IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 20, 2019, accepted June 28, 2019, date of publication July 2, 2019, date of current version July 17, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2926306

Toward the Health Measure for Open Source Software Ecosystem Via Projection Pursuit and Real-Coded Accelerated Genetic

LEI WANG, JING WAN, AND XINSHU GAO

Department of Management Science and Engineering, Nanjing Forestry University, Nanjing 210037, China
NFU Academy of Chinese Ecological Progress and Forestry Studies, Nanjing Forestry University, Nanjing 210037, China

Corresponding author: Lei Wang (leiwangchn@163.com)

This work was supported in part by the Humanity and Social Science Youth Fund of Ministry of Education of China under Grant 18YJCZH170, in part by the Jiangsu Overseas Visiting Scholar Program for University Prominent Young and Middle-aged Teachers and Presidents, Six Talent Peaks Project in Jiangsu Province under Grant RJFW-029 (2019), and in part by the Youth Innovation Fund of Science and Technology of Nanjing Forestry University under Grant CX2016031.

ABSTRACT The benign development of Open-source Software Ecosystem (or OSSE) helps to fuse the wisdom of the community. It can facilitate the development and solve the urgent application needs of large-scale complex software systems. To guarantee that an OSSE is stable and effective for supporting the application development, health assessment for an OSSE has become a research hotspot. In this paper, starting from a new perspective, the OSSE is compared with the ecosystem in the natural world. An OSSE health measure method is proposed by integrating projection pursuit and real-coded accelerated genetic algorithm. First, according to the snowball sampling data collection method and the grounded theory, the data is collected and processed. Second, by designing evaluation indicators and utility functions, the projection pursuit classification model of the natural ecosystem is evaluated and combined with a real-coded accelerated genetic algorithm, thereby designing the health measure model. The experimental results suggest the effectiveness of the proposed approach.

INDEX TERMS Open-source software ecosystem, health measure, projection pursuit model, real-coded accelerated genetic algorithm.

I. INTRODUCTION

To date, open-source has become more and more popular in software technology innovation and crowd-based software development, as its usage, modification, and distribution are not subject to license restrictions [1], [2]. To facilitate the application development on some open source platforms, software ecosystem has attracted much attention, such as the R software ecosystem, Eclipse software ecosystem, Android software ecosystem, etc [3]. In sum, the software ecosystem is a collection of projects developed by a specific community [4], which is a set of software products with some degree of symbiotic relationship [5].

As for open-source software development, an Open-source Software Ecosystem (OSSE) is the interaction of a group of participants on a common open source software technology platform, bringing many software solutions or services. Each participant is motivated by a range of interests or business models and has a symbiotic relationship with other participants and the entire ecosystem. At the same time, the structure of the technology platform also allows different people to participate and contribute [3].

Unlike traditional software projects which focus on progress and budget, an OSSE pays more attention to its own state that is, the state of health which is the ability of the ecosystem to continuously develop and maintain structure and function over time [6]. Therefore, how to guarantee the health of an OSSE arises as a primary concern and attracts significant attention nowadays [7].

The importance of OSSE is irreplaceable in helping to develop complex software or to realize the intellectualization and colocalization of software development [8]. Therefore, this requires OSSEs to be developed benignly [9]. To guarantee the health of OSSEs, health measure is the most basic step [8]. Some researchers have attempted to create their own operationalization, but these typically get stuck in the concept

The associate editor coordinating the review of this manuscript and approving it for publication was Wen-Long Chin.

phase [10], [11]. Therefore, it is still difficult to really measure the health status of an OSSE from its conceptual level to quantitative study. However, there are many other factors that affect the health of open-source software, which makes it a challenging work to measure the health of OSSEs [11], [12].

In a software ecosystem, each user is an independent individual. Each person has a distinguished way of thinking, which results in different coding styles in the same project. Even if submissions come from the same person, he would probably have different ideas of the project as time goes by. What's more, there are not only a lot of artifacts, but also a large number of actors (concerning community administrators, developers, and users) in the OSSE. Therefore, this will introduce a large amount of data being submitted. New ideas are generated and new codes are submitted every day, so the speed of accumulation of codes and some related data will be multiple times faster. When measuring an OSSE's health, we need to extract useful data from a huge dozen. Therefore, it is very difficult to measure the health of an OSSE considering these factors. Faced with the large-scale data, how to deal with these data quickly and effectively is a problem that should be solved. Such a problem will even lead to a computational overload. In summary, measuring the health of an OSSE is a challenging work [7].

Different from the existing software execution quality assurance problems [13], [14], the research on software system health assurance has been going on for a long time, from the initial definition of software systems to the current research on the health management of the OSSE [15], [16]. Kotonya *et al.* consider a software system as a system on intercommunicating components based on the software part of a computer system (a computer system is regarded as a combination of hardware and software). In general, a software system contains a number of separate programs, configuration files (used to set up these programs), system documentation (describes the structure of the system), and user documentations (explains how to use the system) [15]. In recent years, people begin to pay close attention to the health of the software systems. The healthiness of software is defined as *a degree of a healthy software ecosystem, which means that a firm in a healthy software ecosystem can easily reach its financial goal better than other firms in other software ecosystems* [16].

Relevant existing works have studied how to analyze the health of open-source software and software ecosystems. Van Lingen, *et al.* focus on content management systems and their health [17]. In their research, they analyze and compare the ecosystem health status among the three most popular open source content management system platforms (i.e., WordPress, Joomla, and Drupal). They first measure the health characteristics of software ecosystems, then use the HTML analysis mechanism for measurement and analysis. Finally, they complete the health analysis of the OSSE. Manikas *et al.* defined the health status by placing their research focus on the differences between software ecosystems (SECO) and natural ecosystems, which is combined with previous works [7]. S. Jansen explores a health factor based on project level [12]. The work builds the framework of the OSSE health measurement based on three pillars of productivity, robustness and niche creation to analyze. The framework can help to improve ecosystem activities, assess the health of an ecosystem, or identify weaknesses of an ecosystem. Spauwen *et al.* [18] study the Apple's App Store ecosystem based on the works presented in [19], [20], and information that developers' relationships with platforms and other developers over time can be used as an indicator of the robustness of a SECO, which is one of the determinants of a SECO's health. They further investigate the motivation of developers to participate in certain Free and Open Source Software (FOSS) projects from the perspective of a developer's behaviors. The works of [10], [21] investigate project health by looking at factors like developer activity. These works provide some metrics on the project level of the Open-source Ecosystem Health Operationalization (or OSEHO).

In sum, relevant existing works cannot systematically address the challenges of measuring the health of the OSSE [8], [11], [12], [17], [22]. Some works only investigated the importance of the healthy development of the OSSE, but do not propose how to achieve this or how to effectively measure the health of an OSSE [9]. Moreover, these studies do not fully demonstrate whether OSSEs can be developed in a benign manner, and it cannot explain why an OSSE has a relatively good state of health [4]. Especially, researchers prefer to conduct a qualitative study of a feature, and then come to a corresponding conclusion, which often makes its results difficult to be applied universally [12]. Some scholars have conducted quantitative research, but their research is often one-sided [8]. Some are incomplete in the construction of the indicator system, as those indicators cannot fully measure OSSE health [23]. Some works use insufficient data for measurement. For example, the data they needed is very scattered and sparse, which makes it difficult to integrate data together and choose appropriate metrics from data [12], [24]. However, it is necessary and important to measure the health of OSSEs. Whether in building the indicator system or in data processing, we must have a well-round method to measure the health of OSSEs comprehensively and systematically. From this point of view, there is still few quantitative research on the health of the OSSE. Therefore, we need to conduct more research in this area to meet the needs, which is more conducive to the development of the OSSE.

This paper focuses on the quantitative study of the health measurement of the OSSE. Based on the previous OSSE research works, we compare the OSSE with the natural ecosystem and get some insights from natural ecosystem health measurement methods. A novel health measure method for OSSE is proposed in this paper by incorporating projection pursuit model and real-coded accelerated genetic algorithm.

