# 开源软件开发者价值评估体系及其实证研究

游兰, 田明炎, 周烨, 陈智军, 王伟, 金红, 曾星, 崔海波

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.240100169

# 开源软件开发者价值评估体系及其实证研究

游　兰[1]　田明炎[1]　周　烨[1]　陈智军[1]　王　伟[2]　金　红[1]　曾　星[1]　崔海波[1]
1 湖北大学计算机与信息工程学院　武汉 430062
2 华东师范大学(普陀校区)数据科学与工程学院　上海 200062
(yoyo@hubu.edu.cn)

**摘　要**　如何科学客观地评估开源软件开发者的价值是开源领域面临的一个重要问题。现有研究方法存在评估指标较单一、指标权重难以确定等问题。针对这些问题,依据开源生态大数据分析,结合主客观评估方法,提出了一种多维度、多层次的开源软件开发者价值评估体系。综合考虑开发者在项目管理、编程、团队协作、学习、敬业度等方面的表现,通过5个一级指标、12个二级指标和7个三级指标,较全面和客观地评估开源软件开发者的能力和价值。采用Critic方法确定各维度指标的权重,解决了经验权重导致的准确性不高的问题。最后,采用Github 2020年全域开源生态数据,展开了多组实证研究,验证了开源社区开发者价值评估体系的有效性和可行性,为开源软件人才的培养、发现和管理提供了一种客观、科学且操作性较强的衡量方法。实验代码可从Github平台获取[1)]。

**关键词**:开源软件;开发者;价值评估指标体系;指标权重;实证研究分析
**中图分类号**　TP311.5

# Value Assessment System Oriented for Open-source Software Developers and Its Empirical Research

YOU Lan[1],TIAN Mingyan[1],ZHOU Ye[1],CHEN Zhijun[1],WANG Wei[2],JIN Hong[1],ZENG Xing[1] and CUI Haibo[1]
1 School of Computer Science and Information Engineering,Hubei University,Wuhan 430062,China
2 School of Data Science and Engineering,East China Normal University(Putuo Campus),Shanghai 200062,China

**Abstract**　Assessing the value of open-source software developers in a scientific and objective manner is an important issue in the open-source field. Existing research methods encounter challenges,including the use of limited evaluation metrics and the complexities associated with determining metric weights. To mitigate these issues,this paper proposes a multi-dimensional and multi-level assessment system for open-source software developers. The system is informed by an analysis of big data from the open-source ecosystem and combines both subjective and objective evaluation methods. By considering developers' performance in project management,programming,team collaboration,learning,and dedication,the proposed system comprehensively and objectively assesses their values using five primary indicators,twelve secondary indicators,and seven tertiary indicators. The Critic method is employed in this paper to determine the weights of various dimensions,overcoming the issue of low accuracy caused by experiential weights. Finally,multiple empirical studies are conducted using GitHub's 2020 global open-source ecosystem data to validate the effectiveness and feasibility of the open-source community developer value assessment system. This research provides an objective,scientific,and practical method for measuring the talent,discovery,and management of open-source software developers. The experimental code can be Obtained from theGithub platform[1)].

**Keywords**　Open-source software,Developer,Value evaluationsystem,Indicator weight,Empirical research analysis

## 1　引言

当今数字化时代,开源软件已经成为软件领域中不可忽视的力量。开源协作与共享为软件的开发与创新提供了一种新的模式[1]。目前,全球规模最大的开源软件平台GitHub,拥有超过900万的开发者,已经成为软件源代码托管和发现的主要平台。在开源项目中,开发者扮演着十分重要的角色,其价值和贡献对于开源软件的质量和发展至关重要。软件开发者价值的准确评估不仅能指导人力资源决策,更能最大程度地激发开发者的潜力,促进更有效的开源协作。因此,如何

1) https://github.com/OS-HUBU/DevValSys

到稿日期:2024-01-23　返修日期:2024-06-21
基金项目:国家自然科学基金(61977021);湖北省重点研发计划(2022BAA044);湖北省教育厅科学研究计划(Q20211010)
This work was supported by the National Natural Science Foundation of China(61977021),Key Research and Development Program of Hubei Province(2022BAA044) and Scientific Research Project of Education Department of Hubei Province(Q20211010).
通信作者:陈智军(chenzj@hubu.edu.cn)

准确客观地评估开源软件开发者的价值是开源软件领域面临的一个重要问题。

围绕开源软件开发者的价值评估问题，已有一些学者展开了研究，找到了一些指标如关注人数、管理项目数量等，但都存在评估指标比较单一、指标权重难以确定等问题，导致评估结果的准确性有待提高。针对这些问题，本研究依据开源生态大数据分析，结合主客观评估方法，提出了一种开源软件开发者价值评估体系，有助于客观公正地评估开源软件开发者的能力和价值。该方法采用 Critic 方法确定各维度的指标权重，以确保评估计算的合理性。本文以 Github 开源生态数据为实验基础，展开了大量实证研究，验证了开源社区开发者价值评估体系的有效性和可行性。

本文的主要贡献与创新包括以下几个方面：

1)基于开源生态大数据分析，提出了一种开源软件开发者价值评估体系。通过研究分析开发者在开源社区的行为和贡献以及参与的活动，结合主客观评估方法思想，将评估指标体系细分为 5 个一级指标、12 个二级指标和 7 个三级指标，其中绝大多数指标采用的是开源软件开发者的行为和贡献的客观量化度量。本评估体系从不同维度展现了开发者的各项能力和潜在价值。

2)针对现有评估方法指标权重难以确定的问题，采用 Critic 赋权法，通过计算各指标之间的关联程度以及相对重要性，较客观地确定开发者价值评估体系指标的权重，从而保证评估结果的客观性和准确性。

3)以 Github 开源生态大数据为基础，探究了本文评估方法中不同指标间的相关性，分析了 Github 全域开发者的价值等级分布情况。同时，分析 Github 项目影响力公开排行榜中 Top 排名项目中开发者的价值可以发现，在公认排行榜中排名靠前的开发者，在本文的价值评估体系中的价值排名也同样靠前，从而验证了本文方法的有效性和可行性。

本文提出的开源软件开发者价值评估体系，为开源软件人才的培养、发现和管理提供了一种客观、科学且操作性较强的衡量方法，将为新一代“开源协作”软件开发模式的应用和普及奠定基础。

## 2 相关研究

开源软件项目是将软件的源代码公开并允许任何人自由地查看、使用、修改和分发，其核心理念是通过开发者的开放式协作和贡献来共同推进软件质量的持续性改进。开源社区则提供了这样的开放式协作平台，通过“集众智、采众长”让可能互不相识的开发者共同参与到同一个软件项目的开发和维护中，从而使优秀的软件项目能够更加茁壮地成长，为更多的人提供服务。开源社区具有代码共享、分布式开发、贡献量、社区维护、社区驱动的发展等特点，其核心是代码开放和社区共享。代码开放指将源代码公开，帮助其他开发者和专业人士解决技术问题、提高技能水平；社区共享指开源社区的成员会共享自身的技术和知识，与其他人进行交流和学习，拓宽大家的技术视野。

开发者是开源社区的核心，他们通过共同的愿景和目标共同创造并维护软件，通过对开源项目的贡献获得声誉和技术价值，还可以通过参与开源项目来增强自己的技术实力和成就感。开发者在开源社区中存在多种行为，包括贡献代码、解决问题、提出建议、改进建议及参与决策等。

