Hindawi
Scientific Programming
Volume 2020, Article ID 8814247, 13 pages
https://doi.org/10.1155/2020/8814247

Research Article

How to Evaluate the Productivity of Software Ecosystem: A Case Study in GitHub

Zhifang Liao,[1] Xiaofei Qi,[1] Yan Zhang,[2] Xiaoping Fan ,[3] and Yun Zhou[3]

[1]*School of Computer Science and Engineering, Central South University, Changsha 410075, China*
[2]*School of Engineering and Built Environment, Glasgow Caledonian University, Glasgow G4 0BA, UK*
[3]*Department of Information Management, Hunan University of Finance and Economics, Changsha 410075, China*

Correspondence should be addressed to Xiaoping Fan; xpfan@csu.edu.cn

Received 23 March 2020; Accepted 4 May 2020; Published 3 August 2020

Academic Editor: Chenxi Huang

With the development of open source community, the software ecosystem has become a popular perspective in the research on software development process and environment. Software productivity is an important evaluation indicator of the software ecosystem health. A successful software ecosystem relies on long-term and stable production activities by the users, which ensures that the software ecosystem can continuously provide the value needed by users. Therefore, the measurement of software ecosystem productivity can help maintain the user development efficiency and the stability of the software ecosystem. However, there is still little literature on the productivity of open source software ecosystems. By analogy with the natural ecosystem, this paper gives the relevant definitions of software ecosystem productivity and analyzes the factors affecting the productivity of software ecosystem. Based on the factors of the ecosystem productivity and their interrelationships, this paper establishes a software ecosystem productivity model and takes the GitHub platform as an example for detailed analysis and explanation. The results show that the model can better explain the factors affecting the productivity of software ecosystems. It is helpful for the research on the measurement of the software ecosystem health and the software development efficiency.

1. Introduction

The software ecosystem (SECO) consists of a software platform, a set of internal and external developers, and a community of domain experts in service to a community of users that compose relevant solution elements to satisfy their needs [1]. With the fast development of open source software, the development model of an independent company has been replaced by the collective intelligence cooperative development model gradually. And it formed complex whole. More and more researchers are starting to study the complex whole generated from this collaborative model from the perspective of software ecosystems. At the same time, how to assess the health status of the software ecosystem has become a significant research content in the software ecosystem.

The concept of software ecosystem productivity mainly comes from the natural ecosystem, where ecosystem health refers to the stability and sustainability of an ecosystem. The health of an ecosystem can be defined by three characteristics: productivity, organizational structure, and resilience [2]. By analogy with the natural ecosystem, Manikas K et al. [3] define the software ecosystem health as the ability to maintain variables and productivity over time. They think that the actor's productivity and robustness influence the ecosystem. Jansen et al. [4] use productivity, robustness, and niche to assess the health of open source ecosystems. As an important indicator of the health of software ecosystems, the software ecosystem productivity provides evaluation criteria for the development efficiency, production activities, and health status of software ecosystems. Quantitative evaluation of the software ecosystem productivity can provide theoretical basis for managers of ecosystem-related organizations to make better decisions. Currently, the detailed concept of the software ecosystem productivity is not very clear, so it is difficult to clarify the meaning of software ecosystem productivity and implement it on a specific open source platform. Secondly, there is a lack of general measurement models and methods for evaluating the software ecosystem productivity. It is difficult to quantify and explain the software ecosystem productivity specifically. Therefore, it is necessary to construct a general software ecosystem productivity assessment model. The purpose of the productivity assessment model is to provide reference basis for participants to choose appropriate software ecosystems for production activities.

In order to solve the above problems, this paper mainly makes three contributions. Firstly, this paper proposes the definition of the software ecosystem productivity. Based on the concept and evaluation method of the natural ecosystem productivity, the concept and influencing factors of software ecosystem productivity are determined. Then, according to the model of the natural ecosystem productivity, the quantitative model of software ecosystem productivity analysis is constructed by analyzing the influence of various factors on productivity. Finally, for detailed illustration, the typical open source software platform GitHub is selected for empirical study as an example.

The structure of this paper is as follows: Section 2 mainly introduces the related work of the software ecosystem productivity and health measurement. Section 3 introduces the research questions and dataset. Section 4 defines the productivity of software ecosystem and gives the software ecosystem productivity model. Section 5 is a case study of the productivity in GitHub software ecosystems. Section 6 verifies the productivity model. Finally, the paper is concluded in the last section.

2. Related Work

In 2003, Messerschmitt and Szyperski [5] proposed the concept of software ecosystems. They defined the software ecosystem as an online community organization that has a common interest in core software technologies. Later, many researchers defined the software ecosystem from different perspectives and backgrounds. Based on a detailed study of the existing literature in the field of the software ecosystem, in 2013, Manikas and Hansen [1] defined a software ecosystem as the interaction of a set of actors on top of a common technological platform that results in a number of software solutions or services.

In recent years, there are many different research directions in the field of software ecosystem. Researchers have studied software ecosystems from many aspects, such as software ecosystem health and software ecosystem architecture. Software ecosystem health is still a research topic that many people pay attention to. At present, there are many intelligent tools or models for assessing the health of personal [6] or natural ecosystems [2]. However, in the software ecosystem health, productivity is an important evaluation indicator. How to quantify the software ecosystem productivity is still a problem to be solved [7]. In terms of health measurement of software ecosystem, Manikas and Hansen [3] reviewed the existing literature on health status of software system, compared it with the concepts of natural ecosystem and commercial ecosystem, and defined the health status of software ecosystem as the ability of ecosystem to sustain development and maintain variability and efficiency. Gamalielsson et al. [8] used developer responsiveness as an open source ecosystem health indicator for a single project. Amorim et al. [9] proposed a conceptual framework to actively support SECO health participants in the public sphere. It is based on the business ecosystem health indicators including productivity, robustness, and niche creation defined by Iansiti and Levien [7]. The productivity in this framework mainly refers to the influence of architects on platform functions and management. In their work, Eclipse and KDE were used as examples to describe architecture practices that promote and improve ecosystem health. Compared with our work, it discussed the impact of different roles of software architects on ecosystem health. However, our work aims to measure productivity through the historical data of the software ecosystem. Berk et al. [10] proposed a SECO-SAM model for comprehensive evaluating software ecosystem strategy. This model involves the SECO biology, lifestyle, environment, and healthcare organizations. The productivity is a basic level of software ecosystem health in SECO biology, but they only discussed the SECO lifestyle and environment in the case study. Franco [11] proposed a QuESo model to measure the health of software ecosystem based on sustainability, maintenance, process maturity, network health, and resource health. This study mainly introduced the indicators about software ecosystems. These studies mostly use the productivity indicator but do not mention any approach. Jansen [4] provided a multilevel and comprehensive Open Source Ecosystem Health Operationalization (OSEHO) using three pillars, being productivity, robustness, and niche creation pillars, to assess the health of open source ecosystems. And the pillars are separated into three layers, being the theory level, the network level, and the project level. It provides insight into the indicators but does not provide a detailed method for their operation. Liao et al. [12] proposed to measure the sustainability of open source software ecosystem from the aspects of openness, stability, activity, and scalability and applied the evaluation method to Stack Overflow. And Liao et al. [13, 14] also defined the indicators affecting GitHub ecosystem health from the perspective of vitality, organizational structure, and elasticity and proposed the GitHub ecosystem health prediction method. They also forecasted the lifespan length of projects and proposed a prediction model to estimate the project lifespan in open source software ecosystems.

