DaSEA – A Dataset for Software Ecosystem Analysis

Petya Buchkova
pebu@itu.dk
IT University of Copenhagen
Copenhagen, Denmark

Joakim Hey Hinnerskov
jhhi@itu.dk
IT University of Copenhagen
Copenhagen, Denmark

Kasper Olsen
kols@itu.dk
IT University of Copenhagen
Copenhagen, Denmark

Rolf-Helge Pfeiffer
ropf@itu.dk
IT University of Copenhagen
Copenhagen, Denmark

ABSTRACT

Software package managers facilitate reuse and rapid construction of software systems. Since evermore software is distributed via package managers, researchers and practitioners require explicit data of software dependency networks that are opaquely formed by dependency relations between software packages. To reason about increasingly complex software products and ecosystems, researchers and practitioners rely either on publicly available datasets like the seemingly unattended libraries.io [14] or they mine problem-specific data from software ecosystems repeatedly and non-transparently. Therefore, we present the DaSEA dataset, which contains metadata of software packages, their versions, and dependencies from multiple ecosystems (currently six programming languages and five operating system package managers). Alongside the dataset, we provide an extensible open-source tool under the same name that is used to create updated versions of the DaSEA dataset allowing studies of evolution of software ecosystems.

1 INTRODUCTION

Contemporary development and provision of software relies heavily on software reuse. Software is usually build with the help of reusable components, which we call *packages* in this paper. Software ecosystems [17] emerge usually around specific package managers (PMs), which download and setup required dependencies from package registries [22]. For example, Alire[1], FPM[2], Nimble[3], Cargo[4], or vcpkg[5] and Conan[6] are PMs for the Ada, Fortran, Nim, Rust, and the C/C++ programming languages respectively. Homebrew, Chromebrew, Ports, or PkgSrc are PMs for operating systems (OSs), like macOS, ChromeOS, FreeBSD/OpenBSD, or NetBSD and others.

Nowadays, software reuse via dependency on external packages is so fundamental to development that some consider it a *"crime to start writing code before you investigate what packages you can reuse"* [22] and software quality models, such as, the SIG/TÜViT Evaluation Criteria for Trusted Product Maintainability [1] recommend using *"libraries and frameworks over 'homegrown' implementations of standard functionality"* [25]. However, the advantages of software reuse are attended by drawbacks, such as, security or legal issues [9, 24]. Practitioners and researchers require data about dependency networks in ecosystems to build tool support and to study effects of software reuse. For example, nexB's ScanCode or CAST Highlight's transitive license checker require package dependency information[7,8] to check for issues arising from software reuse. Likely, such products rely on problem-specific collection of dependency data from various ecosystems. Problem-specific one-off collection of software dependency data exists in research too, e.g., [3, 15, 16, 20, 21]. However, this is problematic since it strains resources and might impair reproducibility.

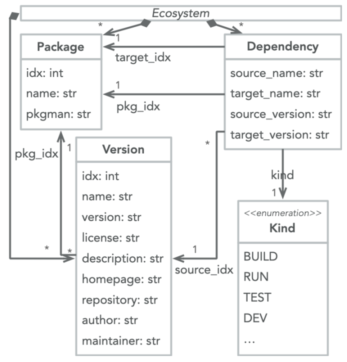
Research of software ecosystems, see e.g., [2, 5–8, 12, 18, 23, 26, 27], relies heavily on the libraries.io [14] dataset. However, that dataset has not been updated in more than two-years (researchers request an updated dataset[9] since start 2021), it never followed a regular release schedule[10], it does not appear to be properly maintained anymore (issues remain unresolved and unanswered[11]), it lacks actual dependency links for some ecosystems (e.g., for Nimble packages), not in all cases the dataset can be directly traced back to the information sources (e.g., for Maven[12]), and importantly, it cannot be reproduced or updated only with the help of the provided tool[13] and documentation[14], see Sec. 2.