As illustrated in Fig. 1, we design the evaluation indicators and utility function for OSSE health measurement, which

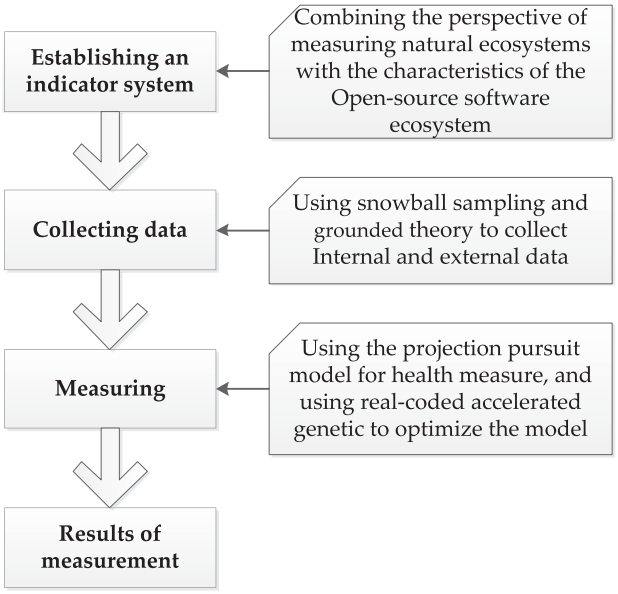

FIGURE 1. Flow chart of technical implementation of OSSE health measure.

is inspired by the natural ecosystem health measure methods. We then adopt the snowball sampling data collection method and the grounded theory to select samples from the fast-accumulating OSSE data space. Finally, we integrate the projection pursuit classification model and real-coded accelerated genetic algorithm to measure the health of OSSEs. It's worth noting that, qualitative problems in this paper are finally addressed by a quantitative way. The indicators we selected are combined with the widely used natural ecosystems health measurement indicators and the specific characteristics of software ecosystem. Experiment results suggest the effectiveness of the proposed approach. Therefore, the approach proposed in this paper can assess the health of an OSSE intuitively and efficiently.

The remainder of this paper is organized as follows. We briefly introduce preliminary knowledge in section II, which mainly includes the snowball sampling method and the grounded theory. In section III, we present how metrics are built for measuring the OSSE health. We describe the acquisition and processing of data in section IV. In section V, we present the method of measuring the health of the OSSE by integrating the projection pursuit classification model and real-coded accelerated genetic algorithm. In section VI, we compare the proposed approach with other methods by simulation experiments. Finally, we conclude by identifying some important future directions in Section VII.

II. PRELIMINARY

In the evaluation of any OSSE's health, the process of data analysis is very important. This section provides a brief overview of the snowball sampling data collection method and grounded theory which are cornerstones of the OSSE health measuring approach proposed in this paper.

A. SNOWBALL SAMPLING

There are two types of snowball sampling methods, one is probability and the other is non-probability [25]. A probability approach is a special form of snowball sampling, namely "*s* stage *k* name snowball sampling" [26], [27]. Coleman [26] defined the snowball sampling method as "*One method of interviewing a man's immediate social environment is to use the sociometric questions in the interview for sampling purposes.*" The non-probability approach collects samples from people who are unlikely to be selected in standard sampling methods in order to study the characteristic ions of individuals in the population [25]. This method collects data through link tracking, and through a layer of links, finally obtains the data that one wants to collect. This paper uses the non-probability approach of snowball sampling.

B. GROUNDED THEORY

The grounded theory is proposed by Glaser *et al.* [28]. It is a qualitative research method that develops and inductively guides the grounded theory according to a phenomenon by a systematic procedure. Researchers generally do not have theoretical assumptions before the start of the study. They start with real observations, summarize the empirical summaries from the original data, and then rise to the theory of the system. This establishes a substantive theory from the bottom up. That is, to find the core concepts reflecting the essence of the phenomenon of things based on systematic collection of data, and then construct relevant social theories through the connection between these concepts [29].

Take OpenStack as an example, we illustrate how data are collected from commercial organizations involved in the open source community. First, after collecting the information, documents and forum discussions on the official website, the snowball sampling method is used to collect articles and some news reports from the project participants' personal social platforms. Although there is no guarantee that the complete knowledge of OpenStack business participation can be obtained, by manually filtering unauthenticated information, the method can successfully collect most of the data which can be used.

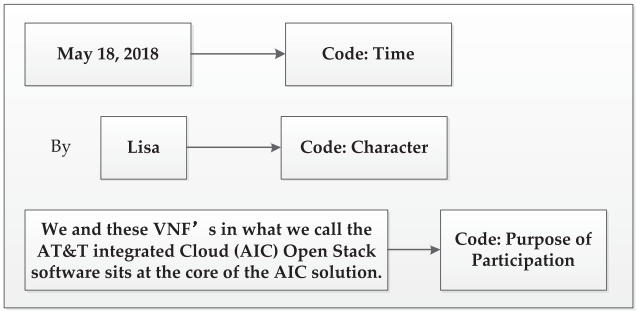

FIGURE 2. The process of encoding materials.

Second, as illustrated in Fig. 2, according to the grounded theory, after simply reading the data and materials collected,

we filter out the data needed and encode the key information. There is no need to entangle the concept or to forcibly find similarities between unclear information when encoding. In the process of encoding, we can browse information, then summarize similar points, assumptions or ideas, then compare it with unencoded or uncharacterized information in subsequent encoding, by which concepts can thus be formed and named. When the encoding is completed, all concepts are abstracted and summarized again, and the information is formed according to existing categories, concepts, and codes.

C. METHOD OF HEALTH MEASUREMENT

Because the software ecosystem resembles the natural ecosystem, there are also some similarities when we analyze their health and selecting evaluation metrics. When health measurement of the natural ecosystem is carried out, such as the health evaluation of agricultural water in the Heihe River Basin, the projection pursuit classification model can be used to establish a specific evaluation indicator system. In this case, following the principle of being practical and quantifiable, the water level first is subdivided by year. Then, real-coded accelerated genetic algorithm is used to calculate the standard value of different indicators, which will then be used for projection. Finally, the model of projection value and evaluation grade is established to measure the health according to the size of grade value.

The health measurement models used for the software ecosystem are built in two dimensions: developers and users. In terms of developers, there are four aspects of measurement metrics based on the history of code submission, which includes: (1) using regression analysis of new external personnel's entry rate to measure attractiveness of the project; (2) using survival analysis of developers contributions to measure continuity of the project; (3) using component analysis of code contribution distribution to measure team diversity; and (4) using regression analysis of codes ownership to measure normativeness of the project. In terms of users, their activities are modeled and the indicator system of hierarchical structure is designed. The software ecosystem is viewed as a whole, and user activities are measured from three aspects: application attractiveness, user loyalty, and user mobility. Specially, the entry volume of new users represents application attractiveness, i.e.

$$NU^{(t)} = \left| N_1^{(t)} \cup N_2^{(t)} \cup N_3^{(t)} \ldots \cup N_n^{(t)} \right|$$

in which N_n represents the number of users who use the n-th software in less than a week. Average online time of users represents user loyalty, i.e.

$$AT = \frac{\sum_{i=1}^{m} T_i}{m}$$

in which T_i represents the length of the software used by the i-th user, m is the number of users. Platform liquidity index at adjacent time points represents user mobility, i.e.

$$UD = \sum_{i=1}^{m} D_i$$

in which D_i represents the amount of users who flow into the i-th software from other software.

III. METRICS FOR OSSE HEALTH MEASUREMENT

In section 3.1, we introduce three different directions of ecological environmental quality assessment, which are safety assessment, risk assessment, and health assessment. In section 3.2, based on the evaluation type mentioned in section 3.1, we propose a new evaluation type which could be applied to OSSE's health, mainly from three facets: commercial quality, product quality, and collaborative quality. In section 3.3, we establish specific indicator evaluation criteria.

A. ECOLOGICAL ENVIRONMENT QUALITY ASSESSMENT

In most cases, the quality of an environmental ecosystem is assessed from safety, risk, and health, which provides a comprehensive result.

1) ECOSYSTEM SAFETY ASSESSMENT

The ecosystem safety assessment mainly evaluates its health and integrity. The result reflects the harm caused by unstable ecosystems that human beings have suffered from in living and producing activities, while healthy ecosystems are stable and sustainable. An unhealthy ecosystem could also cause harm to people living in it both physically and mentally, which is not good for the stability of societies as well. Therefore, the safety assessment, which is holistic, hierarchical and dynamic, is indispensable in measuring an ecosystem's quality.

2) ECOSYSTEM RISK ASSESSMENT

The ecosystem risk assessment mainly demonstrates the possibility of risk in the local environment, especially the damage to ecological environments or other species caused by human activities, which causes adverse reactions or dangerous ecological effects. In the basic theory of ecological risk, the applications of ecology, environmental chemistry, and environmental toxicology are very extensive. In general, quantitative methods are used to find out basic harmful effects on the environment.

