The Role of Dependency Networks in Developer Participation Decisions in Open Source Software Ecosystems: An Application of Stochastic Actor-Oriented Models

Abstract

Open source software relies on the contributions of developers who participate voluntarily in projects. While prior research has investigated social characteristics, relations, and connections that influence a developer's participation, we argue that the technical relations and connections of projects, which emerge through dependencies between packages in software ecosystems, play a focal role in that decision as well. We empirically test these assertions by applying stochastic actor-oriented models to an affiliation network in the JavaScript software ecosystem. Our results show that while the number of dependencies of a project does not influence participation, developers are more likely to participate in projects to which their own projects have dependency relations. This study thereby contributes to the understanding of antecedents that influence developers' participation decisions by highlighting the importance of project interdependencies in software ecosystems.

Keywords: Open Source Software, Software Ecosystems, Longitudinal Network Analysis, Dynamic Network Evolution, Affiliation Networks.

1. Introduction

Unlike software development in organizations, *open source software (OSS)* projects are usually undertaken by a decentralized community of developers who collaborate via development platforms to produce the software (Fang & Neufeld, 2009; Lindberg et al., 2016). Often, these developers are not paid (Crowston, 2011; Roberts et al., 2006), although a fraction is employed by companies specifically to help in OSS development (von Krogh et al., 2012). Thereby, OSS projects depend on the continuous, voluntary participation of distributed developers (Mockus et al., 2002; Roberts et al., 2006).

Previous research has therefore investigated what factors influence a developer's decision to participate in a project. In doing so, studies have focused on individual-related factors that lead to intrinsic and extrinsic motivations (von Krogh et al., 2012), as well as project-related factors, such as organizational sponsorship or license restrictions (Stewart et al., 2006). Researchers also have taken the social structure of projects and their community into account by applying techniques from social network analysis (e.g., Grewal et al., 2006; Hahn et al., 2008; Oh & Jeon, 2007).

While prior research has shown that social relations and connections are important antecedents of participation, existing studies have ignored another essential component: *technical relations and connections*, that is, dependencies that arise in software ecosystems through the reuse of software packages (Decan et al., 2019; Haefliger et al., 2008). We argue that these technical connections play an important role in developers' participation decision due to four key reasons.

First, packages that are reused extensively by others are important for the health and stability of the entire ecosystem, making them more attractive for developers who want to gain reputation and become visible in the community (Hu et al., 2012). Second, by reusing packages, developers can work on tasks they actually enjoy working on (Haefliger et al., 2008), therefore making packages with more reuse more attractive for potential participants. Third, developers tend to support other projects that depend on their provided package (Bogart et al., 2016). In some cases, the provided packages are even spun off from larger projects, which are then maintained by the same developers (Valiev et al., 2018). Fourth, developers tend to participate in projects that they use themselves (Shah, 2006), which is reflected in a dependency towards the used package.

The aim of this study is to empirically test these assertions. To do so, we theorize the role of dependency networks for developers' decisions based on prior literature. Then, we adopt a dynamic network modeling approach and analyze the affiliation network of packages in a large OSS ecosystem over a period of four months. We apply stochastic actor-oriented models (Snijders, 1996) to investigate the effect of software dependencies on the evolution of affiliation

networks between developers and OSS projects. We find that while the number of up- and downstream dependencies of a project does not affect its ability to attract participants, developers tend to contribute to projects that are either up- or downstream dependencies of that particular project. These findings contribute to the literature on developer participation in OSS projects by focusing on the technical connections in the form of package interdependencies of OSS projects in software ecosystems. This offers a more complete view than the prevailing focus on social relations and connections only.

The remainder of this paper is structured as follows. In Section 2, we provide an overview of related work on developer participation and dependency networks in OSS ecosystems, and we develop our hypotheses. In Section 3, we describe our data collection process and network construction, and give a brief overview about stochastic actor-oriented models. In Section 4, we report the results of our analysis. Finally, in Section 5, we discuss our results, implications, and limitations.

2. Theoretical Background

2.1. Developer Participation in Open Source Software Projects

OSS development is driven by a decentralized community of developers that mostly contribute voluntarily to projects and collaborate via online development platforms and management software such as GitHub (Fang & Neufeld, 2009; Roberts et al., 2006). OSS projects rely on the continuous participation of their community members (Roberts et al., 2006; Shah, 2006) and need to attract and retain developers (Butler, 2001; Crowston et al., 2003) in order to stay viable. Therefore, the question of what motivates developers to participate in a particular OSS project has been central to OSS research (Roberts et al., 2006).

Previous research has focused on individual characteristics of developers as well as project-related aspects that influence developers' participation decisions, and various project- and individual-related factors have been identified. For example, project factors include license restrictions (Stewart et al., 2006), organizational sponsorship (Shah, 2006; Stewart et al., 2006), and the modularity of a project's codebase (Baldwin & Clark, 2006). Individual factors that drive participation include fun or enjoyment (Shah, 2006), learning and developing skills (von Hippel & von Krogh, 2003), or increasing reputation (Hu et al., 2012) and career advancements (Lerner & Tirole, 2002).

Moreover, due to the community-based model of developing OSS and the importance of social relations, connections, and structures (Grewal et al., 2006), researchers early on have adopted a network perspective on OSS and have investigated the effect of network structures on participation. Related to the social network of developers, for example, previous collaborations with the project initiators increase the likelihood of joining a project (Hahn et al., 2008), and the decision to remain involved in a project is influenced by other neighboring developers (Oh & Jeon, 2007).

