Advances in Complex Systems
Vol. 17, Nos. 7 & 8 (2014) 1550008 (24 pages)

DOI: 10.1142/S0219525915500083

World Scientific
www.worldscientific.com

HOW DO OSS PROJECTS CHANGE IN NUMBER AND SIZE? A LARGE-SCALE ANALYSIS TO TEST A MODEL OF PROJECT GROWTH

FRANK SCHWEITZER*, VAHAN NANUMYAN,
CLAUDIO J. TESSONE and XI XIA

Chair of Systems Design, ETH Zurich,
Weinbergstrasse 58, 8092 Zurich, Switzerland
**fschweitzer@ethz.ch*

Received 1 February 2015
Revised 11 March 2015
Accepted 19 March 2015
Published 7 May 2015

Established open source software (OSS) projects can grow in size if new developers join, but also the number of OSS projects can grow if developers choose to found new projects. We discuss to what extent an established model for firm growth can be applied to the dynamics of OSS projects. Our analysis is based on a large-scale data set from SourceForge (SF) consisting of monthly data for 10 years, for up to 360,000 OSS projects and up to 340,000 developers. Over this time period, we find an exponential growth both in the number of projects and developers, with a remarkable increase of single-developer projects after 2009. We analyze the monthly entry and exit rates for both projects and developers, the growth rate of established projects and the monthly project size distribution. To derive a prediction for the latter, we use modeling assumptions of how newly entering developers choose to either found a new project or to join existing ones. Our model applies only to collaborative projects that are deemed to grow in size by attracting new developers. We verify, by a thorough statistical analysis, that the Yule–Simon distribution is a valid candidate for the size distribution of collaborative projects except for certain time periods where the modeling assumptions no longer hold. We detect and empirically test the reason for this limitation, i.e., the fact that an increasing number of established developers found additional new projects after 2009.

Keywords: Yule–Simon distribution; entry–exit dynamics; open source software; SourceForge.

1. Introduction

Open source software (OSS) communities share with other organizations, such as social online platforms [16] or research and development networks [18], the feature that they are inherently *dynamic* because of the continuous *entry* of new members (developers, users, firms) and *exit* of established members. While this entry and exit

*Corresponding author.

dynamics usually resemble small perturbations that do not challenge the existence of the organization, it can also lead to large cascades of members leaving [3], in particular if these depend on the contribution of those who left. Hence, these processes have the potential to destabilize an organization. On the other hand, the entry–exit dynamics plays an important role in *knowledge exchange* between organizations. New members can bring new knowledge, information, skills or methods that help organizations to innovate. Members leaving, on the other hand, make space for newcomers and at the same time transfer knowledge they gained to other organizations.

The economist Schumpeter saw the *creative destruction* process induced by newcomers as an important element to renew, and to develop, the economic system [11]. Consequently, economists have for a long time focused on the role of entry and exit of *firms* in industrial organization [9]. For example, they found positive correlations between the entry rate of firms and innovation rates [6]. A particular strand of research was devoted to the impact of newcomers on the *size distribution* of firms. This is a long-standing topic in industrial organization since Gibrat (see [15]) introduced the *law of proportionate growth*, i.e., $\dot{x} = \beta x$, where x represents the firm size as measured by the number of employees, to explain the empirical size distribution of firms. His assumptions lead to a *log–normal distribution* which is valid only if the number of firms is kept constant. An important extension was made by Simon [12], who combined the model of proportionate growth with assumptions about firm's entry. This yields another type of skew size distribution, which he called the *Yule* distribution [19], but is now commonly called the *Yule–Simon distribution*. It is characterized by a power-law tail, $f(x) \propto x^{-\gamma}$, for large values of x.

The debate about whether the firm size distribution is best described by a log–normal, Yule–Simon, or a power-law distribution is still ongoing and the answer largely depends on the dataset analyzed. Therefore, we focus more on the theoretical insights obtained from these investigations. In particular, we ask to what extent an *economic model*, i.e., the Simon model for the entry and subsequent growth of *firms*, can be utilized to describe the dynamics in other types of social organizations, for example *OSS communities*.

It would indeed add to the importance of the Simon model if we find that it also describes the empirical findings in the dynamics of OSS communities. On the other hand, a formal model of the entry dynamics and growth of OSS communities which focuses on the choice of developers is a rather new and important contribution to better understand the complex processes in sociotechnical systems [17, 5]. Precisely, the novel contribution of our paper is *not* in the development of the model, but in the discussion to what extent an existing economic model describes the dynamics in OSS communities and how it could be extended for this purpose.

The methodological approach to test a model of firm growth for OSS communities is based on some implicit analogies. With OSS *community*, we refer to a specific platform that hosts possibly hundreds of thousands of OSS projects, such as `Sourceforge.net` (SF) or `Github.com`. That is, the community is comprised

of *projects* each of which has a number of *developers* contributing. We note that such as system is best described as a bipartite network as discussed in Sec. 2.4. Continuing with the analogy to industrial organization, this system is equivalent to a particular *industrial sector* (also called market). This market is comprised of thousands of *firms* each of which has a number of *employees*. The *size* of the firm is given by the number of employees, as the size of the project is given by the number of developers.

With respect to the dynamics, we observe a continuous entry of new firms/projects that have into the market/platform, but likewise also a continuous exit, e.g., if firms go bankrupt or projects collapse. But it is not the firms/projects that drive the dynamics. The real drivers are the underlying constituting elements, i.e., the employees/developers, that create new firms/projects or join existing ones, or decide to quit. This leaves a considerable degree of freedom. Employees/developers usually decide individually which firm/project to join or whether to establish a *new* firm/project. Only the latter choice leads to an increase in the total number of firms/projects, while the former still results in an increase in the size of a given firm/project. Interestingly, on the system's level this individual choice can be described by a certain probability to found a new firm/project, which is constant and the same across the population. While this does not reflect the individual motivation, it is sufficient to describe the dynamics at the system's level.

Hence, with a focus on the *OSS community* we are not so much interested in the individual dynamics of specific projects which would be better captured in case studies [20]. Instead, we want to investigate systemic properties that result from a large number of projects. Such an approach does not necessarily address a number of issues that may be also of interest in the study of OSS communities, such as the motivation of developers [14], their individual activity [10] or their specific role/skills in the project.

Our paper is organized as follows: Before we propose in, Sec. 4, a model to capture the dynamics of projects (and indirectly also of developers), in Sec. 2 we look at the macroscopic properties of the community, obtained by aggregating over all projects. In Sec. 3, we also analyze in detail the entry and exit rates of projects and developers and particularly focus on the size dependence of growth rates. The model we develop leads to a prediction for the size distribution of projects, which is compared with empirical data in Sec. 5. There, we also discuss reasons for deviations from the prediction and possible extensions of our work.

2. Aggregated Data Analysis

2.1. *Dataset description*

The dataset used in our study was acquired from SF, which was one of the world's largest OSS development website until `Github.com` became predominant. Our analyzed dataset contains 89 monthly snapshots from January 2003 to June 2012, in which information about all the developers and projects hosted on SF is recorded.

In their early years, SF frequently changed the format in which information about the project was stored. This leads to disruptions in our dataset because of corrupt data, in particular snapshots between February 2003 and October 2004 and for January 2005 are missing. Also, snapshots for July–September 2007 were removed by the SF archive provider because of data corruption [8]. Eventually, in February 2006, SF launched an autopurge service to remove inactive projects, which resulted in abnormal dropdowns in the number of projects. Starting from June 2010, SF automatically created a project for each developer. These "projects" do not represent real activities of the developers, so we removed them from the analysis. Also, in total 3 developers/projects were manually removed from the dataset. These three points had an extremely high number of links, were created by machines, and were used for the purpose of advertising or testing.

Nevertheless, the remaining dataset is large and reliable enough for our analysis. Table 1 gives an overview of the total number of projects, $N_p(t)$, and developers, $N_d(t)$, in the first and the last month recorded in our dataset. We also have information about the relationship between projects and developers, in particular about the *entry date* (month) in which a developer joined a project. We then assume that a *link* between the developer and the project was created. The total number of links between developers and projects, $K(t)$, is also reported in Table 1. We note that, due to the lack of data, the programming language used is available only for about 40% of all projects.

