Intertwining Ecosystems: A Large Scale Empirical Study of Libraries that Cross Software Ecosystems

Kanchanok Kannee, Supatsara Wattanakriengkrai, Ruksit Rojpaisarnkit, Raula Gaikovina Kula, Kenichi Matsumoto

Nara Institute of Science and Technology, Japan

Abstract
An increase in diverse technology stacks and third-party library usage has led developers to inevitably switch technologies. To assist these developers, maintainers have started to release their libraries to multiple technologies, i.e., a cross-ecosystem library. Our goal is to explore the extent to which these cross-ecosystem libraries are intertwined between ecosystems. We perform a large-scale empirical study of 1.1 million libraries from five different software ecosystems, i.e., PyPI for Python, CRAN for R, Maven for Java, RubyGems for Ruby, and NPM for JavaScript to identify 4,146 GitHub projects that release libraries to these five ecosystems. Analyzing their contributions, we first find that a significant majority (median of 37.5%) of contributors of these cross-ecosystem libraries come from a single ecosystem, while also receiving a significant portion of contributions (median of 24.06%) from outside their target ecosystems. We also find that a cross-ecosystem library is written using multiple programming languages. Specifically, three (i.e., PyPI, CRAN, RubyGems) out of the five ecosystems has the majority of source code is written using languages not specific to that ecosystem. As ecosystems become intertwined, this opens up new avenues for research, such as whether or not cross-ecosystem libraries will solve the search for replacement libraries, or how these libraries fit within each ecosystem just to name a few.

1. Introduction and Motivation

Popular use of third-party libraries has become prominent in contemporary software engineering \(\mathbb{I}\), which is evident by the emergence of different library repositories like NPM, PyPI, CRAN, Maven, and so on. These massive repositories also depend on each other, thus forming a complex software ecosystem of dependencies, across these different technology stacks.
For various reasons, a developer may realize that a library used in their applications requires replacement. One reason may be due to the availability of newer versions that fix defects, patch vulnerabilities, and enhance features. In such cases, the developer seeks an appropriate replacement and has lead to various efforts in library recommendation [2]. Another case is when a developer is forced to switch programming languages. With an increase of the diversity in technology stacks and third-party library usage [3], developers eventually will face the need to switch a programming language and their subsequent libraries specific to that language [4, 5, 6]. In this case, developers would have to search for a replacement library that they are familiar with.

In the case of switching technologies (i.e., programming languages), maintainers now provide an option by releasing a version specific for that ecosystem. We refer to this phenomenon as a cross-ecosystem library. Figure 1 depicts a
cross-ecosystem library that serves two different ecosystems (cf. Figure 1b and Figure 1c). The library is Bokeh which is a popular interactive visualization library for modern web browsers. Figure 1 shows the official GitHub repository that hosts the common repository, while there are two versions of the library hosted on both the official PyPI and NPM registry. In 2022, the Python release of Bokeh had over 2.94K dependent repositories, and 590 other libraries in the ecosystem that rely on this library. It has 120 releases and was first released on October 25th, 2013. The GitHub repository that hosts the cross-ecosystem library is mainly implemented in Python (i.e., 56.7%) and TypeScript (i.e., 41.1%). According to its homepage, BokehJS is written primarily in TypeScript and not in JavaScript. Furthermore, Bokeh attracts has 603 contributors to its GitHub repository.

Our goal is to explore the extent to which these cross-ecosystem libraries are intertwined within these ecosystems. By studying developer contributions and programming language diversity, we set out to investigate the extent to which cross-ecosystem libraries by mining five of the most popular and widely adopted library ecosystems (i.e., CRAN, Maven, PyPI, RubyGems, and NPM). Our study is a large-scale quantitative analysis of 1,110,059 libraries to identify 4,146 cross-ecosystem library with 567,864 contributors. After preliminary results showed that cross-ecosystem libraries are being depended upon, we formulated two research questions that analyze cross-ecosystem library repository on GitHub:

• (RQ1) What percent of contributors to a cross-ecosystem library repository are from different ecosystems?

Motivation: Prior work [7] shows that the ecosystem plays an important role in the sustained activities of a software project. Hence, we would like to test the assumption that a cross-ecosystem libraries may require involvement from a target ecosystem.