While these network studies highlight the importance of social interactions of developers, they largely neglect the technical relations and connections in the form of package interdependencies between projects. In the following, we focus on these interdependencies that arise in projects embedded in software ecosystems.

2.2. Dependency Networks in Software Ecosystems

OSS is not built from scratch but relies on reuse of code and already implemented functionality (Haefliger et al., 2008; Sojer & Henkel, 2010). This functionality is typically provided via *packages*, which are "reusable code or set of components that can be included in other applications by using dependency management tools" (Kikas et al., 2017). Modern programming languages ease the process of reuse by providing package managers that allow developers to publish and use packaged software components (Cox, 2019). Adding a package to a project creates a dependency relationship, making the project dependent on the package to function (Bogart et al., 2016). This practice results in so-called *software ecosystems*, "large collections of interdependent software components that are maintained by large and geographically distributed communities of collaborating contributors" (Decan et al., 2019).

From a package's perspective, dependency relationships exist in two directions. In the ecosystem, the package has other packages depending on it, so-called *downstream dependencies*, and it might also depend on other packages itself, which results in *upstream dependencies* (Valiev et al., 2018).

The reuse of packages allows developers to save time and to implement proven solutions to their software (Haefliger et al., 2008; von Hippel & von

Krogh, 2003). The functionality provided by the reused package enables them to focus on tasks they actually like to work on (Haefliger et al., 2008). Therefore, projects with extensive reuse of packages, reflected in the number of upstream dependencies, should become more attractive for developers, which leads to our first hypothesis:

H1a: Developers tend to participate in packages with more upstream dependencies.

Furthermore, the number of downstream dependencies reflects the importance and value of a package in the software ecosystem. This is because the more other packages depend on the focal package, the higher the number of downstream dependencies, which also makes it a proxy for a package's user base (Valiev et al., 2018). Since one driving factor of developer participation is the potential gain in reputation (Hu et al., 2012), important and valued packages in the software ecosystem should become more attractive to participate in. Hence, we propose:

H1b: Developers tend to participate in packages with more downstream dependencies.

However, dependencies can create issues in case of breaking changes (i.e., a change in a package that potentially causes other packages to fail). In order to counter breaking changes, package providers usually support and coordinate with their dependents by announcing changes or helping to migrate to another version (Bogart et al., 2016). Moreover, smaller packages are often split off from larger projects, which are then maintained by the same developers (Valiev et al., 2018). Therefore, developers also become affiliated with a package's upstream dependencies. Hence, we propose:

H2a: Developers tend to participate in a package when they are also affiliated with its upstream dependencies.

Furthermore, existing studies have shown that developers tend to participate in projects that they use themselves (Shah, 2006). These developers, often referred to as "peripheral developers", are thereby motivated to improve the package for their own use and are important for the quality assessment and enhancement of a project (Setia et al., 2012). For example, developers participate by submitting bug reports or pull requests to the dependent package (Setia et al., 2012). Hence, we propose that this also holds true for the reuse of packaged software components:

H2b: Developers tend to participate in a package when they are also affiliated with its downstream dependencies.

3. Research Method

3.1. Data and Network Construction

In order to test our hypotheses, we focus on the JavaScript ecosystem, which is one of the largest software ecosystems in the world (Decan et al., 2019). Data was collected from two data sources to construct the dependency network between packages and capture the development activities. Meta and dependency data was collected from the npm registry[1], whereas development activities were collected from GitHub for the package-related repositories. Due to API restrictions, we used the event archives provided by GHArchive[2]. Both datasets were linked by matching repository URLs provided in the packages' metadata.

In network studies, setting boundary conditions is important (Marsden, 2005). For our boundary specification and sampling strategy, we followed the "expanding selection" approach (Doreian and Woodard, (1992), which starts with a fixed set of nodes and adds further nodes linked to the initial set. Thus, we started our data collection with the selection of 3,000 packages identified from Libraries.io's list of top ranked packages[3] in March 2022. Based on that initial set, we added packages to the set that were listed as a runtime dependency between 2019 and 2022. This time restriction helped in avoiding adding abandoned or by now irrelevant packages. The process was repeated for every newly identified package which allowed us to identify all relevant packages further down the dependency tree. In sum, this resulted in a total of 12,678 packages.

To reduce the number of nodes and thereby make the size of the network feasible for the analysis, we performed additional steps for the selection of our final sample of packages and related developers. First, we excluded all packages that shared a repository with other packages to make sure that developer activities were specifically targeted at a particular package. Second, we checked for the development activity during the observation and only included packages with activity in every period. From the resulting set of packages, we randomly selected 250 packages. Third, we collected all developers participating in the sampled packages and only included developers with at least 5 activities (i.e., comments, commits, actions related to

[1] registry.npmjs.org
[2] gharchive.org
[3] libraries.io/search?order=desc&platforms=npm&sort=rank

issues and pull requests) during the observation. This resulted in a set of 1,172 developers.

