Analysing Socio-Technical Congruence in the Package Dependency Network of Cargo

Anonymous Author(s)

1 INTRODUCTION

Today's software development is increasingly relying on software package libraries distributed through open source software package managers (such as Cargo, npm, Maven, CRAN and many more). Rather than writing software from scratch, developers often choose to depend on existing software packages. At the same time, collaborative online development platforms like GitHub make software development an inherently social phenomenon [8, 21].

The internal structure of a software package manager can be considered as a kind of *socio-technical* dependency network. Packages depend on other packages that are required for installing and deploying them. Software developers are *technically* contributing to these packages (e.g., by making commits, pull requests or issue reports to the package's git repository). Software developers are also *socially* active (e.g., by commenting through GitHub on the development activities of packages). The phenomenon of socio-technical congruence (a.k.a. Conway's law) [7, 13] assumes a tight connection between the package dependency network structure and the communication structure of its community of contributors. There is, however, very little research studying such socio-technical congruence at the ecosystem level [22], or how the socio-technical congruence evolves over time [5].

My PhD research project that started *very* recently (June 1, 2019) aims to empirically study, at the scale of the entire ecosystem, the socio-technical relationship between software package dependencies and the interaction patterns of the contributors to those packages. As such, I aim to gain a better understanding of how the socio-technical congruence within open source software packaging ecosystems evolves over time, and how this affects the health of the package dependency network and its community of contributors.

To validate the socio-technical congruence hypothesis at the level of package dependency networks of open source software package managers, I have started to study three exploratory research questions:

- **RQ1** How does the dependency network structure influence social activity?
- **RQ2** How does the social activity of a contributor to a specific package increase his likelihood to start or stop depending on this package?
- **RQ3** How does the social activity of a contributor to a specific package increase his likelihood to start or stop becoming technically active for this package?

In the first phase of my research, I will focus on packages distributed through the Cargo package manager and developed on GitHub, which is by large the biggest online collaborative software development platform.

I will focus on different types of *technical* development activity (e.g., commits, pull requests and issue reports through GitHub) and *social* communication activity (e.g., commit comments, pull request comments and issue comments through GitHub).

More research questions, activity types and packaging ecosystems will be explored in a later phase of my PhD research, on the basis of the results obtained for these questions for the Cargo case study. These follow-up questions will focus on the expected benefits that socio-technical congruence will have on the health of the packaging ecosystem community, such as increased productivity, responsiveness, contributor intake and retention.

2 BACKGROUND

Software ecosystems are large collections of interconnected software components with complex socio-technical interaction patterns [19, 20]. They have become the norm in geographically distributed OSS development. Typical well-studied ecosystems are software library package managers [9, 11, 15] allowing to reuse software libraries for specific programming languages (e.g., npm, PyPI, CRAN, CPAN, RubyGems, Maven, Cargo). Their technical dependency networks grow at a rapid pace and may contain fragile packages that have a high transitive impact [10]. Ecosystem-specific policies, values and technical choices play an important role in how such networks evolve over time [4].

Social issues are at least as important as technical ones. Social coding platforms can lead to effective work coordination strategies [8] and have become indispensable collaborative environments for software ecosystems [13]. Researchers have studied the social aspects of how developer teams interact and evolve [18], how newcomers progress in a software project [25, 26], how the core team grows over time [23], how developer teams get renewed [6], and how socio-technical patterns affect software success or failure [24].

Social and technical issues are tightly interwoven and should be addressed conjointly, because of Conway's law stating that the software structure mimicks the communication and coordination structure of the community developing it [3, 7, 13, 17]. New models are required to better understand such socio-technical congruence at the ecosystem level [22]. In addition, the temporal dimension needs to be taken into account since the contributor and technical relationships evolve over time [5]. Combining social and technical information leads to better prediction models, for example to detect faults in software components [1, 2], to predict whether a project participant will become a developer [12], and to recommend software experts [16]

3 METHODOLOGY

The libraries.io monitoring service provides access to package dependency metadata for 36 different package managers. Among these, I have selected the Cargo package manager as a case study. Cargo is the default package manager for packages (a.k.a. "crates") for the Rust programming language. Cargo is fairly recent (created in 2014), and the development history of most of its packages is available on GitHub. Is shown in [10], Cargo is growing fast in number of packages, package releases, dependencies and contributors.