In risk assessment, we first need to clearly define objectives and key indicators of the evaluation and make a plan. Then, qualitative and quantitative methods are used to evaluate the environmental risk, identifying relevant indicators for analysis and excluding irrelevant ones that do not cause harm. After a detailed assessment of the hazard, an assessment result will be formed, which uses quantitative indicators to show the extent of the risk.

3) ECOSYSTEM HEALTH ASSESSMENT

As is mentioned in the safety assessment, a healthy ecosystem must be stable and sustainable. Stability means an ecosystem can stabilize species living in it. And sustainability means not only the current development but also the future of the ecosystem could be steadily maintained, with parts damaged being able to be self-repaired and return to a healthy state within a limited period of time. The health assessment can effectively evaluate these two and other auxiliary indicators, and it is also important in the overall quality assessment.

B. SOFTWARE ECOSYSTEM HEALTH ASSESSMENT

As stated in section 3.1, the evaluation of natural ecosystems can be divided into three types: safety, risk, and health. Since different types have different evaluation processes, the results will greatly deviated if only these three indicators are borrowed and embedded in the OSSE health measurement. The choice of parameters and criteria needs to be based on the actual situation of the OSSE.

1) ESTABLISHMENT OF CLASSIFICATION INDICATORS

The essential difference between natural ecosystems and software ecosystems is that one is originally born in nature, and the other is created by human beings and controlled by subjective consciousness. Through the analogy between natural ecosystems and software ecosystems, the key characteristic in software ecosystem quality assessment is sustainability [30]. Sustainability means that users are provided with what they need, the number of users and developers grow steadily in systems, and a certain market share is achieved as well. Therefore, when assessing natural ecosystems, it mainly focuses on the three aspects of safety, risk and health, emphasizing the status of the ecosystem itself and external threats to it; but when evaluating the software ecosystem, we must not only start from these three perspectives, but also add human factors, that is, the potential connection between people. Therefore, when evaluating the software ecosystem, it mainly considers the three aspects of commercial quality, product quality and collaborative quality, which is not only considering the situation of the system itself and the influence of the outside world, but also considering the connection between people in the whole process [31].

The commercial quality can be broken down into four indicators: scale, diversity, market share, and features of a business model. They mainly evaluate a number of products, market share, overall size and characteristics in the software ecosystem. For example, these criteria could assess whether new products meet the needs of customers and the changes of the market, or how to maximize the customer's loyalty to the product which is the customer retention rate, or how to attract more users to expand the market share.

The product quality mainly focuses on evaluating various solutions in the software ecosystem. In addition to performance, safety, simplicity, and reliability, users also need to evaluate both the assembly and openness of the product at the same time. To get scores for evaluation, researchers usually conduct a survey and let users score on these six indicators to obtain indicator coefficients.

The collaborative quality concentrates on the degree of collaboration of different development communities in an OSSE. For example, the flexibility of collaboration between development communities, maturity of software development processes, support the situation of democratic decision-making, feedback cycles of user problems, and the degree to which the solution is certified and released.

2) CLASSIFICATION RESULTS

The sustainability of the software ecosystem is reflected in the degree of organizational cooperation, market share and the degree of user satisfaction, which evaluate the sustainability of the software ecosystem from different aspects. According to the classification indicators in 3.2.1, it can evaluate the software ecosystem from three directions of commercial quality, product quality, and collaborative quality. Fig. 3 further breaks down these three criteria.

C. INDICATOR EVALUATION SYSTEM

According to the commercial quality, product quality, and collaborative quality, the health status of an OSSE can, therefore, be measured. Since some abstract data is difficult to obtain, it is necessary to extract data by methods in section 2 or by feedback from questionnaires. Therefore, this paper will adopt the semi-quantitative selection indicator principle, take the health of the OSSE as the target layer, and take the three aspects of sustainable development as the criteria. As shown in Table 1, a total of 14 basic indicators are selected to establish a health evaluation system of the OSSE. The indicators we selected are positive, i.e. the bigger the better, except feedback cycle. The feedback cycle represents the time for developers to feedback the problem. The shorter the time, the more developers are actively maintaining the stability of system, and the healthier the system.

As can be seen from Table 1, we build a comprehensive evaluation framework for the OSSE. It should be noted that, in addition to the basic information from developers, users should also be investigated for some specific data which is introduced in section 4. In section 5, we will introduce the practical application method of the specific projection pursuit model and specific cases of real-coded accelerated genetic algorithm in detail.

IV. DATA ACQUISITION AND PROCESSING

In this section, we introduce how to acquire and process the OSSE data to measure its health status from two aspects, namely internally and externally. In section 4.1, how internal data can be collected and processed is presented. In section 4.2, we introduce how external data is collected and processed. These two approaches are applied to different data sources and data types, and they do not interfere with or complement each other. Only when these two separate approaches of data collection are completed, can the health evaluation be carried out. It should be noted that people

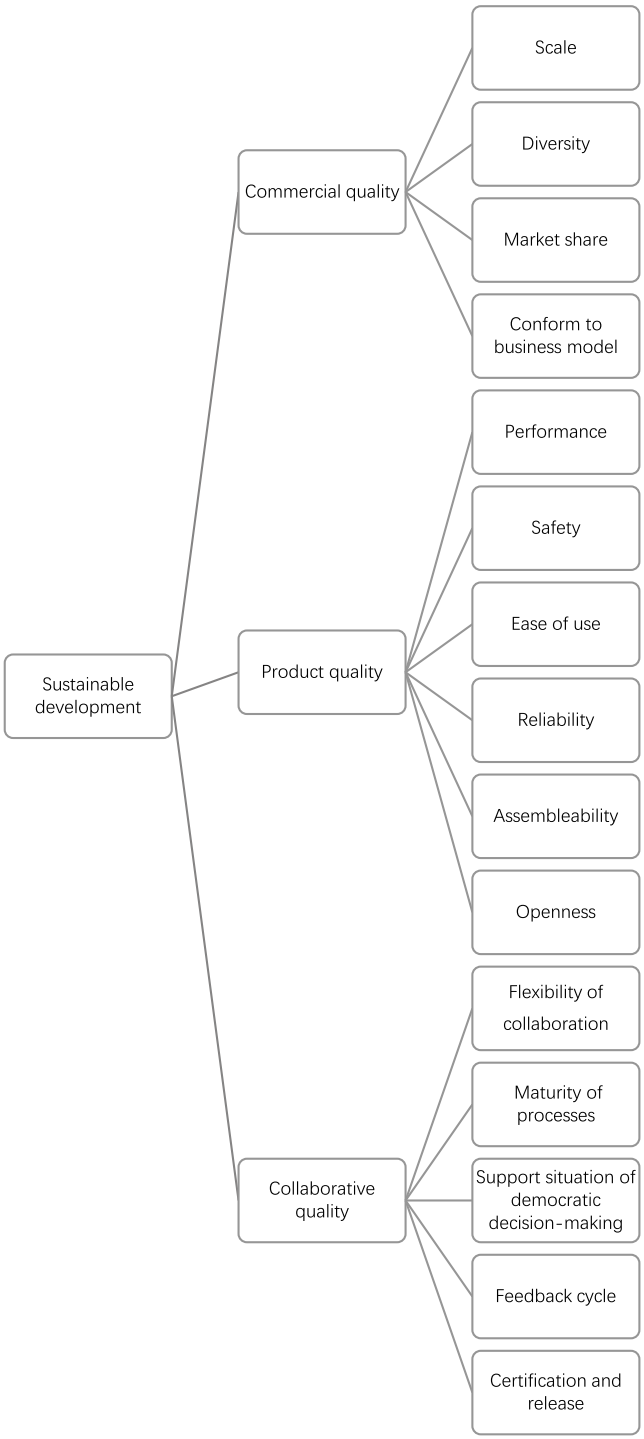

FIGURE 3. Classification results of software ecosystem sustainability.

involved are not limited to only one identifier, i.e., either users or developers, as they could have both roles when they participate in the open-source community and do something. Therefore, we must establish a mapping schema between the participant and the identifier to distinguish.

A. ACQUIRE AND PROCESS INTERNAL DATA

In open-source software platforms, every software has revisions and improvements over time. There store rich data of historical development process, which provides research materials for this paper. In open-source communities, users and developers are two basic player groups and also primary providers of data. For example, developers participate in the OSSE by submitting codes in version control systems, and users may report problems to problem tracking systems. They can also communicate and interact with each other via email.

Historical data in the development is generated by their participation and activities, which are divided into three categories: the history of code submissions in the version control system, record of problem reports in the problem tracking system, and participant information in the mailing list. These data are usually stored in different databases. Therefore, the acquisition and processing of internal data can be implemented in four steps, with the flow chart shown in Fig. 4.