开发者和开源社区之间形成了一种相互依存的关系。开发者在开源社区中的贡献和参与不仅对个人有益，而且对整个社区和开源项目的发展也能起到积极的推动作用。他们的技术贡献和经验分享丰富了开源项目的功能并提高了其质量，促进了软件技术的进步和创新，为用户提供了更好的软件体验。同时，开源社区也为开发者提供了合作和交流的机会，促进了技术的共享和合作，形成了一个互利共赢的生态系统。

### 2.1 开源软件开发者的价值评估现状

开源社区的发展壮大离不开活跃的开发者群体，而开发者对于开源社区的贡献和价值是不可或缺的。因此，评估开发者的价值能力是开源社区管理者和企业所关注的问题之一。目前，已经有很多关于开发者价值评估的研究，其中涉及的内容包括评价方法和工具、评估指标和模型等方面。

围绕开发者的价值评估，已经有一些相关研究。Wu 等[2]提出了一种从分析开发者对核心技术文件的贡献出发，探析“核心-外围开发者”的一种算法，该算法以 9 个 Apache 项目为基础，分析了开发者对项目的贡献度，并以此有效地区分核心开发者和外围开发者。将该算法得到的名单与传统区分方案的名单进行了相似度比较，以此来评估各方法的优劣。之后建立分类模型，证明了文献[2]方案指标相比传统方案具有一定的优势，并分析了全局环境下影响开发者地位的关键因素。该研究的不足之处在于需要进一步考虑不同的项目特征和个人行为，兼顾定量分析和定性分析，利用多种指标来合理评价一名开发者的地位，以集成为一个统一的评价体系。

Tang 等[3]提出了一种开源社区软件开发者人力资源价值评估模型，基于 GitHub 软件开发者的行为操作数据建立软件开发者人力资源价值评估体系。该模型使用 CNN-LSTM 混合神经网络进行软件开发者人力资源价值评估并预测未来价值，选取其中对评估影响较大的非货币性且可数值化的参数，后续应考虑把一些文本参数量化后作为特征。这些影响软件开发者价值的因素没有考虑到某些活动行为异常的开发者，也没有考虑到指标之间的相关性因素，只是单纯地计算数量的多少，因此评估不够全面、客观。

Yang 等[4]研究了开源社区的开发进程，分析了开源领域能够定义开发者能力的一些指标，以及在开源领域什么样的项目才算优秀项目。另外还考虑到所分析的开源软件数目不够多，且没有考虑更多的影响因素。GitHub 开源软件需要考虑更多的影响因素和多种影响因素之间的相关性，因此其开发过程非常复杂。

除此之外，还有一些相关研究主要关注开发者在开源社区中的活跃度和贡献度的评估。然而，以上研究有些仅仅是基于代码贡献量或关注人数数量的评估模型，这些模型已经不能满足现代开源社区对开发者多方面价值的评估需求。还有部分工作在研究开发者评估指标和模型时采用了不同的方法，如基于统计方法的研究、基于机器学习的研究和基于网络分析的研究等，这些方法虽然提出了开发者多方面价值的评估指标，但是没有对指标进行细分，没有深入研究这些指标所包含的意义，也没有对这些指标之间的重要性进行区分和权重分析，虽然可以略微提高评估模型的准确性，但得出的结果

可能会受到多方面的影响。Oliva[5]的研究发现少数核心开发者没有在代码的编写和开发上进行过多的操作，更多的是进行项目的管理和技术经验上的指导，这彰显了本文提出的多种评估指标和指标权重划分的必要性。因此，仅根据代码量、关注量的多少并不能得到准确的评估结果。

总的来说，开源软件开发者的价值评估研究才刚刚开始，面临着不少挑战。1）缺乏客观性。开发者的价值评估往往容易受主观因素的影响，评估指标比较单一。2）评价指标权重缺乏依据。现有评价方法往往通过经验系数来确定指标因素的重要性。3）评估方法复现困难。大量开源社区数据存在信息缺失，很多评估方法难以实践应用。

随着开源软件的不断发展和普及，开发者价值评估指标和体系也在不断更新和演进。建立一个多维度视角的开源软件开发者价值评估体系对于激励和推动开放式协作和共享具有重要意义，但也面临着很多困难。

### 2.2　critic 赋权法

Critic 权重法是一种客观赋权法，也是一种多标准决策分析方法，用于确定各决策标准在决策中的重要性，进而确定最佳决策方案。它主要基于专家判断和统计分析，通过计算各因素之间的关联程度来确定权重。该方法的优点是无需对专家提问或者填写问卷，减少了由专家主观意识和选择偏差引入的误差，且该方法可以较为准确地反映各因素之间的相对重要性。

Critic 方法的核心是两项指标：波动性和冲突性指标。波动性用标准差表示，数据标准差越大说明波动越大，权重就越高；冲突性用相关系数表示，指标之间的相关系数值越大，说明冲突性越小，则其权重就越低。计算权重时，将波动性与冲突性指标相乘，并且进行归一化处理，即得到最终的权重。

其详细步骤如下：

1）获取分析的数据 $\boldsymbol{X}$

假设现有一组数据，有 $m$ 个待评价对象，$n$ 个评价指标，构成原始数据矩阵 $\boldsymbol{X}$，如式(1)所示：

$$\boldsymbol{X}=\begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{pmatrix} \tag{1}$$

2）对数据进行标准化

数据标准化的主要目的就是消除数据量纲的影响，使所有的数据能用统一的标准去衡量，计算式如式(2)、式(3)所示。其中 $x_{ij}$ 表示原始数据，$x_i$ 和 $x_j$ 分别是原始数据的最小值和最大值。

对于正向指标：

$$x'_{ij}=\frac{x_{ij}-\min(x_j)}{\max(x_j)-\min(x_j)} \tag{2}$$

对于逆向指标：

$$x'_{ij}=\frac{\max(x_j)-x_{ij}}{\max(x_j)-\min(x_j)} \tag{3}$$

3）计算数据信息的两项指标 $S_j$，$A_j$

其中 $S_j$ 是第 $j$ 个指标的标准差，表示对比度。

$S_j$ 的计算式如式(4)所示：

$$S_j=\sqrt{\frac{\sum_{i=1}^{m}(x_{ij}-\bar{x}_j)^2}{(n-1)}} \tag{4}$$

其中，$\bar{x}$ 为每个指标数据的均值。

$A_j$ 是第 $j$ 个指标的冲突性，表示与其他指标的相关性的倒数之和。冲突性的计算式如式(5)所示：

$$A_j=\sum_{i=1}^{n}1-r_{ij} \tag{5}$$

其中，$r_{ij}$ 表示第 $i$ 个指标与第 $j$ 个指标的相关系数。

$C_j$ 是第 $j$ 个指标的信息量，其计算式如式(6)所示：

$$C_j=S_j\times A_j \tag{6}$$

4）计算最终的权重 $W$

$$W=\frac{C_j}{\sum_{j=1}^{n}C_j} \tag{7}$$

Critic 权重法利用了数据的波动性和冲突性，完全依靠数据自身的客观属性，但并非数字越大就越重要，以达到科学评价的目的。本研究主要通过 Critic 方法来帮助有效确定开发者评估指标的权重，从而保证评估结果的客观性和准确性。