2.2. *Aggregated growth rates*

The most simple aggregated statistics is given by the total number of projects, $N_p(t)$, developers, $N_d(t)$ and links, $K(t)$, and how these numbers change over time measured in *months*. Figure 1(left) shows their evolution. As we clearly observe in the log–linear plot, all of these quantities follow an *exponential growth* dynamics:

$$\frac{\Delta X}{\Delta t} = \omega X, \quad X(t) \propto \exp\{\omega t\}. \tag{1}$$

This is also known as the *law of proportionate growth* and indicates that the SF community became more attractive the larger it was, which reamplified the growth for many years. The respective *growth rates* ω with Δt=1 month are given in Table 2.

Table 1. Summary of available data for the first and the last monthly snapshot of the dataset. $N_d(t)$: *total number of developers* at time t, $N_p(t)$: *total number of projects* at time t, $K(t)$: *total number of links* between developers and projects at time t.

Time	01/2003	07/2012
$N_d(t)$	77.050	339.140
$N_p(t)$	54.234	357.555
$K(t)$	106.840	576.238

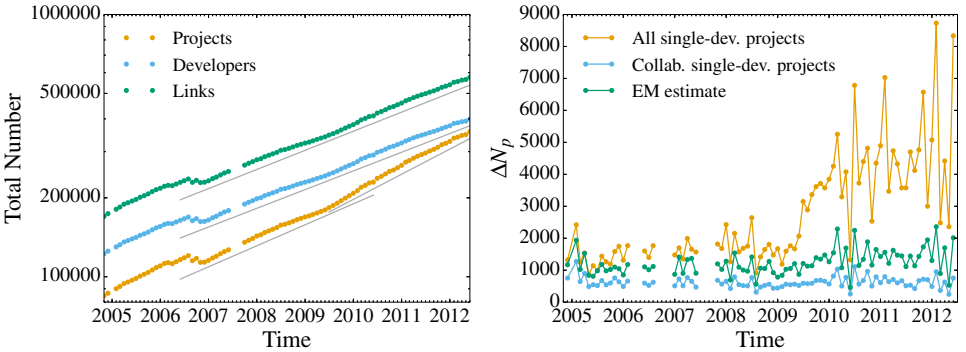

Fig. 1. (Color online) (Left) Total number of projects, $N_p(t)$ (yellow), total number of developers, $N_d(t)$ (blue), and total number of links, $K(t)$ (green), over time measured in months. The solid lines indicate fits of the growth rates given in Table 2. (Right) Number of new single-developer projects (yellow) and multideveloper projects (blue) per month, over time. Solid lines represent the median obtained over a rolling window of one year. The expectation–maximization (EM) estimate (green) is discussed in Sec. 5.2. The missing points in 2006 correspond to the time periods when autopurge was used excessively (see Sec. 2.1).

Table 2. Regression results for left panel of Fig. 1.

Variable	Growth rate ω (%)	R^2	p-value
$K(t)$	1.30	>0.99	2.80e−99
$N_d(t)$	1.27	>0.99	8.18e−99
$N_p(t)$	1.54	>0.99	8.06e−82
$N_p(t<2010)$	1.33	>0.99	6.19e−55
$N_p(t>2010)$	1.81	>0.99	2.30e−33

We note that the exponential growth remains despite of the data disruptions explained in Sec. 2.1. A closer look at Fig. 1 and Table 2 reveals that, in the log–linear plot, both $N_d(t)$ and $K(t)$ grow at about the same growth rate, constant over time. For $N_p(t)$, however, we observe a significant change in the growth rate at about 2010. Before 2010, $N_p(t)$ grew at a rate comparable to the other aggregated quantities, but the growth significantly increased afterwards. Remarkably, this increase does not become visible in the growth of the total number of links. Hence, the network between developers and projects (to which the links refer) becomes sparser after 2010.

We argue that the increasing growth rate for projects results from the fact that developers started their own single-developer projects. These could be either new developers entering SF or developers who already registered at SF for another project and now create another one. This conjecture is explored in Fig. 1(right) which plots the number of new projects per month that have only one developer together with the respective number of new projects per month that have more than one developer. We observe a significant increase of single-developer projects

at about 2010, while the number of new multideveloper projects per month remain about the same over time.

2.3. *Change in programming languages*

One of the reasons for the observed change towards more single-developer projects could be in the rise of *scripting languages* for programming, such as `PHP` and, more recently, `Python`. Such programming languages have been widely adopted in particular for single-developer projects, as we verify in our dataset. We already mentioned that only about 40% of all projects list their programming language and some projects, especially large ones, also use more than one programming languages. Precisely, in January 2003 information about the programming languages was available for 35.089 projects, which increased to 187.168 projects in July 2012. There are in total 106 programming languages listed in the dataset, but more than 80% of all projects use one of the major 7 languages `C`, `C#`, `C++`, `Python`, `PHP`, `Java` and `JavaScript`. Each of the remaining 99 languages has a share of less than 1% and is ignored in the following.

The importance of the major 7 languages changed considerably over time, as shown in Fig. 2(left). Despite the fact that this refers only to a subset of projects, we can observe that `C` lost nearly 10% market share in 7 years (from 25% down to 15%), which is a loss of 40% of its original total market share against the other 6 languages even if the *absolute* number of projects using `C` has increased. `Java`, on the other hand, increased its market share by about 10%. But the largest shares are taken by `JavaScript` and `C#`.

Figure 2(right) plots, for each of the 7 main programming languages, how the share of single-developer projects changes over time. We first note the trend towards more single-developer projects for *all* of these 7 languages, but with noticeable language preferences. In July 2012, 76% of all projects using `C#` are single-user projects, followed by `PHP` with a share of 74% single-developer projects and `Python`

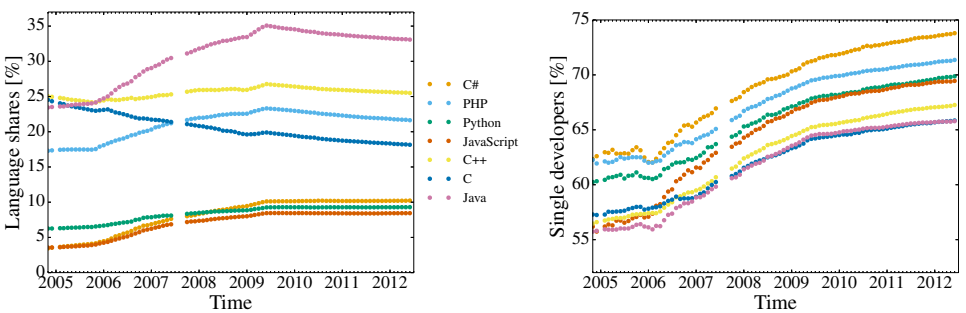

Fig. 2. (Color online) (Left) Share of the 7 most popular programming languages (normalized to 100%) over time measured in months, for all projects with available information about programming languages. (Right) Share of single-developer projects (normalized for each of the 7 most popular programming languages separately) over time measured in months.

and `JavaScript` with 72%. That is, developers who prefer to work on their own, have a clear preference for these languages.

2.4. *The bipartite network of developers and projects*

We now take a closer look at the developers and their projects. Both form a *bipartite network*, i.e., a network where links exist between *different* types of nodes. As explained above, we consider a *link* between a developer and a project if this developer has registered for the project regardless of her subsequent activity. Thus, instead of a weighted network where the weight of the links reflects the contribution, in this paper we only consider an *unweighted* network. A sketch of this bipartite network is shown in Fig. 3, where 10 developers contribute to eight different projects. That is, links between developers only exist through projects, and links between projects only through developers.

Nevertheless, we can draw two projections of this *bipartite network*, also shown in Fig. 3, one with respect to the *developers* and one with respect to the *projects*. In these projection, a link between *developers* appears if both of them contribute to the *same* project, and a link between *projects* appears if both of them have the same developer contributing. We emphasize that the bipartite network and its projections are *aggregated* over a given time interval, i.e., a link essentially reflects that two developers contributed to the same project in the same time interval (and not necessarily at the same time).

