A Second Look at the Dynamics of the JavaScript Package Ecosystem

Kevin de Haan, Gregory Neagu, Frederic Sauve-Hoover, Abram Hindle
Department of Computing Science
University of Alberta
Edmonton, Canada
Email: {kdehaan,neagu,rsauveho,abram.hindle}@ualberta.ca

***Abstract*—In recent years, the tools and packages most commonly involved with JavaScript development have evolved rapidly. Newer packages such as Angular and React have experienced a marked increase in popularity among developers, while frameworks such as jQuery have begun to phase out.[1] For this reason we take a second look at a 2016 paper by Wittern, Suter and Rajagopalan [11] to see what aspects of the JavaScript package ecosystem have changed, and if previously observed trends have remained constant. In the original paper the authors use the *node package manager* (`npm`) to gain insight into the JavaScript ecosystem as a whole, and data from projects publicly hosted on `GitHub` to observe an alternative measure of popularity. We adhere to the same methods of analysis, and extend the data to capture more recent information up to April 1st 2019. Ultimately, this second look aims to discover if recent years have had any significant effects on ecosystem-wide trends, and provide developers with further insight into how packages are used and evolve.**

***Index Terms*—JavaScript; Node.js; node package manager; software ecosystem analysis**

I. Introduction

Software ecosystems are environments that form as projects develop in parallel, becoming interconnected as contexts and dependencies span companies and communities [5]. Research on these systems has increased rapidly in the recent past [9], investigating their characteristics and behaviour as they develop [6]. Understanding how software ecosystems evolve is important from both a software as well as a business standpoint [7], and is valuable for informing developers how technologies are used over time [10]. An understanding of software ecosystems can inform decisions on when to adopt frameworks and how long to support them, as well as provide insight into how changes to software propagate throughout the community [11]. Additionally, determining the characteristics of software ecosystems can help clarify why some frameworks flourish while others fail, and guide developers in the creation of new tools [10]. Furthermore, because software projects are overwhelmingly a collaborative effort, a complete understanding of a single project often requires knowledge of the ecosystem supporting it [1].

[1]https://insights.stackoverflow.com/survey/2016#technology-most-popular-technologies, https://insights.stackoverflow.com/survey/2017#technology-_-frameworks-libraries-and-other-technologies, https://insights.stackoverflow.com/survey/2018#technology-_-frameworks-libraries-and-tools

This paper is a replication of *A Look at the Dynamics of the JavaScript Package Ecosystem* [11] that performs extensive analysis of the *node package manager* (`npm`) ecosystem. `npm` provides a set of open source tools that allow developers to describe packages for `Node.js`, an asynchronous JavaScript runtime environment designed for network applications[2]. The services provided by `npm` include a command line interface for maintaining `package.json` files, the primary method to describe package metadata such as the name, description, version, and dependencies of a given package. `npm` also allows developers to publish their packages to a public registry, permitting anyone to download and use their software. Packages hosted on `npm` will often depend on other `npm` packages, forming an elaborate JavaScript package ecosystem. Since the publishing of the original paper, the usage and scale of `npm` has only grown, and now hosts more than three times as many packages (over 750,000 as of April 1st 2019) and handles over ten times as many weekly package downloads (now over ten billion per week). Additionally, the major frameworks used in JavaScript development have undergone a rapid transformation as packages such as Angular and React are adopted[1]. The core contributions we make are as follows:

- We replicate and verify the results found in the original paper for the window of October 1st 2010 to September 1st 2015.
- We extend the analysis to the time period of September 2nd 2015 to April 1st 2019, and evaluate whether patterns and trends noted in the original paper are still observable.
- We investigate whether the continued evolution of the JavaScript package ecosystem has affected the relationships between various measures of package popularity.
- We determine if the ongoing maturation of the `npm` ecosystem has resulted in tangible changes to version numbering or adoption practices.

II. Data Collection

The window of data analyzed within this paper is October 1st 2010 (as in the original paper) to April 1st 2019. We collected from three publicly available data sets. Two of these, the `npm` registry and the `GitHub` repository platform, are from the same source as in the original paper. To find

[2]https://nodejs.org/en/about/

repositories relying on npm, we used the Google BigQuery github_repos data set, updated weekly[3]. By using this set we are able to analyze GitHub data without being constrained to the currently available window provided by the GHTorrent project [3]. The final data set encompasses 797,940 packages and **#VALUE** applications.