Results: A significant majority (median of 37.5%) of contributors come from a single ecosystem instead of from both ecosystems. Interestingly, we find there is a significant portion of contributors do not belong to any ecosystem (median of 24.06%).

• (RQ2) How diverse are the programming languages for a cross-ecosystem library repository?

Motivation: To complement RQ1, the motivation for RQ2 is to specifically analyze which programming languages are implemented in a cross-ecosystem library. As shown in the motivating example, some libraries are only front-end implementation. Hence, we would like to understand the extent to which these libraries are implemented for the different programming languages that serve the ecosystem (e.g., NPM for

https://github.com/bokeh/bokeh
https://pypi.org/project/bokeh/
https://www.npmjs.com/package/@bokeh/bokehjs
Table 1: Overview of dataset for the preliminary analysis.

| Lib. Releases | As of 12 Jan 2020 |
|---------------|------------------|
|               | with GitHub Repo URL | initial libraries |
| NPM           | 818,787          | 2,357,829         |
| PyPI          | 138,001          | 420,350           |
| CRAN          | 5,551            | 21,526            |
| Maven         | 36,762           | 456,756           |
| RubyGems      | 110,958          | 176,987           |
| **Total**     | **1,110,023**    | **3,433,448**     |
| # cross-ecosystem library pairs | 4,146 |

Results: Three (PyPI, CRAN, RubyGems) out of the five ecosystems show the majority of the source code is written using languages that are not specific to an ecosystem (i.e., Python, Java, JavaScript, R, and Ruby).

In addition to the insights revealed by the study, our contributions are two-fold. The first contribution is a large quantitative study that covers over 1.1 million libraries, 0.5 million contributors from five different software ecosystems. The second contribution is a replication package with all data and scripts that contains all the algorithms used in the study. Our preliminary appendix is available at [https://zenodo.org/record/6983864#.YvVmGuxBy3I](https://zenodo.org/record/6983864#.YvVmGuxBy3I).

2. Preliminary Analysis

To motivate the importance of a cross-ecosystem library on the ecosystem, we performed a preliminary study to answer the question *(P1)* How dependent is the ecosystem to a cross-ecosystem library release?

Data Collection. We selected five popular and well-studied software ecosystems. NPM is a package manager for the JavaScript programming language that was recently purchased by Microsoft via GitHub. PyPI is the library ecosystem that serves the Python programming language, which is interpreted high-level general-purpose programming language. CRAN is the library ecosystem that serves the R programming language, which is a free software environment for statistical computing and graphics. Maven is the library ecosystem that serves the Java programming language, which is a general-purpose programming language that follows the object-oriented programming paradigm and can be used for desktop, web, mobile, and enterprise applications. RubyGems is the library
ecosystem that serves the Ruby programming language, which is a dynamic, open-source programming functional programming language with a focus on simplicity and productivity.\footnote{https://www.ruby-lang.org/en/}

To identify a cross-ecosystem library, we use the GitHub library repository as the linking heuristic. For instance, as shown in our motivating example, the GitHub library repository URL serves as a link between two analogical libraries. Our assumption is that GitHub should be the common platform by which cross-ecosystem libraries may decide to host their libraries. We also use this filter as a quality check, to ensure that library quality.

Similar to prior work \cite{8, 9, 10}, we then queried the libraries.io dataset for library ecosystems that indeed listed a GitHub repository URL as their library repository. For each library ecosystem, we collect a list of all libraries from the Libraries.io dataset. We mined the dataset version (1.6.0)\footnote{https://libraries.io/}. Once we were able to collect the list of libraries that hosted their library repository on GitHub, we then proceed to cross-reference between libraries that are hosted in different library ecosystems. The results are shown in Table \ref{tab:cross-ecosystem-libraries}, where we extracted a total of 1,110,023 from 3,433,448 libraries. As shown in Figure \ref{fig:cross-ecosystem-diagram} we noticed that the cross-ecosystem library is usually comprised in pairs. In other words, we identify a cross-ecosystem library as having releases to two ecosystems (e.g.,

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{cross-ecosystem-diagram.png}
\caption{Identifying the 4,146 cross-ecosystem libraries from the five ecosystems. For example, we identified 22 libraries across CRAN and PyPI.}
\end{figure}