## 3　开源软件开发者的价值评估分析

评估一个开发者的价值能力可以从多个维度进行考量，本研究认为应从项目管理能力、编程能力、学习能力、团队协作能力、敬业度这 5 个方面展开评估。主要依据如下：

1）项目管理能力是刻画开发者能力价值的一个十分重要的方面。项目管理能力指开发者在开源社区中的项目管理、维护及解决问题的能力。例如，在 GitHub 这样庞大的开发者社区中，存在着大量的开源软件项目[6-10]，而开发者既可以创建自己的项目，也可以参与其他项目并做出贡献。项目管理能力的展示不仅涉及对项目本身的管理，还包括与团队成员的协作和交流能力。有效的项目管理能力不仅对项目的顺利进行至关重要，也对团队的协作和项目的成功有着积极影响。具备优秀的项目管理能力的开发者能够更好地规划和执行项目，确保项目按时交付、质量可控，并提高团队的工作效率和成果质量。因此，对开发者的项目管理能力进行评估和衡量，对于识别优秀的开发者、建立高效的团队以及推动项目的成功具有重要意义。综合考虑开发者在项目管理方面的表现，有助于更全面地评估开发者的价值和潜力。

2）编程能力是软件开发者的核心竞争力[11-13]，开发者的编程能力代表开发者在编写和维护代码方面的技术能力。作为一个面向开发者的社交平台，开源社区扮演着代码仓库和版本控制系统的角色，吸引着全球范围内的开发者共同协作和贡献代码。很多的开源社区行为和操作都能直接体现出开发者的编程能力。首先，开源代码库中的项目和提交记录直接反映了开发者的编码活动，如参与其他项目的数量、参与其他项目提交的 pr 数量以及被合并的 pr 数量；其次，开发者的编程能力还可以通过编写的代码的影响力和受欢迎程度来反映，如 GitHub 中创建项目的数量、创建项目的 fork 总数和 watch 总数等。GitHub 开发者的编程能力是一个综合性的评估，需要综合考虑其代码贡献、项目影响力和协作能力等多个因素。

3）学习能力是开源软件开发者学习新技术、掌握新知识和适应变化的能力，能够体现其持续发展潜力。在高速发展的软件技术领域中，持续学习和更新知识是开发者保持竞争力和适应性的关键。现有的开源社区行为和操作很难直接反

映开发者的学习能力，但可以从侧面间接体现。例如，通过分析其代码库和项目中使用的技术栈和工具，可以了解开发者对不同类型技术的驾驭情况。优秀的开发者通常会涉猎多个技术领域，开发者掌握的编程语言的数量越多，也就意味着开发者的学习能力越强。因此，开发者掌握的编程语言数量可以一定程度上反映其学习能力。

4)团队协作能力指开发者与其他人共同完成同一任务的能力，反映了协作配合程度和团队贡献程度，这对于开放式的开源软件协作尤为重要。当前 Github 平台上的一个开源项目，通常由一群身处不同空间环境且互不相识的开发者基于互联网平台合作研发和管理，他们的协作力和配合度对于该软件的质量至关重要。

开源社区本质上为开发者提供了一个技术社交平台，而技术社交行为通常能间接体现开发者的团队协作能力。首先，通过开发者与其他开发者之间的协作行为的次数，能够了解开发者的历史协作情况和团队协作经验，如共同参与的项目数量、在项目中的角色和贡献度等。其次，开发者与他人之间互动操作的紧密程度也能反映团队协作的能力，例如开源社区中的问题讨论、代码审查、合并请求等。这些都能够反映开发者是否积极参与团队的讨论和决策过程，是否真实地参与共同解决问题，是否具备沟通交流能力，以及是否推动项目进展，并且能体现开源软件开发者的团队协作能力。

5)开发者敬业度是专业态度的体现，是人才培养和评价的重要指标。由于开放式无限制的在线协作和管理，开源软件开发者的敬业度很难采用传统人事方法来评价，但在开源社区中可以通过分析开发者在不同时间段的活动来评估。在开源社区评估开发者的敬业度可以考虑以下几个因素。首先，通过分析开发者的活跃天数来了解他们在开源社区的参与程度，开发者在多少天内有活跃行为，如代码提交、评论、问题讨论等，可以反映他们对软件项目的持续关注和参与情况。其次，开发者在开源社区的工作量也是评估敬业度的重要指标，包括创建的项目数量、发表的 issue 数量、提交的代码量等。高敬业度的开发者通常在开源社区上有丰富的工作产出，积极参与项目并提供有价值的贡献。评估开发者的敬业度可以考虑活跃天数、工作量等因素。敬业度越高，表明开发者在开源社区平台上展现出更高的活跃度和更大的贡献，具备更高的价值和影响力。

近年来，互联网上涌现出了许多受欢迎的开源社区平台，比如 GitHub，Git Lab，Bit bucket 等。每个开源社区的术语都基本类似，这些术语的具体使用可能因不同的社区或平台而异，但概念含义是相似的。为了方便后续描述，本文统一了开源社区通用术语[14-17]。

**定义 1**(仓库，Repository，简称 Rep) 在 GitHub 上托管的项目的存储空间，包含项目的代码、文档和其他资源。

**定义 2**(分支，Branch) 基于主分支或其他分支创建的代码分支，用于独立开发和测试功能，最终可以将其合并到主分支或其他分支中。

**定义 3**(提交，Commit) 在版本控制系统中，表示一次代码变更的记录。提交包含了修改的文件、代码注释和作者等信息。

**定义 4**(拉取请求，Pull Request，简称 PR) 开发者向项目仓库提交的一种请求，希望将自己的代码变更合并到项目中。PR 包含了解决的问题、所做的更改、变更的目的等相关信息，供项目的维护者或其他贡献者审查和讨论。

**定义 5**(合并，Merge) 将分支或拉取请求的代码变更合并到目标分支中，使两者的代码保持一致。

**定义 6**(问题，Issue) 在项目中遇到的问题、错误或建议的记录。问题可以用于报告 bug、提出新功能请求或进行讨论，并可以分配给相关的开发者进行处理。

**定义 7**(评论，Comment) 对问题、拉取请求或其他仓库对象进行讨论和交流的文字内容。评论允许开发者之间进行沟通、提供反馈或解答问题。

**定义 8**(关注，Watch) 在开源社区上关注一个仓库，以便接收关于仓库活动的通知，如新的问题、拉取请求、提交等。

**定义 9**(复刻，Fork) 在开源社区上复制一个仓库，以便在自己的账户下拥有一份独立的副本，forks 即为个人仓库的收藏者。

**定义 10**(追随者，Followers) 指关注某个开源项目或特定开发者的用户或成员。

## 4 开发者价值评估体系

之前的研究提出了一种开源社区软件开发者人力资源价值评估体系，并提出了 6 种一级指标，但是对于如何通过 GitHub 上的行为去量化定义二级指标，并未给出确定的形式化定义，并且所提出的二级指标不能很好地展现一级指标的特点，同时，提出的二级指标过于广泛，在开源社区里没有具体的活动去展现。而本文评估指标体系是一个多维度、多层次指标体系，细分为 5 个一级评估指标、12 个二级评估指标和 7 个三级评估指标，如图 1 所示。