The software ecosystem currently lacks appropriate management theory, support tools, and solid experience. Although researchers have proposed a measurement framework for software ecosystem health, they have not quantified the model to a specific platform and have not had in-depth analysis of the specific conditions of various factors of software ecosystem health. The software ecosystem productivity plays a huge role in assessing the health of the software ecosystem and improving the productivity of the software ecosystem. This paper studies the relevant factors affecting software ecosystem productivity and verifies it by constructing a software ecosystem productivity model.

3. Research Questions and Data Gathering

3.1. Research Questions. The purpose of this paper is to study the productivity of the open source software ecosystem. Through defining the representation of productivity in GitHub and analyzing related factors, we proposed a specific, concrete operational approach and analysis model to evaluate productivity. To achieve that, the research questions answered by the paper are as follows:

Q1 What is the definition of the productivity of open source software ecosystem? And what can explain the productivity on the GitHub platform? Based on the natural ecosystem, we defined the software ecosystem productivity including software primary productivity and software secondary productivity. Also, we compared the productivity of open source software ecosystem with the productivity of natural and business ecosystems.

Q2 What are the factors that affect productivity? What are the specific effects of these factors on primary productivity and secondary productivity? Based on the definition in Q1 and the calculation model of the natural ecosystem productivity, we found out the factors affecting the productivity of the ecosystem and proposed a hypothetical productivity model. It was analyzed and verified in the example study latter.

Q3 How should we measure the productivity in open source software ecosystems? Dose this hypothetical open source software ecosystem productivity model hold? To answer this question, we used an example study of GitHub to explain this approach and constructed the model in Section 5.

Q4 Can this evaluation method and productivity model be applied to ecosystems of GitHub and other platforms? In Section 6, the applicability of the productivity model on other platforms is verified with the same evaluation method.

3.2. Platform Introduction. As the largest and fastest-growing open source ecosystem in recent years, GitHub has attracted millions of developers to release open source projects. At the same time, it opened APIs to provide a convenient way for researchers to obtain the required data. Therefore, this paper focused on projects in the GitHub open source ecosystem. GitHub is an open source community that performs code changes based on pull requests. In open source projects, users can perform a variety of actions, including forking, adding stars, watching, creating issues, commenting, pulling requests, pushing, and making commits. Users can be divided into owners, core developers, and noncore developers. Owners can execute all activities and assign privileges to other developers. Core developers can directly perform submission activities on code by pushing their changes after assigning permissions. Noncore developers submit code changes by pulling requests and can only perform submission activities after core developers review. GitHub is a project hosting platform for open source and proprietary software projects. It is also the most popular open source library at present. GitHub platform saves a lot of historical data of development process. These data provide material for the study of project status. Project data is easy to obtain, and the API of GitHub website provided data crawling routes with high data integrity.

Therefore, this paper took GitHub ecosystem as an example to build a software ecosystem productivity model and verify the accuracy and universality of the model.

4. Definition and Methodology

In this section, the definition of software ecosystem productivity is provided firstly. Then, the method used in the experiment is explained in detail.

4.1. Productivity Definition. Q1 What is the definition of the productivity of open source software ecosystem? And what can explain the productivity on the GitHub platform?

In the natural ecosystem, biological productivity refers to the ability to produce substances at different life levels, such as individuals, groups, ecosystems, regions, and even biospheres. It determines the overall material cycle and energy flow and is also an important indicator of the health status of the system [15]. The concepts of biological productivity include primary productivity (GPP) and secondary productivity. Primary productivity in ecosystem refers to the fixed solar energy or organic matter manufactured by plants. Secondary productivity refers to the ability of consumers to metabolize the substances manufactured by primary production and stored energy and form their own substances and energy through assimilation. Additionally, in human ecology, many other energy sources, such as wind energy, are also used. There are many different affected factors and measurement methods for the different types of energy [16]. In business ecosystem, the productivity is defined as how the ecosystem effectively converts raw materials into living organisms. It is the capacity of the ecosystem to transform inputs into new products and functionalities with low cost [7]. And it emphasizes the delivery of innovations and the lower costs. The difference between the definition of productivity in natural ecosystems and business ecosystems is that natural ecosystems measure the ability to produce a product, mainly based on the total number, while business ecosystems pay more attention to efficiency and measure how to produce more products at a lower cost.

By drawing an analogy between natural ecosystem and business ecosystem, this paper gives the definition of software ecosystem productivity with more focus on the production and information. The definition of productivity in this paper differs from that in business ecosystem. It focuses on the ability of software ecosystem to produce ecological products and does not reflect the innovation of the software ecosystem. And it can be measured from activity data in the ecosystem.

Definition 1(software ecosystem total productivity). The total productivity of software ecosystem (SEP) refers to the amount of information generated by the interaction and collaboration of participants, platforms, and supporters in the ecosystem.

Taking the GitHub *platform* as an example, the user's main contributions include building code repository, committing codes, presenting issues, and commenting on different issues and commits. These behaviors produce a series of interactive information. This information brings vitality to the production activities of the ecosystem. It becomes the total production of the ecosystem.

Definition 2(software ecosystem primary productivity). The software ecosystem primary productivity (SEPP) refers to all information produced by participants that affects ecosystem products in a software ecosystem.

Different software platforms have different functions, and the information affecting ecosystem products is also different. For *example*, in the GitHub platform, ecosystem products are mainly codes. Therefore, the information affecting software ecosystem products produced by participants mainly includes committing codes, making an issue, and repairing defects.

Definition 3(software ecosystem secondary productivity). Software ecosystem secondary productivity (SESP) refers to the valuable information that the participants produce and the ecological products which directly affect the software eco-products in the software ecosystem.

This information is usually further processed by the previously generated information, such as reviewing and merging pull requests, closing issues and commits, or commenting on commits and issues in the GitHub. The commit generates a large amount of code information. After the code is approved and merged, it can be merged into the original code repository. It affects the project version and forming valuable information for the software ecosystem. Similarly, after the issue, commit, or pull requests behavior are generated, the user will comment on these issues and commit to exchange information. The information on the impact of these production activities on the production activities of users is the subproductivity of software ecosystem. In the platform-based open source software ecosystem, users are the main participants of the platform. A series of activities on the platform interact with the platform to generate information. This information promotes the normal development of the platform and becomes the energy to maintain the normal operation of the software ecosystem.

4.2. Productivity Model Hypothesis. *Q2* What are the factors that affect productivity? What are the specific effects of these factors on primary productivity and secondary productivity?