\footnote{https://libraries.io/}

\footnote{https://zenodo.org/record/3626071#.YdRu_hNBzFo}
Table 2: Summary statistics of #Dependent showing statistical difference including significance where cross-ecosystem is more dependent than regular libraries.

| cross-ecosystem pair | # Dependent (regular) | Mean   | Median | Significance |
|----------------------|------------------------|--------|--------|--------------|
| NPM ∩ PyPI           | 2664.71(761.39)        | 53(1)  | * S    |
| PyPI                 | 405.35(29.43)          | 26(0)  | -      |
| NPM ∩ Maven          | 2884.94(761.39)        | 71(1)  | -      |
| Maven                | 75.06(59.35)           | 5(0)   | * N    |
| NPM ∩ RubyGems       | 4003.08(761.39)        | 19(1)  | -      |
| RubyGems             | 2291.94(335.90)        | 16(0)  | -      |
| CRAN ∩ PyPI          | 21.43(19)              | 12(0)  | -      |
| Maven ∩ PyPI         | 94.21(59.35)           | 20(0)  | -      |
| PyPI                 | 91.42(29.43)           | 23(0)  | -      |
| Maven ∩ RubyGems     | 269.47(59.35)          | 34(0)  | * S    |
| RubyGems             | 1104.91(335.90)        | 56(0)  | * S    |
| PyPI ∩ RubyGems      | 132.66(29.43)          | 7(0)   | * S    |
|                      | 569.63(335.90)         | 8(0)   | * S    |

The effect sizes level: N(negligible), and S/small

*: p-value < 0.05

NPM ∩ PyPI).

Approach. To answer this preliminary question, we collected dependent metrics, which is the number of other libraries that declare the library as a dependency. Note that for the analysis, we compare against a pair of ecosystems (i.e., NPM ∩ PyPI). For example, Font-Awesome[^10] is one of the top open-source libraries on GitHub. The library has a dependency score for both the PyPI (i.e., 29 dependents) and Maven (i.e., 22 dependents). We used the libraries.io dataset to append to our existing dataset.

For evaluation, we set up an experiment to compare the cross-ecosystem library against other regular libraries. As a result, we will report the statistical summary (i.e., mean, median, and standard deviation). Furthermore, we statistically validate our results using the Mann-Whitney U test[^11],[^12] which is a non-parametric statistical test. To show the power of differences between metrics from cross-ecosystem libraries and regular libraries, we investigate the effect size using Cliff’s δ, which is a non-parametric effect size measure[^13]. The interpretation of Cliff’s δ is shown as follows: (1) δ < 0.147 as Negligible, (2) 0.147 ≤ δ < 0.33 as Small, (3) 0.33 ≤ δ < 0.474 as Medium, or (4) 0.474 ≤ δ as Large. To analyze Cliff’s δ, we use the cliffsDelta package[^11].

[^10]: https://github.com/fortawesome/Font-Awesome
[^11]: https://github.com/neilernst/cliffsDelta
Table 3: Summary statistics of appended data to the 1,110,023 libraries to answer our RQs.

| RQ   | Median | Max  | Total   |
|------|--------|------|---------|
| RQ1 -# Contrib. per cross-ecosystem lib. | 8      | 1,727 | 49,674  |
|      | per lib. | 4    | 16,606  | 567,864 |
| RQ2 -# Lang. per lib.     | 2      | 44   | 120     |

Results. Table 2 shows the evidence that the cross-ecosystem library influences ecosystems. We find that the cross-ecosystem library has more dependents than the regular library for every pair of cross-ecosystem libraries, e.g., the median of the number of dependents for the cross-ecosystem library that was released to the NPM and PyPI ecosystem: 53 dependent < 1 dependents. Statistically, we find that the number of dependents between cross-ecosystem and regular libraries are significantly different (p-value < 0.05), with a negligible to small association as the reported effect size. Furthermore, as shown in Table 2, we show that cross-ecosystem library combinations that include RubyGems, which is the pair of Maven with RubyGems and PyPI with RubyGems had a significantly larger number of dependents than regular libraries.

Summary: We find that cross-ecosystem libraries are relied upon more when compared to regular libraries, especially for Maven, RubyGems, and PyPI.