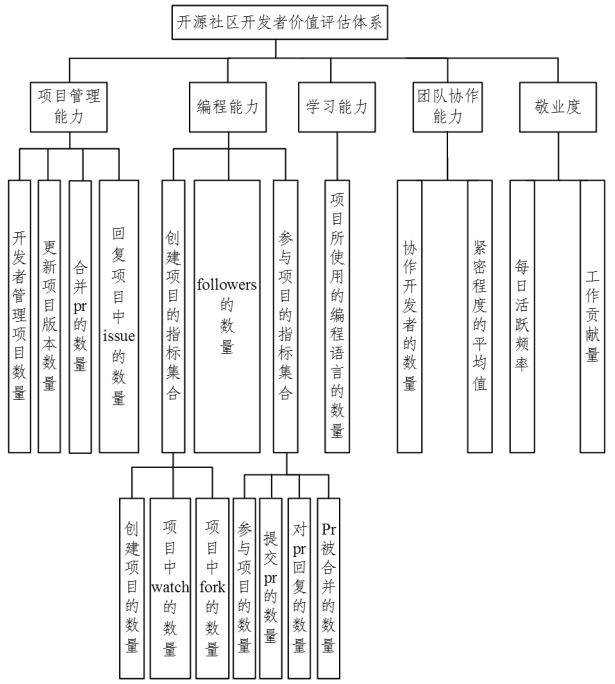


图 1 开源社区开发者价值评估体系

Fig. 1 Value assessment system of open source community developer

表 1 中列出了开源社区开发者价值评估指标的定义和描述，这些度量指标从 5 个不同的方面诠释了开发者价值评估的多维属性。

表 1　开发者价值评估度量指标

Table 1　Developer value assessment metrics

| 指标 | | | 描述 |
|---|---|---|---|
| 项目管理能力 | 管理项目数量 | rep_sum | 开发者开源社区中管理多少个项目 |
| | 更新项目版本数量 | release_sum | 开发者管理的项目中更新版本的次数 |
| | 合并 pr 数量 | pr | 管理的项目中合并 pr 的数量 |
| | 回复项目中 issue 的数量 | issue_sum | 管理的项目中回复 issue 数量 |
| 编程能力 | followers 的数量 | followers | 开发者的 Followers 的数量 |
| | 创建项目的总数 | my_rep | 开发者创建项目的数量 |
| | 项目的 watch 总数 | watches | 创建的项目被用户关注的数量 |
| | 项目被 Fork 的总数 | forks | 开发者项目 fork 的数量 |
| | 参与项目的总数 | part_rep | 开发者参与其他项目的数量 |
| | 提交 pr 的总数 | pr_sub | 在参与项目中提交 pr 的数量 |
| | 对 Pr 回复的总数 | pr_review | 参与的项目中 pr review 的数量 |
| | Pr 被合并的总数 | pr_merge | 在参与项目中 pr 被合并的数量 |
| 学习能力 | 掌握编程语言的数量 | language | 开发者在项目中使用编程语言的数量 |
| 团队协作能力 | 开发者的协作开发者数量 | A1 | 项目中有过协作关系的开发者数量 |
| | 与开发者的协作程度 | A2 | 项目中与协作者的紧密程度的平均值 |
| 敬业度 | 每日活跃频率 | WA | 开发者一年中活跃天数占全年的比例 |
| | 工作贡献度 | WC | 开发者一年贡献量占平均贡献量的比例 |

开发者价值评估综合计算式如下：

$$S_{\text{开发者}}=X_1\times S_{\text{项目管理能力}}+X_2\times S_{\text{编程能力}}+X_3\times S_{\text{学习能力}}+X_4\times S_{\text{团队协作能力}}+X_5\times S_{\text{敬业度}} \tag{8}$$

其中，$X_1$，$X_2$，$X_3$，$X_4$，$X_5$ 表示该一级指标的权重因子，$S_{\text{项目管理能力}}$，$S_{\text{编程能力}}$，$S_{\text{学习能力}}$，$S_{\text{团队协作能力}}$，$S_{\text{敬业度}}$ 表示 5 个一级指标得分。这 5 个指标的得分能够多维度体现出开发者不同方面的综合能力，最后将这 5 项能力的得分通过加权求和得到开发者价值评估的总得分 $S_{\text{开发者}}$。

$S_{\text{开发者}}$ 表示开发者价值评估的总得分，由 5 个指标的加权求和得到。每个一级指标的满分是 100 分，这样经过加权求和之后得到的总分 $S_{\text{开发者}}$ 的满分也是 100 分。通过对 $S_{\text{开发者}}$ 的得分划分区间，最后可对开发者的得分进行评级。

该评级的区间划分表达式如式(9)所示：

$$f(S_{\text{开发者}})=\begin{cases}D, & 0\leqslant S_{\text{开发者}}\leqslant 20\\ C, & 20<S_{\text{开发者}}\leqslant 40\\ B, & 40<S_{\text{开发者}}\leqslant 60\\ A, & 60<S_{\text{开发者}}\leqslant 80\\ S, & 80<S_{\text{开发者}}\leqslant 100\end{cases} \tag{9}$$

$X_1$，$X_2$，$X_3$，$X_4$，$X_5$ 表示各项能力得分的权重因子，该权重因子是由上述的 critic 赋权法所得到的，其得出的权重因子结果客观且符合现实。表 2 列出了 5 个一级指标的权重因子结果。

表 2　开发者价值评估的一级指标权重

Table 2　Weight of primary indicators for developer value assessment

| | 项目管理能力 | 编程能力 | 学习能力 | 团队协作能力 | 敬业度 |
|---|---|---|---|---|---|
| 权重 | 0.2221 | 0.1511 | 0.2663 | 0.1315 | 0.2300 |

$X_1=0.2221$，$X_2=0.1511$，$X3=0.2663$，$X_4=0.1315$，$X_5=0.2300$

在开发者价值评估体系中，评估指标的选择和定义是构建准确可靠的评估体系的关键步骤[18]。下文将详细介绍一级评估指标的定义和计算过程。为了准确评估开发者的价值能力，需要选择具有代表性和可度量性的指标，从而有助于较为全面地覆盖开发者在开源社区上的关键能力和贡献领域。

### 4.1　开发者项目管理能力

项目管理能力维度包含开发者管理项目的数量、更新项目版本的数量、合并项目 pr 的数量、项目中回复 issue 的数量 4 个二级指标。

开发者项目管理能力的定义如下：

$$S_{\text{项目管理能力}}=W_{\text{rep\_sum}}\times f(Z_{\text{rep\_sum}})+W_{\text{release\_sum}}\times f(Z_{\text{release\_sum}})+W_{pr}\times f(Z_{pr})+W_{\text{issue\_sum}}\times f(Z_{\text{issue\_sum}}) \tag{10}$$

其中，$S_{\text{项目管理能力}}$ 表示开发者项目管理能力；$W_{\text{issue\_sum}}$ 表示该影响因素的权重因子；$Z_{\text{issue\_sum}}$ 表示开发者管理项目中亲自回复的 issue 的数量，例如某个开发者总计回复了多少个 issue；$f(Z_{\text{issue\_sum}})$ 表示以该影响因素为自变量的函数值；函数 $f(x)$ 即为相应三级指标的实际映射方式，例如某个开发者总计回复了多少个 issue，根据回复 issue 的数量给予赋值。

利用上文讲到的 critic 方法步骤，最终可以得到相应影响因子的权重，如表 3 所列。

表 3　项目管理能力 CRITIC 权重确定法的计算结果

Table 3　Calculation results of critical weight determination method for project management capability

| 项 | 指标变异性 | 指标冲突性 | 信息量 | 权重 |
|---|---|---|---|---|
| 管理仓库的数量 | 0.010 | 2.859 | 0.028 | 0.2069 |
| 合并 pr 的数量 | 0.013 | 2.515 | 0.032 | 0.2400 |
| 更新版本的数量 | 0.008 | 2.920 | 0.023 | 0.1730 |
| 回复 issue 的数量 | 0.020 | 2.542 | 0.051 | 0.3801 |