The productivity index of software ecosystem can be used to evaluate the development efficiency of software projects and the health of software ecosystem. Participants carry out production activities in the ecosystem. The contribution of participants is also the main energy source of open source software ecosystems. Productivity in software ecosystem is mainly generated by the interaction between software participants and platforms. The main factors affecting the productivity of software ecosystem include platform factors and user factors. Platform factors are mainly external factors that have great impact on the software ecosystem, including project popularity and project development language. Participant factors mainly include the number of participants in the software ecosystem and the willingness of participants to contribute. This paper mainly studies whether there is a certain relationship between the activities of the participants and the productivity of the software ecosystem and whether this relationship can be quantified in a similar way to the quantitative model of natural ecosystem productivity.

In the measurement model of productivity in natural ecosystems, the main influencing factors of productivity are illumination radiation, that is, the input of energy. The calculation model of primary productivity is usually obtained by multiplying the coefficient of the factors affecting the conversion of illumination energy by the amount of effective illumination. In terrestrial ecosystems, geographic detection platforms usually use GLOPEM model algorithm [17] to retrieve primary productivity data from various satellite remote sensing data. Primary ecosystem productivity can be expressed as

$$\mathrm{GPP} = \mathrm{PAR} \times \mathrm{FPAR} \times \varepsilon. \tag{1}$$

In equation (1), *GPP* represents primary ecosystem productivity, PAR is photosynthetically active radiation, FPAR is the ratio of photosynthetically active radiation absorbed by vegetation, and ε is the actual light utilization rate based on the concept of GPP. The multiplication of PAR and FPAR is the photosynthetically effective radiation absorbed by vegetation, that is, the light utilization rate of vegetation.

This paper mainly analyzed the influence of factors of the number and activities of participants on productivity in software ecosystem. In software ecosystem, the contribution activities of participants are the main energy source of open source software ecosystem. Therefore, according to the abovementioned quantitative model of natural ecosystem productivity, the software ecosystem productivity model is similarly represented as a linear function of participants. We hypothesize a quantitative model of software ecosystem productivity as equation (2). Then, we verify it in the example study:

$$\mathrm{SEPP} = Ac * Pe + C. \tag{2}$$

In equation (2), SEPP represents the primary productivity of software ecosystems, and it is a linear function related to the number of users and the willingness to contribute. *Pe* is the number of participants, *Ac* is the introduced parameter, representing the activity factor, and *C* is a constant. In different natural ecosystems, the primary productivity of ecosystems will vary depending on the effective radiation ratio of vegetation to absorb photosynthesis. Similarly, in different software ecosystems, different programming languages, project lifetimes, project followers, etc. are not exactly the same for the effective information production rate of the ecosystem. These different factors together constitute the *Ac*. And the default minimum productivity of a participant is 1. Thus, the value of SEPP is the number of participants when C is negative number and $Ac * Pe$ is less than C absolute value.

Similarly, according to the definition of secondary productivity, secondary productivity is transformed from primary productivity. And it is also a linear function related to the number of users and willingness to contribute. This hypothesis can be expressed as

$$\text{SESP} = \text{SEPP} * Cr + C1. \tag{3}$$

In equation (3), SESP is the secondary productivity of software ecosystem, Cr is the productivity conversion rate, and $C1$ is the constant. According to the primary productivity formula, the relationship model of secondary productivity can be transformed as follows:

$$\text{SESP} = Ac' * Pe + C2. \tag{4}$$

In equation (4), Ac' is the conversion parameter of productivity, which is obtained by multiplying the active factor by the conversion rate. $C2$ is a constant; the value is equal to $C1$ multiplied by the conversion rate with a constant added. It can also be predicted that the secondary productivity of software ecosystem is linearly related to the number of users and their willingness to contribute.

In Sections 5 and 6, we specifically analyzed the influencing factors of software productivity for different platforms and made an empirical study on the feasibility and universality of the abovementioned hypothetical model to verify whether the models can express the impact of participants on the ecosystem productivity.

4.3. Methodology. Q3 How should we measure the productivity in open source software ecosystems? Dose this hypothetical open source software ecosystem productivity model hold?

The research method analyzed the factors affecting the productivity. Then, the productivity model of software ecosystem was verified and established. It was divided into three steps.

Step 1 (data collection): firstly, the appropriate data were selected and the ecosystem was divided. In order to verify the feasibility of the model, the typical open source software platform GitHub was selected for empirical research. We took GitHub platform as an example in Section 4. Because project popularity and development language are important platform factors affecting ecosystem, in order to eliminate the impact of these factors in the ecosystem, we divided the ecosystem with different types of platforms, mainly using development language as the index. For platforms that cannot divide ecosystem according to the development language, we adopt other dividing standards such as project type and project popularity.

Step 2 (correlation analysis): in order to analyze the impact of user activities on software ecosystem productivity, the relationship between productivity and ecosystem participants in software ecosystem was analyzed. The data of productivity and participants in the ecosystem were analyzed in a month-long observation period, and the relationship between different types of productivity in the ecosystem and those produced by people in a unit time were analyzed. The data used in this paper are spaced monthly and the data variables are equidistant, the person formula is applicable to measure the coefficient of linear relationship between the fixed distance variables, and the data scale is suitable for the calculation of the person correlation coefficient, so this paper used the person correlation coefficient to analyze the correlation between the participant data and the software productivity data:

$$r_{\text{xy}} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\left(\sqrt{\sum_{i=0}^{n}(x_i - \overline{x})^2}\right)\left(\sqrt{\sum_{i=0}^{n}(y_i - \overline{y})^2}\right)}. \tag{5}$$

In equation (5), i is the first month of time, x is the number of software ecosystem participants, and y is productivity of this software ecosystem. r_{xy} represents the correlation coefficient between productivity and participants, $r_{\text{xy}} \in (-1, 1)$; r_{xy} is larger, which indicates that the software ecosystem productivity is more positively correlated with the participants. The negative value means that the software ecosystem productivity is negatively correlated with the participants. And r_{xy} tends to 0, which means that there is no correlation between them.

Step 3 (model construction): the construction of this model was divided into two parts. The first is regression analysis. According to the prototype of the software ecosystem productivity model, the linear relationship between productivity and participants was judged. Under the condition of linear relationship, the initial regression equation between productivity and participants was obtained by the least square method [18]. Because the productivity created by different types of producers is different in the same ecosystem, the specific ecosystem presents different regression models because of the different activity of participants.