3. Findings

Motivated by the preliminary results, which show evidence that cross-ecosystem libraries are relied upon by the ecosystem, we now continue to answer each research question that relates to their contributions and diversity of programming language composition.

3.1. Data Preparation

As shown in Table 3, extending our dataset from the preliminary study, we mined and collected contributor information (RQ1) and programming language (RQ2). It is important to note that RQ1 analyzes all cross-ecosystem libraries against regular libraries, while RQ2 only analyzes the cross-ecosystem libraries.

To answer RQ1 and RQ2, we use the GitHub API to collect all contributors that made commits to all repositories. We use the API request https://api.github.com/repos/{owner}/{repo}/commits to collect all the commit information.

After collecting all contributors for each library, we then merged the contributors’ list based on the ecosystem, so that we have a merged listing of contributors. For RQ2, we use the GitHub API to collect the lines of code that each programming language. We use the API command https://api.github.com/repos/{owner}/{repo}/languages to retrieve all
we show how contributions for a cross-ecosystem library originate from a single ecosystem.

these informations. This process was extremely time-consuming, taking almost two months to slowly collect all this information.

3.2. Distribution of Contributors (RQ1)

Approach. To answer RQ1, we investigate whether a cross-ecosystem library attracts contributions from the targeted ecosystems. Taking our example shown in Figure 1, we would like to understand the extent to which both the PyPI and NPM ecosystems support the GitHub Bokeh repository. In terms of our motivating example, we calculate the percentage of the 533 contributors from the Bokeh repository who have also made contributions to a regular PyPI or NPM repository. For data collection, we first collected all contributors from the 1,110,059 projects. For our analysis, we calculate a percentage of contributors from each cross-ecosystem library into these three classifications:

- **Both (%)** Percentage of contributors that have made prior contributions to libraries that belong to both ecosystems.
- **Single (%)** Percentage of contributors that have made prior contributions to a single ecosystem.
- **Independent (%)** Percentage of contributors who do not have any contributions to any ecosystem libraries.
Table 4: Statistical significance test results related to RQ1.

| Contributions (%) | Single-Both * | Both-Neither * |
|-------------------|---------------|-----------------|
| S                 | -             | N               |

The effect sizes level: small(S) and negligible(N)

*: p-value < 0.05

Since we are dealing with a large dataset, we used the PySpark arrays to classify and merge contributors into these three groups. Hence, for a cross-ecosystem library, we calculate these ratios with the hypothesis that most contributors would come from a single ecosystem.

To statistically confirm our classification of contributors, we use McNemar’s Chi-Square test [14]. This is a non-parametric statistical test to use to find the change in proportion for paired data and to find the change in proportion for our paired data. We test the null hypothesis that ‘the percentage of contributions classified is the same’. We also measure the effect size using Cohen’s $d$, a non-parametric effect size measure by Cohen [15]. Effect size is analyzed as follows: (1) $d < 0.2$ as Negligible, (2) $0.2 \leq d < 0.5$ as Small, (3) $0.5 \leq d < 0.8$ as Medium, or (4) $0.8 \leq d$ as Large.

Most contributors originate from a single ecosystem. From Figure 3 we make two observations. First, we find that there is a higher percentage of contribution (a median of 37.5%) that is supported by only one ecosystem. (see Figure 3). It is important to note that our analysis does not identify the originating ecosystem.

The second observation is that a significant portion of contributors (median of 24.06%) that do not belong to either ecosystem. The result shows that these contributors may be just specific to the project itself. This is more than contributors that belong to both ecosystems (median of 20%). The result indicates that these libraries may need to have their own base of contributors in order to sustain maintenance activities.

In terms of statistical significance, Table 4 shows evidence that the null hypothesis on whether ‘percentage of contributions from three types are the same’ is rejected. We see significant differences (p-value < 0.05) where a library that has a higher percentage of single ecosystem contributors will have a lower percentage of contributors from both ecosystems. Furthermore, libraries with a higher percentage of independent contributors will have fewer contributors from both ecosystems.