从表 3 可以看到，回复 issue 的数量这个二级指标变异性最大、指标冲突性第二，因此权重最高为 0.3801；指标变异性中更新版本的数量最少，所以其权重最低，仅为 0.1730。

$W_{\text{rep\_sum}}=0.2069$，$W_{pr}=0.2400$，$W_{\text{release\_sum}}=0.1730$，$W_{\text{issue\_sum}}=0.3801$

接下来，分别给出式(10)中映射函数 $f(x)$ 的计算方式。这里给出对应上述 4 个二级指标的映射函数的表达式，如

式(11)—式(14)所示：

$$f(Z_{issue\_sum})=\begin{cases}0.5, & 1\leqslant issue\leqslant 5\\ 1, & 6\leqslant issue\leqslant 9\\ 3, & 10\leqslant issue\leqslant 19\\ 5, & 20\leqslant issue\leqslant 29\\ 10, & 30\leqslant issue\leqslant 49\\ 20, & 50\leqslant issue\leqslant 99\\ 40, & 100\leqslant issue\leqslant 249\\ 80, & 250\leqslant issue\leqslant 499\\ 100, & 500\leqslant issue\leqslant 999\\ 150, & 1000\leqslant issue\leqslant 1\,999\\ 200, & issue\geqslant 2\,000\end{cases} \tag{11}$$

$$f(Z_{pr})=\begin{cases}1, & 1\leqslant pr\_merge\leqslant 9\\ 3, & 10\leqslant pr\_merge\leqslant 19\\ 7, & 20\leqslant pr\_merge\leqslant 39\\ 10, & 40\leqslant pr\_merge\leqslant 59\\ 15, & 60\leqslant pr\_merge\leqslant 79\\ 20, & 80\leqslant pr\_merge\leqslant 99\\ 40, & 100\leqslant pr\_merge\leqslant 249\\ 80, & 250\leqslant pr\_merge\leqslant 499\\ 100, & 500\leqslant pr\_merge\leqslant 999\\ 150, & pr\_merge\geqslant 1\,000\end{cases} \tag{12}$$

$$f(Z_{release\_sum})=\begin{cases}1, & 1\leqslant release\leqslant 2\\ 2, & 1\leqslant release\leqslant 5\\ 5, & 6\leqslant release\leqslant 9\\ 15, & 10\leqslant release\leqslant 19\\ 25, & 20\leqslant release\leqslant 29\\ 40, & 30\leqslant release\leqslant 49\\ 60, & 50\leqslant release\leqslant 69\\ 80, & 70\leqslant release\leqslant 99\\ 100, & 100\leqslant release\leqslant 299\\ 150, & 300\leqslant release\leqslant 999\\ 200, & release\geqslant 1\,000\end{cases} \tag{13}$$

$$f(Z_{rep\_sum})=\begin{cases}10, & 1\leqslant rep\_sum\leqslant 13\\ 20, & 14\leqslant rep\_sum\leqslant 40\\ 30, & 41\leqslant rep\_sum\leqslant 70\\ 40, & 71\leqslant rep\_sum\leqslant 100\\ 50, & 50\leqslant rep\_sum\leqslant 999\\ 100, & rep\_sum\geqslant 1\,000\end{cases} \tag{14}$$

通过上面4个二级指标的映射函数的表达式，可以分别得出各二级指标数量对应的区间得分值，然后通过式(10)得到项目管理能力的评估得分值。

本指标的设计考虑到部分开发者在开源领域的工作量特别大，该指标的值域没有上限。为了方便将该指标的得分统计到价值评估体系当中，本文通过不同值域区间的人数比例，对该指标进行了等级映射。70%的用户为D级20分，10%的人为C级40分，10%的人为B级60分，7%的人为A级80分，3%的人为S级100分。通过这种等级映射划分，少数能力值超高或超低的开发者都能映射到统一的价值评估体系中，如表4所列。

表4 项目管理能力得分区间占比

Table 4 Proportion of score range for project management capability

| 分数区间 | 百分比 | 有效百分比 | 累积百分比 |
|---|---|---|---|
| 0～6 | 71.8 | 71.8 | 71.8 |
| 6～10 | 12.3 | 12.3 | 84.0 |
| 10～15 | 8.2 | 8.2 | 92.2 |
| 15～20 | 2.4 | 2.4 | 94.6 |
| 20以上 | 5.4 | 5.4 | 100.0 |

### 4.2 开发者编程能力

在编程能力维度上，采用创建项目的集合、followers、参与他人项目的集合作为开源社区开发者价值评估的3个二级评价指标。其中，创建项目的集合指标细分为创建项目的总数、项目的watch总数、项目的fork总数3个三级指标；参与他人项目的集合细分为参与项目的总数、提交pr的总数、对pr回复的总数、pr被合并的总数4个三级指标。

开发者编程能力的定义如下：

$$\begin{aligned}S_{编程能力}=&W_{my\_rep}\times f(Z_{my\_rep})+W_{watches}\times f(Z_{watches})+\\&W_{forks}\times f(Z_{forks})+W_{part\_rep}\times f(Z_{part\_rep})+\\&W_{pr\_sub}\times f(Z_{pr\_sub})+W_{pr\_merge}\times f(Z_{pr\_merge})+\\&W_{pr\_review}\times f(Z_{pr\_review})+W_{followers}\times\\&f(Z_{followers})\end{aligned} \tag{15}$$

其中，$S_{编程能力}$表示开发者编程能力值，$W_{factor}$表示三级指标factor的权重系数；$Z_{factor}$表示该开发者在三级指标factor上的数量，比如某个开发者创建项目的数量；$f(Z_{factor})$表示三级指标factor，是以三级指标factor为自变量的函数值；函数$f(x)$即为相应三级指标的实际映射方式，例如某个开发者创建了多少个项目，依据创建的项目数量给予得分。

使用critic权重确定方法得出所有三级指标的权重，结果如表5所列。

表5 编程能力CRITIC权重确定法计算结果

Table 5 Calculation results of programming ability CRITIC weight determination method

| 项 | 指标变异性 | 指标冲突性 | 信息量 | 权重 |
|---|---|---|---|---|
| my_rep | 0.005 | 5.874 | 0.030 | 0.175 |
| watches | 0.003 | 5.539 | 0.018 | 0.106 |
| forks | 0.005 | 5.717 | 0.030 | 0.171 |
| part_rep | 0.002 | 6.843 | 0.015 | 0.087 |
| Pr_sub | 0.002 | 6.846 | 0.012 | 0.071 |
| pr_merge | 0.003 | 6.493 | 0.022 | 0.127 |
| pr_review | 0.004 | 6.635 | 0.025 | 0.144 |
| followers | 0.003 | 6.614 | 0.021 | 0.120 |

由表5可知，编程能力的各影响因素的权重分别如下：

$W_{my\_rep}=0.175$，$W_{watches}=0.106$

$W_{forks}=0.171$，$W_{part\_rep}=0.087$

$W_{pr\_sub}=0.071$，$W_{pr\_merged}=0.127$

$W_{pr\_review}=0.144$，$W_{followers}=0.120$

从表5可以看到，创建项目数量和forks数量这个二级指标变异性最大、指标冲突性较小，因此他们的权重最高，分别为0.175和0.171；指标变异性中参与项目的数量和提交pr数量最少，因此其权重最低，分别仅为0.087和0.071。