Based on the selected packages and identified developers, we constructed the affiliation network between developers and packages. An affiliation network, also referred to as two-mode or bipartite network, is a graph with two distinct node types (i.e., developers and packages), where edges are only allowed between different types of nodes (Koskinen & Edling, 2012). To construct the network, we collected all activities of the selected developers towards the sampled packages for each observation period and created edges between developer and package in case there existed an activity in the particular period.[4]

We observed the affiliation network from February until May 2021. We opted for the four-month window to reduce time heterogeneity and keep the amount of change between periods at a sufficient level for analysis, which is also consistent with previous research (e.g., Hahn et al., 2008; Tang et al., 2020). Thereby, we created snapshots of the network state at the beginning of each month. This resulted in a total of four observations. Figure 1 shows the state of the affiliation network at the last observation point. The layout was generated using the Fruchterman-Reingold algorithm (Fruchterman & Reingold, 1991).

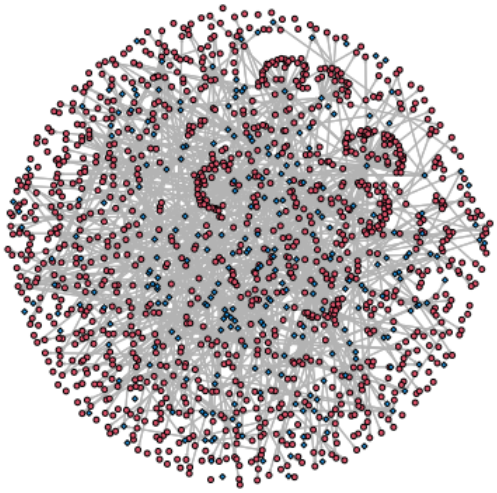

Figure 1. Affiliation network at observation t_4 (developers as red circles; packages as blue squares).

3.2. Data Analysis

We applied stochastic actor-oriented modeling (SAOM) (Snijders, 1996, 2001, 2005) by using the R package RSiena (Ripley et al., 2022). In the following, we briefly summarize the underlying assumptions behind SAOMs. For a detailed description, we refer to Snijders (1996, 2001) and Snijders et al. (2010).

SAOMs are the most advanced predictive models for dynamic networks that allow the testing of various mechanisms influencing the evolution of a network (Cornwell, 2015). SAOMs assume a continuous change process of the network structure, which is represented by Markov chain models (Holland & Leinhardt, 1977) with a continuous-time parameter, although the network is observed at discrete points in time (Snijders, 1996, 2001). The change process consists of two sub-processes: the (1) change opportunity process and the (2) change determination process (Snijders et al., 2010). Thereby, when an actor has the opportunity to change ties, determined by the rate function, the probabilities of change are determined by the evaluation (or objective) function (Snijders et al., 2010). The evaluation function thereby represents the relative attractiveness of establishing a tie (Conaldi et al., 2012). The evaluation function includes effects related to the structural properties of the network (endogenous effects) and effects based on the attributes of an actor in the network (exogenous effects) (Snijders et al., 2010).

3.3. Model Development

We followed the guidelines for model development provided by Snijders et al. (2010) and Ripley et al. (2022). Thereby, we specified the model for the dynamics in the affiliation network via forward selection of theoretically grounded effects and tested these effects using the score-type test proposed by Schweinberger (2012). During the selection process, we checked the t-ratios for convergence for each effect, which indicate the stability of parameter estimates across simulations and should be below an absolute value of 0.1 (Kalish, 2020; Snijders et al., 2010). Furthermore, we checked the overall maximum convergence of the estimated models during the selection process, which should be below the threshold of 0.25 (Ripley et al., 2022). We started with endogenous effects, followed by exogenous effects. Results were then validated by performing a backward selection. We also tested for time heterogeneity (J. A. Lospinoso et al., 2011) and accounted for the composition change of developers (Huisman & Snijders, 2003).

In terms of endogenous effects, by default, an effect for *outdegree* (density), which represents the tendency of an actor to have ties at all, is included in

[4] All data and scripts to construct the data as well as the analysis results can be found here: https://tinyurl.com/yupp59rn

the model (Snijders et al., 2010). Also, an effect for the tendency toward *transitivity* should be included in the model (Snijders et al., 2010). In two-mode networks, transitivity is expressed by the number of four-cycles (Robins & Alexander, 2004). This effect reflects the extent to which actors make the same choices as their peers (Ripley et al., 2022). Another set of effects that should be accounted for during the model selection process are degree-related effects (Snijders et al., 2010). Both in- and out-degree are important positional characteristics of nodes that drive network dynamics (Snijders et al., 2010) and should be included when high dispersion in in- and out-degrees is present (Ripley et al., 2022). Based on our theoretical foundation, we included the *in-degree popularity* effect, which accounts for the tendency of dispersion in in-degrees of packages (i.e., number of participating developers). The *out-degree activity* effect, which reflects the tendency of dispersion in out-degree of developers (i.e., the number of packages a developer participates in), was also significant and contributed to the convergence of the overall model and was therefore also included.

Furthermore, we included several exogenous actor-specific effects. We started by including control effects. Related to packages, we controlled for *community interest* (measured as the number of GitHub stars given to a package during a period), *release activity* (measured as the number of version releases of a package during a period), *license restrictiveness* (by adopting the categorization of Lerner and Tirole (2005) into (1) permissive, (2) restrictive, and (3) highly restrictive licenses), and *age* (measured as the number of months from the package's creation date until the end of a period). These effects were modeled as receiver effects, which means that actors with higher values of the covariate tend to have higher in-degrees (Snijders et al., 2010). Related to developers, we controlled for the overall *activity of a developer* by measuring the number of activities a developer performed during a period. These include comments made on issues or pull requests, status changes of issues or pull requests (e.g., opening or closing), and pushing commits to the repository. This effect was modeled as a sender effect, which means that actors with higher covariate values tend to have higher out-degrees (Snijders et al., 2010).