[12] https://spark.apache.org/docs/latest/api/python/
3.3. Programming Language Diversity (RQ2)

**Approach.** To answer RQ2, we collect and analyze the proportion of programming languages used in a cross-ecosystem library. Using our running example, we would like to understand the extent to which the programming language implemented to correspond to the software ecosystem. For example, we expect a PyPI library like Bokeh should have a higher proportion of its code written in Python. For our analysis, we compare the percentage of the programming language against the five programming languages that correspond with each ecosystem. This is Python for PyPI, Java for Maven, JavaScript, R for CRAN, and Ruby for RubyGems. Finally, we grouped all other languages as others.

A cross-ecosystem library is written using multiple programming languages. Figure 4 shows evidence that a majority of code written for three out of the five ecosystems is from other languages. Specifically for PyPI it is 40.45%,
Table 5: A breakdown of the top five languages for the Others category in Figure 4.

| Other | Percent (%) |
|-------|-------------|
| HTML  | 63.75       |
| Shell | 23.92       |
| CSS   | 10.17       |
| Makefile | 2.15    |
| C, C++, C#, PHP, TypeScript | 0.01 |
| Total | 100.00      |

Table 6: Contingency Table that shows statistical significance test results that relate to RQ2.

| Lines of Code (%) | Python | Java | JavaScript | R | Ruby |
|-------------------|--------|------|------------|---|------|
| NPM               | *L     | *L   | *N         | *L| *L   |
| PyPI              | *M     | *L   | *M         | *L| *L   |
| CRAN              | *L     | *L   | -          | *L| *L   |
| Maven             | -      | -    | *N         | - | -    |
| RubyGems          | *L     | *L   | *S         | *L| *S   |

The effect sizes level: large(L), medium(M), small(S) and negligible(N)

*: p-value < 0.05

77.12% for cross-ecosystem library that serves the CRAN ecosystem and 20.74% from RubyGem libraries. The second related finding is that JavaScript is a primary programming language, especially for NPM (85.01%), Maven (94.45%), and RubyGem (7.85%) libraries. Table 5 shows a deeper analysis of the other languages shown in Figure 4. We can see that many of the code for a cross-ecosystem library is written in HTML, which could be accounted for either documentation and other kinds of meta files that are needed in a library. This result suggests that a significant portion of a cross-ecosystem library is for documentation and other configuration files (Shell, CSS, and makefiles).

Table 6 shows the statistical results, where the null hypothesis of whether ‘cross-ecosystem library written in corresponding programming languages’ is rejected. We find each ecosystem is significant and mostly uses other languages besides programming languages that serve the ecosystem. Additionally, we find that NPM and Maven have significantly used JavaScript more than other languages.

Summary: Three (PyPI, CRAN, RubyGems) out of the five ecosystems show the majority of the source code is written using languages that are not specific to an ecosystem (i.e., Python, Java, JavaScript, R, and Ruby).

4. Implications

Based on our findings, we now discuss implications and present possible research directions for each stakeholder.
For Library Maintainers. Should maintainers consider releasing to different ecosystems? The good news that is highlighted in the preliminary study is that cross-ecosystem libraries are already relied upon by the different ecosystems. For maintainers that are considering to release to multiple ecosystems, they should ensure that already have an existing community of contributors that continue to support them. This is because RQ1 indicates that most cross-ecosystem libraries are not likely to receive contributions from both ecosystems. On the other hand, another interesting result in RQ1 was that a cross-ecosystem library has contributions that are independent of the ecosystem. In terms of programming language, we show that a cross-ecosystem library is written in a diverse set of programming languages, sometimes not related to the target ecosystem. Hence, these maintainers should be competent with multiple programming languages. Furthermore, we see that our research opens up new research directions, as we are not sure of the motivations why maintainers decide to support multiple ecosystems, and whether or not they see this phenomenon as being beneficial in terms of solving the problem by replacement libraries, and attracting both new contributors or users. We consider all these are future work.

For Library Users and Contributors. Will cross-ecosystem libraries solve the need to find replacement libraries? From the results of the preliminary study, we can conclude that these libraries that decided to release to multiple ecosystems have an already established user base of clients in the original ecosystem. This result might suggest that the libraries are quality candidates for replacement, and maybe users are more comfortable using a library that comes from the same maintainers of a library that they are already familiar with. In terms of RQ1, since the results show that these libraries still rely heavily on contributions from the original ecosystem, contributors might consider supporting these libraries, this might also include learning a new programming language, as shown in RQ2. The results do suggest that with the increase in technology stacks and languages, contributors, users and maintainers may need to become proficient in multiple programming languages. However, our results do not report that clients are indeed choosing a cross-ecosystem library, so this would have to be future work. Another interesting avenue is how these libraries will compete with already established libraries in the new ecosystem that provide the same functionalities.