Additionally, researchers and practitioners request specifically dependency data for the C/C++ ecosystem, see e.g. [18][15], for which currently no public dataset seems to exist. To alleviate the burden of repeated problem-specific mining of software ecosystem data and to address the issues with the libraries.io dataset, we create the DaSEA dataset[16] (CC BY-SA 4.0 licensed) to provide dependency network data for system programming languages and OS package managers. Alongside the dataset, we provide an extensible tool under the same name[17] (AGPL-3.0 licensed). The tool can be used to create updated versions of the DaSEA dataset allowing studies on evolution of software ecosystems. In the remainder, we illustrate the DaSEA data model and distribution format in Sec. 3, we present the data collection process and information sources in Sec. 4, we sketch potential use cases of the dataset (Sec. 5), and we discuss limitations and future improvements of the dataset in Sec. 6.

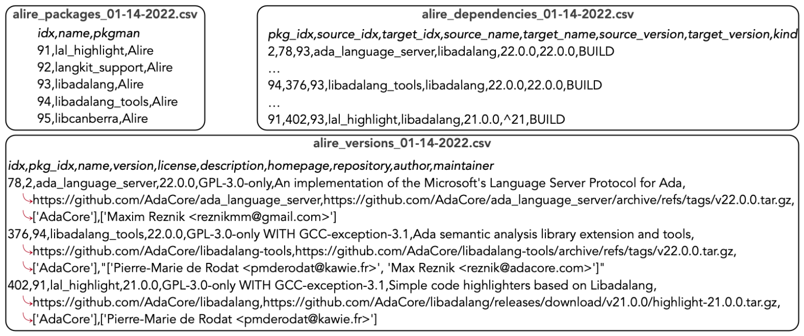
2 RELATED WORK & BACKGROUND

We require software ecosystem metadata, like versions of packages, their licenses, origin, dependencies etc., e.g., to research license incompatibilities in system programming language ecosystems and to study critical projects [18]. Such metadata might be inferred from previously released datasets, e.g., GHTorrent [13], which mirrors event streams and persistent data of GitHub, Debsources [4], which organizes historical source code and metadata of Debian source packages, or the Software Heritage Graph Dataset [19], which captures the development history of projects on GitHub, GitLab, Debian, and PyPI. However, neither it is directly clear which parts of a software project from a forge like GitHub or GitLab form a package that is distributed on a PM, e.g., multiple packages might be built from one source code repository, nor if sources form a reusable package at all. Additionally, besides Debian source packages exist binary packages with metadata that potentially diverges from their source version.

The libraries.io dataset [14] is driving research of software ecosystems, e.g., [2, 5–8, 12, 18, 23, 26, 27]. However, besides the mentioned



(a) Schema of the data model in UML



```
alire_packages_01-14-2022.csv
idx,name,pkgman
91,lal_highlight,Alire
92,langkit_support,Alire
93,libadalang,Alire
94,libadalang_tools,Alire
95,libcanberra,Alire
```

```
alire_dependencies_01-14-2022.csv
pkg_idx,source_idx,target_idx,source_name,target_name,source_version,target_version,kind
2,78,93,ada_language_server,libadalang,22.0.0,22.0.0,BUILD
…
94,376,93,libadalang_tools,libadalang,22.0.0,22.0.0,BUILD
…
91,402,93,lal_highlight,libadalang,21.0.0,^21,BUILD
```

```
alire_versions_01-14-2022.csv
idx,pkg_idx,name,version,license,description,homepage,repository,author,maintainer
78,2,ada_language_server,22.0.0,GPL-3.0-only,An implementation of the Microsoft's Language Server Protocol for Ada,
   ↳https://github.com/AdaCore/ada_language_server,https://github.com/AdaCore/ada_language_server/archive/refs/tags/v22.0.0.tar.gz,
   ↳['AdaCore'],['Maxim Reznik <reznikmm@gmail.com>']
376,94,libadalang_tools,22.0.0,GPL-3.0-only WITH GCC-exception-3.1,Ada semantic analysis library extension and tools,
   ↳https://github.com/AdaCore/libadalang-tools,https://github.com/AdaCore/libadalang-tools/archive/refs/tags/v22.0.0.tar.gz,
   ↳['AdaCore'],"['Pierre-Marie de Rodat <pmderodat@kawie.fr>', 'Max Reznik <reznik@adacore.com>']"
402,91,lal_highlight,21.0.0,GPL-3.0-only WITH GCC-exception-3.1,Simple code highlighters based on Libadalang,
   ↳https://github.com/AdaCore/libadalang,https://github.com/AdaCore/libadalang/releases/download/v21.0.0/highlight-21.0.0.tar.gz,
   ↳['AdaCore'],['Pierre-Marie de Rodat <pmderodat@kawie.fr>']
```

(b) Excerpts from Alire ecosystem data in CSV serialization syntax.