Based on the aggregated description, we can define the *degree* k_i of a developer i as the number of links she has, i.e., the total number of projects she was involved over that time period. Likewise, we can also define the degree x_r of a project r as the total number of developers that contributed to this project over that time period. x is also called the *size* of the project, as measured by the number of developers. The network of developers can then be described by a *degree distribution* $f(k)$ which gives the fraction of developers with degree k in the population of all developers, during the observation period. Likewise the degree distribution, later also called *size distribution*, $f(x)$ gives the fraction of projects with x developers, during the

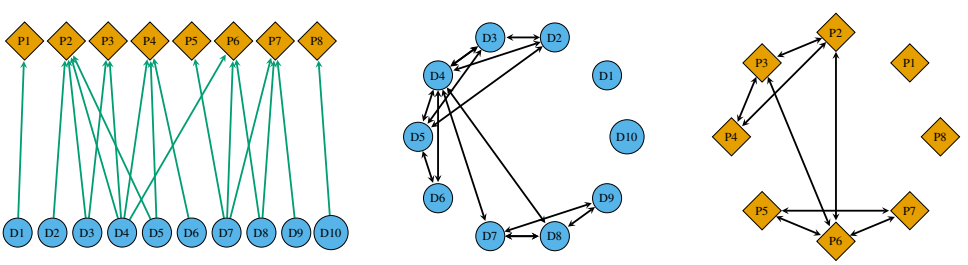

Fig. 3. (Left) Example of a bipartite network where 10 developers labeled $D1, \ldots, D10$ contribute to eight projects labeled $P1, \ldots P8$. (Middle) Projection of the network of developers (linked by common projects) and (right) projection of the network of projects (linked by common developers).

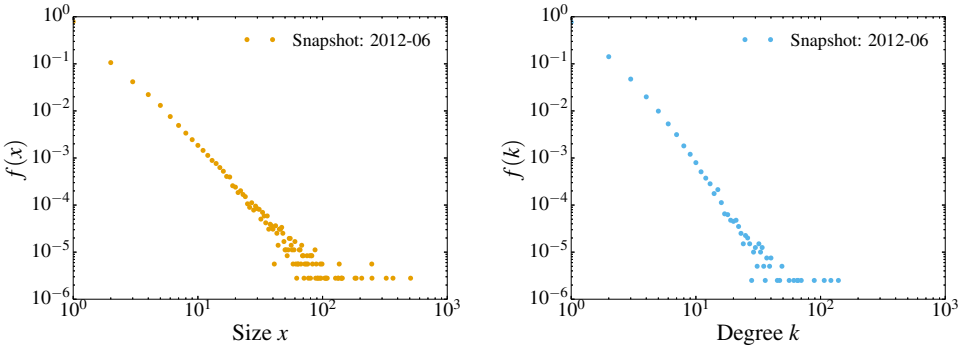

Fig. 4. (Left) Degree (size) distribution of projects (i.e., number of developers per project), $f(x)$ and (right) degree distribution of developers (i.e., number of projects a developer contributes to), $f(k)$, for the monthly snapshot of June 2012.

observation period. Both distributions are plotted in Fig. 4 for the snapshot of June 2012, which is the last snapshot of our dataset.

We observe that both are very skew distributions, reflecting the fact that there is a considerable probability to find projects of large sizes, or developers involved in very many projects. The distributions resemble the known *scale-free* distributions (such as power-law distributions), which indicates that there is no characteristic scale (size, number of projects) for projects or developers. In fact, these are *not* pure power-law distributions (note the bend in the shape and a rather limited range), but the specific type will be discussed in Sec. 5.

3. The Growth of OSS Projects: A Microscopic Perspective

3.1. *Entry and exit dynamics*

In this section, we analyze the dynamics of projects and of the developer community in more detail, by looking at the available data about birth and death of projects and entry and exit of developers, instead of the aggregated growth. Figure 5(left) shows the number of *new* projects per months, as well as the number of *removed* projects per month, while Fig. 5(right) shows the corresponding numbers for *developers*. We call the underlying processes "entry" and "exit" of projects or developers, respectively.

We can immediately observe that the number of entry events largely exceeds the number of exit events, for each month, both for projects and developers, with the exception of the large exit spikes observed in 2006–2007 because of the project cleanup initiated by SF (see Sec. 2.1). The reason for the dominance of the entry dynamics in normal time periods is that most projects or developers are not really *removed* if they become *inactive*. In fact, it is not trivial to determine whether a project or a developer is really inactive. Often, the activity just decreases considerably, but does not cease to exist. Also, the fact that there is no activity in a given

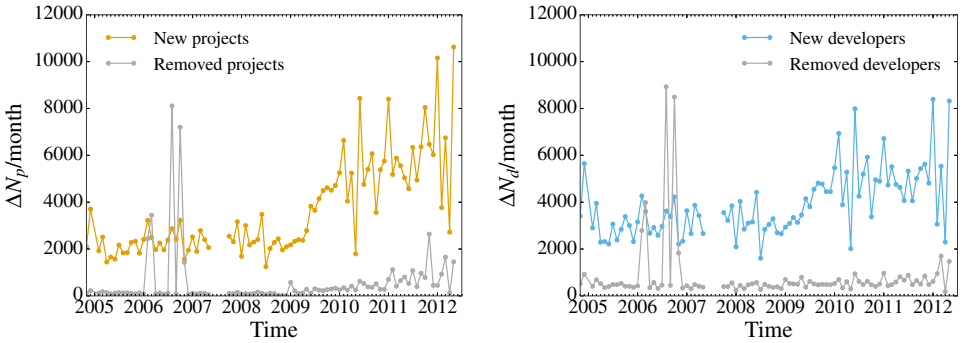

Fig. 5. (Color online) (Left) Number of new projects (yellow) and removed projects (gray) per month. (Right) Number of new developers (blue) and removed developers (grey) per month.

time period does not imply that there will be also no activity in the future. We have discussed this question in detail in [10]. In this paper, we do not speculate about inactivity and just take the computed exit rates as a matter of fact. For the modeling in Sec. 4, we take advantage of their very low numbers and will simply neglect the exit dynamics.

Eventually, we also note the occasional large fluctuations in the exit rates, both for projects and developers. These are the results of extraordinary efforts by SF to clean up the project and developer base, e.g., by testing and implementing the new autopurge system after turning off the old one (see Sec. 2.1). During and shortly after this switch, either extremely higher or lower numbers of projects and developers were detected as inactive and removed.

The second important observation is the *growth* of the monthly *entry rate*s over time, both for projects and developers (indeed, for projects, we could also note an increase of the exit rates over time). High occasional fluctuations might result from seasonal factors (holidays) or high media attention. This growth on an average is in line with the exponential growth observed both for projects and developers on the aggregated level as discussed in Sec. 2.2. The *law of proportionate growth* tells us that SF, for the observed time interval, became more attractive the bigger it was. Hence, the monthly entry rates shall depend on the current numbers of projects or developers, respectively. Figure 6 plots these *relative monthly entry rates*:

$$g_p(t) = \frac{N_p(t) - N_p(t - \Delta t)}{N_p(t)}, \quad g_d(t) = \frac{N_d(t) - N_d(t - \Delta t)}{N_d(t)}, \tag{2}$$

both for projects and developers. We note that, despite some considerable fluctuations, they tend to vary around long-term stationary values $\bar{g}_p$, $\bar{g}_d$ in a first-order approximation (i.e., we do not discuss a nonlinear dependency on N).