The second is the construction of the real model. The regression equation of each ecosystem is inconsistent. To determine a regression model applicable to most projects, this study used truth discovery methods [19]. Truth discovery is a method to measure the reliability of multisource information and estimate the real information. The flow of this algorithm was shown in Algorithm 1. By this method, we calculated the reliability of the regression equation based on each project and obtained a general regression equation with a greater accuracy.

```
Input: Data from $n$ project: $\{Ac_1, \ldots\ldots, Ac_n\}$, $\{c_1, \ldots\ldots, c_n\}$
Output: Truths $Ac^{*\,(t)}$, $c^{*\,(t)}$
(1) Initialize the truths
      $Ac^{*\,(1)} = \sum_{i=1}^{n} Ac_i/n$
      $c^{*\,(1)} = \sum_{i=1}^{n} c_i/n$
(2) repeat
(3)    for $i \leftarrow 1$ to N do
(4)      for $i \leftarrow 1$ to $T$ do
(5)         $w_i = \log(\sum_{i=1}^{n}((Ac^{*\,(t-1)} - Ac_i)^2) + ((c^{*\,(t-1)} - c_i)^2)/(Ac^{*\,(t-1)} - Ac_i)^2 + (c^{*\,(t-1)} - c_i)^2)$
(6)         $Ac^{*\,(t)} = \sum_{i=1}^{n}(w_i * lip_i)/\sum_{i=1}^{n} w_i$
(7)         $c^{*\,(t)} = \sum_{i=1}^{n}(w_i * c_i)/\sum_{i=1}^{n} w_i$
(8)      end for
(9)    end for
(10) until Convergence criterion is satisfied;
(11) return $Ac^{*\,(t)}$, $c^{*\,(t)}$
```

Algorithm 1: Truth discovery.

The specific algorithm was described as follows. Firstly, the initial settings of $\{Ac_1, \ldots\ldots, Ac_n\}$, $\{c_1, \ldots\ldots, c_n\}$ are determined by the regression equation of each item and sorted. The results of the first iteration are the average value of $Ac^{*\,(1)}$ and $c^{*\,(1)}$. Secondly, the weight (w_i) of each item in the overall ecosystem is calculated, where i represents the ecosystem number i, c_i represents the activity of the ecosystem i obtained through the regression equation, $Ac^{*\,(t)}$ represents the activity result of the t-time iteration, and $c^{*\,(t)}$ represents the constant result of the t-time iteration. Finally, the results of the t-time iteration are calculated.

In this paper, we used the truth discovery algorithm to get the real software ecosystem productivity quantification model under a specific software ecological platform. The specific experimental process was described in detail in Section 5.

5. Case Study

This paper focused on the analysis of user factors influence on ecosystem productivity. This section verified the relationship between the number of users, activities, and productivity. In this section, we took GitHub as an example to quantify the software ecosystem productivity model and conduct empirical research.

5.1. Data Collection. The first task was data collection and preprocessing. The main external factors affecting ecosystem productivity include the function type, the popularity of the project, and the popularity of the development language. Based on the type of development language, this paper chose seven of the most popular languages and divided GitHub platform into several technological ecosystems to analyze the productivity model.

In order to scientifically compare the popular languages of each platform, we ranked the top 10 languages in the four platforms of Stack Overflow, GitHub, TIOBE, and IEEE. As shown in Table 1, we reversed the popularity of these languages. The first language in each platform had 10 points, the second had 9 points, and so on. The scores of the four platforms were added, and we selected the top 7 languages in terms of popularity. Then, technical ecosystems of the platform were divided by language as the main factor, and the ecosystem productivity was analyzed. The final scores were JavaScript, Java, C, C++, C#, PHP, and Python, the seven different development languages for analysis.

Table 1: Comparison of language popularity on different platforms.

Score	Stack Overflow	GitHub	TIOBE	IEEE
10	JavaScript	JavaScript	Java	Python
9	Java	Python	C	C++
8	C#	Java	C++	C
7	PHP	Ruby	Python	Java
6	Python	PHP	VB.net	C#
5	Html	C++	C#	PHP
4	C++	css	PHP	R
3	css	C#	JavaScript	JavaScript
2	SQL	Go	SQL	Go
1	ASP.NET	C	Swift	Ass

Using the seven popular languages mentioned above, the platform was divided into software ecosystems of different languages. According to the definition of software ecosystem productivity in this paper, the data of user contribution activities were used. GitHub platform saves a lot of historical data of development process, and the API of the website provides high data integrity for data crawling. As shown in Table 2, this paper used several projects with the highest attention in GitHub and collected 70 projects with the largest number of stars in JavaScript, Java, C, C++, C#, PHP, and Python.

5.2. Correlation Analysis. After data processing was completed, the correlation between productivity and participants was analyzed. This paper attempted to find out the relationship between the number of participants and productivity through the statistical analysis of average user activities.

Table 2: Projects of the largest number of stars in different languages on GitHub.

JavaScript	Java	C++	C#	C	Python	PHP
Vue	Java-design-patterns	TensorFlow	Shadowsocks-windows	Linux	Awesome-python	Laravel
React	RxJava	Electron	CodeHub	Netdata	System-design-primer	Symfony
D3	Elasticsearch	Swift	CoreFX	Redis	Public-apis	Faker
JavaScript	Spring-boot	Bitcoin	PowerShell	Git	Models	Composer
React-native	Retrofit	NW.js	Wox	Ijkplayer	Youtube-dl	CodeIgniter
Angular.js	Interviews	x64dbg	CoreCLR	Php-src	Flask	DesignPatternsPHP
Font-Awesome	OkHttp	Protobuf	Roslyn	Wrk	Thefuck	SecLists
Create-react-app	Guava	OpenCV	Dapper	How-to-Make-a-Computer-Operating-System	Httpie	Framework
Node	MPAndroidChart	Caffe	WaveFunctionCollapse	the_silver_searcher	Django	Guzzle

In a specific software ecosystem, it is necessary to select data that can represent the ecosystem productivity and make correlation analysis with the number of users. In GitHub ecosystem, the most important contribution of users usually comes from the pull requests behavior, so PR data is a good representation of the software ecosystem productivity, while valuable PR is usually merged into the project code base. So, the merged PR was used as the secondary productivity after transformation. The correlation between the productivity and the ecological participants in software ecosystem projects was analyzed.

In this paper, the productivity data of each project in unit time and the data of participants in statistical time threshold were counted according to the observation period, and the correlation coefficient of person was used to analyze the correlation between participant data and software productivity data. The results are shown in Figure 1.

As shown in Figure 1, in the GitHub platform, a highly positive correlation can be found between the total PR and the total number of participants. Among them, the lowest degree of association of a development language is 0.922, while the highest level is JavaScript ecology, up to 0.986. The number of PR that has been merged and the number of participants are also highly positively correlated, and the degree of association ranges from 0.811 to 0.967. Therefore, it can be concluded that, in GitHub ecosystem, there is a highly positive correlation between the productivity of software ecosystem and the number of participants in ecosystem, and the number of participants directly affects the productivity.

Based on the above analysis of the correlation between productivity and the number of users, it was concluded that there is a highly positive correlation between the productivity of software ecosystem and the number of participants in the ecosystem. The number of participants directly affects the value of productivity. Because productivity comes from the interaction of all participants in the ecosystem, the main influencing factors of software productivity are the number and activity of producers in the ecosystem. Therefore, this paper used the number of users and the activity of users to build a model of the impact of participants on productivity.