Figure 1: The schema of the DaSEA data model and a concrete instance in serialization syntax.

challenges like the lack of updates and attention, see Sec. 1, metadata from system programming language ecosystems like C/C++, Fortran, Ada, etc. is absent, only one OS ecosystem is included (Homebrew), and for some ecosystems dependency links are missing (Nimble). Additionally, the libraries.io dataset is ill-suited for studying certain aspects of ecosystem evolution, since most package metadata is provided only for the latest version of a package.

Since the libraries.io tool[13] is open-source (AGPL-3.0), we try to create an updated dataset and we attempt to extend it to include data from system programming language ecosystems. However, our attempts fail. Given the official documentation[14], the project's source code, and ca. 60 man-hours distributed over three persons, we reach only to instantiate a functionally incomplete instance of the libraries.io web-application. That requires over 30 manual steps, of which only 11 are currently documented. For example, we experience issues with, failing download of sample data, incomplete instructions of system initialization, or failing and undocumented data collection tasks, etc. Since we and others[18,19] fail to create an updated libraries.io dataset, and since contributions of data for new ecosystems are not necessarily merged into the tool[20], we decide to create a minimal and extensible tool that allows to create and update a dataset similar to libraries.io. Our focus on reproducibility and updatability sets the DaSEA dataset and tool apart from previous work.

3 DATA MODEL

The schema of the DaSEA data model is illustrated as an UML class diagram in Fig. 1a. Per ecosystem (abstract class), the dataset stores metadata about `Package`s, `Version`s, `Dependency` links, and their `Kind`. Packages are uniquely identifiable in an ecosystem (`idx`), they carry a `name`, and refer to the PM of origin (`pkgman`). Each `Version` of a `Package` is uniquely identifiable (`idx`), refers to the package of which it is a version (`pkg_idx`), and it is characterized by a `name` and a version specification (`version`). Due to heterogeneity of version schemes, e.g., numbers, dates, etc., version specifications are stored as strings. Additionally, a version's `license`, `description`, `homepage`, `repository`, `author`, and `maintainer` are stored if available. `Dependency` links connect specific `Version`s of packages (`source_idx`) with the required package (`target_idx`). `source_version` stores the version specification of the dependent package (usually a precise version number like `22.0.0`) and `target_version` stores the version constraint of the package that is required (often a version range like `^21`). Additionally, the names of the `Package` and the `Version` that are connected by a dependency link are stored (`target_name` and `source_name` respectively). Dependency `Kind`s are stored via values of an enumeration (only an excerpt is illustrated), e.g., a dependency exists at, build-, run-, development-time, etc.

The data model is implemented in an object-oriented style, that facilitates extension with ecosystem-specific fields, e.g., location of bug-trackers, creation time of a version, etc. Instances of the data model are store and distributed as CSV files, since this versatile format can be easily imported, processed, and analyzed by a plethora of tools, see the examples in Sec. 5.

The dataset is distributed as a BZip2 compressed TAR archive[16], which contains directories named after PMs (in minuscule), e.g., `alire`, `conan`, etc. Each directory contains three CSV files named `<package_manager>_[packages|versions|dependencies]_<date>.csv`, where `<date>` is formatted as `%m-%d-%Y`. Fig. 1b illustrates the naming scheme of the CSV files for the Alire ecosystem together with an excerpt of data. It shows, that each of the three versions of `ada_language_server`, `libadalang_tools`, and `lal_highlight` have a build-time dependency to `libadalang`, either to its precise version `22` (`ada_language_server` and `libadalang_tools`) or to any minor version above `21` (`lal_highlight: ^21`).