By studying this package dependency network since its inception I hope to gain insights into how the ecosystem growth affects the socio-technical congruence over time.

I relied on a datadump of libraries.io [14] to recover the entire temporal evolution of the Cargo package dependency network. For each package, I extracted metadata such as the package name, release number and release date, maintainer, package dependencies and their versioning constraints. I also retrieved the (optional) link to the corresponding GitHub repositories. Packages for which no GitHub link was found, or for which the link was incorrect were filtered out. The libraries.io dataset I used was quite big and there were 48597 unique package dependency relation and 14491 package with more than 66106 releases. I also found that 1571 of records contain no repository address, 413 of them have repository address other than GitHub and 3500 of them contains duplicate repository address. Finally I ended up with 9954 GitHub repositories to download metadata.

From the GitHub repository associated to each remaining package I gathered relevant technical and social activity. This information includes all contributors and their role, the “social” commenting activities they were involved in (separated into commit comments, pull comments, pull review comments and issue comments), and the “technical” development activities they were involved in (commits, pull requests and issue requests). The downloaded metadata was also fairly big including 942183 commits, 145053 pull requests, 266033 issues, 90228 unique comments-contributor. It is also worth to mention that 3170 repositories have no comments and comments follow the pareto rule which mean more than 80 percent of comments belong to less than 20 percents of the repositories.

To explore and analyse the extracted temporal socio-technical package dependency network, I use Python scripts and notebooks, and rely of existing data analytics and statistical libraries.

4 ANALYSIS AND RESULTS

In order to study the socio-technical congruence hypothesis for the Cargo package manager, I started analysing its package dependency network and the associated technical and social (commenting) activities of its GitHub contributors. To do so, I focused on the three preliminary research questions presented in Section 1.

RQ1 - How does the dependency network structure influence social activity? In this research question I am looking for any proof of relation between technical network structure and social interaction between contributors (E.g., are contributors more likely to be active in commenting on packages they depend on than on other packages?) in order to find an answer for this question I created a notebook containing a dependency network and contributors comment of different types and investiged temporal and structural relation between comment activity of developers and dependency network.

RQ2 - How does the social activity of a contributor to a specific package increase his likelihood to start or stop depending on this package? As for this case I am looking for any kind of evidence showing that if there is any sort of relationship between social activity of the contributors of packages and temporal or structural pattern of dependency between packages. To do this in fact I basically tried to find a relation between any kind of social activity including issue comments, commit comment, pull review comment and pull comments of package maintainers and the dependency of each version of that package to find whether there is a tendency to put comment before start using a package or is there any behaviorial pattern between dependency network and social activity. If I can find any sign that shows its more likely for a developer to comment on package before start using it, based on this we can conclude social interaction can lead to technical dependency. My rudimental results shows that although not all activities will result in a technical dependency but there is priliminary evidence that shows there are a lot of cases that developer start using a package after commenting and to be more precise the result shows contributors tend to put comment on issues and pulls before depending on a package. the graph 1 shows my finding in Cargo ecosystem.

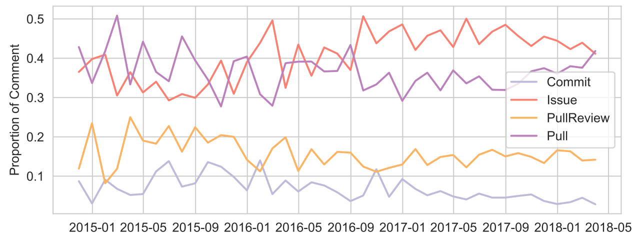

Figure 1: Proportion of comments on unique dependent packages before an after dependency

I also found that there are early signs of increase in commenting activity of developers before using a package as dependency and after june 2017 this trend has a drastic growth between contributors in Cargo ecosystem. the graph 2 shows the emprical result of my investigations.