The first step is grabbing the information about development communities. The original data can be obtained from different software databases, such as SVN and CVS. This step mainly gets data from the version control system, the open mail archive, and the problem tracking system. It should be noted that there is no need to extract and filter data at this step, and the data obtained at this step is called Step 1 data.

The second step is extracting and filtering the data. After the raw data is collected at the first step, there will be a large amount requiring further processing, in which we mainly take three actions: first, extracting code submission history of the version control system to get the content and number of submissions; second, decompressing and extracting contents of the mail archive; third, regularizing the data in problem reports to extract problem feedback cycle and handle irrelevant information. The data obtained at this step is called Step 2 data.

The third step is integrating the data. Because systems have different task focuses on different development processes, formats of the recorded information are thus different. As the same participant may have different identities in different development process systems, the identification cannot be directly used as the statistical unit when we examine the project development status. Instead, the mappings between individual participants and identities should be established, and the measurement should be conducted with the individual participants as the unit, so as to reduce the error of the measurement results. Therefore, we need to integrate data source information from different information sources, test participants with multiple identities (login), and infer participants' identities (such as internal employees or external contributors) on this. The data processed before was obtained from the historical information of the software. However, this information is not comprehensive, such as collaboration flexibility in the indicators, which we cannot accurately obtain from the data above. Therefore, we need to conduct a questionnaire survey among developers to obtain their specific working content in the development process. The data obtained at this stage is called Step 3 data.

The fourth step is obtaining measurable data. So far, we have integrated participants and identifiers in the

TABLE 1. OSSE health evaluation system.

Target layer	Criteria layer	Criterion sub-layer	Description	Source	Type
Open Source Software Ecosystem Health Indicator	Commercial Quality	Scale	Number of registered users (unit: ten thousand)	System inquiry	Positive
		Diversity	Number of open source code categories	System inquiry	Positive
		Market share	Number of users of this system / total number of users of open source software	System inquiry	Positive
		Active users	Number of users who log in at least once a week	System inquiry	Positive
	Product Quality	Open source code usage	Percentage of code provided by developers being applied to the software=Code successfully applied/Total number of submissions	Developer survey	Positive
		Stability	Ability to resist viruses, i.e., whether be attacked by a virus in using	Developer survey and user survey	Positive
		Simplicity	How easy it is for users to use the system	User survey	Positive
		Openness	Proportion of subsystems in the system that are solved by third parties	Developer survey and user survey	Positive
		Software assembly	To meet the individual needs of users, how much software is open to the user to freely combine and how satisfied users are	User survey	Positive
	Collaborative Quality	Collaboration flexibility	Degree of cooperation between different development groups	Developer survey	Positive
		Process maturity	Satisfaction of collaboration	Developer survey	Positive
		Democratic decision making	In the development process, when solving problems, take the level of brainstorming	User survey	Positive
		Feedback cycle	Time period after a problem occurs	Developer survey	Negative
		Certification and issuance	User's comprehensive evaluation of software solutions released by developers	User survey	Positive

original data. Then, we remap the identifiers to the original data to get a record containing developer and user information, which provides a basis for measuring the activity of the participants. In this process, we get the system inquiry data and some information related to the survey from the historical events submitted by developers, obtain the information about developer survey and user survey from the record information for user problem report, and obtain data related to the indicators from the developer's questionnaire. The data obtained at this step is called Step 4 data.

We organize information obtained by different indicators to obtain comprehensive data about health indicators. We can get the data of scale, diversity, market share, active users, open source code usage, stability, openness, collaboration flexibility, process maturity, democratic decision making, feedback cycle, which provides the basis for measuring the health of OSSEs.

B. ACQUIRE AND PROCESS EXTERNAL DATA

In this section, we introduce methods of acquiring and processing external data. We start with three steps of data collection and then show data coding and analysis, and data induction.

1) DATA COLLECTION

This paper will use data collected on the Internet which are related to the health of OSSEs. Then through a random survey of developers and users, other data are obtained and combined with data from the Internet. In this paper, we can gather information from the Internet through the following steps:

First, we go to their official websites to briefly browse the information of the last six months and then extract some useful information, such as historical information of projects.

Second, we adopt the snowball sampling mechanism to collect scattered text data about open-source projects. Main steps are as follows: (1) searching for the name of the open-source community investigated in a search engine, and collecting the first retrieved web pages; (2) viewing the results of the first 100 web pages links in the collected web package. If we find any documents on these pages, we will need to save and check it again in the document which contains links and Uniform Resource Locators (URLs); (3) repeating document searching for new links and URLs, and then cycling many times. The collection is complete when there are no new documents and links.

Third, all the documents collected in the previous step are subjected to secondary filtering to manually remove

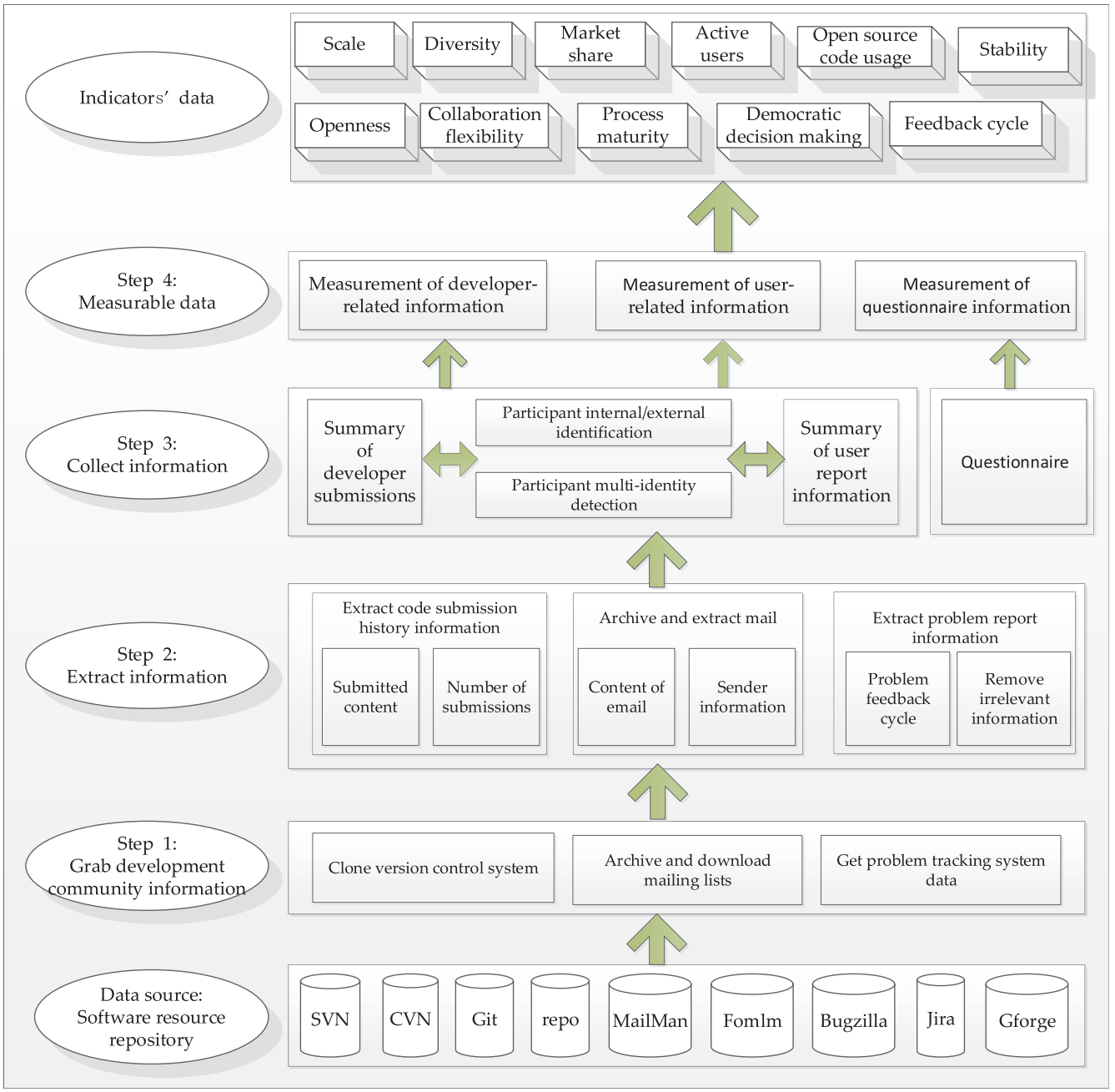

FIGURE 4. **Internal data and processing phase flow chart.**

information that has a high repetition rate and is not related to keywords. For example, if there is only one specific type of keyword or technology and then there is no information related to the indicators, it needs to be removed.