Some data is duplicated across the classes `Package`, `Version`, and `Dependency`. For example, `Version`s duplicate a package's `name` and `Dependency` links duplicate the names and version information from the linked entities. Though redundant, such a representation allows users of the dataset to focus analysis on certain aspects of the dataset without joining CSV files or database tables. For instance, the first example in Sec. 5 illustrates how packages with many dependents can be identified via SQL without the need to join database tables.

Unlike the libraries.io dataset, we decide to store information about licenses, authors, etc. on `Version` instances, since that permits to study evolution of packages over time.

4 DATA GATHERING

The DaSEA dataset is created by mining metadata from various PMs and registries, which usually constitute ecosystems [22]. Per ecosystem, Python scripts[21] collect metadata of packages, their versions, and dependencies from suitable sources, convert it into the uniform data model (Sec. 3), and serialize it to CSV files. We call these Python scripts *miners*. Except of the shared data model, miners are completely independent of each other.

The DaSEA dataset is gathered from heterogeneous sources in various ways. The miners for vcpkg, FPM, and Homebrew parse the respective package registries, which are conveniently distributed in JSON format via web-APIs[22,23,24]. The Nimble miner works similarly. First, it receives the package registry as JSON object via a web-API call[25], second, it shallowly clones package repositories that are referenced in the package registry, and third, it parses nimble files, that store version and dependency related metadata from these repositories. The process is staged, since the Nimble package registry does not store information about versions of packages and their dependencies centrally.

Alire[26] and Chromebrew[27] are mined by cloning the package registries (organized as Git repositories) and by parsing metadata respectively from TOML and Ruby files. Similarly, the Conan miner clones the Git repository that holds the package registry[28], parses package-related metadata from YAML files, and extracts version and dependency-related metadata from the output of the conan CLI tool.

The Cargo miner converts the daily database dump from crates.io[29], which is distributed as a set of CSV files, into the DaSEA data model.

On Free-[30] and OpenBSD[31], the current ports trees and on NetBSD[32] the current pkgsrc tree are downloaded and extracted. In these ecosystems, metadata is stored in Makefiles, which respective miners extract with the OS-specific version of the make tool. Ports and pkgsrc trees are mined in virtual machines (VMs) since the make tools are slightly different across the BSDs. The DaSEA tool manages VMs via vagrant[33] and VirtualBox[34].

Metadata from packages and their versions, like names, version numbers/constraints, authors, repositories, etc. are dumped directly from the respective data source into the corresponding fields of the data model (Fig. 1a) and are serialized to CSV without further processing. The rationale behind this decision is: most PMs provide unchecked free-form fields for such data. For example, the authors of ada_language_server, libadalang_tools, and lal_highlight provide links to release archives instead of links to VCS repositories, see Fig. 1b, or authors identify themselves inhomogeneously via real names, email-addresses, nicks, or combinations of these. Since we want to preserve highest degree of utility of the gathered data and since we cannot anticipate potential applications of the DaSEA dataset, we leave processing of such *raw* data to the users. However, miners enrich the gathered metadata by explicitly creating dependency links, see class Dependency in Sec. 3. Usually, PMs express dependency links via string references, which our miners interpret according to official documentation. Internally, miners construct lookup tables to convert string-based package and version identifiers to integers, see all *idx fields in Fig. 1a. Besides mining metadata from heterogeneous sources, first class dependency links are the main feature of the DaSEA dataset. They facilitate ecosystem analysis as illustrated by the following examples.

5 USAGE EXAMPLES

In this section, we present four examples that illustrate how various tools can be applied to query the DaSEA dataset, to reason about it, and to visualize it. All code examples are in Python, they are optimized for space usage and not for coding style, and they are explained in more detail in the official documentation[35].

a) SQL: Identify relevant packages

Number of dependents is a basic metric for assessing the relevance of a package in an ecosystem. In Lst. 1, we demonstrate how to identify the pkgsrc package (on NetBSD) with most direct dependents via an SQL query (lines six to eight).

For brevity, Dependency CSV data is imported into an SQLite in-memory database with Pandas' to_sql function. The latter creates the database schema automatically. The actual SQL query identifies the node with the highest in-degree in the dependency network by searching for the most frequent target_name of all dependency links. Grouping by a Dependency's target_name (line seven), encodes the search for the number of dependent packages instead of the number of dependent versions.