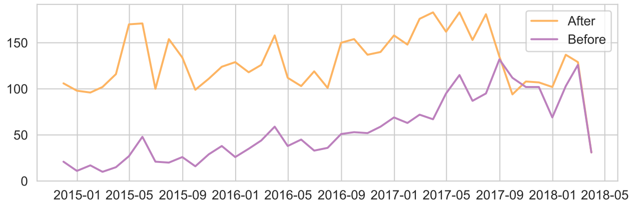

Figure 2: Number of packages with social activity of developer before start using it

RQ3 - How does the social activity of a contributor to a specific package increase his likelihood to start or stop becoming technically active for this package? The prerliminary result of my emprical study shows that package maintainers mostly start commenting on a issue before contributing to package. This shows that a contribution generally starts with an issue in some part of a package. the graph shows contributors tendency to comment before contributing to a package.

5 CONCLUSION

Given that I started my PhD research very recently, I have only been able to report very preliminary results about the socio-technical congruence of the Cargo packaging ecosystem and how it evolves over

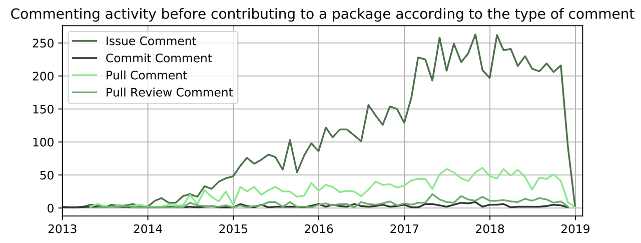

Figure 3: Comment avtivity before contributing to a package

time. During my 4-year PhD studies, I intend to gain a deeper understanding of the dynamics of this phenomenon, and complement this by studying the expected positive effects of such congruence on the health of the ecosystem community, such as increased popularity, productivity, responsiveness, contributor intake and retention. I will also carry out similar studies on other packaging ecosystems (e.g. npm, PyPI, Maven, CRAN, RubyGems), in order to compare their socio-technical congruence and the effect of ecosystem-specific policies and values.

REFERENCES

[1] P. Bhattacharya, M. Iliofotou, I. Neamtiu, and M. Faloutsos. 2012. Graph-based analysis and prediction for software evolution. In *International Conference on Software Engineering (ICSE)*. 419–429. https://doi.org/10.1109/ICSE.2012.6227173

[2] Christian Bird, Nachiappan Nagappan, Harald Gall, Brendan Murphy, and Premkumar Devanbu. 2009. Putting It All Together: Using Socio-technical Networks to Predict Failures. In *International Symposium on Software Reliability Engineering (ISSRE '09)*. IEEE Computer Society, Washington, DC, USA, 109–119. https://doi.org/10.1109/ISSRE.2009.17

[3] Kelly Blincoe, Francis Harrison, Navpreet Kaur, and Daniela Damian. 2019. Reference Coupling: An exploration of inter-project technical dependencies and their characteristics within large software ecosystems. *Information and Software Technology* 110 (2019), 174 – 189. https://doi.org/10.1016/j.infsof.2019.03.005

[4] Christopher Bogart, Christian Kästner, James Herbsleb, and Ferdian Thung. 2016. How to Break an API: Cost Negotiation and Community Values in Three Software Ecosystems. In *International Symposium on Foundations of Software Engineering (FSE)*. ACM, 109–120. https://doi.org/10.1145/2950290.2950325

[5] Marcelo Cataldo and James D. Herbsleb. 2008. Communication Networks in Geographically Distributed Software Development. In *ACM Conference on Computer Supported Cooperative Work (CSCW '08)*. ACM, New York, NY, USA, 579–588. https://doi.org/10.1145/1460563.1460654

[6] Eleni Constantinou and Tom Mens. 2017. Socio-technical evolution of the Ruby ecosystem in GitHub. In *IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 34–44. https://doi.org/10.1109/SANER.2017.7884607

[7] M. Conway. 1968. How do Committees Invent? *Datamation Journal* (April 1968), 28–31.

[8] Laura A. Dabbish, H. Colleen Stuart, Jason Tsay, and James D. Herbsleb. 2012. Social coding in GitHub: transparency and collaboration in an open software repository. In *Int'l Conf. Computer Supported Cooperative Work*. 1277–1286.

