness, perhaps in reaction to security flaws and the growing significance of upholding safer, more effective dependency management procedures.

2) Pattern Analysis:

Although there are discernible variations in the net change in dependency over time, a distinct trend of stabilizing appears in the latter years. This suggests a move away from prior times of major change, when developers were adopting contemporary dependency management techniques (most notably in the early years of 2005–2006), and toward a more consistent and systematic approach to dependency management. Libraries' dependence utilization is now changing steadily rather than drastically. A useful indicator of how the Maven Central ecosystem's use of external dependencies has changed is the average net change in dependencies per library. This indicator shows the general pattern of libraries either becoming more dependent on dependencies over time or becoming less dependent. By monitoring these changes, one can determine whether library ecosystems are becoming more complicated or simpler, providing a better understanding of how dependency management techniques have developed over time.

3) Conclusion:

A useful indicator of how the Maven Central ecosystem's use of external dependencies has changed is the average net change in dependencies per library. This indicator shows the general pattern of libraries either becoming more dependent on dependencies over time or less dependent. By monitoring these changes, one can determine whether library ecosystems are becoming more complicated or simpler, providing a better understanding of how dependency management techniques have developed over time.

REFERENCES

1) **Cossette, B., & Walker, R. J. (2012).** "Seeking the ground truth: A retroactive study on the evolution and migration of software libraries." *Proceedings of the 20th ACM SIGSOFT International Symposium on the Foundations of Software Engineering*, pp. 1–11.
2) **Decan, A., Mens, T., & Claes, M. (2017).** "An empirical comparison of dependency network evolution in seven software packaging ecosystems." *Empirical Software Engineering*, vol. 22, no. 6, pp. 2942–2973.
3) **Gkortzis, A., & Spinellis, D. (2021).** "The landscape of programming languages and ecosystem evolution: The case of Java." *IEEE Transactions on Software Engineering*, vol. 47, no. 12, pp. 2724–2740.
4) **Valiev, M., Mens, T., & Goeminne, M. (2018).** "A quantitative analysis of dependencies in open-source software." *Proceedings of the 15th International Conference on Mining Software Repositories (MSR)*, pp. 209–219.
5) **Di Cosmo, R., Jeannerod, C., & Zacchiroli, S. (2017).** "Measuring and analyzing software dependencies." *Empirical Software Engineering*, vol. 22, no. 5, pp. 2647–2674.
6) **da Mota Silveira Neto, P. A., et al. (2022).** "A deep dive into the impact of COVID-19 on software development." *IEEE Transactions on Software Engineering*, vol. 48, no. 9, pp. 3342–3360, doi: 10.1109/TSE.2021.3088759.