A. Package Metadata

this one is probably mostly for @Fred

B. Applications using npm Packages

@Greg I'm gonna need you to put something here

III. Ecosystem Evolution

Created in 2009, npm has grown rapidly in popularity and scope over the last ten years, and as of the original paper showed no signs of slowing down [11]. We investigate the state of the npm ecosystem since September 1st 2015, and look for any signs of deterioration in the health of the ecosystem. Periods of stagnating growth would suggest that developer interest is waning, while steady activity would indicate that the ecosystem as a whole is healthy and will continue to evolve. To search for these potential indicators in the npm package environment, we investigate the number of packages created and updated over time, as well as system-wide trends of dependencies within packages. In figure 1 [INSERT DATA COMMENTARY HERE]. Figure 2 [MORE DATA COMMENTARY].

To better visualize the status of inter-package dependencies, we constructed a directed dependency graph using dependants as out-degrees and dependencies as in-degrees. Based on this dependency graph, Figure 3 displays the percentage of packages with various amounts of dependencies. [MIGHT NEED TO CHANGE BASED ON DATA: While the average number of dependencies of packages continues to increase, the percentage of packages with zero to three dependencies has actually increased since the end of the original paper's reporting period [11]. This suggests some changes to developer behaviour, likely either as a deliberate effort by programmers seeking to avoid the perils of complicated dependency trees [4], or as a natural result of the ever-changing ways in which JavaScript is used for project development.

IV. Package Popularity

A. Relationships between Measures

B. Distinct Package Types

C. Popularity Over Time

1) Identifying Top Packages:

2) Popular Package Dynamics:

3) Comparing Popularities:

[3]https://github.com/fhoffa/analyzing_github/

V. Version Numbering and Package Adoption

A. Attribution of Version Numbers

B. Adoption by Version Number

VI. Related Work

VII. Conclusion

References

[1] Kelly Blincoe, Francis Harrison, and Daniela Damian. Ecosystems in github and a method for ecosystem identification using reference coupling. In *Proceedings of the 12th Working Conference on Mining Software Repositories*, MSR '15, pages 202–207, Piscataway, NJ, USA, 2015. IEEE Press.

[2] Alexandre Decan, Tom Mens, and Philippe Grosjean. An empirical comparison of dependency network evolution in seven software packaging ecosystems. *CoRR*, abs/1710.04936, 2017.

[3] Georgios Gousios. The ghtorrent dataset and tool suite. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR '13, pages 233–236, Piscataway, NJ, USA, 2013. IEEE Press.

[4] Riivo Kikas, Georgios Gousios, Marlon Dumas, and Dietmar Pfahl. Structure and evolution of package dependency networks. In *Proceedings of the 14th International Conference on Mining Software Repositories*, MSR '17, pages 102–112, Piscataway, NJ, USA, 2017. IEEE Press.

[5] Mircea Lungu, Michele Lanza, Tudor Grba, and Romain Robbes. The small project observatory: Visualizing software ecosystems. *Science of Computer Programming*, 75(4):264 – 275, 2010. Experimental Software and Toolkits (EST 3): A special issue of the Workshop on Academic Software Development Tools and Techniques (WASDeTT 2008).

[6] Konstantinos Manikas and Klaus Marius Hansen. Software ecosystems - a systematic literature review. *J. Syst. Softw.*, 86(5):1294–1306, May 2013.

[7] David Messerschmitt and Clemens Szyperski. *Software Ecosystem: Understanding an Indispensable Technology and Industry*. 01 2003.

[8] Amantia Pano, Daniel Graziotin, and Pekka Abrahamsson. What leads developers towards the choice of a javascript framework? *CoRR*, abs/1605.04303, 2016.

[9] M. Seppnen, S. Hyrynsalmi, K. Manikas, and A. Suominen. Yet another ecosystem literature review: 10+1 research communities. In *2017 IEEE European Technology and Engineering Management Summit (E-TEMS)*, pages 1–8, Oct 2017.

[10] Alexander Serebrenik and Tom Mens. Challenges in software ecosystems research. In *Proceedings of the 2015 European Conference on Software Architecture Workshops*, ECSAW '15, pages 40:1–40:6, New York, NY, USA, 2015. ACM.

[11] Erik Wittern, Philippe Suter, and Shriram Rajagopalan. A look at the dynamics of the javascript package ecosystem. In *Proceedings of the 13th International Conference on Mining Software Repositories*, MSR '16, pages 351–361, New York, NY, USA, 2016. ACM.

Fig. 1: Neat you can include PDFs