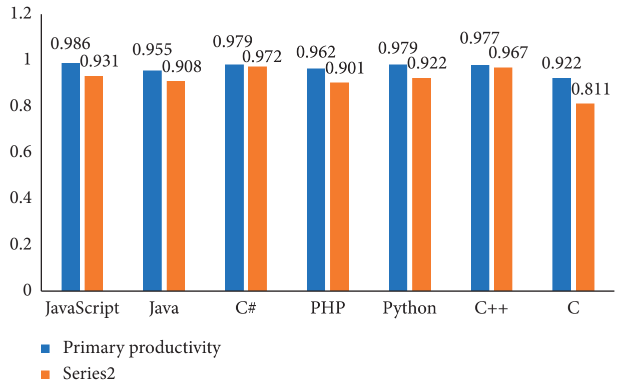
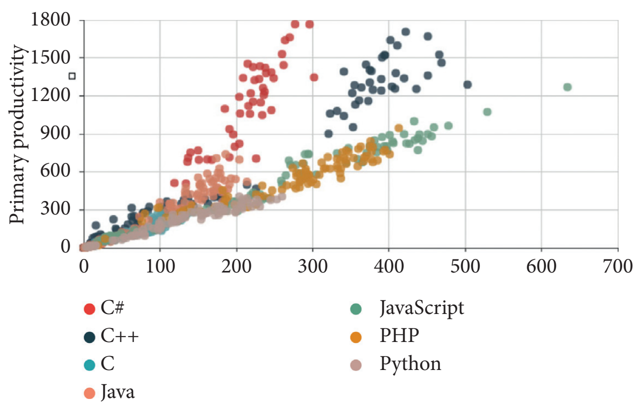


Figure 1: The correlation between the number of participants and productivity on the GitHub platform. The blue column represents the correlation between primary productivity and the number of participants, while the orange column represents the relationship between secondary productivity and the number of participants.

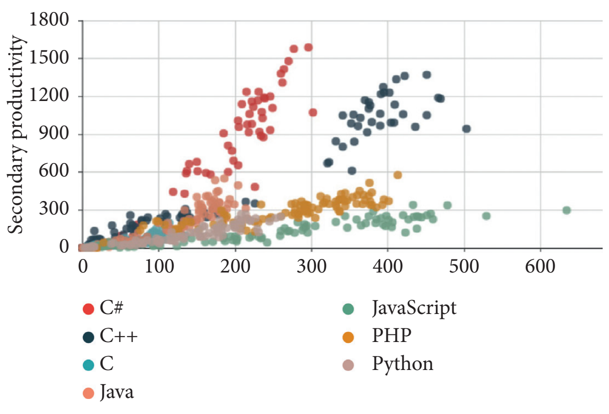
5.3. Model Construction

5.3. Model Construction. According to the above correlation analysis, it was found that there is a clear positive correlation between ecosystem productivity and the number of participants. Therefore, it was necessary to judge whether productivity and the number of participants have linear function relations and carry out regression analysis. On the GitHub platform, the impact model of participants on productivity was constructed by using seven representative languages. In Figure 2, we found that there is a linear relationship between productivity and the number of participants. Specific projects or ecosystems present different regression models depending on the activity of participants.

As shown in Figure 2, in every ecosystem, productivity is a linear function related to the number of participants. Primary and secondary productivity models can be met in both primary and secondary productivity. Therefore, the



(a)



(b)

Figure 2: Correlation analysis between productivity and participant. (a) The correspondence between the number of participants and the primary productivity in each observation period on the GitHub platform. The dots of different colors represent the correspondence between the participants of different languages and productivity. (b) The correspondence between the number of participants and the secondary productivity.

Table 3: The linear function of productivity related to the number of participants.

Languages	Equation of primary productivity	Equation of secondary productivity	Ac	C1	Ac*Cr	C2
C#	$y = 5.46x - 31.985$	$y = 4.66x - 29.407$	5.46	31.985	4.665	29.407
C++	$y = 3.37x - 24.101$	$y = 2.69x - 31.796$	3.37	24.101	2.690	31.796
C	$y = 2.05x - 9.555$	$y = 0.97x - 6.834$	2.05	9.555	0.967	6.834
Java	$y = 2.94x - 13.341$	$y = 1.97x - 29.100$	2.94	13.341	1.974	29.100
JavaScript	$y = 2.05x - 5.639$	$y = 0.56x - 10.923$	2.05	5.639	0.557	10.923
PHP	$y = 1.93x + 11.439$	$y = 0.99x + 23.920$	1.93	11.439	0.987	23.920
Python	$y = 1.65x + 2.464$	$y = 0.94x - 19.103$	1.65	2.464	0.941	19.103

primary regression equation was obtained by the least square method, as shown in Table 3.

Since the regression equations of each project are inconsistent, the lowest activity factor, Ac, was 1.65 and the highest was 5.46. The constant C1 ranged from −31.99 to 11.44. Through analysis, it was found that the active factor can usually indicate the willingness of users in the ecosystem. To obtain a regression model for most projects, the participant's impact model on productivity was constructed using the real-discovery approach described in Section 3:

$$\text{SEPP} = 2.22 * Pe - 9.44, \tag{6}$$

$$\text{SESP} = 1.20 * Pe - 21.11. \tag{7}$$

In equation (6), SEPP is the primary productivity of GitHub ecosystem, and in equation (7), SESP is the secondary productivity. Pe represents the number of participants in the software ecosystem. The primary activity was 2.22, the secondary activity was 1.20, the constant C1 of primary productivity was −9.44, and the constant C2 of secondary productivity was −21.11. And in this paper, the default minimum productivity of a user was 1. The value of SEPP and SESP is the number of users when C is a negative number and Ac*Cr is less than C absolute value.

Through the analysis of the software ecosystem productivity of GitHub platform, it was found that the software productivity model proposed in Section 4 can be applied to multiple software ecosystems. Because of the different functions of the specific ecosystem and the different user groups, the user's active degree will be different. Therefore, when using the model, we need to analyze the productivity of the software ecosystem according to the user's active degree on different platforms. Usually, the average activity of users can be used to replace the active factors derived from the inversion of productivity and the number of participants.

6. Verification Analysis

Q4 Can this evaluation method and productivity model be applied to ecosystems of GitHub and other platforms?

To answer this question, we verified the model in other ecosystems in this section, which include three different class ecosystems in GitHub. Then, the ecosystems in Stack Overflow and Bugzilla were verified.

6.1. Verification in Other Ecosystems of GitHub. In this section, we verified the model in the three different class ecosystems in GitHub. In biology, the range of ecosystems can be large or small, and ecosystem productivity can represent the production capacity of individuals, groups, ecosystems, regions, and even biosphere. Similarly, this paper selected three software ecosystems of different sizes and types in GitHub for verification. They are the single software product ecosystems, the software development team ecosystems, and the language ecosystems. The relationship between model productivity and actual productivity was verified by statistical calculation.