3.2. *Size dependent growth rates of projects*

So far, we have discussed the *law of proportionate growth* on the aggregated level, both with respect to the absolute numbers N_d, N_p and K, Eq. (1), and the relative

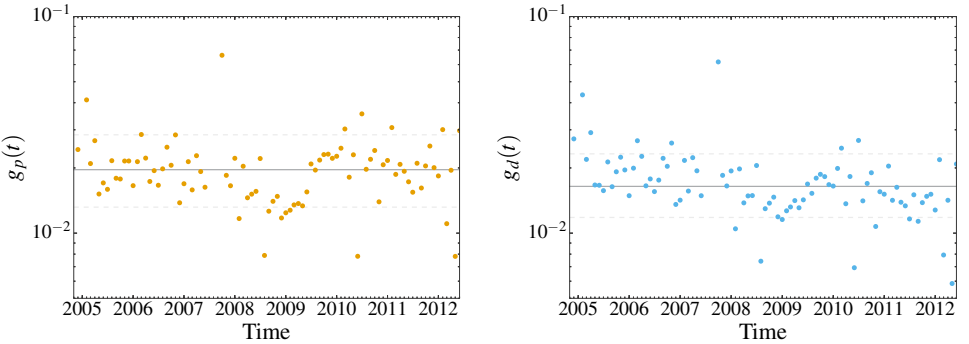

Fig. 6. Relative monthly entry rate of projects, $g_p(t)$ (left), and of developers, $g_d(t)$ (right). The horizontal lines represent the median of all the values in the time series (solid line) and the dashed lines, 10% and 90% quantiles. The values read for g_p: 10%: 0.0131, median: 0.0196 and 90%: 0.0284 and for g_d: 10%: 0.0118, median: 0.0164 and 90%: 0.0232.

growth rates, g_d and g_p, Eq. (2). But we can also refer to the individual project level, to verify this dynamics.

Recall that the size x_r of a project r is defined by the number of developers contributing to it (x_r was also called the degree of the project because, in the bipartite network, links exist between developers and the project). Then, the growth dynamics on the individual project level reads as

$$\frac{x_r(t+\Delta t) - x_r(t)}{\Delta t} \propto x_r^{\gamma}(t). \tag{3}$$

If $\gamma = 1$, we have a growth strictly proportional to size, which is also known as preferential attachment in network theory, i.e., nodes (projects) receive new links (developers) proportional to the number of existing links. $\gamma > 1$ would indicate a super-linear growth.

As we have seen on the aggregated level, growth rates heavily fluctuate for each month. Therefore, for the individual project level, we choose the time window $\Delta t =$ 12 months, i.e., large enough to cancel out these fluctuations. We then compute the average growth rate $\bar{g}(x)$ of projects with similar size x for each year, which is shown in Fig. 7(left). We verify that the annual growth rate indeed increases with the size of the project, as described in Eq. (3), and we barely notice differences in this dependence for different years.

For a closer inspection of the law of proportionate growth, we estimate the exponent γ, Eq. (3), from the data separately for each year. We find that γ varies indeed between 1.23 and 1.35 during the seven years period, hence the growth is slightly super-linear for all times. However, because γ is very close to 1, we can still argue that the law of proportionate growth approximately holds, but there is a higher-order dependence on the project size ($\sim x^{1/4}$) which may enter the proportionality factor β, i.e., $\dot{x} = \beta(x)x$.

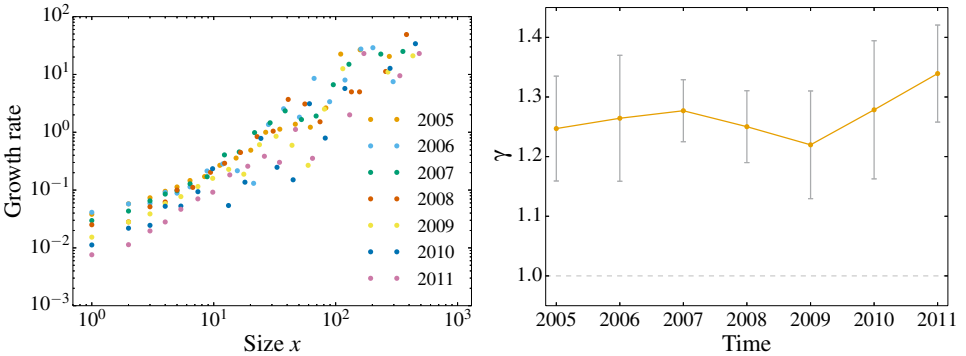

Fig. 7. (Color online) (Left) Averaged annual growth rate $\bar{g}(x)$ over project size x (measured by the number of developers) for each year separately. (Right) Exponent γ, Eq. (3) for each year, where error bars indicate the standard error.

4. A Dynamic Model of Project Growth

4.1. *Dynamic assumptions*

In the following, we focus on the dynamics at the *project* level, only. The number of projects of a given size x (measured by the number of developers contributing to the project) at a given time t is given by $n(x,t)$. Each project of size x at time t belongs to the same *size class* $Y_t = x$. Time t is assumed to be discrete (measured in months). The total number of projects, $N_p(t)$, and the total number of developers, $N_d(t)$, contributing to projects are defined as

$$N_p(t) = \sum_{x=1}^{x_{ax}} n(x,t), \quad N_d(t) = \sum_{x=1}^{x_{\max}} xn(x,t). \tag{4}$$

For our dynamic assumptions, we follow the model of Simon [12] for the entry of new firms and the growth of existing firms. That means in our model, the *entry of new developers* is assumed to be the only source for (i) establishing *new projects* and (ii) *enlarging* existing ones. Precisely, we neglect the possibility that also established developers already involved in other projects found a new project. This is supported by the empirical finding that most developers are only involved in one project, as showing by the degree distribution of developers in Fig. 4(right). But we will come back on the validity of this assumption in Sec. 5.

We further neglect *fragmentation*, or *fork* processes, i.e., an existing larger project splits up into two (or more) smaller projects, which could be also seen as the foundation of a new project of size $x \geq 1$. We argue that such events exist but are comparably rare so that we cannot sufficiently calibrate our model against such data, and simply neglect these events.

The empirical finding of Fig. 1 tells us that the number of developers has increased exponentially. We can include this in three different ways: (i) Choosing a *linearly increasing* number of new entrants per time interval, (ii) rescaling

the time interval linearly such that the number of new entrants per time interval is constant and (iii) replacing time t simply by the total number of developers N_d. We have chosen the latter as the most elegant way. Hence from $N_d(t) \propto \exp(\omega t)$, we find the transformation $t \to \ln N_d/\omega$. From now on $N_d \equiv N$ measures time in discrete steps, $N, N+1 \ldots$.

For the change of $n(x,t)$, we discuss the following processes:

(i) *A new project is founded*: Here the assumption is that the project starts in the smallest size class $x = 1$. There is a certain (conditional) probability,

$$P_{0,1}^{N+1}(Y_{N+1} = 1 \,|\, Y_N = 0) = p_0, \quad (x = 1), \tag{5}$$

that we find in the next time step $N+1$ a new project of size 1 (where its former size 0 indicates that the project did not exist yet at time N). This probability is denoted as $p_0 \in (0,1)$ and assumed to be constant in time, except for the very first time step $N = 0$ at which no projects exist yet. So one has to be founded with certainty:

$$P_{0,1}^{1} = (Y_1 = 1 \,|\, Y_0 = 0) = 1, \tag{6}$$

i.e., we start the dynamic process with one project that is of the smallest possible size 1. Because at each time step only one new developer enters, the largest possible size of any project cannot be larger than N, i.e., we set $x_{\max} = N$.

(ii) *An established project grows*: Here the assumption is that the project only grows by attracting *one new developer* at a time. This event is described by the probability,

$$P_{x-1,x}^{N+1}(Y_{N+1} = x \,|\, Y_N = x-1) = K(N)(x-1)^{\alpha} n(x-1,N), \quad (x = 2, \ldots, N), \tag{7}$$

i.e., the (conditional) probability of a project in size class $(x-1)$ to grow at time N is proportional to the number of projects in that size class, $n(x-1,N)$. However, the new developer may have a *preference* for larger or smaller projects, i.e., the probability to choose from size class $(x-1)$ is also proportional to $(x-1)^{\alpha}$. The value $\alpha = 0$ would recover the case of *no size preference*, which was discussed, e.g., in [15]. The value $\alpha = 1$ would be a preference directly proportional to the existing size, which was discussed in [12] to cope with Gibrat's *law of proportionate growth*. In the following, because of analytical tractability we will only consider the case $\alpha = 1$. Its empirical evidence is discussed in the following section, while in Sec. 5 consequences for different values of α are discussed.