Finally, we included exogenous actor-specific receiver effects as well as dyadic effects for the influence of the dependency network. First, we measured the number of *up- and downstream dependencies* of a package in a period. Second, we included dyadic effects accounting for the relation between a developer and a package. The dyadic covariate thereby reflects if the developer also *participates in an upstream or downstream dependency of the targeted package* in the period. Therefore, the effect expresses the extent to which participation becomes more likely if a developer also participates in an upstream or downstream dependency.

Because the estimation operations of RSiena are not scale-independent, it is advised to scale covariates to achieve standard deviations between 0.1 and 10 (Ripley et al., 2022). Therefore, we log-transformed the values of our actor-specific covariates. Table 1 summarizes the relevant effects used in this study.

Table 1. Summary of endogenous network effects and exogenous actor-specific covariates.

Parameter	Description
Outdegree (Density)	Tendency of developers to participate in packages.
Transitivity	Tendency of developer pairs to participate in the same package.
Package Popularity	Tendency of popular packages to attract more developers.
Developer Participation	Tendency for developers that participate in more packages to engage in extra packages.
Package License	Tendency of developers to participate in packages with specific license restrictiveness.
Developer Activity	Tendency of developers with a higher level of activity to participate in more packages.
Community Interest	Tendency of packages with higher community interest to attract more developers.
Release Activity	Tendency of packages with more releases to attract more developers.
Package Age	Tendency of packages with higher age to attract more developers.
Package Upstream Dependencies	Tendency of packages with more upstream dependencies to attract more developers.
Package Downstream Dependencies	Tendency of packages with more downstream dependencies to attract more developers.
Participation in Upstream Dependency	Tendency of developers to participate in a package if they also participate in an upstream dependency of that package.
Participation in Downstream Dependency	Tendency of developers to participate in a package if they also participate in a downstream dependency of that package.

4. Results

First, Table 2 summarizes network descriptives. Over all periods, the network's density is relatively low. During all four periods, the network's density, and its average degree declines. This is a result of the

joining actors in periods 2 to 4. Therefore, the number of possible ties increases, but the number of ties that are actually established does not increase accordingly.

Table 2. Network descriptives.

Period	1	2	3	4
Density	0.005	0.004	0.003	0.003
Avg. Degree	1.211	0.909	0.736	0.647
No. Ties	758	807	788	758
Miss. Fraction	0.466	0.242	0.087	0.000
Joined Actors	–	262	182	102

Second, we report the tie changes of the networks in subsequent observations (Table 3). For example, between observation 1 and 2 (1 → 2), 88 developers newly participated in packages (0 → 1), 325 developers discontinued their participation (1 → 0), and 433 developers continued to participate in the related packages. The Jaccard index represents the amount of change between two observations (Snijders et al., 2010). For SAOMs, the suggested value is between 0.2 and 0.9 (Conaldi et al., 2012). In our case, the values of the Jaccard coefficients for tie changes between observations are between 0.45 and 0.51.

Table 3. Network tie changes between periods.

	0 → 0	0 → 1	1 → 0	1 → 1	Distance	Jaccard
1 → 2	155654	88	325	433	413	0.51
2 → 3	221043	150	362	445	512	0.47
3 → 4	266517	195	345	443	540	0.45

4.1. Model Estimates

We estimated the models using the procedure of method of moments (Snijders, 1996). Table 4 presents the results of the models. We report parameter estimates and standard errors for rate effects, endogenous network effects, actor-specific and dyadic covariates. Model 1 includes the effects of the endogenous network structure. Model 2 adds to Model 1 the effects of exogenous actor-specific covariates for our control variables. Model 3 adds to Model 2 the effects of the dependency network related covariates (i.e., our hypotheses). All models were run for 3,000 iterations in phase 3. In all models, t-ratios for convergence for each effect are below the suggested threshold of |0.1| (Kalish, 2020; Snijders et al., 2010) and the overall maximum convergence ratios are below the suggested value of 0.25 (Ripley et al., 2022).

In the following, we report the estimates for Model 3 in more detail. In general, the rate function indicates the expected number of opportunities that developers have to change their affiliation with a project (Conaldi et al., 2012). Hence, the parameter estimates can be interpreted as the number of changes developers make regarding their affiliations over time. For example, developers make on average 0.8 changes in the last period. This rate remains relatively stable over time and indicates that developers are reluctant to change their affiliation with a project. Furthermore, the developer's activity level has a positive and significant effect on the number of change opportunities (0.05; $p < 0.01$), indicating that more active developers tend to change their affiliation more often.

The evaluation function controls for the subjective utility for developers when changing their affiliation (Conaldi et al., 2012). In terms of endogenous network effects, we observe that the estimate for the out-degree of the developers is negative and highly significant (-5.52; $p < 0.001$). This indicates that developers show a lower tendency to participate in new packages over time. Also, package popularity has a small but positive and significant effect (0.04; $p < 0.001$), which indicates that already popular packages are more likely to attract additional developers. Transitivity is positive but not statistically significant (0.52; $p < 0.1$), which does not indicate a significant tendency towards clustering in the network. Thus, developers do not seem to follow their previous collaborators to new packages in the future. The parameter estimate for developer participation is not significant.