The query yields pkgtools/cwrappers as the package with most dependents. 18,228 of the 18,231 packages depend directly on it.

```python
1 import pandas as pd, sqlalchemy as sl
2
3 ddf = pd.read_csv("ports/netbsd9/netbsd9_dependencies_01-27-2022.csv")
4 db_engine = sl.create_engine('sqlite://')  # in memory DB
5 ddf.to_sql("Dependency", db_engine)
6 query = """SELECT target_name, COUNT(target_name) AS indegree FROM Dependency
7 GROUP BY target_name ORDER BY indegree DESC
8 LIMIT 1;"""
9 print(pd.read_sql(query, db_engine))
```

Listing 1: SQL query to identify most required package.

b) Pandas: Identify license changes

Changing licenses between package versions may pose legal issues for dependents[36]. In Lst. 2, we show how to leverage the data analysis library Pandas[37] to infer those packages from the Conan ecosystem that possess more than one license over their version history (line four). Pandas DataFrames (line three) are a tabular data representation comparable to spreadsheets. They can be queried in a similar fashion (line four).

The code of line five organizes the query results for display. The results in Lst. 3 show, that 12 Conan packages change licenses across versions. The respective package names are listed to the left and an unordered set of corresponding licenses is listed to the right. Note, ordering the changing licenses chronologically, would require a modification of Lst. 2.

```python
1 import pandas as pd
2
3 df = pd.read_csv("conan/conan_versions_01-27-2022.csv")
4 rdf = df.groupby("pkg_idx").filter(lambda x: len(set(x.license)) > 1)
5 print(rdf.groupby("name").apply(lambda x: set(x.license)))
```

Listing 2: Pandas query to identify Conan packages that change licenses over versions.

```
bzip2                                      {['bzip2-1.0.6'], ['bzip2-1.0.8']}
freetype                                          {['MIT'], ['bzip2-1.0.8']}
gtk                                         {['MIT'], ['LGPL-2.1-or-later']}
mbedtls                              {['GPL-2.0', 'Apache-2.0'], ['Apache-2.0']}
mosquitto                                            {['MIT'], ['EPL-2.0']}
opencv                          {['MPL-2.0', 'LGPL-3.0-or-later'], 'Apache-2.0']}
openssl                                         {['OpenSSL'], ['Apache-2.0']}
poco                                        {['Apache-2.0'], ['bzip2-1.0.8']}
proj                                                 {['MIT'], ['GPL-2.0']}
rmlui                                             {['MIT'], ['bzip2-1.0.8']}
sentry-crashpad                                 {['OpenSSL'], ['Apache-2.0']}
zbar                                        {['LGPL-2.1'], ['LGPL-2.1-only']}
```

Listing 3: Conan packages with changing licenses.

c) NetworkX: Compute a centrality metric

Various centrality metrics encode various notions of *importance* of a node in a graph. Packages that –if removed– would affect the Conan ecosystem the most can be detected via *Betweenness Centrality* [10]. Amongst others, the Python library NetworkX[38] offers an implementation of a betweenness centrality algorithm. The DaSEA dataset has to be converted into a format that is accessible by NetworkX. Lst. 4 shows, how to convert the Conan Dependency data into an adjacency list, a text file of space separated node identifiers (lines three to six). Out of this, NetworkX constructs a directed graph for which the betweenness centrality is computed (lines eight and nine). The script prints package identifiers (pkg_idx) and the respective centrality score for the three most central packages (lines 10 and 11). The results (not illustrated) indicate that openssl, libcurl, and at-spi2-core are the packages with highest betweenness centrality in the Conan C/C++ ecosystem. The prominent position of openssl in the ecosystem hints at the impact of the *Heartbleed bug*[39] in 2014.

```
1  import pandas as pd, networkx as nx, numpy as np
2
3  ddf = pd.read_csv("conan/conan_dependencies_01-27-2022.csv")
4  ddf = ddf[(~ddf.pkg_idx.isnull()) & (~ddf.target_idx.isnull())]
5  adjl = ddf[["pkg_idx", "target_idx"]].to_numpy()
6  np.savetxt("/tmp/conan.adjl", adjl, fmt="%u", delimiter=" ")
7
8  g = nx.read_adjlist("/tmp/conan.adjl", nodetype=int, create_using=nx.DiGraph)
9  betweennes_ranks = nx.betweenness_centrality(g)
10 print(list(sorted(betweennes_ranks.items(), key=lambda item: item[1],
11                   reverse=True))[:3])
```