$K(N)$ is a proportionality constant that has to satisfy the condition that all probabilities sum up to 1. Using $\alpha = 1$ from now on, we find

$$\sum_{x=1}^{N} K(N) x n(x,N) + p_0 = 1 \tag{8}$$

Because of Eq. (4), $\sum_{x=1}^{N} xn(x,N) = N$, and we get from Eq. (8)

$$K(N)N + p_0 = 1 \Rightarrow K(N) = \frac{1-p_0}{N}. \tag{9}$$

Note that this would not hold if $\alpha \neq 1$. Equation (9) allows us to rewrite Eq. (7) as

$$P_{x-1,x}^{N+1}(Y_{N+1} = x \,|\, Y_N = x-1) = (1-p_0)\frac{(x-1)n(x-1,N)}{N}, \quad x = 2, \ldots, N. \tag{10}$$

Our kinetic assumptions as seen from the perspective of the developer, are summarized as follows: At each time step $N+1$, *one new developer arrives*. This developer has *two options*: (i) With probability p_0, she chooses to found a *new* project and (ii) with probability $(1-p_0)$, she chooses to join one of the projects that exist at time N, i.e., $N_p(N)$. Without any preference for larger projects, she will choose a project from size class x with a probability $(1-p_0)n(x,N)/N$. But with the assumed size preference, $\alpha = 1$, the proportional weight x comes into account.

We emphasize that some of the dynamic processes one could think of are deliberatively neglected, e.g., we neglect that several developers join one or different projects during the same time interval (which can be solved by changing the time resolution). More importantly, we also neglect that developers switch between projects, i.e., some projects lose and some projects gain in developers, but the total number of developers does not change. Such dynamics can be seen as reallocation processes among projects, and are neglected here.

Further we do not consider that *existing projects shrink*, i.e., loose in size if developers leave. Since we have opted out reallocation processes, it would mean that developers become inactive. Again, there is empirical evidence for this process (see Fig. 5). But the number of developers exiting is (i) rather constant in time, and (ii) much smaller than the number of new developers arriving. Therefore, we will consider this in our model as a rescaling of the arrival rate of new developers and will not explicitly model the shrinking process.

Eventually, we also do not consider that an *established project ceases to exist*, either. Such processes can happen in two ways: (1) The project goes extinct and (ii) two existing projects *merge* into a new one, with a larger project size. Again, in our model we neglect both processes. Projects often become *inactive*, but are rarely deleted, and for *mergers and acquisitions* the same argument as for the project forks apply.

With these considerations, the total number of projects at time N is given by:

$$\sum_{x=1}^{N} n(x,N) \approx 1 + p_0(N-1) = Np_0. \tag{11}$$

The first term, 1, results from the fact that there exists a new project in the first time step. Then, during every time step from $N = 2$ up to N a new project appears with probability p_0. Hence, if $p_0 \ll 1$ and N is large, this gives approximately $1 + p_0(N-1)$, which is Np_0.

4.2. *The rate equation*

We now start formalizing the above assumptions by developing a rate equation for the relevant quantity $n(x, N)$. This can change by two processes: (i) *Gain*: A project of size $x-1$ is chosen by the developer and thus advances to the next size class, leading to an *increase* in $n(x, N)$ and (ii) *loss*: A project of size x is chosen by the developer and thus advances to the next size class, leading to a *decrease* in $n(x, N)$.

$$n(x, N+1) - n(x, N) = (1-p_0)\left[\frac{(x-1)n(x-1, N)}{N} - \frac{xn(x, N)}{N}\right], \quad (x = 2, \ldots, N+1). \tag{12}$$

The left-hand side (LHS) of Eq. (12) describes the *net inflow* of projects into size class x. Similarly we get for the number of new projects at time $N+1$:

$$n(1, N+1) - n(1, N) = p_0 - (1-p_0)\frac{n(1, N)}{N}. \tag{13}$$

The *gain* comes from funding new projects with a probability p_0, whereas the *loss* results from the fact that a project of size 1 grows into size 2.

In the following we will only consider "steady-state" distributions, i.e., we assume that each size class grows *proportionally* with N:

$$\frac{n(x, N+1)}{n(x, N)} = \frac{N+1}{N}, \quad \forall\, x, N, \quad (\text{if } x < N). \tag{14}$$

With $\Delta n(x, N) = n(x, N+1) - n(x, N)$, we rewrite Eqs. (12) and (13) as

$$\Delta n(x, N) = (1-p_0)\left[\frac{(x-1)n(x-1, N)}{N} - \frac{xn(x, N)}{N}\right], \quad (x = 2, \ldots, N+1), \tag{15}$$

$$\Delta n(1, N) = p_0 - (1-p_0)\frac{n(1, N)}{N}.$$

From Eq. (14) it follows that

$$n(x, N+1) = \left(1 + \frac{1}{N}\right) n(x, N), \quad \Delta n(x, N) = \frac{n(x, N)}{N}. \tag{16}$$

Plugging Eq. (16) into Eqs. (12) and (13), we have

$$\begin{aligned} \Delta n(x, N) &= \frac{n(x, N)}{N} \\ &= (1-p_0)\left[\frac{(x-1)n(x-1, N)}{N} - \frac{xn(x, N)}{N}\right], \quad x = 2, \ldots, N, \\ \Delta n(1, N) &= \frac{n(1, N)}{N} = -p_0 - (1-p_0)\frac{n(1, N)}{N}, \end{aligned} \tag{17}$$

which simplifies to the following set of equations:

$$\begin{aligned} 0 &= (1-p_0)(x-1)n(x-1, N) - (1-p_0)xn(x, N) - n(x, N), \\ 0 &= Np_0 - (1-p_0)n(1, N) - n(1, N). \end{aligned} \tag{18}$$

4.3. *The size distribution of projects*

In order to solve Eq. (18), we define the new parameter ρ as

$$\rho = \frac{1}{1-p_0}, \tag{19}$$

where $1 < \rho < \infty$, because of $p_0 \in (0,1)$. To interpret ρ (see [12]), we keep in mind that p_0 actually decides how much of the growth (one developer per time unit) is spent on new projects as compared to established projects. In Sec. 5.1, we will test this relation against our empirical data.

From Eq. (18), we find for the stationary solution for $n(1,N)$ (denoted by *):

$$n^*(1,N) = \frac{Np_0}{2-p_0} = \frac{\rho}{\rho+1}Np_0, \tag{20}$$

whereas we find for the stationary solution of $n(x,N)$:

$$n^*(x,N) = \frac{(1-p_0)(x-1)}{1+(1-p_0)x}n^*(x-1,N) = \frac{(x-1)}{\rho+x}n^*(x-1,N). \tag{21}$$

We can solve this equation in an iterative manner, to find

$$n^*(x,N) = \frac{(x-1)}{(\rho+x)}\frac{(x-2)}{(\rho+(x-1))}\cdots\frac{1}{(\rho+2)}n^*(1,N), \quad (x=2,\ldots N). \tag{22}$$

To further compact this expression, we make use of the so-called *gamma function* $\Gamma(z)$ with the property $\Gamma(z+1) = z\Gamma(z)$ that, for integer z, results in

$$\Gamma(z) = (z-1)!. \tag{23}$$

The denominator of Eq. (22) can then be expressed as

$$\Gamma(x+\rho+1) = (x+\rho)\Gamma(x+\rho) = (x+\rho)(x+\rho-1)\cdots(2+\rho)\Gamma(\rho+2). \tag{24}$$

With this and the expression for $n^*(1,N)$, Eq. (20), we can rewrite Eq. (22) as

$$\begin{aligned} n^*(x,N) &= \frac{\Gamma(x)\Gamma(\rho+2)}{\Gamma(x+\rho+1)}f^*(1,N) \\ &= (\rho+1)\frac{\Gamma(x)\Gamma(\rho+1)}{\Gamma(x+\rho+1)}f^*(1,N) = \rho\mathcal{B}(x,\rho+1)Np_0, \end{aligned} \tag{25}$$

where $\Gamma(x)\Gamma(\rho+1)/\Gamma(x+\rho+1) = \mathcal{B}(x,\rho+1)$ is the *beta function*.