For Software Ecosystems and Researchers. How will cross-ecosystem libraries impact ecosystem-level topics like governance, and management?. From a research perspective, the growing intertwining between different ecosystems will bring forth interesting implications at the ecosystem level. For instance, to what extent do these libraries abide by the specific rules, and regulations that are enforced by each ecosystem? and how are bug fixes and specifically security vulnerabilities propagated through different ecosystems? Since RQ1 states that their libraries are more likely to receive contributions from the original ecosystem, does that mean that the boundaries between these two ecosystems become
closer? Also, RQ2 states that since these libraries are composed of different programming languages, does this mean that software ecosystems do not need to be defined by programming language, and how do very specific communities like the ‘Pythonic’ culture of the PyPI community react to such statements? All these questions should have implications and open up new potential research directions for the software engineering community.

5. Threats to Validity

In this section, we outline the threats to the validity of our study.

Construct Validity - A key threat in the construct validity exists in the matching approach using the GitHub URL cross-ecosystem library. Although we do miss out on other cross-libraries that may exist between two ecosystems, except for CRAN, the GitHub platform is the most central repository as evident by their popularity for PyPI, Maven, and RubyGems. Also, due to this heuristic, we only are able to detect cross-ecosystem libraries hosted on GitHub, which might be a bias against ecosystems that might not share their projects on GitHub. Since GitHub is one of the largest platform of Open Source Software, especially for the NPM ecosystem, we are confident of our results.

Internal Validity - We discuss two threats to internal validity. The first threat is in regard to the tool selection for the statistical analysis. By using different tools and techniques, results may be different, e.g., statistical testing. However, we are confident that the threat is minimum since all employed tools and techniques are used in prior works and are well-known to the software engineering community. We are also aware that our results are dependent on the dataset. However, by using two different reliable data sources (libraries.io, and GitHub API), we are confident in our results.

External Validity - The results of our analyses are limited to the five ecosystems: NPM, PyPI, CRAN, Maven, and RubyGems, which threaten the generalization of our claims to other ecosystems. Still, we believe that our analyses could easily be applied to other ecosystems and are seen as immediate future work.

6. Related work

In this section, we discuss key ideas from related works and how our work aligns and complements the existing literature.

On Analogical Libraries and Library Recommendation These studies relate to the analogical libraries from various ecosystems. Cossette and Walker [2], highlighted that analogical library recommendation tools should beware of false predictions as the consequences are serious. Ouni et al. [16] proposed an approach to recommend libraries by using a search base algorithm. They suggested that the history of library usage and semantics of each library can be used to improve the efficiency of the recommendation technique. Chen et al. [5] presented a new approach to recommend analogical libraries from tags of millions
of Stack Overflow questions by using the word embedding technique. From the
study by Teyton et al. [17], they identified sets of analogical libraries by using
library migration graphs that show how existing libraries have performed mi-
grations among them. These migration graphs allow developers to discover and
select library replacements easier. Studies such as [18, 19] created a library rec-
ommendation system in order to developers capture unaware suitable libraries
based on the set of used dependencies in the project. Chen et al. [20] created
a knowledge base of analogical libraries which mined from tags of millions of
Stack Overflow questions. They then implemented a proof-of-concept web ap-
lication to recommend an analogical library. Several studies focused on helping
developers to migrate the libraries and API of their project to new ecosystems.
Zhong et al. [21] presented a tool that maps API from Java to C# called MAM
(i.e., Mining API Mapping). This tool can reduce the errors (e.g., compilation
error, defect) during the library migration to the new ecosystem. Teyton et al.
[6] automatically extracted function mappings from the analysis of software li-
braries that previously performed the library migration. Velázquez-Rodríguez
and De Roover [22] proposes MUTAMA, a multi-label classification approach
to the Maven library tagging problem based on information extracted from the
byte code of each library. In our work, we perform an empirical study on the
cross-ecosystem library which is a library that is cross-cutting different library
ecosystems. Prior works, instead, focus on libraries from the same ecosystem.