Listing 4: Computation of Betweenness Centrality with NetworkX.

d) Gephi: Visualize an ecosystem

Fig. 2 illustrates the vcpkg C/C++ ecosystem with a Fruchterman–Reingold layout [11]. Nodes are scaled to Eigenvector Centrality, which indicates the high influence of the packages vcpkg-cmake and vcpkg-cmake-config (center). Highlighted are dependents of these two. Visual inspection of ecosystem dependency networks allows to get acquainted with certain properties of the ecosystem, e.g., the amount of independent packages (mainly outer ring in Fig. 2), cliques of packages (center ring to the left and right), etc.

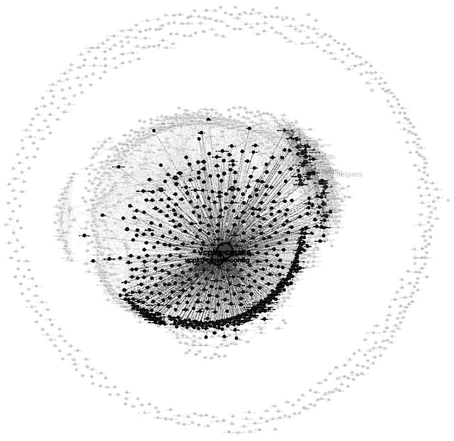


Figure 2: Visualization of the vcpkg C/C++ ecosystem. Nodes are scaled to Eigenvector Centrality.

Gephi[40] is a network visualization tool. It can import CSV files that contain a graph's *nodes* and *edges*. Indexes of nodes and relation ends have to be stored in columns labeled Id, Source, and Target respectively. Lst. 5 converts the DaSEA package and dependency CSV files from vcpkg accordingly.

```
import pandas as pd

pdf = pd.read_csv("vcpkg/vcpkg_packages_01-27-2022.csv")
pdf.rename(columns={"idx": "Id", "name": "Label"}, inplace=True)
pdf.to_csv("/tmp/vcpkg_gephi_nodes.csv", index=False)

ddf = pd.read_csv("vcpkg/vcpkg_dependencies_01-27-2022.csv")
ddf = ddf[(~ddf.pkg_idx.isnull()) & (~ddf.target_idx.isnull())]
ddf.pkg_idx = ddf.pkg_idx.astype(np.uint)
ddf.rename(columns={"pkg_idx": "Source", "target_idx": "Target"}, inplace=True)
ddf.to_csv("/tmp/vcpkg_gephi_edges.csv", index=False)
```

Listing 5: Conversion of data for visualization with Gephi.

6 LIMITATIONS & IMPROVEMENTS

Currently, packages and their versions are indexed per ecosystem, i.e., cross-ecosystem analyses have to first harmonize indexes accordingly. Only the data for Alire, Conan, FPM, and Cargo contains information about historical versions of packages, due to how package registries and PMs provision metadata. For the other ecosystems, the most recent package versions at the time of mining are provided. We plan to release a script, that allows to merge updated versions of the DaSEA dataset into a single one.

As the data model in Fig. 1a illustrates, dependencies in the DaSEA dataset link versions of a package to required packages accompanied by a version constraint (target_version). That is, dependency links do not link versions of packages directly. This mimics how dependency links are modeled in the libraries.io dataset. If required, users have to programmatically create version-to-version dependency links by interpreting the provided version constraints.

In future, we plan to extend the DaSEA dataset with dependency networks from more ecosystems. Currently, we are working on implementing miners for Nix, APT on Ubuntu, Gentoo's portage, and Maven with Maven Central. Additionally, we hope that the open nature of the DaSEA tool encourages contributions from the community.

To conclude, in this paper, we present the DaSEA dataset. We are not aware of a similar dataset that contains metadata including package dependencies for the Conan and vcpkg (C/C++), Alire (Ada), FPM (Fortran), Nimble (Nim), Chromebew (ChromeOS) and the three BSDs in a unified format. The Rust (Cargo) metadata in the DaSEA dataset substantially updates the information provided by libraries.io.