In terms of exogenous effects of actor-specific covariates, we first focus on the effects of interest related to the effect of a package's dependency network on developer participation. We find that the number of up- and downstream dependencies of a package does not influence its ability to attract developers, with both estimates being not significant (both *H1a* and *H1b* are not supported). However, the estimates for both dyadic effects of up- (1.93; $p < 0.001$) and downstream dependencies (1.34; $p < 0.01$) are both positive and significant. This indicates that developers are more likely to participate in a package if they also participate in another package that is a down- or upstream dependency of that specific package. Hence, we observe support for both *H2a* and *H2b*.

We conclude by reporting estimates for our control variables. For packages, the estimates for community interest (0.20; $p < 0.001$) and package age (-0.20; $p < 0.01$) are significant. Furthermore, the estimate for developer activity is positive and highly significant (0.47; $p < 0.001$), which indicates that developers with a higher level of activity tend to participate in more packages. In contrast to prior findings, both release activity (0.13; $p < 0.1$) and license restrictiveness (0.31; $p > 0.1$) of a package are not significantly influencing a package's ability to attract developers.

Table 4. Estimated stochastic actor-oriented models for affiliation networks.

	Model 1		Model 2		Model 3	
Effects	Estim.	S.E.	Estim.	S.E.	Estim.	S.E.
Rate Function						
Rate of Network Change 1	0.72***	(0.04)	0.77***	(0.04)	0.77***	(0.04)
Rate of Network Change 2	0.74***	(0.04)	0.81***	(0.04)	0.81***	(0.04)
Rate of Network Change 3	0.70***	(0.04)	0.80***	(0.04)	0.80***	(0.04)
Effect of Dev. Activity on Rate	-0.10***	(0.02)	0.05*	(0.02)	0.05**	(0.02)
Evaluation Function						
Endogenous Network Effects						
Outdegree (Density)	-5.27***	(0.14)	-5.48***	(0.12)	-5.52***	(0.13)
Transitivity (Four-Cycles)	0.50	(0.38)	0.54*	(0.25)	0.52†	(0.29)
Package Popularity	0.05***	(0.00)	0.04***	(0.00)	0.04***	(0.01)
Developer Participation	0.27***	(0.04)	0.05	(0.04)	0.04	(0.04)
Exogenous Actor-specific Covariates						
Developer Activity			0.49***	(0.04)	0.47***	(0.04)
Community Interest			0.19***	(0.04)	0.20***	(0.05)
Release Activity			0.12	(0.08)	0.13†	(0.08)
License Restrictiveness			0.31	(0.33)	0.31	(0.33)
Package Age			-0.18*	(0.08)	-0.20**	(0.08)
Upstream Dependencies					0.05	(0.05)
Downstream Dependencies					0.08	(0.06)
Dyadic Covariates						
Participation in Upstream Dependency					1.93***	(0.56)
Participation in Downstream Dependency					1.34**	(0.46)
Wald χ^2 Statistics (df)	320.19*** (4)		231.63*** (5)		14.64*** (4)	
Gen. score χ^2 Statistics (df)	394.84*** (4)		198.98*** (5)		19.20*** (4)	

† $p < 0.1$, * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Convergence t-ratios for all effects < |0.1|. Overall maximum convergence ratios < 0.13

4.2. Goodness-of-Fit

The simulation-based goodness-of-Fit (GOF) test for the estimated models tests the hypothesis that the model which generated the observed data is equal to the fitted model (J. Lospinoso & Snijders, 2019). The approach implemented in RSiena takes an auxiliary statistic, that is, a feature of the data not included in the model and therefore not a function of the estimation and compares it with the observed data and their distribution (J. Lospinoso & Snijders, 2019). We tested auxiliary statistics for outdegree and indegree distributions. Both Model 1 and 2 performed poorly for both in- and outdegree distributions, but the fit for Model 3 meets the criteria of $p > 0.05$ for the Mahalanobis distance-combination, indicating a good model fit (Kalish, 2020; J. Lospinoso & Snijders, 2019).

Figure 2 shows the results of the GOF tests for Model 3. Observed values are indicated by the number connected by the red line. The simulated statistics are represented by the violin plots. The dotted lines represent the 95th percentile bands. Wald-type tests and score-type tests for the joint significance of the added effects as reported in Table 4 also indicate an improvement of model fit and strong significance of the added effects ($p < 0.001$).

5. Discussion

In this study, we developed and estimated a dynamic network model for the analysis of the evolution of an affiliation network in a large OSS ecosystem. With this model and the test of associated hypotheses, we contribute to the literature on the participation decisions of developers by focusing on the role of technical relations and connections in the form of package interdependencies, thus introducing (package-based) project dependencies as important antecedents for developer participation decisions.