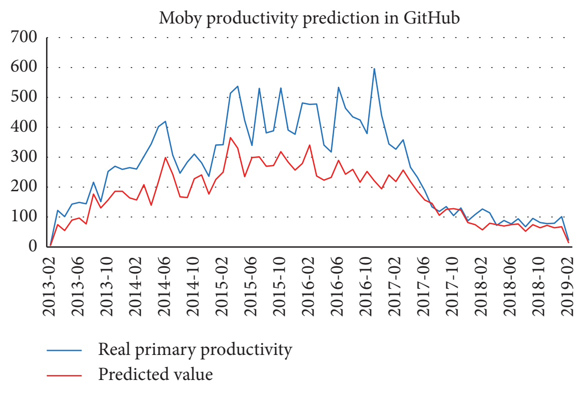
Three different ecosystems, Moby, GitBook, and Ruby, were selected to validate the model. Moby is an open source project dedicated to promoting the movement of software containerization. In the Moby project, users and software frameworks, components, and other software products gather to form a software ecosystem. The GitBook team is mainly a development team for text editors using Git technology. In the GitBook ecosystem, users and software development environments, software products, services, and others are condensed together through a team to form a software ecosystem. Ruby is a simple and fast object-oriented scripting language. It is a popular project development language in GitHub. In the Ruby ecosystem, users, software products, and development environment form a software ecosystem with the same development language.

Figure 3 shows that the software ecosystem primary productivity was calculated by the productivity model. It was consistent with the trend of the actual productivity of the software ecosystem, and the quantity is roughly the same. And through correlation analysis, the correlations between model productivity and measured productivity in the single software product ecosystem, the software team ecosystem, and the language ecosystem were higher than 90%. However, there was a difference between the predicted and the measured value. Because this model analyzed the characteristics of multiple ecosystems, the large dataset obscured the characteristics of specific ecosystems. It caused the predicted value to be different from the average productivity of the platform. The predicted value is greater than the measured value; this means that the productivity of this ecosystem is lower than the average productivity of this platform, and user activities should be improved. The predicted value is less than the measured value; this means that the users of this ecosystem are more active, and the productivity is higher than the average productivity of the platform. When the productivity of the ecosystem is continuously higher than the prediction model, the model should be adjusted according to the characteristics of this software ecosystem to ensure its accuracy. This fully proved that the abovementioned composition model of software ecosystem primary productivity is applicable to GitHub and other software ecosystems. It was also found that the number of users is limited by the environmental capacity of the ecosystem during the stable operation of the platform. In the software ecosystem, the transition from primary productivity to secondary productivity takes time. It takes some collaboration with other users to convert primary productivity to secondary productivity. Hence, the prediction of secondary productivity was not very good in the last few months. Also, the prediction of secondary productivity of these three ecosystems in GitHub is analyzed. The result is shown in Figure 4.

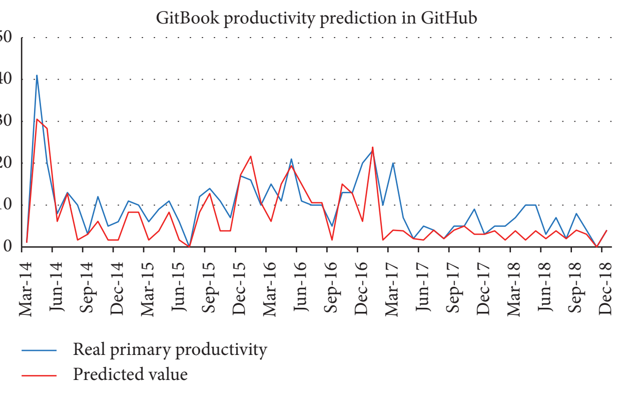
During the verification process, we found the impact of factors other than the number of users and activity on the software ecosystem productivity. On the GitHub platform, the software primary productivity problem had a high correlation with the number of participants, but when the ecosystem is small and the data volume is sparse, the accuracy of the model will be greatly reduced. During the productivity verification experiment of a single project team, it was found that each user participating in the PR submission would have one or two problems, but a few core developers would generate a large number of submissions during certain observation periods, resulting in partial errors in the model. The reason why the secondary productivity and the number of participants are lower than the primary productivity was that the large number of submissions generated by these few core developers is often incorporated into the code base, so the primary productivity depends more on the activities of the core developers. In the initial phase of the project, the contribution rate of core users is usually high. However, as the project progresses, the number of noncore developers participating in the project will increase, and the proportion of secondary productivity converted from primary productivity will gradually increase. Therefore, in the next step of the work, the impact of the core developer's user activities will be considered.

In the experiment, we also found that, in the process of stable operation of the platform, there is no sudden effect of external force. And after the number of users reaches a certain level, it will remain in a range for a long time. Therefore, the most important thing for primary productivity is to improve the participants and active level. And in these open source software ecosystems, the transition from primary productivity to secondary productivity takes time and needs to be discovered and collaboratively completed by other users to drive primary productivity into secondary productivity. Therefore, the amount of secondary productivity is not very good in the last few months. And because the ecological secondary productivity is converted from primary productivity, the secondary productivity is almost zero when the primary productivity is low, so the secondary productivity model is too dependent on the core user for the product of a small project team. Therefore, in order to increase secondary productivity, it is necessary to increase primary productivity, conversion factor, and number of core users.

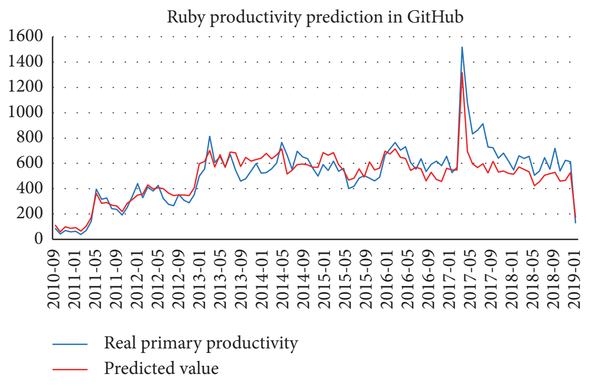
6.2. Verification in Other Ecosystems of Other Platforms. To verify the universal applicability of this model, we also verified the model in the ecosystems of other ecosystems. Using the same method, the software ecosystem productivity of Stack Overflow platform and Bugzilla platform was obtained. Stack Overflow is a standard Q&A website on computer science and programming topics. On the Stack Overflow platform, users can perform a variety of different activities such as questions, answers, votes, and comments. Users participate in group intelligence collaborative activities such as questions and answers to form an open source ecosystem. It has been highly popular with software



(a)

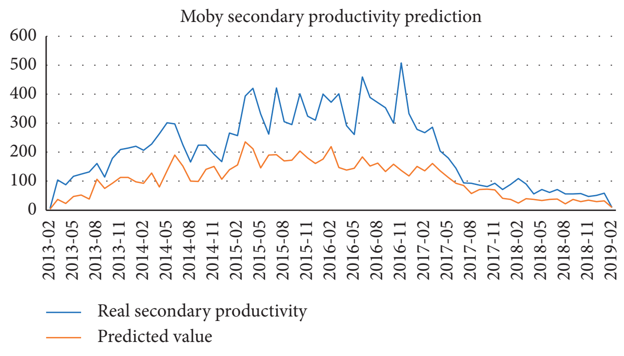


(b)

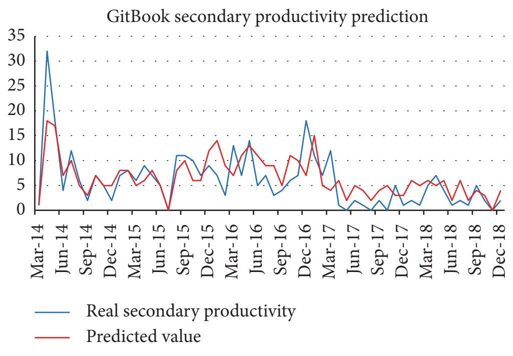


(c)

Figure 3: Model predictive value vs. real primary productivity. (a) The Moby project primary productivity predictions compared to real primary productivity. (b) The GitBook team's primary productivity predictions compared to real primary productivity. (c) The Ruby language ecosystem primary productivity predictions compared to real primary productivity. The red line indicates the predicted value. The blue line represents the real value. The abscissa axis is the different month and the ordinate axis is the productivity of the ecosystem.