Lastly, we aim at releasing continuous updates of the DaSEA dataset semiannually.

REFERENCES

[1] [n. d.]. SIG/TÜViT Evaluation Criteria Trusted Product Maintainability, Version 12.0. https://www.softwareimprovementgroup.com/wp-content/uploads/2020-SIG-TUViT-Evaluation-Criteria-Trusted-Product-Maintainability.pdf. Accessed: 2020-10-27.

[2] Rabe Abdalkareem, Vinicius Oda, Suhaib Mujahid, and Emad Shihab. 2020. On the impact of using trivial packages: An empirical case study on npm and pypi. *Empirical Software Engineering* 25, 2 (2020), 1168–1204.

[3] Thorsten Berger, Rolf-Helge Pfeiffer, Reinhard Tartler, Steffen Dienst, Krzysztof Czarnecki, Andrzej Wąsowski, and Steven She. 2014. Variability mechanisms in software ecosystems. *Information and Software Technology* 56, 11 (2014), 1520–1535.

[4] Matthieu Caneill, Daniel M Germán, and Stefano Zacchiroli. 2017. The Debsources Dataset: two decades of free and open source software. *Empirical Software Engineering* 22, 3 (2017), 1405–1437.

[5] Alexandre Decan, Tom Mens, and Eleni Constantinou. 2018. On the evolution of technical lag in the npm package dependency network. In *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 404–414.

[6] Alexandre Decan, Tom Mens, and Eleni Constantinou. 2018. On the impact of security vulnerabilities in the npm package dependency network. In *Proceedings of the 15th International Conference on Mining Software Repositories*. 181–191.

[7] Alexandre Decan, Tom Mens, and Philippe Grosjean. 2019. An empirical comparison of dependency network evolution in seven software packaging ecosystems. *Empirical Software Engineering* 24, 1 (2019), 381–416.

[8] Jens Dietrich, David Pearce, Jacob Stringer, Amjed Tahir, and Kelly Blincoe. 2019. Dependency versioning in the wild. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 349–359.

[9] Ruian Duan, Ashish Bijlani, Meng Xu, Taesoo Kim, and Wenke Lee. 2017. Identifying open-source license violation and 1-day security risk at large scale. In *Proceedings of the 2017 ACM SIGSAC Conference on computer and communications security*. 2169–2185.

[10] Linton C Freeman. 1977. A set of measures of centrality based on betweenness. *Sociometry* (1977), 35–41.

[11] Thomas MJ Fruchterman and Edward M Reingold. 1991. Graph drawing by force-directed placement. *Software: Practice and experience* 21, 11 (1991), 1129–1164.

[12] Mehdi Golzadeh. 2019. Analysing socio-technical congruence in the package dependency network of Cargo. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1226–1228.

[13] Georgios Gousios and Diomidis Spinellis. 2012. GHTorrent: GitHub's data from a firehose. In *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*. IEEE, 12–21.

[14] Jeremy Katz. 2020. *Libraries.io Open Source Repository and Dependency Metadata*. https://doi.org/10.5281/zenodo.3626071

[15] Maria Kechagia, Diomidis Spinellis, and Stephanos Androutsellis-Theotokis. 2010. Open source licensing across package dependencies. In *2010 14th Panhellenic conference on informatics*. IEEE, 27–32.

[16] Riivo Kikas, Georgios Gousios, Marlon Dumas, and Dietmar Pfahl. 2017. Structure and evolution of package dependency networks. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. IEEE, 102–112.

[17] Mircea Lungu, Michele Lanza, Tudor Gîrba, and Romain Robbes. 2010. The small project observatory: Visualizing software ecosystems. *Science of Computer Programming* 75, 4 (2010), 264–275.

[18] Rolf-Helge Pfeiffer. 2021. Identifying Critical Projects via PageRank and Truck Factor. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. 41–45. https://doi.org/10.1109/MSR52588.2021.00017

[19] Antoine Pietri, Diomidis Spinellis, and Stefano Zacchiroli. 2019. The Software Heritage graph dataset: public software development under one roof. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 138–142.