Our results show that developers not only contribute to packages they use themselves (*H2b*), but also to packages that make use of their own packages (*H2a*). This shows that projects benefit from their dependencies in both directions through contributions made by developers of interdependent projects. While previous research already mentioned need-driven motivation as one antecedent for participation (Shah, 2006), this empirically shows for the first time that users of a package do not free-ride but also contribute

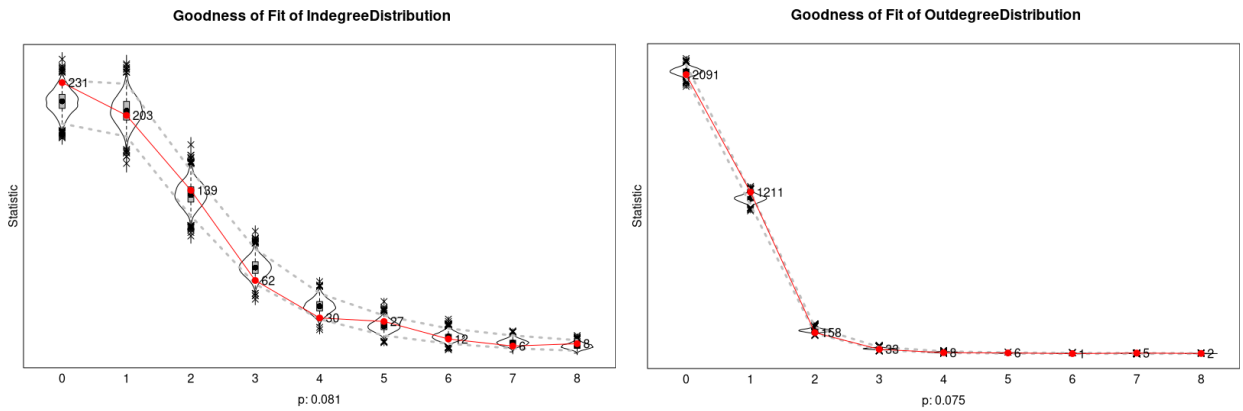

Figure 2. Goodness of fit of Model 3 for indegree and outdegree distributions.

back. Moreover, our findings show that package providers contribute and provide help to their dependent packages. Even though we find that the likelihood of contribution increases, the actual type of the contribution remains an open question and provides opportunities for future research. However, the number of dependencies by themselves do not influence a developer's decision; a large number of up- or downstream dependencies does not equal more attractiveness, thus we did not find support for *H1a* and *H1b*.

Furthermore, our results show that developers only rarely change affiliations, as reflected in the rate effects. Given that the participation in a new project comes with associated costs related to required knowledge, skill, and necessary time to get involved and familiar with a project (von Krogh et al., 2003), this is not surprising.

In comparison to prior studies, our results support previous findings related to the influence of community interest on a project's attractiveness (Subramaniam et al., 2009) and its decreasing ability to attract developers with growing age (Chengalur-Smith et al., 2010). Interestingly, we did not find an effect of license restrictiveness. This might be related to the fact that most of the analyzed packages are released under the MIT license and, in general, we did not see a great variety of used licenses in the overall JavaScript ecosystem.

From a research perspective, our study demonstrates the benefits and potential insights that can be gained by applying dynamic network models to affiliation networks in OSS projects. From a practical perspective, our results highlight that community efforts should be directed not only towards a project itself, but also to interdependent projects that build upon or are used by the focal project. This may also help to counter negative effects such as breaking changes.

As with all research, this study has several limitations. First, we did not include all available packages in the JavaScript ecosystem. However, by following the selected sampling approach, we were able to identify and analyze the most important and used packages during our observation. Furthermore, we only focused on one specific ecosystem. Hence, future research could analyze if the shown mechanisms are also present and influential in other software ecosystems.

Second, we focused only on effects driving the structural evolution and formation of the affiliation network and neglected the co-evolutionary aspect of its structure on potential outcomes, such as a project's sustainability and success. Hence, future research should build upon this study by including project-related outcomes and their interplay with both social and technical network structures.

Third, the dependency network has only partially been included in our analysis by projecting it as actor and tie variables. Future research could therefore explicitly include its structure by investigating the co-evolution of the dependency and affiliation network.

Fourth, we used digital trace data, which entails potential validity problems (Howison et al., 2011). Even though we performed several checks to increase our data's confidentiality, we cannot ensure complete accuracy due to the secondary nature of our data sources.

6. Conclusion

In sum, our study theoretically and practically contributes to our understanding of antecedents of

developer participation in OSS by introducing and highlighting the role of technical interdependencies of projects in a software ecosystem. Thereby, we underline the importance of a socio-technical lens on the OSS phenomena that considers the social as well as technical structures and provide several opportunities and directions for future research.

7. References

Baldwin, C. Y., & Clark, K. B. (2006). The Architecture of Participation: Does Code Architecture Mitigate Free Riding in the Open Source Development Model? *Management Science*, *52*(7), 1116–1127. https://doi.org/10.1287/mnsc.1060.0546

Bogart, C., Kästner, C., Herbsleb, J., & Thung, F. (2016). How to break an API: cost negotiation and community values in three software ecosystems. *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 109–120. https://doi.org/10.1145/2950290.2950325

Butler, B. S. (2001). Membership Size, Communication Activity, and Sustainability: A Resource-Based Model of Online Social Structures. *Information Systems Research*, *12*(4), 346–362. https://doi.org/10.1287/isre.12.4.346.9703

Chengalur-Smith, I., Sidorova, A., & Daniel, S. (2010). Sustainability of Free/Libre Open Source Projects: A Longitudinal Study. *Journal of the Association for Information Systems*, *11*(11), 657–683. https://doi.org/10.17705/1jais.00244

Conaldi, G., Lomi, A., & Tonellato, M. (2012). Dynamic Models of Affiliation and the Network Structure of Problem Solving in an Open Source Software Project. *Organizational Research Methods*, *15*(3), 385–412. https://doi.org/10.1177/1094428111430541

Cornwell, B. (2015). *Social Sequence Analysis: Methods and Applications*. Cambridge University Press.