The corrected, normalized *Yule–Simon distribution*, which holds also for $x = 1$, is then

$$f(x,N) = \frac{n^*(x,N)}{Np_0} = \rho\mathcal{B}(x,\rho+1) = \frac{\rho\Gamma(\rho+1)}{(x+\rho)^{\rho+1}}, \quad (x=1,2,\ldots). \tag{26}$$

For large x we get for $f(x,N)$,

$$f(x,N) = \rho\mathcal{B}(x,\rho+1) \approx \rho x^{-(\rho+1)}, \quad x\to\infty, \tag{27}$$

which has the form of a power law if x is large enough. Therefore, the power law approximates the Yule–Simon distribution only in its upper tail. To get Zipf's law, one has to assume $p_0 \to 0$, as $\rho = 1/(1-p_0) \approx 1$. However, as noticed, e.g., by Krugman [7], for $p_0 \to 0$ the convergence to the steady state is infinitely slow.

5. Discussion

5.1. *Comparison with the Yule–Simon distribution*

We have now a theoretical prediction for the size distribution of projects, Eq. (26), and we have the respective empirical data for different years. Therefore, as a first step, we evaluate the kind of distribution that was already plotted in Fig. 4.

Before doing so, we have to argue whether the theoretical prediction and the empirical data really describe the same kind of projects. Our theoretical model is based on the assumption that *all* projects entering the system are *potentially* available to grow in the number of developers, i.e., developers can simply join them. This, however, cannot be confirmed for all single-developer projects listed in the database. Here, we have to consider that developers host their projects on SF not just to invite collaboration, but for various reasons, e.g., for archival purposes or just for distribution. While we cannot access the intrinsic reasons for a project to be hosted on SF, we argue that all new projects appearing on SF every month can be divided into *two classes*: (i) *Collaborative* projects, i.e., projects that are meant to grow also by the contribution of other developers joining the project and (ii) *noncollaborative* projects, that are not aimed at attracting other developers and thus do *not* grow in size as measured by the number of developers, but maybe grow in their lines of code submitted by the project holder. That is, we conjecture that there is a sizable number of single-developer projects that, from their very beginning, are not captured by our model that applies only to *collaborative* projects, i.e., projects with the potential to grow in the number of developers.

Consequently, when comparing the predicted size distribution with empirical data, we have to take into account that $f(1, N)$, i.e., the normalized density of projects of size 1, will need to be *corrected* to subtract the *noncollaborative* projects, $f^{\not c}(1, N)$ (where $\not c$ stands for noncollaborative), and to consider only the *collaborative* ones, $f^{c}(1, N)$. The procedure for this necessary correction will be described further below. The resulting corrected size distribution will then be indicated by $f^{c}(x, N)$.

As the null hypothesis for the size distribution $f(x, N)$, we test for the Yule–Simon distribution for which the maximum likelihood of the parameter ρ can be computed numerically [4]. We perform a Kolmogorov–Smirnov (KS) test to determine the significance level (p-value) for which the empirical distribution matches the Yule–Simon distribution. If $p = 0$ it means that the two distributions do not match under any circumstances. The higher the p-value, the more likely it is that the null hypothesis cannot be *rejected*. That means we cannot exclude that the Yule–Simon distribution is the right distribution, but there might be also other candidate distributions that could be considered (which we abstain from).

Applying the KS test in its simplest form to skew distributions usually results in very high p-values, simply because the mass of the distribution is mostly concentrated in the head while the tail is weighted less. The problems resulting from this naive approach have been discussed in detail in [1]. These authors also propose

a more reliable, but computationally more demanding, goodness-of-fit test suitable for heavy-tailed distributions which we adopt here.

We first turn to the degree distribution of *developers*, an example of which is shown in Fig. 4(right). Our goodness-of-fit test reveals that the, rather steep, "broad" developer degree distribution, $f(k)$, does *not* follow a Yule–Simon distribution ($p = 0$). But since we never made a hypothesis about this and did not develop a model for it, we just take this as a fact.

With respect to the *project size distribution*, $f(x)$, we have to test the null hypothesis of the Yule–Simon distribution for each *monthly* snapshot, an example of which is shown in Fig. 4(left). Simply fitting the Yule–Simon distribution to the empirical one and calculating the parameter $\rho_{\text{all}} = 3.88$ according to [4] would lead to the result shown in Fig. 8(lower right). The (dashed-line) fit is visually worse, it does not capture the tail well because the (uncorrected) value of $f(1, N)$ is over-represented.

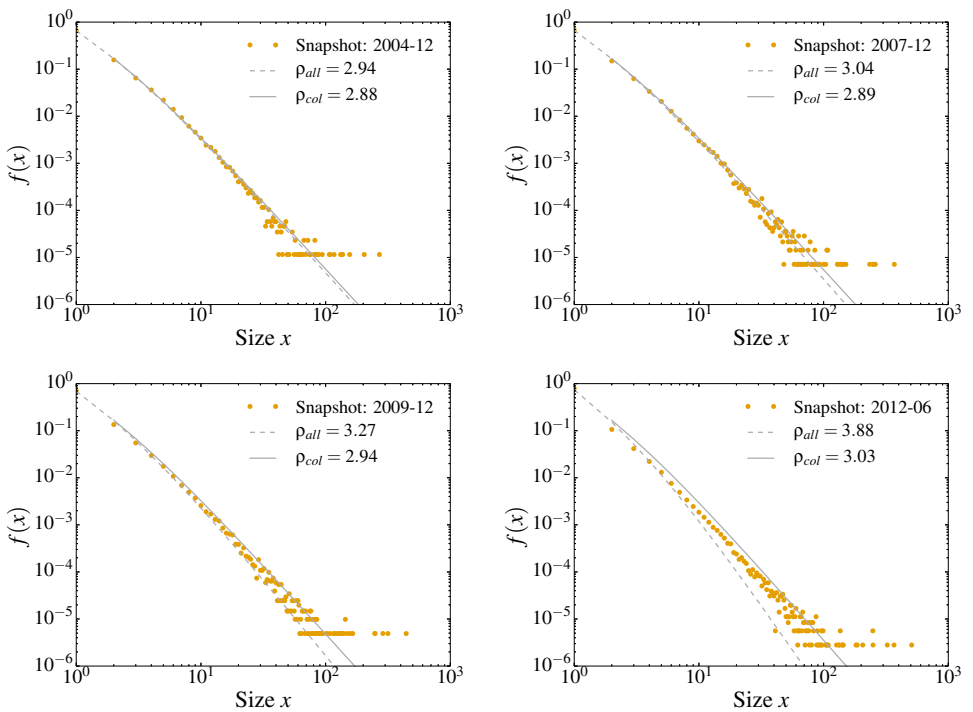

Fig. 8. (Color online) Project size distribution $f(x)$ for different monthly snapshots. For each snapshot, a fit (dashed gray line) of the Yule–Simon distribution is plotted, for which the parameter ρ_{all} was numerically obtained. The goodness-of-fit test however rejects the hypothesis that the Yule–Simon distribution fits the empirical one for most of the snapshots. A second fit (solid gray line) of Yule–Simon distribution is plotted, for which the value for single-developer projects is taken as unknown, and latent variable is also plotted, for which the same hypothesis cannot be rejected for most of the snapshots.

Therefore, in the next step we correct $f(1, N)$ as follows: Given the value ρ_{all} obtained from all projects, we first predict the value $f^{c'}(1, N)$ for the collaborative single-developer projects. Then, we do a new fit of the Yule–Simon distribution which leads to a corrected value ρ'_{col}. With this new value, we do a better prediction of $f^{c''}(1, N)$ and so forth. This method is known as the *EM algorithm* [2], where the number of single-developer projects $f^c(1, N)$ is used as an unknown, *latent* variable. EM is an iterative algorithm that consists of alternating expectation steps (E) and maximization (M) steps. Expectation refers to predicting $f^c(1, N)$, while maximization refers to calculating the appropriate ρ_{col}. We halt the algorithm when the change of ρ_{col} is smaller then a given threshold $\epsilon = 10^{-4}$. This leads to the much better (solid gray line) fit shown in Fig. 8(lower right). Taking all corrected distributions of Fig. 8 into account, we observe that the parameter ρ_{col} stays almost constant over time with values around $\rho_{\text{col}} \approx 3$ (which contrasts with the uncorrected distributions where ρ increases).