[20] Gede Artha Azriadi Prana, Abhishek Sharma, Lwin Khin Shar, Darius Foo, Andrew E Santosa, Asankhaya Sharma, and David Lo. 2021. Out of sight, out of mind? How vulnerable dependencies affect open-source projects. *Empirical Software Engineering* 26, 4 (2021), 1–34.

[21] Steven Raemaekers, Arie Van Deursen, and Joost Visser. 2013. The maven repository dataset of metrics, changes, and dependencies. In *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, 221–224.

[22] Diomidis Spinellis. 2012. Package management systems. *IEEE software* 29, 2 (2012), 84–86.

[23] Jacob Stringer, Amjed Tahir, Kelly Blincoe, and Jens Dietrich. 2020. Technical Lag of Dependencies in Major Package Managers. In *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 228–237.

[24] Christopher Vendome. 2018. Assisting Software Developers With License Compliance. (2018).

[25] Joost Visser, Sylvan Rigal, Rob van der Leek, Pascal van Eck, and Gijs Wijnholds. 2016. Building Maintainable Software, Java Edition: Ten Guidelines for Future-Proof Code.

[26] Ahmed Zerouali, Eleni Constantinou, Tom Mens, Gregorio Robles, and Jesús González-Barahona. 2018. An empirical analysis of technical lag in npm package dependencies. In *International Conference on Software Reuse*. Springer, 95–110.

[27] Ahmed Zerouali, Tom Mens, Gregorio Robles, and Jesus M Gonzalez-Barahona. 2019. 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, 589–593.

REFERENCED URLS

[1] https://alire.ada.dev/
[2] https://fpm.fortran-lang.org/en/index.html
[3] https://nimble.directory/
[4] https://doc.rust-lang.org/cargo/ and the package registry https://crates.io/
[5] https://vcpkg.io/
[6] https://conan.io/
[7] https://github.com/golang/go/issues/36785#issuecomment-683388282
[8] https://www.castsoftware.com/products/highlight/capabilities#Software-Composition-Analysis
[9] https://github.com/librariesio/libraries.io/issues/2744
[10] https://libraries.io/data
[11] https://github.com/librariesio/libraries.io/pull/2130
[12] https://github.com/librariesio/libraries.io/blob/main/app/models/package_manager/maven.rb
[13] https://github.com/librariesio/libraries.io
[14] https://github.com/librariesio/libraries.io/blob/main/docs/development-setup.md
[15] https://github.com/ossf/criticality_score/issues/53#issuecomment-748556336
[16] https://zenodo.org/record/6369420
[17] https://github.com/DaSEA-project/DASEA/blob/main/README.adoc
[18] https://github.com/librariesio/libraries.io/issues/2835
[19] https://github.com/librariesio/libraries.io/issues/2820
[20] https://github.com/librariesio/libraries.io/issues/2043
[21] https://github.com/DaSEA-project/DASEA/tree/main/dasea/miners
[22] https://vcpkg.io/output.json
[23] https://raw.githubusercontent.com/fortran-lang/fpm-registry/master/index.json
[24] https://formulae.brew.sh/api/formula.json
[25] https://raw.githubusercontent.com/nim-lang/packages/master/packages.json
[26] https://github.com/alire-project/alire-index/tree/stable-1.1/index
[27] https://github.com/skycocker/chromebrew/tree/master/packages
[28] https://github.com/conan-io/conan-center-index
[29] https://static.crates.io/db-dump.tar.gz
[30] received with `portsnap`, which fetches from https://cgit.freebsd.org/ports
[31] https://cdn.openbsd.org/pub/OpenBSD/6.9/ports.tar.gz
[32] ftp://ftp.NetBSD.org/pub/pkgsrc/current/pkgsrc.tar.gz
[33] https://www.vagrantup.com/
[34] https://www.virtualbox.org/
[35] https://github.com/DaSEA-project/DASEA/blob/main/docs/using_the_dataset.md
[36] https://www.theregister.com/2021/03/25/ruby_rails_code/
[37] https://pandas.pydata.org/
[38] https://networkx.org/
[39] https://nvd.nist.gov/vuln/detail/CVE-2014-0160
[40] https://gephi.org/