Cox, R. (2019). Surviving Software Dependencies. *Communications of the ACM*, *62*(9), 36–43. https://doi.org/10.1145/3347446

Crowston, K. (2011). Lessons from Volunteering and Free/Libre Open Source Software Development for the Future of Work. In M. Chiasson, O. Henfridsson, H. Karsten, & J. I. DeGross (Eds.), *Researching the Future in Information Systems* (Vol. 356, pp. 215–229). Springer. https://doi.org/10.1007/978-3-642-21364-9_14

Crowston, K., Annabi, H., & Howison, J. (2003). Defining Open Source Software Project Success. *Proceedings of the Twenty-Fourth International Conference on Information Systems (ICIS)*.

Decan, A., Mens, T., & Grosjean, P. (2019). An empirical comparison of dependency network evolution in seven software packaging ecosystems. *Empirical Software Engineering*, *24*, 381–416. https://doi.org/10.1007/s10664-017-9589-y

Doreian, P., & Woodard, K. L. (1992). Fixed list versus snowball selection of social networks. *Social Science Research*, *21*(2), 216–233. https://doi.org/10.1016/0049-089X(92)90016-A

Fang, Y., & Neufeld, D. (2009). Understanding Sustained Participation in Open Source Software Projects. *Journal of Management Information Systems*, *25*(4), 9–50. https://doi.org/10.2753/MIS0742-1222250401

Fruchterman, T. M. J., & Reingold, E. M. (1991). Graph drawing by force-directed placement. *Software: Practice and Experience*, *21*(11), 1129–1164. https://doi.org/10.1002/spe.4380211102

Grewal, R., Lilien, G. L., & Mallapragada, G. (2006). Location, Location, Location: How Network Embeddedness Affects Project Success in Open Source Systems. *Management Science*, *52*(7), 1043–1056. https://doi.org/10.1287/mnsc.1060.0550

Haefliger, S., von Krogh, G., & Spaeth, S. (2008). Code Reuse in Open Source Software. *Management Science*, *54*(1), 180–193. https://doi.org/10.1287/mnsc.1070.0748

Hahn, J., Moon, J. Y., & Zhang, C. (2008). Emergence of New Project Teams from Open Source Software Developer Networks: Impact of Prior Collaboration Ties. *Information Systems Research*, *19*(3), 369–391. https://doi.org/10.1287/isre.1080.0192

Holland, P. W., & Leinhardt, S. (1977). A dynamic model for social networks. *The Journal of Mathematical Sociology*, *5*(1), 5–20. https://doi.org/10.1080/0022250X.1977.9989862

Howison, J., Wiggins, A., & Crowston, K. (2011). Validity Issues in the Use of Social Network Analysis with Digital Trace Data. *Journal of the Association for Information Systems*, *12*(12), 767–797. https://doi.org/10.17705/1jais.00282

Hu, D., Zhao, J. L., & Cheng, J. (2012). Reputation management in an open source developer social network: An empirical study on determinants of positive evaluations. *Decision Support Systems*, *53*(3), 526–533. https://doi.org/10.1016/j.dss.2012.02.005

Huisman, M., & Snijders, T. A. B. (2003). Statistical Analysis of Longitudinal Network Data With Changing Composition. *Sociological Methods & Research*, *32*(2), 253–287. https://doi.org/10.1177/0049124103256096

Kalish, Y. (2020). Stochastic Actor-Oriented Models for the Co-Evolution of Networks and Behavior: An Introduction and Tutorial. *Organizational Research Methods*, *23*(3), 511–534. https://doi.org/10.1177/1094428118825300

Kikas, R., Gousios, G., Dumas, M., & Pfahl, D. (2017). Structure and Evolution of Package Dependency Networks. *Proceedings of the 14th International Conference on Mining Software Repositories (MSR)*. https://doi.org/10.1109/MSR.2017.55

Koskinen, J., & Edling, C. (2012). Modelling the evolution of a bipartite network—Peer referral in interlocking directorates. *Social Networks*, *34*(3), 309–322. https://doi.org/10.1016/j.socnet.2010.03.001

Lerner, J., & Tirole, J. (2002). Some Simple Economics of Open Source. *The Journal of Industrial Economics*, *50*(2), 197–234. https://doi.org/10.1111/1467-6451.00174

Lerner, J., & Tirole, J. (2005). The Scope of Open Source Licensing. *The Journal of Law, Economics, and*

Organization, *21*(1), 20–56. https://doi.org/10.1093/jleo/ewi002

Lindberg, A., Berente, N., Gaskin, J., & Lyytinen, K. (2016). Coordinating Interdependencies in Online Communities: A Study of an Open Source Software Project. *Information Systems Research*, *27*(4), 751–772. https://doi.org/10.1287/isre.2016.0673

Lospinoso, J. A., Schweinberger, M., Snijders, T. A. B., & Ripley, R. M. (2011). Assessing and accounting for time heterogeneity in stochastic actor oriented models. *Advances in Data Analysis and Classification*, *5*, 147–176. https://doi.org/10.1007/s11634-010-0076-1

Lospinoso, J., & Snijders, T. A. B. (2019). Goodness of fit for stochastic actor-oriented models. *Methodological Innovations*, *12*(3). https://doi.org/10.1177/2059799119884282

Marsden, P. V. (2005). Recent Developments in Network Measurement. In P. J. Carrington, J. Scott, & S. Wasserman (Eds.), *Models and Methods in Social Network Analysis* (pp. 8–30). Cambridge University Press.