2) DATA CODING AND ANALYSIS

In this paper, data analysis is based on the classical grounded theory. When collecting data, we can analyze simultaneously. For example, it is natural for us to carry out simple data analysis when browsing, which are complementary. In this section, we mainly introduce steps of data coding and analysis.

The first step is to browse the materials collected and find out the information related to the health metrics of the OSSE. When there is any uncertain information, we can put it in another data package and wait for subsequent filtering. For example, if we find the information regarding user evaluation, it can be quantified in the following steps.

The second step is to encode the information. We can identify keywords in the data collected, and different keywords can be annotated with different codes. After simply reading data and materials collected, data needed is filtered out. When encoding information at key information being studied, there

is no need to forcibly find similarities for information with unclear commonality. By browsing the information in the process of coding and summarizing the similarities or the same assumptions or ideas, we can compare the un-coded or un-characterized information with coded and characterized information and compare the similarities to form and name the preliminary concept. Although the existence of mixed theory requires a variety of information encoding, some data is a single indicator, which can still facilitate the research on measuring health state. After simple coding, all integrated concepts are integrated and summarized again, and the concept of second coding integration can be put into theory.

The third step is to form a concept. At the end of the second step, the data has been sorted into different theories and concepts, but some data information has been left out or temporarily put in the data package without being qualitative. Because the data of different indicators can be compared with each other, it is necessary for these data to compare them with the summarized concept. For data of the same concept or basically similar, their coding can be reintegrated into a topic which is the data related to health indicators. This process uses the continuous comparison method in the grounded theory, and the data coding obtained by this method is more accurate and in order.

The fourth step is to classify the code. After data code is completed, the concept and coding of the third step of integration are checked. All the data and theories that are coded and conceptualized are re-integrated into the theory, that is, the understanding framework of the mixed pattern.

In addition, the data in the measurement system should be collected as much as possible, and we should expand the collection process and collect it through multiple channels. In this way, the obtained data information has the characteristics of universality and diversity.

3) DATA INDUCTION

After using snowball sampling data collection mechanism to collect data and encoding and analyzing the data based on the classical grounded theory, it is necessary to comprehensively sum up the data once or more time. First, the data of the same type is packaged and summarized. A simple and comprehensive scanning process is performed in each of the different types of data to reintegrate the data of similar concepts. In secondary integration, we can set a similarity ratio to merge or filter similar concepts. After the secondary integration of multiple adjacent packets is completed, the data is restored from the packet mode to the common coding mode. At this point, data induction has completed the vast majority. Finally, we have to proofread the summarized data to prevent errors in packaging or secondary integration leading to inductive errors.

In the introduction above, we can obtain information related to measuring OSSE health indicators on the Internet. However, since the users' evaluation of the software at different times will be different, the information obtained above may be one-sided. Therefore, we need to conduct a questionnaire survey among the users to obtain a comprehensive evaluation of the software. Then we integrate the information collected earlier with the information obtained from the survey to get more accurate data.

V. OSSE HEALTH MEASUREMENT

We start to measure the health of an OSSE after data processing is completed. In section 5.1, we introduce the projection pursuit model. In section 5.2, we introduce the application of real-coded accelerated genetic algorithm (RAGA) in modeling to optimize the projection index function.

A. PROJECTION PURSUIT MODEL FOR EVALUATION

The projection pursuit model is generally applied to the health evaluation of a natural ecosystem. In this paper, we compare an OSSE with a natural ecosystem and perform combinations of some indicators both quantitatively and qualitatively, then the model is applied to measure an OSSE's health status. The essence of the projection pursuit model is to project high-dimensional data into low-dimensional subspace by some transformation. It first uses the projection index function to measure the projection exposure probability of a certain structure. Then it finds out projection values that the projection index function is optimized, which can reflect high-dimensional data structures or features. Finally, it analyzes structural features of the high dimensional data by the projection value. The construction and optimization of the projection index function is key to the success of the application. When solving high-dimensional data problems, traditional methods are computationally intensive, which limits the in-depth research and wide application to some extent [32]. Therefore, we use RAGA to deal with this problem and propose a novel optimization model. Modeling steps are as follows:

Step 1: Pre-processing the high-dimensional collected data to determine the input.

We evaluate the health of OSSEs by setting a set of indicators, i.e., $\{x*(i,j)|i=1,2,\cdots n; j=1,2,\cdots p\}$, where $x*(i,j)$ is j-th index value of i-th OSSE, and n and p are the OSSE size and number of evaluation indicators of health status, respectively. Because different data have different meanings, that is, some values are the bigger the better, while others are the smaller the better, we normalize these data, which can eliminate dimensions of each index value and unify variation ranges of each index value [33]–[35].

For the indicators that bigger is better:

$$x(i,j)=\frac{x*(i,j)-x_{\min}(j)}{x_{max}(j)-x_{min}(j)}. \tag{1}$$

For the indicators that smaller is better:

$$x(i,j)=\frac{x_{\max}(j)-x*(i,j)}{x_{max}(j)-x_{min}(j)}. \tag{2}$$

In the formula (1), (2), $x_{max}(j)$ and $x_{min}(j)$ are maximum and minimum values of j-th index, respectively. $x(i,j)$ is the normalized sequence of index eigenvalues.

Step 2: Constructing a projection indicator function $Q(a)$ to measure the health status of OSSEs [36].

The projection pursuit method is to integrate the p-dimensional data $\{x*(i,j)|i=1,2,\cdots n; j=1,2,\cdots p\}$ into the one-dimensional projection value $z(i)$ with $a = \{a(1), a(2), \cdots a(p)\}$ as the projection direction [37], i.e.,

$$z(i) = \sum_{j=1}^{p} a(j)x(i,j), \tag{3}$$

in which a is a unit length vector.

Then, we sort the one-dimensional projection value according to the value of $z(i)$. When we select the projection value, the scattering characteristics of the projection value $z(i)$ should be as follows to rationalize the projection indicator function $Q(a)$: the overall projection points should spread as much as possible and the local projection points should be as dense as possible; then aggregating into several clusters will achieve the best result. Therefore, the projection indicator function can be formulated as:

$$Q(a) = S_Z \times D_Z, \tag{4}$$

in which S_z is the standard deviation of the projection value $z(i)$ [38]; D_z is the local density of the projection value $z(i)$, i.e.:

$$S_z = \frac{\sqrt{\sum_{i=1}^{n} [z(i) - E(z)]^2}}{n-1}, \tag{5}$$

$$D_z = \sum_{i=1}^{n}\sum_{j=1}^{n} [R - r(i,j)] \times u[R - r(i,j)], \tag{6}$$

in which $E(z)$ is the average of the sequence $\{z(i)\ |i = 1, 2, \cdots n\}$; R is the window density of the local density. In order to avoid the deviation of sliding average being too large and to increase too much with the value of n, R can be determined according to experience. In the actual operation, we can take $0.1S_z$ as R. $r(i,j)$ is the distance between $z(i)$ and $z(j)$; $u(t)$ is the unit step function that equals 1 if $t \geq 0$, and 0 otherwise.

Step 3: Optimizing the projection indicator function.

When given a collected data set for each indicator value, the projection index function $Q(a)$ changes only as the projection direction a changes. It is possible to estimate the optimal projection direction by solving the maximum value of the projection index function, which is the highest possible to exposure feature structure of high dimensional data, i.e.,

$$\max [Q(a)] = S_z \times D_z, \tag{7}$$

$$s.t. \sum_{j=1}^{p} a^2(j) = 1 \quad (a(j) \in [0,1]). \tag{8}$$

This is a complicated nonlinear optimization problem [39], in which the variable $a(j)$ is used as the optimized variable. According to the definition of the function of u and $r(i,j)$, the objective function $Q(a)$ is discontinuous or indivisible at some points. The conventional optimization methods are difficult to deal with this problem. However, the real-coded accelerated genetic algorithm (RAGA), which simulates the survival of the fittest and the intra-group chromosome information exchange mechanism, is a general global optimization method [40]. It can enhance the stability and accuracy of calculation results by multiple substitution and superposition operations. At the end of step 3, we need to use RAGA to iterate the data, as detailed in the next section.