[9] Alexandre Decan, Tom Mens, and Maelick Claes. 2017. An empirical comparison of dependency issues in OSS packaging ecosystems. In *International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 2–12. https://doi.org/10.1109/SANER.2017.7884604

[10] 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 (February 2019), 381–416. https://doi.org/10.1007/s10664-017-9589-y

[11] Jens Dietrich, David J. Pearce, Jacob Stringer, Amjed Tahir, and Kelly Blincoe. 2019. Dependency Versioning in the Wild. In *International Conference on Mining Software Repositories (MSR)*.

[12] Mohammad Gharehyazie, Daryl Posnett, and Vladimir Filkov. 2013. Social Activities Rival Patch Submission For Prediction of Developer Initiation in OSS Projects. In *Int'l Conf. Software Maintenance*.

[13] J. D. Herbsleb and R. E. Grinter. 1999. Architectures, coordination, and distance: Conway's law and beyond. *IEEE Software* 16, 5 (1999), 63–70.

[14] Jeremy Katz. 2018. Libraries.io Open Source Repository and Dependency Metadata (Version 1.4.0) [Data set]. http://doi.org/10.5281/zenodo.2536573.

[15] R. Kikas, G. Gousios, M. Dumas, and D. Pfahl. 2017. Structure and Evolution of Package Dependency Networks. In *International Conference on Mining Software Repositories (MSR)*. 102–112. https://doi.org/10.1109/MSR.2017.55

[16] Ghadeer A. Kintab, Chanchal K. Roy, and Gordon I. McCalla. 2014. Recommending Software Experts Using Code Similarity and Social Heuristics. In *Proceedings of 24th Annual International Conference on Computer Science and Software Engineering (CASCON '14)*. IBM Corp., Riverton, NJ, USA, 4–18. http://dl.acm.org/citation.cfm?id=2735522.2735526

[17] I. Kwan, A. Schroter, and D. Damian. 2011. Does Socio-Technical Congruence Have an Effect on Software Build Success? A Study of Coordination in a Software Project. *IEEE Trans. Soft. Eng.* 37, 3 (May 2011), 307–324. https://doi.org/10.1109/TSE.2011.29

[18] Luis Lopez-Fernandez, Gregorio Robles, Jesus Gonzalez-Barahona, and Israel Herraiz. 2009. Applying Social Network Analysis Techniques to Community-driven Libre Software Projects. In *Integrated Approaches in Information Technology and Web Engineering: Advancing Organizational Knowledge Sharing*. IGI Global, Chapter 3, 28–50. https://doi.org/10.4018/978-1-60566-418-7.ch003

[19] Mircea Lungu. 2009. *Reverse Engineering Software Ecosystems*. Ph.D. Dissertation. University of Lugano.

[20] Konstantinos Manikas and Klaus Marius Hansen. 2013. Software Ecosystems: A Systematic Literature Review. *J. Systems and Software* 86, 5 (May 2013), 1294–1306. http://dx.doi.org/10.1016/j.jss.2012.12.026

[21] Tom Mens, Marcelo Cataldo, and Daniela Damian. 2019. The Social Developer: The Future of Software Development. *IEEE Software* 36 (January–February 2019). https://doi.org/10.1109/MS.2018.2874316

[22] M. Palyart, G. C. Murphy, and V. Masrani. 2018. A Study of Social Interactions in Open Source Component Use. *IEEE Transactions on Software Engineering* 44, 12 (dec 2018), 1132–1145. https://doi.org/10.1109/TSE.2017.2756043

[23] Gregorio Robles, Jesus M. Gonzalez-Barahona, and Israel Herraiz. 2009. Evolution of the core team of developers in libre software projects. In *Int'l Conf. Mining Software Repositories*. IEEE Computer Society, 167–170.

[24] Didi Surian, Yuan Tian, David Lo, Hong Cheng, and Ee-Peng Lim. 2013. Predicting Project Outcome Leveraging Socio-Technical Network Patterns. In *European Conf. Software Maintenance and Reengineering*.

[25] Minghui Zhou and Audris Mockus. 2011. Does the initial environment impact the future of developers?. In *Int'l Conf. Software Engineering*. ACM, 271–280. https://doi.org/10.1145/1985793.1985831

[26] Minghui Zhou and Audris Mockus. 2012. What make long term contributors: willingness and opportunity in OSS community. In *Int'l Conf. Software Engineering*. IEEE Press, 518–528.