Mockus, A., Fielding, R. T., & Herbsleb, J. D. (2002). Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, *11*(3), 309–346. https://doi.org/10.1145/567793.567795

Oh, W., & Jeon, S. (2007). Membership Herding and Network Stability in the Open Source Community: The Ising Perspective. *Management Science*, *53*(7), 1086–1101. https://doi.org/10.1287/mnsc.1060.0623

Ripley, R. M., Snijders, T. A. B., Boda, Z., Vörös, A., & Preciado, P. (2022). *Manual for RSiena*. University of Oxford, Department of Statistics; Nuffield College. http://www.stats.ox.ac.uk/~snijders/siena/RSiena_Manual.pdf

Roberts, J. A., Hann, I.-H., & Slaughter, S. A. (2006). Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects. *Management Science*, *52*(7), 984–999. https://doi.org/10.1287/mnsc.1060.0554

Robins, G., & Alexander, M. (2004). Small Worlds Among Interlocking Directors: Network Structure and Distance in Bipartite Graphs. *Computational & Mathematical Organization Theory*, *10*(1), 69–94. https://doi.org/10.1023/B:CMOT.0000032580.12184.c0

Schweinberger, M. (2012). Statistical modelling of network panel data: Goodness of fit. *British Journal of Mathematical and Statistical Psychology*, *65*(2), 263–281. https://doi.org/10.1111/j.2044-8317.2011.02022.x

Setia, P., Rajagopalan, B., Sambamurthy, V., & Calantone, R. (2012). How Peripheral Developers Contribute to Open-Source Software Development. *Information Systems Research*, *23*(1), 144–163. https://doi.org/10.1287/isre.1100.0311

Shah, S. K. (2006). Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development. *Management Science*, *52*(7), 1000–1014. https://doi.org/10.1287/mnsc.1060.0553

Snijders, T. A. B. (1996). Stochastic actor-oriented models for network change. *The Journal of Mathematical Sociology*, *21*(1–2), 149–172. https://doi.org/10.1080/0022250X.1996.9990178

Snijders, T. A. B. (2001). The Statistical Evaluation of Social Network Dynamics. *Sociological Methodology*, *31*(1), 361–395. https://doi.org/10.1111/0081-1750.00099

Snijders, T. A. B. (2005). Models for Longitudinal Network Data. In P. J. Carrington, J. Scott, & S. Wasserman (Eds.), *Models and Methods in Social Network Analysis* (pp. 215–247). Cambridge University Press.

Snijders, T. A. B., van de Bunt, G. G., & Steglich, C. E. G. (2010). Introduction to stochastic actor-based models for network dynamics. *Social Networks*, *32*(1), 44–60. https://doi.org/10.1016/j.socnet.2009.02.004

Sojer, M., & Henkel, J. (2010). Code Reuse in Open Source Software Development: Quantitative Evidence, Drivers, and Impediments. *Journal of the Association for Information Systems*, *11*(12), 868–901. https://doi.org/10.17705/1jais.00248

Stewart, K. J., Ammeter, A. P., & Maruping, L. M. (2006). Impacts of License Choice and Organizational Sponsorship on User Interest and Development Activity in Open Source Software Projects. *Information Systems Research*, *17*(2), 126–144. https://doi.org/10.1287/isre.1060.0082

Subramaniam, C., Sen, R., & Nelson, M. L. (2009). Determinants of open source software project success: A longitudinal study. *Decision Support Systems*, *46*(2), 576–585. https://doi.org/10.1016/j.dss.2008.10.005

Tang, T. (Ya), Fang, E. (Er), & Qualls, W. J. (2020). More Is Not Necessarily Better: An Absorptive Capacity Perspective on Network Effects in Open Source Software Development Communities. *MIS Quarterly*, *44*(4), 1651–1678. https://doi.org/10.25300/MISQ/2020/13991

Valiev, M., Vasilescu, B., & Herbsleb, J. (2018). Ecosystem-Level Determinants of Sustained Activity in Open-Source Projects: A Case Study of the PyPI Ecosystem. *Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 644–655. https://doi.org/10.1145/3236024.3236062

von Hippel, E., & von Krogh, G. (2003). Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science. *Organization Science*, *14*(2), 209–223. https://doi.org/10.1287/orsc.14.2.209.14992

von Krogh, G., Haefliger, S., Spaeth, S., & Wallin, M. W. (2012). Carrots and Rainbows: Motivation and Social Practice in Open Source Software Development. *MIS Quarterly*, *36*(2), 649–676.

von Krogh, G., Spaeth, S., & Lakhani, K. R. (2003). Community, joining, and specialization in open source software innovation: A case study. *Research Policy*, *32*(7), 1217–1241. https://doi.org/10.1016/S0048-7333(03)00050-7