Step 4: Selecting an excellent scheme and establishing a clustering model [41]. Substituting the optimal projection direction a obtained in step 3 into formula (3), the projection value $z(i)$ of each scheme is obtained.

The right endpoint value is taken for each evaluation standard interval of each evaluation index, and then the evaluation standard sample is generated and performed normalization processing. The RAGA-PP model is used to integrated projection to obtain the projection value of the standard sample. We set the lowest level of evaluation to 1 and set the highest level to N to obtain the scatter diagram of the projection value of the standard sample. According to each status to divide its corresponding projection value $z*(i)$, we establish a projection pursuit level evaluation model $y = f(z)$. We can sort the ranking values from large to small, which means the larger the ranking value, the better the solution.

Because of some indicators data collected by user surveys, we cannot collect quantitative indicators data, and then it cannot be directly substituted into the calculation in actual data collection. When the questionnaire or evaluation feedback from users, we can select dummy variables instead of detailed variables. In this paper, we choose a number from 0 to 9 to replace the specific data of users' survey, so that we can better to deal with data.

B. APPLICATION OF RAGA IN MODELING

By referring to the literature [42]–[44], we find that real-coded accelerated genetic algorithm has a strong practicability. This paper uses real-coded accelerated genetic algorithm to optimize the projection pursuit model, which can be divided into eight steps. For example, we want to use it to solve Max: $f(X)$.

Step 1: Linear transformation. Introducing a random number with $y \in [0, 1]$, called the uniform mutation operator, which helps to jump out of the local optimal solution in the later stage of the evolution process.

Step 2: Random generation of initial parent groups. Generating N groups of random number in range [0, 1].

Step 3: Evaluation of the adaptive ability of the parents individual. According to step 2, the corresponding fitness function $f(x)$ is obtained, and the smaller the value is, the stronger the individual adaptability is, and the better individual.

Step 4: The probability of selection of the parent individual. Selection is the key to genetic algorithm, which reflects the idea of survival of the fittest.

TABLE 2. The first 6 sets of simulation data.

Indications	Sca	Div	MS	AU	OSCU	Sta	Sim	Ope	SA	CF	PM	DEM	FC	CI
1	10	698	0.2	7.8	5	7	5	0.6	5	6	6	6	3	6
2	8.5	573	0.17	6	6	8	6	0.8	5	7	6	8	5	7
3	11.5	822	0.23	8.1	6	7	6	0.7	6	5	6	6	1	6
4	5	287	0.1	2.3	5	6	3	0.4	3	3	4	5	7	5
5	6	432	0.12	2.9	7	7	7	0.6	7	7	8	7	7	7
6	9	765	0.18	7.1	7	6	7	0.7	7	8	7	6	3	7

Step 5: Hybridization of parent individual. The new population selected in step 4 is hybridized according to the probability of crossover probability.

Step 6: Variation of offspring individual. The new population generated in step 5 is mutated according to the probability of mutation, resulting in a new generation of population.

Step 7: Evolutionary iteration. Algorithms are transferred to step 3 into the next evolutionary process, re-running: evaluation, selection, hybridization, and mutation. This repeated evolution increases the individual's ability to adapt until the optimal individual's optimization criterion value is less than a specified value or the optimal individual's optimization criterion value no longer increases. Then the evolutionary iteration is terminated and the algorithm ends.

Step 8: Speeding up the cycle. The above seven steps constitute Standard Genetic Arithmetic (SGA). Re-running the SGA algorithm by taking the variable change interval of the excellent individual generated by the first and second iterations as the new initial change interval of the variable. In order to accelerate the cycle, the change interval of the excellent individual will gradually adjust and shrink. The distance from the best advantage will be closer and closer until the optimal individual optimization criterion function value is less than a certain set value or the algorithm reaches a predetermined number (periods) of accelerations. Then the entire algorithm will stop running. At this time, the average of the best individual or excellent individual in the current group is designated as the result of RAGA. The RAGA method can be used to gradually adjust and compress the search space of the algorithm and control the parameters.

VI. EXPERIMENTS AND DISCUSSION

In this section, we compare the proposed methods and indicators with other methods. In section 6.1, we verify the method used in this paper through an experiment. In section 6.2, we give some discussions about the method proposed in this paper.

A. EXPERIMENTAL EVALUATION

Measurement of the health of OSSE not only benefits the development of the software itself but also facilitates investment decisions by external investors. There are many popular software systems in the world, such as Google, Linux, OpenStack, Docker and so on. To this end, we select six software ecosystems as targets and combine the above indicators and methods to measure the health of the six systems to determine the pros and cons. Table 2 shows the basic indicator values for the six systems, where MS is market share, AU is active users, OSCU is open source code usage, SA is software assembly, CF is collaboration flexibility, PM is process maturity, DEM is democratic decision making, FC is feedback cycle, CI is certification and issuance.

We select that the initial population size of father (n) is 400, the crossover probability (Pc) is 0.80, the mutation probability (Pm) is 0.80, the number of excellent individuals is 20, and the number of accelerations is 7. In this paper, to determine the best Pc and Pm, these values were adaptively adjusted by using fitness function. When they equal 0.8, the algorithm convergence faster and the OSSE measure results appear more effective. Optimal projection direction for indicators variable are (0.3215, 0.6577, 0.3147, 0.2002, 0.1997, 0.2223, 0.1333, 0.2226, 0.2455, 0.1525, 0.0887, 0.1175, 0.1907, 0.1637) respectively.

Bring the best projection direction into formula (3), we can get the projection values of each system, which are (1.9167, 2.1640, 2.3824, 0.1907, 1.7412, 2.3756). According to the order of magnitude of projection values, we know System 3 > System 6 > System 2 > System 1 > System 5 > System 4.

We then conduct a questionnaire survey for the six systems to know experts' evaluation of the health of these systems. The results show that System 3 is the healthiest, accounting for 0.53; the next is system 6, accounting for 0.37; and the last one is system 2, accounting for 0.10.

It is generally believed that System 3 is the healthiest, which consistent with the results we calculated. In the results of the survey, we find that experts consider System 6 to be the healthiest accounting for 0.37. It is not difficult to find that the index values of System 3 and System 6 are excellent, and projection values of two systems in our calculated projection values are also very close. This shows that these are two systems with similar health, so it is difficult to directly identify which is the healthiest. However, the method used in this paper can be discerned and the result consistent with the opinions of most experts.

We sample again at different times and get data in Table 3. Optimal projection direction for indicators variable are (0.1891, 0.6638, 0.3420, 0.2474, 0.0872, 0.2402, 0.2589, 0.0711, 0.2185, 0.1306, 0.1584, 0.1772, 0.2426, 0.1669) respectively. Projection values of each system are (1.8361, 2.1052, 2.4275, 0.2426, 1.7970, 2.2814). According to the order of magnitude of projection values, the health order of

TABLE 3. The second 6 sets of simulation data.

Indications	Sca	Div	MS	AU	OSCU	Sta	Sim	Ope	SA	CF	PM	DEM	FC	CI
1	10	700	0.19	7.9	5	7	5	0.6	5	6	6	6	3	6
2	9	589	0.17	6.3	6	8	6	0.8	5	7	6	8	5	7
3	11.5	834	0.24	8.1	6	7	6	0.7	6	6	7	6	1	7
4	6	324	0.1	2.5	5	6	3	0.4	3	4	4	5	8	5
5	6.5	501	0.12	3.2	7	7	7	0.6	7	7	8	7	7	7
6	9	798	0.18	7.1	7	6	7	0.7	7	8	7	6	3	7

the six systems is System 3 > System 6 > System 2 > System 1 > System 5 > System 4, which is consistent with the first result.

We apply the method used in this paper to the literature [45]. The optimal projection directions of 10 indicators selected in literature [45] are (0.2316, 0.1706, 0.2780, 0.2631, 0.2839, 0.7314, 0.2330, 0.1540, 0.2157, 0.1750). Hence, projection values of the eight systems are (0.4974, 0.2614, 0.2096, 0.1890, 1.0967, 2.6568, 0.0731, 0.3487). Thus, we can sort the health of these 8 systems, i.e., Heroku > Google App Engine > Azure > OpenShift > CloudFoundry > dotCloud > Engine Yard > Nodejitsu, which is basically consistent with the results of the analysis by Garm Lucassen *et al.* Their results are to be obtained through analyzing, and systems that are similar in health cannot accurately judge the pros and cons, such as OpenShift, CloudFoundry. However, the method proposed in this paper can directly sort the magnitude of projection values, so that the most health system can be judged intuitively, even for a system with similar health.