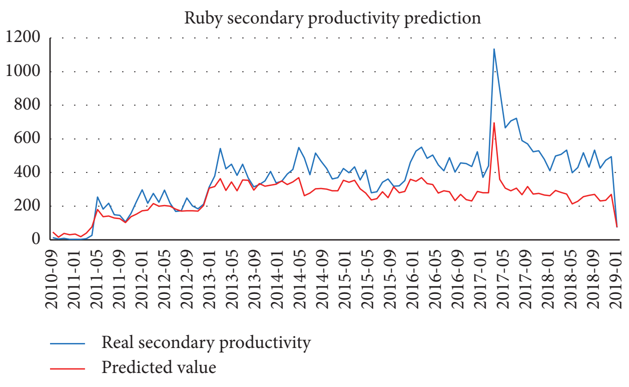


(a)



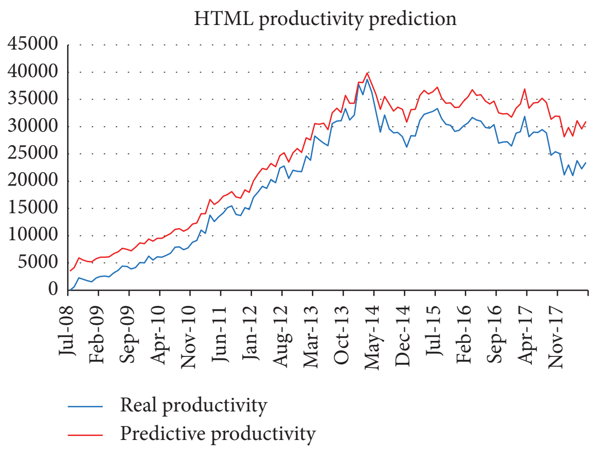
(b)

Figure 4: Continued.

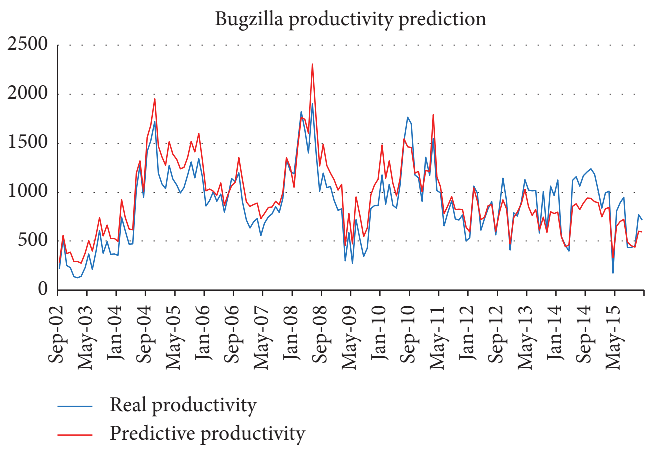


(c)

Figure 4: Model predictive value vs. real secondary productivity. (a) The Moby project secondary productivity predictions compared to real secondary productivity. (b) The GitBook team's secondary productivity predictions compared to real secondary productivity. (c) The Ruby language ecosystem secondary productivity predictions compared to real secondary productivity.



(a)



(b)

Figure 5: Model predictive value vs. real productivity. (a) The productivity predictions compared to real productivity on Stack Overflow. (b) The productivity predictions compared to real productivity on Bugzilla of Mozilla projects.

developers and is considered to be one of the most successful open source ecosystems. Bugzilla is a defect tracking system developed by the Mozilla Foundation (http://www.mozilla.org). Many users, including open source projects (Apache, Open Office for Linux), private projects, and public agencies (NASA, IBM) are using Bugzilla. In Bugzilla, the user's contribution mainly starts from the user's submission of a bug report, which is passed to the defect repair, inspection activity behavior, etc. Therefore, we selected these software ecosystems to verify the productivity model.

Before verification, the ecosystems and their productivity data in these platforms were determined. In the Stack Overflow, the information is in questions and answers. Therefore, the number of questions and answers represent productivity. In the Bugzilla, the user's contribution starts from the bug report to the bug repair. Therefore, the productivity is represented by the number of reports. Then, the ecosystems on the platform were divided. According to the characteristics of the platform, users in Stack Overflow usually gather with different technical fields. Therefore, Stack Overflow was divided according to the programming language. In Bugzilla, there are five types of projects, so the ecosystems were divided in terms of project types.

Then, we determined the activity factor of these ecosystem models according to the method in Section 4.3. First, the correlation between productivity and the number of participants was analyzed. It was found that, on the Stack Overflow, the productivity is highly positively correlated with the number of participants. The lowest correlation was 0.88 and the highest was 0.995. On the Bugzilla, there was also a positive correlation between productivity and participants with a maximum of 0.94 and a minimum of 0.65. Then, according to the method in 4.3, regression analysis and truth discovery were used to adjust the productivity model parameters of the corresponding platform. Finally, the model was verified. On the Stack Overflow platform, the ecosystem of HTML language was selected to verify the model. On the Bugzilla platform, we chose the ecosystem of the single software product Firefox for verification. The result was shown in Figure 5.

The correlation between the model productivity and real productivity of the two platforms was well over 90%. This result verified that the proposed model of software ecosystem productivity is applicable to other platforms. It was proved that the model is able to represent the software ecosystem productivity. And it was also found that, in the software ecosystem, the transition from primary productivity to secondary productivity takes time. It took some collaboration with other users to convert primary productivity to secondary productivity. Therefore, the prediction of secondary productivity was not very good in the first and last few months.

7. Discussion and Conclusions

Based on the definition of natural ecosystem productivity, this paper proposed a definition of software ecosystem productivity. Analogous to the primary and secondary productivity of natural ecosystems, this paper decomposed software ecosystem productivity into primary and secondary productivity according to the different impacts of user activities on ecosystems. Also, the productivity was quantified in different software ecosystems. According to the source of productivity, this paper put forward the view that the number, scale, and activity of participants play the most direct and important role in ecosystem productivity. In the process of verifying the relationship between the number and activity of participants and the productivity of software ecosystem, we found that primary productivity is highly correlated with the number of participants, while secondary productivity was less correlated with the number of participants. This paper presented a model of software ecosystem productivity. The validity of the model was verified by experiments.