On Software Categorization These studies focus on categorizing libraries
by using various kinds of their characteristics. Kawaguchi et al. [23] proposed
MUDABlue, a tool that automatically categorizes software systems by using
the information from source code. MUDABlue allows a software system to be a
member of multiple categories. MUDABlue has three major aspects: (1) it relies
on no other information than the source code, (2) it determines category sets
automatically, and (3) it allows a software system to be a member of multiple
categories. MUDABlue not only correctly determined the categories as specified
by the authors of the software, it determined some additional categories that
were missed by the manual categorization. We have also shown that MUDABlue
is more advanced than the research systems described in the literature.

In this work, we a cross-ecosystem library is a kind of software category
that spans across ecosystems. Hence, it would be interesting to understand how
cross-ecosystem libraries are categorized.

On Open Source Software Retention, Engagement, and Sustained
Contributions - The common motivations to make contributions to OSS
projects are the joy of programming, the identification with a community, career
advancement, and learning [24]. Additionally, Roberts et al. [25] explored the
interrelationships between OSS developer motivations, revealing the motivations
are not always complementary. When comparing motivations between individ-
ual developers and companies, Bonaccorsi and Rossi-Lamastra [26] observed
that companies are more motivated by economic and technological reasons.

In this work, we would like to understand how cross-ecosystem libraries
are able to engage and retain contributions from multiple ecosystems. In this
case, the motivation is to ensure that the library can have sustained activity,
especially if the ecosystem becomes dependent on this library.

7. Conclusion and Future Directions

Developers may inevitably face the need to switch programming languages, which ultimately may result in replacing third-party libraries. An alternative to replacement libraries is for maintainers to release different releases for different ecosystems (i.e., cross-ecosystem library). In a large-scale study that covers over 1.1 million libraries, we find that contributors to a cross-ecosystem library belong to one ecosystem, and more likely to be implemented using multiple programming languages. Insights from this study lay the groundwork and open up new opportunities for library maintainers, users, contributors, and researchers into understanding how these different ecosystems are becoming more intertwined with each other.

Declaration of Interest

Funding: This work has been supported by JSPS KAKENHI Grant Number JP20H05706, JP20K19774

Conflicts of interest: Raula Gaikovina Kula is on the Editorial Board.

Availability of replication package: Our replication package is openly available at https://zenodo.org/record/6983864#.YvVmGuxBy3I

Author contributions:

- Kanchanok Kannee: Conceptualisation, Methodology, Software, Original Writing draft, Visualisation.
- Supatsara Wattanakriengkrai: Investigation, Data collection, Original Writing, Review.
- Ruksit Rojpaisarnkit: Data Curation, Investigation, Original Writing
- Raula Gaikovina Kula: Conceptualisation, Funding Acquisition, review and editing drafts, Supervision, project administration.
- Kenichi Matsumoto : Funding Acquisition, review and editing drafts, Supervision, project administration.

References

[1] R. G. Kula, D. M. German, A. Ouni, T. Ishio, K. Inoue, Do developers update their library dependencies?, Empirical Softw. Engg. 23 (2018) 384–417. URL: https://doi.org/10.1007/s10664-017-9521-5 doi [10.1007/s10664-017-9521-5]
[2] B. E. Cossette, R. J. Walker, Seeking the ground truth: A retroactive study on the evolution and migration of software libraries, in: Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, FSE ’12, 2012.

[3] S. Islam, R. G. Kula, C. Treude, B. Chinthanet, T. Ishio, K. Matsumoto, Contrasting third-party package management user experience, in: 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2021, pp. 664–668. doi:10.1109/ICSME52107.2021.00077

[4] C. Chen, S. Gao, Z. Xing, Mining analogical libraries in q amp;a discussions – incorporating relational and categorical knowledge into word embedding, in: 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), volume 1, 2016, pp. 338–348. doi:10.1109/SANER.2016.21

[5] C. Chen, S. Gao, Z. Xing, Mining analogical libraries in q amp;a discussions – incorporating relational and categorical knowledge into word embedding, in: 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), volume 1, 2016, pp. 338–348. doi:10.1109/SANER.2016.21

[6] C. Teyton, J.-R. Falleri, X. Blanc, Automatic discovery of function mappings between similar libraries, in: 2013 20th Working Conference on Reverse Engineering (WCRE), 2013, pp. 192–201. doi:10.1109/WCRE.2013.671294

[7] M. Valiev, B. Vasilescu, J. Herbsleb, Ecosystem-level determinants of sustained activity in open-source projects: A case study of the pypi ecosystem, in: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2018, 2018, p. 644–655.