Now, with the corrected size distribution $f^c(x, N)$, we can apply our rigorous goodness-of-fit test to each monthly snapshot. The results for the p-values are shown in Fig. 9. We see that the null hypothesis of the Yule–Simon distribution as the empirical one *cannot* be rejected for all times between late 2005 and late 2009. This gives us great confidence both in our modeling assumptions and in the proposed correction to distinguish between collaborative and noncollaborative projects.

At the same time, it leads us to the question what has changed after the end of 2009, to make the fits invalid. As we observe, from 2010 the significance level goes down considerably, although it is hardly really zero. In order to better understand the dynamics from 2010, we will develop another conjecture in the following section.

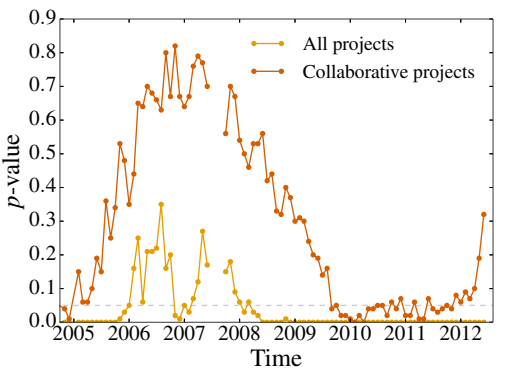

Fig. 9. The p-values for the goodness-of-fit test [1] of the Yule–Simon distribution and the empirical size distribution for each monthly snapshot. *Collaborative projects* refers to the project size distribution with corrected value for the single-developer projects. We observe that the Yule–Simon distribution is a plausible candidate for the size distribution of all projects only for certain periods of time, while for the corrected empirical distribution it is a plausible candidate for most of the times, notably from the late 2005 to the late 2009.

5.2. *Estimations for p_0*

In the previous section, we demonstrated that the Yule–Simon distribution (and the underlying model) is a valid candidate for describing the empirical dynamics of collaborative projects at least for certain time intervals. We can link our findings for the size distribution, which refer to the *systemic* level, back to our assumptions for the *microscopic* dynamics. Recall that in the model of Simon [12] there is only one parameter p_0 that decides whether new developers found new projects, as opposed to joining existing ones. This parameter, as far as the theory goes, is directly linked to the exponent ρ of the distribution, via Eq. (19). Assuming $\rho = 3$, Eq. (19) would give $p_0 = 2/3$, which is quite high if compared, e.g., to the *firm size distribution* where ρ is about 1.2 and p_0 about 0.16.

In this section, we want to find an independent way of estimating p_0. We recall that p_0 essentially describes how much of the total growth goes into newly founded projects. So, if G_{tot} is the growth spent on all existing projects and G_1 is the growth spent on new projects during a given time interval, then $p_0 = G_1/G_{\text{tot}}$ [13].

In our empirical data, $G_{\text{tot}}(t)$ is measured by the total number of *developers* per month who enter SF, $\Delta N_d(t)$ which is shown in Fig. 5(right). $G_1(t)$, on the other hand, is given by the total number of newly founded projects per month, $\Delta N_p(t)$, shown in Fig. 5(left). So, we just divide these monthly entry rates, to obtain $p_0(t) = \Delta N_p(t)/\Delta N_d(t)$ from the empirical data. We do this for the two different datasets: (i) For *all* projects and (ii) for *collaborative* projects. For the latter, we need to correct also $\Delta N_d(t)$ because we have to *exclude* those developers that joined SF to establish a noncollaborative project. These corrections are done based on the empirical data.

The results are plotted in Fig. 10 for the two different data sets: (i) $p_0^a(t)$ for *all* projects and developers and (ii) $p_0^c(t)$ for the collaborative projects and developers. As expected from the above discussion, we see differences for the time periods before

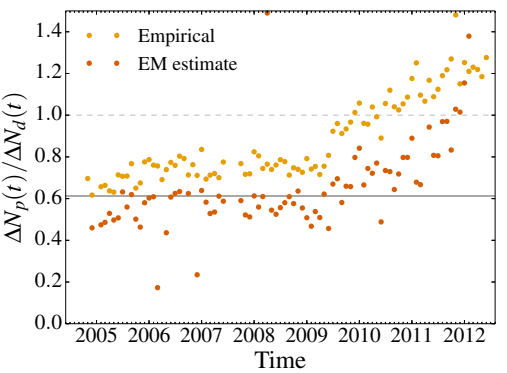

Fig. 10. (Color online) Estimation of the parameter p_0 from the monthly entry rates of projects and developers, both for all projects/developers ($p_0^a(t)$, yellow dots) and for collaborative projects/developers only ($p_0^c(t)$, red dots). The full line is the median of $p_0^c(t)$ at 0.6128.

and after 2010. In fact, we see that after 2010 $p_0^a(t)$ has consistently values *above* 1, which *cannot* be realized from the assumption that only newly entering developers establish new projects. If the latter holds, p_0^a is necessarily bound to values below 1. To explain this, we arrive at our second conjecture that after 2010 an increasing number of *established* developers started to found *new* projects. This, however, is not considered in our modeling assumptions, therefore the prediction derived from the model necessarily fails as we also see from the low p-values in Fig. 9.

If we look at *collaborative* projects, we see that $p_0^c(t)$ follows the same trend as $p_0^a(t)$, just with a shift toward lower values. To find out whether this shift results from a higher entry rate of collaborative projects or a lower entry rate of collaborative developers, we have plotted in Fig. 1(right) $\Delta N_p(t)$ values. We verify that $\Delta N_p(t)$ for collaborative projects is almost constant over all years, i.e., the increase in $p_0^c(t)$ is from the lower entry rate of collaborative developers. Because of the similar trend compared to $p_0^a(t)$ after 2010 we also keep our conjecture that an increasing number of established developers started to found new *collaborative* projects. This violation of our modeling assumption can be confirmed also in Fig. 9, where we see that the p-values for the goodness-of-fit test break down after end of 2009 for the collaborative projects. If we take the median for $p_0^c(t)$ over the whole time period, we find $\bar{p}_0^c \approx 0.61$ which is in a remarkable agreement with the theoretical value $p_0 = 2/3$ obtained from $\rho = 3$.

Eventually, we want to discuss an additional issue in comparing the empirical and theoretical results. By means of the EM method, we found a way to correct $f(1, N)$ such that only collaborative projects are taken into account. The corrected value $f^c(1, N)$ can also be related to the empirical number of collaborative single-developer projects. That is, by tracing their history, we can identify in the data set those single-developer projects that grew at a later point in time. Their monthly growth rate ΔN_p is already plotted in Fig. 1(right) (blue line). We observe that there is a shift between the predicted growth rate of collaborative single-developer projects (green line) and the empirical one, which slightly increases over time from values of 1.5 to 2.

The cause for this mismatch should not be attributed to the predicted value, but rather to the empirical one because it *underestimates* the number of collaborative projects for the following reason. Our empirical classification of collaborative versus noncollaborative projects is based on their observed growth, only. If we classify projects as noncollaborative, we make a mistake because projects may still grow later in time, but this is just not observed. This mistake becomes larger the closer we come to the end of the data set. Therefore, to estimate the magnitude of the mistake, we should look at the oldest projects in the data set, which date back to November/December 2004 (when the data became reliable). We verify that the interval between the time when these projects were created and the time when a second developer joined, can be well described by an exponential distribution, $f(t) = \lambda \exp^{-\lambda t}$. The expected value of this distribution is $\mathbb{E}[t] = 1/\lambda$, which can be

also measured from the data for almost 3000 projects created in these two months, to yield roughly 450 days. The cumulative distribution gives us the probability that those projects will grow before 450 days as $\Pr(t < \mathbb{E}[t]) \approx 0.6$.

This can now be used to calculate the mistake made by classifying projects as noncollaborative during the last 450 days of the data set. It is about 40%, i.e., we miss 40% of the collaborative projects in our estimation. The correction for the observed number thus will be a factor of $1/0.6 \approx 1.6$, which is very close to the observed mismatch of 1.5 to 2. That is, with this rough estimation we can explain fairly well the observed difference between the empirical and EM estimates.