In this paper, the relevant factors of software ecosystem productivity and the composition model were studied, hoping to provide help for further software ecosystem research, software ecosystem health measurement, development efficiency measurement, and so on. This paper researched the impact of indirect factors such as user reviews and external environmental factors on software ecosystem productivity. However, this paper analyzed the number and activity of users as the factors influencing software ecosystem productivity, which is not comprehensive enough.

Some threats affect the validity of the measurement methods proposed in this paper, including external threats and internal threats. The main external threat is the validity of datasets. In this work, parameters of this model were obtained through data analysis. Therefore, the quality of this model is determined by the quality of the used dataset. In this paper, the verification datasets of Stack Overflow and Bugzilla were dumped from their platforms, and the GitHub datasets were obtained through the data service GitHub API V3. The small ecosystem and little data will lead to inaccurate acquisition of parameters in the model, which will threaten the validity of the conclusion. The internal threat is that the model only considers the number of users and the impact of user activities on productivity. However, there may be other factors affecting the accuracy of the model. Due to the characteristics of the large amount of data used in building the model, the characteristics of some specific projects in the ecosystem may have been ignored. And this model considers the productivity of the software ecosystem from a quantitative rather than qualitative perspective, ignoring the impact of different events on the ecosystem. For example, in the project-level ecosystem, some core developers frequently submit code during the observation period. It will threaten the validity of the conclusion.

In future work, we plan to validate the adaptability and reliability of the model in other open source websites and to analyze and verify other factors affecting software ecosystem production, such as the language of projects and life length of projects. It is clear that language affects the willingness of users to contribute to the ecosystem. Also, we consider the use of neural network to improve the model, especially robust multilayer extreme learning machine [20] and plan to use a visualization method [21] and develop a tool to show the impact of various factors in the ecosystem on its productivity and the evolution of the open source software ecosystem.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported by Ministry of Science and Technology: Key Research and Development Project (2018YFB003800), Hunan Provincial Key Laboratory of Finance and Economics Big Data Science and Technology (Hunan University of Finance and Economics) (2017TP1025), China NSF (61876190 and HNNSF 2018JJ2535), and Fundamental Research Funds for the Central Universities of Central South University (2019zzts150).

References

[1] K. Manikas and K. M. Hansen, “Software ecosystems - a systematic literature review,” *Journal of Systems and Software*, vol. 86, no. 5, pp. 1294–1306, 2013.

[2] K. Ma, H. Kong, M. Guan, and B. Fu, “Ecosystem health assessment: methods and directions,” *Acta Ecologica Sinica*, vol. 21, no. 12, pp. 2106–2116, 2001.

[3] K. Manikas and K. M. Hansen, “Reviewing the health of software ecosystems-a conceptual framework proposal,” in *Proceedings of the Fifth International Workshop on Software*

Ecosystem (IWSECO-2013), vol. 987, pp. 26–37, Potsdam, Germany, May 2013.

[4] S. Jansen, "Measuring the health of open source software ecosystems: beyond the scope of project health," *Information and Software Technology*, vol. 56, no. 11, pp. 1508–1519, 2014.

[5] D. G. Messerschmitt and C. Szyperski, *Software Ecosystem: Understanding an Indispensable Technology and Industry*, MIT Press Books, Cambridge, MA, USA, 2003.

[6] M. Chen, Y. Li, X. Luo, W. Wang, L. Wang, and W. Zhao, "A novel human activity recognition scheme for smart health using multilayer extreme learning machine," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1410–1418, 2019.

[7] M. Iansiti and R. Levien, "Keystones and dominators: framing operating and technology strategy in a business ecosystem," *Harvard Business School Working Paper*, vol. 03-061, 2004.

[8] J. Gamalielsson, B. Lundell, and B. Lings, "Responsiveness as a measure for assessing the health of oss ecosystems," in *Proceedings Of the 2nd International Workshop On Building Sustainable Open Source Communities (OSCOMM 2010)*, IFIP, Notre Dame, IN, USA, June 2010.

[9] S. Amorim, J. D. McGregor, E. S. Almeida, C. Flach, and G. Chavez, "The architect's role in software ecosystems health," in *Proceedings Of the 2nd Workshop On Social, Human, and Economic Aspects Of Software (WASHES'17)*, DBLP, Salvador, Brazil, May 2017.

[10] V. Berk, I. V. Den, S. Jansen, and L. Luinenburg, "Software ecosystems: a software ecosystem strategy assessment model," in *Proceedings Of the 4th Software Architecture, European Conference (Ecsa), Companion Volume DBLP*, DBLP, Copenhagen, Denmark, August, 2010.

[11] O. Franco-Bedoya, D. Ameller, D. Costal, and X. Franch, "QuESo a quality model for open source software ecosystems," in *Proceedings Of the 9th International Conference On Software Engineering and Applications (ICSOFT-EA)*, pp. 209–221, Vienna, Austria, August 2014.

[12] Z. Liao, L. Deng, X. Fan et al., "Empirical research on the evaluation model and method of sustainability of the open source ecosystem," *Symmetry*, vol. 10, no. 12, 2018.

[13] Z. Liao, M. Yi, Y. Wang et al., "Healthy or not: a way to predict ecosystem health in GitHub," *Symmetry*, vol. 11, no. 2, p. 144, 2019.

[14] Z. Liao, B. Zhao, S. Liu et al., "A prediction model of the project life-span in open source software ecosystem," *Mobile Network Application*, vol. 24, no. 4, pp. 1382–1391, 2019.

[15] F. Jing and C. A. Ping, "Implications and estimations of four terrestrial productivity parameters," *Acta Phytoecologica Sinica*, vol. 25, no. 4, pp. 414–419, 2001.

[16] X. Luo, J. Sun, L. Wang et al., "Short-term wind speed forecasting via stacked extreme learning machine with generalized correntropy," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4963–4971, 2018.

[17] J. B. Wang, J. Y. Liu, and Q. Q. Shao, "Spatial-Temporal patterns of net primary productivity for 1988-2004 based on Glopem-Cevsa model in the 'Three-River Headwaters' region of Qinghai province, China," *Chinese Journal of Plant Ecology*, vol. 33, no. 2, pp. 254–269, 2009.

[18] X. Zhang, X. Pan, and M. Lu, "Meshless weighted least-square method," *Acta Mechanica Sinica*, vol. 17, no. 3, pp. 270–282, 2003.

[19] Q. Li, Y. Li, J. Gao et al., "Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation," in *Proceedings Of the 2014 ACM SIGMOD International Conference On Management Of Data*, pp. 1187–1198, Snowbird, UT, USA, June 2014.

[20] X. Luo, Y. Li, W. Wang, X. Ban, J.-H. Wang, and W. Zhao, "A robust multilayer extreme learning machine using kernel risk-sensitive loss criterion," *International Journal of Machine Learning and Cybernetics*, vol. 11, no. 1, pp. 197–216, 2020.

[21] Z. Liao, D. He, Z. Chen, X. Fan, Y. Zhang, and S. Liu, "Exploring the characteristics of issue-related behaviors in GitHub using visualization techniques," *IEEE Access*, vol. 6, pp. 24003–24015, 2018.