[8] A. Zerouali, T. Mens, G. Robles, 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), IEEE, 2019, pp. 589–593.

[9] A. Decan, T. Mens, P. Grosjean, An empirical comparison of dependency network evolution in seven software packaging ecosystems, Empirical Software Engineering 24 (2019) 381–416.

[10] M. Alfadel, D. E. Costa, E. Shihab, M. Mkhallalati, On the use of dependabot security pull requests, in: 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), IEEE, 2021, pp. 254–265.

[11] H. B. Mann, D. R. Whitney, On a test of whether one of two random variables is stochastically larger than the other, The annals of mathematical statistics (1947) 50–60.
[12] F. Wilcoxon, Individual comparisons by ranking methods. biometrics bulletin, 1 (6), 80-83, 1945.

[13] J. Romano, J. D. Kromrey, J. Coraggio, J. Skowronek, L. Devine, Exploring methods for evaluating group differences on the nsse and other surveys: Are the t-test and cohen’sd indices the most appropriate choices, in: annual meeting of the Southern Association for Institutional Research, Citeseer, 2006, pp. 1–51.

[14] Q. McNemar, Note on the sampling error of the difference between correlated proportions or percentages, Psychometrika 12 (1947) 153–157. URL: https://doi.org/10.1007/bf02295996 doi:10.1007/bf02295996

[15] J. Cohen, Statistical power analysis for the behavioral sciences, Routledge, 2013.

[16] A. Ouni, R. G. Kula, M. Kessentini, T. Ishio, D. M. German, K. Inoue, Search-based software library recommendation using multi-objective optimization, Information and Software Technology (IST) 83 (2017) 55–75.

[17] C. Teyton, J.-R. Fuller, X. Blanc, Mining library migration graphs, in: 2012 19th Working Conference on Reverse Engineering, 2012, pp. 289–298. doi:10.1109/WCRE.2012.38.

[18] P. T. Nguyen, J. Di Rocco, D. Di Ruscio, M. Di Penta, Crossrec: Supporting software developers by recommending third-party libraries, Journal of Systems and Software 161 (2020) 110460.

[19] F. Thung, D. Lo, J. Lawall, Automated library recommendation, in: 2013 20th Working Conference on Reverse Engineering (WCRE), 2013, pp. 182–191.

[20] C. Chen, Z. Xing, Y. Liu, What’s spain’s paris? mining analogical libraries from q&a discussions, Empirical Software Engineering 24 (2019) 1155–1194.

[21] H. Zhong, S. Thummalapenta, T. Xie, L. Zhang, Q. Wang, Mining api mapping for language migration, in: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1, 2010, pp. 195–204.

[22] C. Velázquez-Rodríguez, C. De Roover, Mutama: An automated multi-label tagging approach for software libraries on maven, in: 2020 IEEE 20th International Working Conference on Source Code Analysis and Manipulation (SCAM), 2020, pp. 254–258. doi:10.1109/SCAM51674.2020.00034

[23] S. Kawaguchi, P. K. Garg, M. Matsushita, K. Inoue, Mudablue: An automatic categorization system for open source repositories, Journal of Systems and Software 79 (2006) 939–953.
[24] A. Hars, S. Ou, Working for free? motivations of participating in open source projects, in: Proceedings of the 34th Annual Hawaii International Conference on System Sciences, 2001.

[25] J. Roberts, I.-H. Hann, S. Slaughter, Understanding the motivations, participation, and performance of open source software developers: A longitudinal study of the apache projects, Management Science 52 (2006) 984–999.

[26] A. Bonaccorsi, C. Rossi-Lamastra, Comparing motivations of individual programmers and firms to take part in the open source movement. from community to business, Knowledge and Policy (2006) 40–64.