To conclude our discussion so far, we find that the Yule–Simon distribution is a valid candidate to describe the empirical size distribution of *collaborative* projects. However, this validity is constrained to certain time periods for which the underlying assumptions of the model can be justified. We observe that in later time periods established developers started to create additional projects. This was not considered in the model assumptions to derive the Yule–Simon distribution, where only newly entering developers are considered to create new projects.

5.3. *Extensions*

In this paper, we have investigated to what extent an established model for the entry of new *firms* and the growth of existing ones originally developed by Simon [12] can be directly applied to OSS communities, where new projects are founded by new developers and existing projects grow by attracting new developers.

The advantages of using the Simon model go alongside with the disadvantages resulting from the limitations arising from the underlying assumptions. We discuss them here, to give hints for further improvements of the model, because we noticed that the Yule–Simon distribution, over large time intervals, has shown to be a promising candidate for the size distribution of collaborative projects. However, each of the suggestions discussed below will modify the original model such that the analytical approach developed can no longer be used and closed-form solutions will most likely not be derived.

The *first* suggestion relates to a known criticism of the Yule–Simon model, namely that not more than one project can be founded or grow at each time step. This does not make problems as long as one is interested in the asymptotic size distribution. But in order to come up with a more realistic dynamics before the steady state is reached, one should consider that projects can be founded and grow in parallel. In particular, one has to consider concurrent activities, i.e., that not only new developers perform an action, but also established developers can decide to have more than one project, e.g., by founding a new one. As a consequence, p_0, the probability to found a new project, shall become a *heterogeneous* parameter, to better reflect individual motivations of developers. By this, we could further distinguish between different personalities, e.g., *founders*, who prefer to start new projects, and *contributors*, who prefer to join existing projects.

As a *second* suggestion, we can consider that new developers may have a *preference* for larger or smaller projects, i.e., the probability to choose from size class $(x-1)$ is also proportional to $(x-1)^{\alpha}$. This was already implemented in the dynamic assumptions of Eq. (7) as a new element not discussed in [12]. The value $\alpha = 0$ would recover the case of *no size preference* (as, e.g., also used to describe the firm growth dynamics [15]), $\alpha = 1$ would be a preference directly proportional to the existing size and $\alpha < 0$ would indicate that projects become *less attractive* with increasing size, probably because of coordination and integration efforts. Hence, the additional parameter α allows us to consider various (monotonous) *size-dependent preferences*. To account for optimal project sizes, this dependence should be also nonmonotonous. In our formal approach, we have set $\alpha = 1$ to favor the mathematical approach by which the project size distribution was derived. Without this restriction to closed-form solutions, an agent-based simulation could explore the impact of size-dependent preferences on the project size distribution.

As a *third* suggestion, we should allow contributors to *switch* between projects, in order to better utilize their skills. This would also have consequences for the *knowledge spillovers* between projects, which is an important consideration for management science and economics. On the formal modeling level, such additional assumptions would change the rate equation approach developed in Sec. 4.2, by adding additional terms for the shrinkage of existing projects (developers leave) and for the growth of existing projects by other than newly registered developers.

A different set of possible extensions points to the way the developer *activity* is counted in. So far, we have assumed that a newly entered developer *immediately* founds a new project or joins an existing one. But developers may have joined SF for different other reasons, e.g., for getting better access to code to reuse outside of SF. This may lead to a mismatch between the number of developers entering SF per month and the assumed activity of these developers inside SF. In the same line, in our analysis developers are assigned to projects, which is indicated by a link, and our modeling approach assumes that such links do not change. However, links do not necessarily mean that developers are actively working for the project, they are only a first (and not necessarily the best) approximation of contributions. Here, we note that already more refined measures are available which are discussed in a subsequent paper [10]. But these measures largely depend on information that is not available from SF, so we will have to resort to `Github.com`.

We conclude that, even with these limitations on the SF data, our analysis about the launch of new projects and their subsequent growth is one of the largest investigations on such data to date. It resulted in new findings about the project size distribution and the degree distribution of developers, about their entry and exit rates and the preferred usage of programming languages. In order to better understand the dynamics that generated these systemic properties, we utilized an established economic model that in this paper has proven to be a valuable candidate also for the modeling of sociotechnical systems such as OSS communities. At the

same time, the model also revealed some shortcomings which helps us to better understand the role of underlying assumptions and their limitations. At the end, not only the (positive) confirmation, but also the (negative) rejection of modeling assumptions both generate important insights into the dynamics of real sociotechnical systems.

Acknowledgments

Early investigations on this topic were supported by the Swiss National Science Foundation (Grant No. CR12I1_125298). We thank Natalia Frey for discussions about the Simon model.

References

[1] Clauset, A., Shalizi, C. and Newman, M., Power-law distributions in empirical data, *SIAM Rev.* **51** (2009) 661–703.
[2] Dempster, A. P., Laird, N. M. and Rubin, D. B., Maximum likelihood from incomplete data via the EM algorithm, *J. R. Stat. Soc., B* **39** (1977) 1–38.
[3] Garcia, D., Mavrodiev, P. and Schweitzer, F., Social resilience in online communities: The autopsy of friendster, in *Proc. First ACM Conf. on Online Social Networks* (*COSN'13*) (ACM, New York, 2013), pp. 39–50.
[4] Garcia, J. M. G., A fixed-point algorithm to estimate the Yule–Simon distribution parameter, *Appl. Math. Comput.* **217** (2011) 8560–8566.
[5] Geipel, M. M., Press, K. and Schweitzer, F., Communication in innovation communities: An analysis of 100 open source projects, *Adv. Complex Syst.* **17** (2014) 1550006.
[6] Geroski, P. A., What do we know about entry? *Int. J. Ind. Organ.* **13** (1995) 421–440.
[7] Krugman, P. R., *The Self-Organizing Economy* (Blackwell, Oxford, 1996).
[8] Madey, G. (ed.), *The SourceForge Research Data Archive* (*SRDA*) (University of Notre Dame, 2012).
[9] Mansfield, E., Entry, Gibrat's law, innovation, and the growth of firms, *Am. Econ. Rev.* **52** (1962) 1023–1051.
[10] Scholtes, I., Mavrodiev, P. and Schweitzer, F., From Aristotle to Ringelmann: A large-scale analysis of team productivity and coordination in open source software projects, *Adv. Complex Syst.* **17** (2014) in review.
[11] Schumpeter, J., *The Theory of Economic Development* (Duncker and Humblot, 1911).
[12] Simon, H. A., On a class of skew distribution functions, *Biometrika* **42** (1955) 425–440.
[13] Simon, H. A. and Bonini, C. P., On the size distribution of business firms, *Econ. Lett.* **48** (1958) 607–617.
[14] Spaeth, S., von Krogh, G. and He, F., Perceived firm attributes and intrinsic motivation in sponsored open source software projects, *Inf. Syst. Res.* **26** (2014).
[15] Sutton, J., Gibrat's legacy, *J. Econ. Lit.* **35** (1997) 40–59.
[16] Szell, M., Lambiotte, R. and Thurner, S., Multirelational organization of large-scale social networks in an online world, *Proc. Natl. Acad. Sci.* **107** (2010) 13636–13641.
[17] Tessone, C. J., Geipel, M. M. and Schweitzer, F., Sustainable growth in complex networks, *Europhys. Lett.* **96** (2011) 58005.
[18] Tomasello, M. V., Perra, N., Tessone, C. J., Karsai, M. and Schweitzer, F., The role of endogenous and exogenous mechanisms in the formation of R&D networks, *Sci. Rep.* **4** (2014) e5679.

[19] Yule, G. U., A mathematical theory of evolution, based on the conclusions of Dr. J. C. Willis, F. R. S., *Philos. Trans. R. Soc. Lond. Series B, Containing Papers of a Biological Character* **213** (1925) 21–87.
[20] Zanetti, M. S., Scholtes, I., Tessone, C. J. and Schweitzer, F., The rise and fall of a central contributor: Dynamics of social organization and performance in the GENTOO community, in *Proc. Sixth Int. Workshop on Cooperative and Human Aspects of Software Engineering* (*CHASE/ICSE '13*) (IEEE, 2013), pp. 49–56.