B. DISCUSSION

By the experiment and comparison, we can find the essential differences between the method proposed in this paper and some other methods.

First, we establish more indicators in this paper. Many other methods used Iansiti *et al.* [23] as the reference standard to set indicators from the three aspects of productivity, robustness and niche creation. Different from most works, Z Liao, *et al.* have compared natural ecosystem to establish four aspects of openness, stability, activity and extensibility to study the health of Stack Overflow [46]. They determine indicators' weight by improving the entropy method to measure health [46], [47]. Although indicators can be dimensioned using the devaluation method, the entropy method cannot determine the influential direction of indicators on the target when determining weight (i.e. positive correlation or negative correlation) [46]. Therefore, there will be certain requirements on selection of indicators. The establishment of indicators in this paper is also compared with natural ecosystem. However, this paper not only considers indicators of M. Iansiti and natural ecosystem, but it has also increased the synergy between commercial quality and product quality and considered positive and negative indicators, which makes the indicators more comprehensive.

Second, we propose a method for collecting and analyzing data. In the relevant existing works, data collection and analysis methodologies for OSSE health measure were seldom investigated [12]. We draw on the method in this paper of measuring the health of natural ecosystem, using snowball sampling to collect data, and using the grounded theory for analysis, and finally can handle the data well.

Finally, we present a method for measuring the health of the OSSE. The projection pursuit model is used in this paper to measure the health of the OSSE. The real-coded accelerated genetic algorithm is adopted to solve the optimization problem in the measuring process. It is possible to rank the healthy status of OSSEs based on the proposed method. It's intrinsically different from the previous method, that only a qualitative analysis approach for the health of open source software was proposed. So far, only a small number of works have made quantitative research on health of OSSEs [12], [17]. In this paper, we adopt a certain method in calculation of indicators' weight and health measurement, which makes the results more objective and accurate.

The experimental results show that the proposed method has obvious operability, and the results needed are more clearly seen. Whether it is used for a single software or a software ecosystem, the health of these can be measured by the method of this paper. Among other methods, we can see that in the study by van Lingen *et al.* [17], the health of the three software systems they finally evaluated was a subjective evaluation. Using the method of this paper, it is concluded that System 3 is more healthy, more objective, and straightforward. In order to verify the accuracy of results, we asked experts to evaluate the health of six systems, and their results are consistent with our results. And in order to test the operability of the method, we also tested eight systems in literature [45] and found that results in the two papers are consistent, but results of this paper are more objective. It can be seen that the method proposed in this paper has more advantageous.

The method proposed in this paper can be used well in both the collection and analysis of the previous data and the health of the OSSE in the later stage. In the stage of data collection and analysis, this paper collects data from both internal and external perspectives, collects data through the non-probability principle of snowball sampling, which is collecting data by linking tracking method, and finally obtains the desired data through a layer of links. And using the grounded theory to analyze the data, so that valuable information can be obtained. In the measurement of the health of the OSSE, this paper combines the projection pursuit model with real-coded accelerated genetic algorithm. The combination of

the two can not only be used to measure the health of the OSSE, but also can be used for the health of other systems, and has a certain reference significance.

As for an OSSE, it is composed of many parts, and there are many participants involved. It concentrates on the wisdom of the group, but this often makes it difficult to measure the health of the OSSE [2], [48]. However, in the data collection and analysis mentioned in this paper, the combination of snowball sampling and grounding theory, no matter how many participants participate, no matter how many components, we can find and collect this information by creating links. More indicators can be easily included later in our OSSE health measure method. We can also apply these methods to data collection and analysis in other industries or fields to get valuable information. In this paper, when measuring the health of OSSE, the projection pursuit model is combined with real-coded accelerated genetic algorithm. The indicators proposed in this paper are represented by specific numerical values so that the most healthy set of data can be calculated. Even in more complex and larger software systems, we can set certain indicators and use specific numerical values to measure their health and even other characteristics.

VII. CONCLUSION AND FUTURE WORK

This paper compares the open-source software ecosystem with the natural ecosystem, as well as their health evaluation methods and metrics. Based on the snowball sampling method and the grounded theory, we integrate the projection pursuit model and real-coded accelerated genetic algorithm for the OSSE health measure, which are widely used in the comprehensive evaluation of the ecological environment. Our study shows that the health of an OSSE can be measured by semi-quantitative methods, which can help developers to conduct a self-health evaluation of an OSSE by collecting user feedback and system parameters. In this paper, the key is the projection pursuit model which projects the high-dimensional data into a low-dimensional subspace. With evaluation indicators selected, this method finds a projection that reflects the structure and features of the original high-dimensional data.

In this paper, we bring a general framework for health evaluation. After a detailed discussion of data collection and processing methods, we prove that the scattered data can be collected through the snowball sampling mechanism, and the grounded theory is used for data filtering and coding. These lay the foundation for calculation of the evaluation model. As for the real-coded accelerated genetic algorithm, the formulation of crossover probability and mutation probability is subjective. Although the fitness function is used to adjust the values of Pm and Pc, the determination of Pm and Pc still has subjective factors. The collected samples are accidental, so the number of samples should be large enough, otherwise it is difficult to prepare to calculate the health of the OSSEs. In future research, we will collect data from the system on a larger scale. We will also explore the essence of OSSEs from more perspectives than developers and users to build enhanced models and obtain more reasonable evaluation mechanisms.

REFERENCES

[1] W.-T. Tsai, W. Wu, and M. N. Huhns, "Cloud-based software crowdsourcing," *IEEE Internet Comput.*, vol. 18, no. 3, pp. 78–83, May/Jun. 2014.
[2] F. R. A. Neto and C. A. S. Santos, "Understanding crowdsourcing projects: A systematic review of tendencies, workflow, and quality management," *Inf. Process. Manage.*, vol. 54, no. 4, pp. 490–506, 2018.
[3] K. Manikas and K. M. Hansen, "Software ecosystems—A systematic literature review," *J. Syst. Softw.*, vol. 86, no. 5, pp. 1294–1306, May 2013.
[4] T. Mens, M. Claes, and P. Grosjean, "ECOS: Ecological studies of open source software ecosystems," in *Proc. IEEE Conf. Softw. Maintenance, Reeng., Reverse Eng. (CSMR-WCRE)*, Feb. 2014, pp. 403–406.
[5] D. G. Messerschmitt and C. Szyperski, *Software Ecosystem: Understanding an Indispensable Technology and Industry*, vol. 1. Cambridge, MA, USA: MIT Press, 2005.
[6] R. Costanza, "Toward an operational definition of ecosystem health," in *Ecosystem Health: New Goals For Environmental Management*. Washington DC, USA: Island Press, 1992, pp. 239–256.
[7] K. Manikas and K. M. Hansen, "Reviewing the health of software ecosystems—A conceptual framework proposal," in *Proc. 5th Int. Workshop Softw. Ecosyst. (IWSECO)*, 2013, pp. 33–44.
[8] O. Franco-Bedoya, D. Ameller, D. Costal, and X. Franch, "Open source software ecosystems: A systematic mapping," *Inf. Softw. Technol.*, vol. 91, pp. 160–185, Nov. 2017.
[9] S. Hyrynsalmi, J. Ruohonen, and M. Seppänen, "Healthy until otherwise proven: Some proposals for renewing research of software ecosystem health," in *Proc. IEEE/ACM 1st Int. Workshop Softw. Health (SoHeal)*, May/Jun. 2018, pp. 18–24.
[10] K. Crowston, J. Howison, and H. Annabi, "Information systems success in free and open source software development: Theory and measures," *Softw. Process, Improvement Pract.*, vol. 11, no. 2, pp. 123–148, Mar. 2006.
[11] S. da Silva Amorim, F. S. S. Neto, J. D. McGregor, E. S. de Almeida, and C. G. von Flach Chavez, "How has the health of software ecosystems been evaluated?: A systematic review," in *Proc. 31st Brazilian Symp. Softw. Eng.*, Sep. 2017, pp. 14–23.
[12] S. Jansen, "Measuring the health of open source software ecosystems: Beyond the scope of project health," *Inf. Softw. Technol.*, vol. 56, no. 11, pp. 1508–1519, Nov. 2014.
[13] H. Wang, L. Wang, Q. Yu, Z. Zheng, and Z. Yang, "A proactive approach based on online reliability prediction for adaptation of service-oriented systems," *J. Parallel Distrib. Comput.*, vol. 114, pp. 70–84, Apr. 2018.
[14] H. Wang, L. Wang, Q. Yu, Z. Zheng, A. Bouguettaya, and M. R. Lyu "Online reliability prediction via motifs-based dynamic Bayesian networks for service-oriented systems," *IEEE Trans. Softw. Eng.*, vol. 43, no. 6, pp. 556–579, Jun. 2017.
[15] G. Kotonya and I. Sommerville, *Requirements Engineering: Processes and Techniques*. Hoboken, NJ, USA: Wiley, 1998.
[16] M. Iansiti and R. Levien, *The Keystone Advantage: What the New Dynamics of Business Ecosystems Mean for Strategy, Innovation, and Sustainability*. Brighton, MA, USA: Harvard Business Press, 2004.
[17] S. Van Lingen, A. Palomba, and G. Lucassen, "On the software ecosystem health of open source content management systems," in *Proc. 5th Int. Workshop Softw. Ecosyst. (IWSECO)*, 2013, pp. 38–49.
[18] R. Spauwen and S. Jansen, "Towards the roles and motives of open source software developers," in *Proc. 5th Int. Workshop Softw. Ecosys. (IWSECO)*, 2013, pp. 62–73.
[19] A. Idu, T. van de Zande, and S. Jansen, "Multi-homing in the apple ecosystem: Why and how developers target multiple apple app stores," in *Proc. Int. Conf. Manage. Emergent Digit. EcoSyst.*, Nov. 2011, pp. 122–128.
[20] S. Hyrynsalmi, T. Mäkilä, A. Järvi, A. Suominen, M. Seppänen, and T. Knuutila, "App store, marketplace, play! An analysis of multi-homing in mobile software ecosystems," in *Proc. IWSECO*, 2012, p. 59.
[21] A. Wiggins, J. Howison, and K. Crowston, "Heartbeat: Measuring active user base and potential user interest in floss projects," in *Proc. IFIP Int. Conf. Open Source Syst.* Berlin, Germany: Springer, 2009, pp. 94–104.
[22] R. Hoving, G. Slot, and S. Jansen, "Python: Characteristics identification of a free open source software ecosystem," in *Proc. 7th IEEE Int. Conf. Digit. Ecosyst. Technol. (DEST)*, Jul. 2013, pp. 13–18.
[23] M. Iansiti and R. Levien, "Strategy as ecology," *Harvard Bus. Rev.*, vol. 82, no. 3, pp. 68–81, Mar. 2004.