同样这一部分也需要映射函数进行计算，对应指标映射函数的表达式如式(16)—式(23)所示：

$$f(Z_{my\_rep})=\begin{cases}10, & my\_rep=0\\60, & my\_rep=1\\65, & my\_rep=2\\70, & my\_rep=3\\75, & my\_rep=4\\80, & my\_rep=5\\85, & my\_rep=6\\90, & my\_rep=7\\95, & my\_rep=8\\100, & my\_rep\geqslant 9\end{cases}\tag{16}$$

$$f(Z_{watches})=\begin{cases}10, & watches=0\\60, & watches=1\\65, & watches=2\\70, & watches=3\\75, & 4\leqslant watches\leqslant 6\\80, & 7\leqslant watches\leqslant 11\\85, & 12\leqslant watches\leqslant 24\\90, & 25\leqslant watches\leqslant 67\\95, & 68\leqslant watches\leqslant 299\\100, & watches\geqslant 300\end{cases}\tag{17}$$

$$f(Z_{forks})=\begin{cases}10, & forks=0\\60, & forks=1\\65, & 2\leqslant forks\leqslant 3\\70, & 4\leqslant forks\leqslant 9\\75, & 10\leqslant forks\leqslant 25\\80, & 26\leqslant forks\leqslant 70\\85, & 71\leqslant forks\leqslant 226\\90, & 227\leqslant forks\leqslant 1\,600\\95, & 1601\leqslant forks\leqslant 2\,999\\100, & forks\geqslant 3\,000\end{cases}\tag{18}$$

$$f(Z_{part\_rep})=\begin{cases}10, & part\_rep=0\\60, & part\_rep=1\\65, & part\_rep=2\\70, & part\_rep=3\\75, & part\_rep=4\\80, & part\_rep=5\\85, & part\_rep=6\\90, & part\_rep=7\\95, & part\_rep=8\\100, & part\_rep\geqslant 9\end{cases}\tag{19}$$

$$f(Z_{pr\_sub})=\begin{cases}10, & pr\_sub=0\\60, & pr\_sub=1\\65, & pr\_sub=2\\70, & 3\leqslant pr\_sub\leqslant 6\\75, & 7\leqslant pr\_sub\leqslant 12\\80, & 13\leqslant pr\_sub\leqslant 26\\85, & 27\leqslant pr\_sub\leqslant 57\\90, & 58\leqslant pr\_sub\leqslant 149\\95, & 150\leqslant pr\_sub\leqslant 512\\100, & pr\_sub\geqslant 513\end{cases}\tag{20}$$

$$f(Z_{pr\_merge})=\begin{cases}10, & pr\_merged=0\\60, & pr\_merged=1\\65, & pr\_merged=2\\70, & 3\leqslant pr\_merged\leqslant 5\\75, & 6\leqslant pr\_merged\leqslant 8\\80, & 9\leqslant pr\_merged\leqslant 15\\85, & 16\leqslant pr\_merged\leqslant 27\\90, & 28\leqslant pr\_merged\leqslant 51\\95, & 52\leqslant pr\_merged\leqslant 121\\100, & pr\_merged\geqslant 122\end{cases}\tag{21}$$

$$f(Z_{pr\_review})=\begin{cases}10, & pr\_review=0\\60, & pr\_review=1\\65, & 2\leqslant pr\_review\leqslant 3\\70, & 4\leqslant pr\_review\leqslant 5\\75, & 6\leqslant pr\_review\leqslant 8\\80, & 9\leqslant pr\_review\leqslant 14\\85, & 15\leqslant pr\_review\leqslant 22\\90, & 23\leqslant pr\_review\leqslant 38\\95, & 39\leqslant pr\_review\leqslant 68\\100, & pr\_review\geqslant 69\end{cases}\tag{22}$$

$$f(Z_{followers})=\begin{cases}10, & followers=0\\60, & followers=1\\65, & 2\leqslant followers\leqslant 3\\70, & 4\leqslant followers\leqslant 7\\75, & 8\leqslant followers\leqslant 14\\80, & 15\leqslant followers\leqslant 24\\85, & 25\leqslant followers\leqslant 43\\90, & 44\leqslant followers\leqslant 92\\95, & 93\leqslant followers\leqslant 351\\100, & followers\geqslant 352\end{cases}\tag{23}$$

### 4.3　开发者学习能力

在学习能力维度,采用项目所使用的编程语言的数量作为开源社区开发者学习能力的评价指标。

开发者学习能力计算式如下:

$$S_{学习能力}=f(Z_{language})\tag{24}$$

其中,$S_{学习能力}$表示开发者的学习能力评估得分,$Z_{language}$表示开发者掌握的编程语言数量,$f(Z_{language})$表示开发者学习能力评估函数。

接下来,分别给出式(24)中映射函数$f(Z_{language})$的计算方式,具体映射函数的表达式如式(25)所示:

$$f(Z_{language})=\begin{cases}10, & Z_{language}=1\\50, & Z_{language}=2\\60, & Z_{language}=3\\65, & Z_{language}=4\\70, & Z_{language}=5\\80, & Z_{language}=6\\90, & 7\leqslant Z_{language}<8\\90, & 9\leqslant Z_{language}<10\\95, & 11\leqslant Z_{language}<15\\100, & Z_{language}\geqslant 16\end{cases}\tag{25}$$

通过上述掌握编程语言的数量与学习能力的映射函数，可以得到开发者学习能力的评估值。

### 4.4 开发者团队协作能力

在团队协作能力维度，采用开发者的协作开发者的数量、与协作者之间协作行为的紧密程度作为开源社区开发者团队协作能力的评价指标。其计算式如下：

$$S_{团队协作能力}=f(A1)*W_1+f(A2)*W_2 \tag{26}$$

式(27)中 $A1$ 表示与开发者协作过的开发者数量，$A2$ 表示开发者与其协作者之间的紧密程度的平均值。$A2$ 由开发者与所有协作者的紧密程度总和(记作 $M$)/协作过的开发者数($A1$)得出，即：

$$A2=M\div A1 \tag{27}$$

协作者的紧密程度由开发者和协作者在某个 issue 或者 pr 中进行评论互动的次数来确定，由于在开源领域中 pr 下的评论相较于 issue 下的评论更能体现开发者的能力，因此对其给予的得分也应该有差异性。通过研究形成以下规则：pr 下的评论互动一次则紧密程度加 2，issue 下的评论互动一次则紧密程度加 1。以此类推，最后算出每一个协作者与开发者之间的紧密程度总和(记作 $M$)。

同样地，使用 critic 权重确定法得出的结果如表 6 所列。

表 6 团队协作能力 CRITIC 权重确定法计算结果

Table 6 Calculation results of CRITIC weight determination method for teamwork collaboration ability

| 指标 | 指标变异性 | 指标冲突性 | 信息量 | 权重 |
|---|---|---|---|---|
| 协作者数量 | 0.027 | 0.938 | 0.026 | 0.7070 |
| 紧密程度平均值 | 0.011 | 0.938 | 0.011 | 0.2930 |

这里给出对应上述两个二级指标的映射函数的表达式，如式(28)和式(29)所示：

$$f(A1)=\begin{cases}5, & A1<3\\ 10, & 3<A1<5\\ 15, & 5<A1<10\\ 25, & 10<A1<30\\ 40, & 30<A1<50\\ 50, & 50<A1<100\\ 75, & 100<A1<150\\ 100, & A1>150\end{cases} \tag{28}$$

$$f(A2)=\begin{cases}5, & A2<3\\ 10, & 3<A2<5\\ 15, & 5<A2<10\\ 25, & 10<A2<30\\ 40, & 30<A2<50\\ 50, & 50<A2<100\\ 75, & 100<A2<150\\ 100, & A2>150\end{cases} \tag{29}$$