[24] M. Goeminne and T. Mens, "Analyzing ecosystems for open source software developer communities," in *Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry*. Northampton, MA, USA: Edward Elgar, 2013, pp. 247–275.
[25] M. S. Handcock and K. J. Gile, "Comment: On the concept of snowball sampling," *Sociol. Methodol.*, vol. 41, no. 1, pp. 367–371, Oct. 2011.
[26] J. S. Coleman, "Relational analysis: The study of social organizations with survey methods," *Hum. Org.*, vol. 17, no. 4, pp. 28–36, 1958.
[27] L. A. Goodman, "Snowball sampling," *Ann. Math. Statist.*, vol. 32, no. 1, pp. 148–170, Mar. 1961.
[28] B. G. Glaser, A. L. Strauss, and E. Strutzel, "The discovery of grounded theory; strategies for qualitative research," *Nursing Res.*, vol. 17, no. 4, p. 364, Jul. 1968.
[29] B. G. Glaser and A. L. Strauss, *Discovery of Grounded Theory: Strategies for Qualitative Research*. Evanston, IL, USA: Routledge, 2017.
[30] D. Dhungana, I. Groher, E. Schludermann, and S. Biffl, "Software ecosystems vs. natural ecosystems: Learning from the ingenious mind of nature," in *Proc. 4th Eur. Conf. Softw. Archit., Companion*, Aug. 2010, pp. 96–102.
[31] B. W. Boehm, J. R. Brown, and M. Lipow, "Quantitative evaluation of software quality," in *Proc. 2nd Int. Conf. Softw. Eng.*, Oct. 1976, pp. 592–605.
[32] A. Yang, H. Lian, X. Jiang, and P. Liu, "Sparse Bayesian variable selection for classifying high-dimensional data," *Statist. Interface*, vol. 11, no. 2, pp. 385–395, Jan. 2018.
[33] H. Wang, L. Wang, Q. Yu, and Z. Zheng, "Learning the evolution regularities for bigservice-oriented online reliability prediction," *IEEE Trans. Services Comput.*, vol. 12, no. 3, pp. 398–411, May/Jun. 2019.
[34] H. Wang, C. Yu, L. Wang, and Q. Yu, "Effective bigdata-space service selection over trust and heterogeneous QoS preferences," *IEEE Trans. Services Comput.*, vol. 11, no. 4, pp. 644–657, Jul./Aug. 2018.
[35] L. Wang, Q. Zhao, Z. Wen, and J. Qu, "RAFFIA: Short-term forest fire danger rating prediction via multiclass logistic regression," *Sustainability*, vol. 10, no. 12, p. 4620, Dec. 2018.
[36] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," in *Proc. SIGMOD Rec.*, 1998, pp. 94–105.
[37] L. Zuoyong, "Projection pursuit theory (PPT) and its progress of application," *Explor. Nature*, vol. 17, no. 63, PP. 47–50, Jan. 1998.
[38] K. Pearson, "LIII. On lines and planes of closest fit to systems of points in space," *London, Edinburgh, Dublin Philos. Mag. J. Sci.*, vol. 2, no. 11, pp. 559–572, 1901.
[39] P. Hall and K.-C. Li, "On almost linearity of low dimensional projections from high dimensional data," *Ann. Statist.*, vol. 21, no. 2, pp. 867–889, 1993.
[40] X. Yang, Z. Yang, G.-H. Lu, and J. Li, "A gray-encoded, hybrid-accelerated, genetic algorithm for global optimizations in dynamical systems," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 10, no. 4, pp. 355–363, Jun. 2005.
[41] X. Z. Fern and C. E. Brodley, "Random projection for high dimensional data clustering: A cluster ensemble approach," in *Proc. 20th Int. Conf. Mach. Learn. (ICML)*, 2003, pp. 186–193.
[42] D. M. Gay, *Correctly Rounded Binary-Decimal and Decimal-Binary Conversions*. Murray Hill, NJ, USA: AT&T Bell Lab, 1990.
[43] C. M. Anderson-Cook, "Practical genetic algorithms," *J. Amer. Stat. Assoc.*, vol. 100, no. 471, pp. 1099–1099, 2005.
[44] Q. Fu, T.-G. Lu, and H. Fu, "Applying PPE model based on raga to classify and evaluate soil grade," *Chin. Geograph. Sci.*, vol. 12, no. 2, pp. 136–141, Jun. 2002.
[45] G. Lucassen, K. van Rooij, and S. Jansen, "Ecosystem health of cloud PaaS providers," in *Proc. Int. Conf. Softw. Bus.* Berlin, Germany: Springer, 2013, pp. 183–194.
[46] Z. Liao, L. Deng, X. Fan, Y. Zhang, H. Liu, X. Qi, and Y. Zhou, "Empirical research on the evaluation model and method of sustainability of the open source ecosystem," *Symmetry*, vol. 10, no. 12, p. 747, Dec. 2018.
[47] Z. Liao, M. Yi, Y. Wang, S. Liu, H. Liu, Y. Zhang, and Y. Zhou, "Healthy or not: A way to predict ecosystem health in Github," *Symmetry*, vol. 11, no. 2, p. 144, Jan. 2019.
[48] K. Mao, L. Capra, M. Harman, and Y. Jia, "A survey of the use of crowdsourcing in software engineering," *J. Syst. Softw.*, vol. 126, pp. 57–84, Apr. 2017.

LEI WANG received the Ph.D. degree in computer science from Southeast University, China. He is currently an Associate Professor with Nanjing Forestry University, China. He is also an Honorary Visiting Scholar with the Department of Computer Science and Engineering, The Chinese University of Hong Kong. His research interests mainly include service computing, software reliability engineering, and data mining. His publications have appeared in well-known journals and popular conferences, including the IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON SERVICES COMPUTING, the *Journal of Parallel and Distributed Computing*, ICSOC, ICWS, and SCC.

JING WAN received the B.S. degree from Nantong University, China. She is currently pursuing the master's degree with Nanjing Forestry University, China. Her research interests mainly include software ecosystem health measure, crowdsourced software engineering, and big data computing.

XINSHU GAO received the B.S. degree from Nanjing Forestry University, China. She is currently pursuing the master's degree with the University of Birmingham, U.K. Her research interests mainly include software ecosystem health measure, software reliability engineering, and big data computing.