### 4.5 开发者敬业度

使用活跃频率和工作贡献度两个三级指标来评估开发者的敬业度。

假设一个软件开发者在过去一年中共参与了 200 天的开发工作，在开源社区上的工作总量(包括 issue_comment，open_issue，open_pr，review_comment 和 pr_merged 的数量)为 1 000，同行平均工作量为 500，则该开发者的活跃频率 $WA$ 可以通过式(30)计算。

$$WA=AD/365 \tag{30}$$

其中，$AD$ 为开发者在一年中活跃的天数。

该开发者的工作贡献度 $WC$ 可以通过式(31)计算。

$$WC=WL/AVG(WL) \tag{31}$$

其中，$WL$ 为工作总量(包括 issue_comment，open_issue，open_pr，review_comment 和 pr_merged 的数量)，$AVG(WL)$ 为同行开发者的平均工作贡献量。

最终，该开发者的敬业度 $LY$ 可以通过式(32)和式(33)展现。

$$LY=WA\times WC \tag{32}$$

$$S_{敬业度}=f(LY) \tag{33}$$

具体的 $f(LY)$ 映射函数的表达式如下：

$$f(LY)=\begin{cases}0, & LY=0\\ 20, & 0<LY<0.02\\ 60, & 0.02\leqslant LY<0.15\\ 65, & 0.15\leqslant LY<0.5\\ 70, & 0.5\leqslant LY<1.5\\ 80, & 1.5\leqslant LY<4.0\\ 85, & 4.0\leqslant LY<8.7\\ 90, & 8.7\leqslant LY<16.8\\ 95, & 16.8\leqslant LY<29.2\\ 100, & LY\geqslant 29.2\end{cases} \tag{34}$$

## 5 实证分析与结论

为了验证开发者价值评估体系的有效性和实用性，本文采用了 GitHub 上的真实开发者数据来进行实证研究。本文的实证研究包括 3 个部分。1)对开发者价值等级进行群体分析，以了解 GitHub 开发者群体的价值等级分布情况[19-20]。2)对评估价值高、中、低 3 类开发者分别展开分析，从开发者的其他开源表现来分析其是否与本文价值评估体系得到的评估结果一致。3)针对公认优秀的开源项目中开发者的价值评估分析，以验证开发者价值评估体系的合理性和准确性。本文实证研究采用的是 2020 年的 Github 全域开源生态数据。4)验证了不同指标之间的相关性，确定不同指标之间的相关性和指标与总分之间的相关性，以此来验证本文指标的选取是否合理。

### 5.1 数据的收集与处理

由于新冠疫情，2020 年全球很多开发者不得不居家办公，在此基础上各种开源软件的开发者和各种活动大幅度增加。而 Github2020 年的日志数据更是达到了 319 GB。因此本文从 GitHub API 获取了 2020 年全年的开发者全域数据，采用 clickhouse 数据库管理。该数据显示，2020 年在 GitHub 上有仓库的开发者人数共计 317 200 人，产生了 109 GB 的日志数据。本文对这些日志数据进行了分析，从中提取了 143 个指标字段，然后对这些指标字段进行了筛选和汇总，以得到本文所需的数据。表 7 列出了部分指标字段的说明。

表 7　分析数据中部分使用的字段

Table 7　Fields partially used in analytical data

| 字段名称 | 字段类型 | 详细描述 | 有效范围 |
|---|---|---|---|
| id | UInt64 | 事件实例的唯一标识 | 全部 |
| type | Enum | 事件实例类型,不同事件实例类型具有不同的有效负载类型 | 全部 |
| action | Enum | 对于关键事件实例的操作 | 全部 |
| actor_id | UInt64 | 触发该事件实例的 GitHub 用户 ID | 全部 |
| actor_login | LowCardinality(String) | 触发该事件实例的 GitHub 用户登录名称 | 全部 |
| repo_id | UInt64 | 被跟踪的事件的所属仓库唯一标识 | 全部 |
| repo_name | LowCardinality(String) | 仓库名称 | 全部 |
| org_id | UInt64 | 该仓库所属组织的唯一标识 | 全部 |
| org_login | LowCardinality(String) | 该组织的登录名称 | 全部 |
| created_at | DateTime | 该事件实例在 GitHub 上的创建时间 | 全部 |
| created_date | Date | 该事件实例在 GitHub 上的创建日期 | 全部 |
| issue_id | UInt64 | 该问题在 GitHub 上的唯一标识 | 问题,问题评论,拉取请求,拉取请求审阅评论模块 |
| issue_number | UInt32 | 该问题在当前仓库中的 ID | 问题,问题评论,拉取请求,拉取请求审阅评论模块 |
| issue_title | String | 问题标题 | 问题,问题评论,拉取请求,拉取请求审阅评论模块 |
| issue_body | String | 问题正文 | 问题,问题评论,拉取请求,拉取请求审阅评论模块 |
| issue_labels | Nested | 该仓库中当前问题的所属标签 | 问题,问题评论,拉取请求,拉取请求审阅评论模块 |
| issue_labels.name | String | 问题标签名称 | 问题,问题评论,拉取请求,拉取请求审阅评论模块 |
| issue_labels.color | String | 问题标签颜色 | 问题,问题评论,拉取请求,拉取请求审阅评论模块 |
| issue_labels.default | UInt8 | 是否为默认问题标签 | 问题,问题评论,拉取请求,拉取请求审阅评论模块 |

为了进行数据分析,本文首先确定了数据分析的用户人数和筛选条件。为了排除在 GitHub 上无任何活动的用户,本文只选取了在 GitHub 上创建或管理过仓库的用户,共计 317 200 名开发者。

### 5.2　开发者价值等级分布情况

采用本文评估体系,分析得到了不同价值等级的开发者人群分布情况,具有如表 8、表 9 所列。

表 8　不同等级的人数分布占比情况

Table 8　Distribution of people at different levels

| 等级 | 人数 | 所占比例/% |
|---|---|---|
| S 级 | 13 | 0.006 |
| A 级 | 4232 | 1.333 |
| B 级 | 25869 | 8.151 |
| C 级 | 74154 | 23.366 |
| D 级 | 213082 | 67.144 |

由此可知,不同价值等级和不同价值分数段的人数分布呈金字塔形,与现实中开发者群体的分布特征相符,即普通或低价值的开发者占大多数,中高价值的开发者占极少数。同时,根据不同分数段的人数分布占比情况分析可以认定,开发者价值评估结果高于 50 的可被认定为优秀的开发者,等级处于 S,A 级的开发者可被认定为顶级开发者。

表 9　不同分数段的人数分布占比情况

Table 9　Distribution of people across different score ranges

| 分数段 | 人数 | 所占比例/% |
|---|---|---|
| 0～10 | 42887 | 13.514 |
| 10～20 | 170195 | 53.630 |
| 20～30 | 48053 | 15.141 |
| 30～40 | 26101 | 8.224 |
| 40～50 | 13684 | 4.311 |
| 50～60 | 12185 | 3.839 |
| 60～70 | 3412 | 1.075 |
| 70～80 | 820 | 0.260 |
| 80～90 | 13 | 0.006 |
| 90～100 | 0 | 0 |

### 5.3　开发者价值分析

#### 5.3.1　开发者群体的价值分析

为了证明开发者价值评估体系的合理性和有效性,本节将从不同等级的开发者中挑选部分样本,分析他们在 GitHub 上的贡献和工作情况,验证他们的工作量和能力是否与本文评估方法的结果相符,如表 10 和表 11 所列。

表 10　不同等级的开发者能力评估

Table 10　Evaluation of developer capabilities at different levels

| 开发者 | 编程能力(得分/等级) | 项目管理能力(得分/等级) | 敬业度(得分/等级) | 学习能力(得分/等级) | 团队协作能力(得分/等级) | 价值评估总分(得分/等级) |
|---|---|---|---|---|---|---|
| DanySK | 91.72/S | 100/S | 100/S | 90/S | 15/D | 84.10/S |
| bgruening | 90.25/S | 100/S | 100/S | 80/A | 34/C | 83.75/S |
| ardalis | 95.68/S | 100/S | 95/S | 60/B | 65/A | 82.25/S |
| sindresorhus | 92.23/S | 100/S | 85/S | 80/A | 39/C | 81.25/S |
| peterbe | 92.91/S | 100/S | 95/S | 65/B | 30/C | 78.58/A |
| miyacorata | 50.49/B | 100/S | 70/A | 10/D | 15/D | 50.2235/B |
| kuroni | 65.76/A | 100/S | 60/B | 10/D | 15/D | 50.2140/B |
| 0x01x0 | 10.00/D | 40/C | 50/B | 60/B | 20/D | 40.0000/C |
| N1cholas | 36.42/C | 40/C | 20/D | 10/D | 20/D | 24.0630/C |
| suyukun666 | 36.72/C | 20/D | 20/D | 10/D | 15/D | 19.0580/D |
| soracom | 38.31/C | 20/D | 0/D | 10/D | 0/D | 12.7465/D |

表 11 不同等级的开发者基本信息

Table 11 Basic information of developers at different levels

| 开发者 | GitHub 管理仓库数量(单位/个) | GitHub 跟随者(单位/人) | 2020 年 GitHub 贡献次数(单位/次) | 2020 年活跃天数 | GitHub 活跃年限 | 个人履历 |
|---|---|---|---|---|---|---|
| DanySK | 215 | 125 | 16843 | 337 | 12 | 在 GitHub 存档计划里贡献了多个仓库的代码,拥有 GitHub 颁发的勋章、博洛尼亚大学博士后研究员 |
| bgruening | 365 | 463 | 5275 | 361 | 13 | 在 GitHub 存档计划里贡献了多个仓库的代码,拥有 GitHub 颁发的勋章 |
| ardalis | 243 | 7100 | 3221 | 285 | 12 | 在 Github 上有 7100 个开发者关注,存档计划里贡献了多个仓库的代码,拥有 GitHub 颁发的勋章。获得 20 次 Microsoft MVP 奖项 |
| sindresorhus | 1100 | 57100 | 3338 | 363 | 14 | 在 Github 上有 57000 个开发者关注,参加了多个项目与软件的开发如 Swift 和 JavaScript、制作 macOS 应用程序、CLI 工具、npm 软件包 |
| peterbe | 268 | 987 | 3338 | 313 | 15 | GitHub 的员工,在 GitHub 存档计划里贡献了多个仓库的代码,拥有 GitHub 颁发的勋章,参与了 GitHub 平台初始的开发 |
| miyacorata | 33 | 41 | 705 | 164 | 9 | 在 GitHub 存档计划里贡献了多个仓库的代码,拥有 GitHub 颁发的勋章 |
| kuroni | 22 | 272 | 617 | 111 | 9 | 在 GitHub 存档计划里贡献了多个仓库的代码,拥有 GitHub 颁发的勋章 |
| 0x01x0 | 72 | 6 | 40 | 113 | 4 | 在 GitHub 存档计划里贡献了多个仓库的代码,拥有 GitHub 颁发的勋章 |
| N1cholas | 15 | 80 | 78 | 57 | 8 | 普通开发者 |
| suyukun666 | 5 | 19 | 12 | 13 | 7 | 普通开发者 |
| soracom | 32 | 9 | 28 | 5 | 10 | 普通开发者 |

通过分析发现,在价值评估体系中得分较高的开发者具有以下特点:在 GitHub 上管理的仓库数量多,2020 年在 GitHub 上的贡献次数多,活跃天数多,工作时间长。此外,这些开发者都参与了 GitHub 存档计划的项目,受到许多开发者的关注,加入了知名的开源组织,参与了知名软件或项目的开发。这些特点表明这些开发者的各项能力均超越了大部分开发者。相反,在价值评估体系中得分较低的开发者在这些指标上的表现都较差,工作量也较小,并且参与的优秀项目极少或几乎没有。

本节分析能够证明价值评估模型的有效性和合理性,该模型能够很好地区分优秀开发者和普通开发者。

### 5.3.2 GitHub 著名开发者的价值分析

为了验证本文评估体系能够准确地评判不同开发者的价值,并进一步检验本文方法的有效性,本节将挑选出一些知名网站中排名靠前的开发者和热门项目中贡献排名第一的开发者,然后通过本文提出的模型对这些开发者的各项能力评分和等级进行计算和分析。

根据 GitHub 统计的 2020 年比较受欢迎的开源项目,本文参考了 GitHub 中文社区[1)]的数据,选取了 5 个热门项目及其中贡献排名第一的开发者(见表 12)和该网站制作的开发者排名榜中前 100 名的开发者(见表 13)。

表 12 5 个热门项目的开发者能力评估

Table 12 Developer competency assessment of five popular projects

| 项目名称 | 项目开发者 | 编程能力(得分/等级) | 项目管理能力(得分/等级) | 敬业度(得分/等级) | 学习能力(得分/等级) | 团队协作能力(得分/等级) | 价值评估总分(得分/等级) |
|---|---|---|---|---|---|---|---|
| freeCodeCamp | raisedadead | 81.58/S | 100/S | 90/S | 50/B | 10/D | 69.2370/A |
| react | gaearon | 85.22/S | 100/S | 90/S | 10/D | 30/C | 61.9830/A |
| vue | yyx990803 | 80.52/S | 100/S | 90/S | 10/D | 15/D | 59.3280/B |
| TensorFlow | yongtang | 51.56/B | 100/S | 85/S | 50/B | 20/D | 64.8840/A |
| Bootstrap | mdo | 84.37/SS | 100/S | 85/S | 50/B | 25/C | 70.4555/A |

表 13 排名前 100 的开发者能力评估

Table 13 Ability assessment of top 100 developers

| GitHub 开发者 | 编程能力(得分/等级) | 项目管理能力(得分/等级) | 敬业度(得分/等级) | 学习能力(得分/等级) | 团队协作能力(得分/等级) | 价值评估总分(得分/等级) |
|---|---|---|---|---|---|---|
| JakeWharton | 92.20/S | 100/S | 90/S | 65/A | 25/C | 76.6800/A |
| ruanyf | 77.90/A | 100/S | 80/A | 50/B | 10/D | 65.0850/A |
| tj | 71.93/A | 100/S | 70/A | 50/B | 25/C | 65.1395/A |
| YYX990803 | 80.52/S | 100/S | 90/S | 10/D | 15/D | 59.3280/B |
| gaearon | 85.22/S | 100/S | 90/S | 10/D | 30/C | 61.9830/A |
| addyosmani | 70.06/A | 100/S | 60/B | 10/D | 40/C | 54.1090/B |
| paulirish | 87.81/S | 100/S | 85/S | 50/B | 15/D | 69.6715/A |
| michaelliao | 74.51/A | 100/S | 80/A | 10/D | 10/D | 54.1765/B |
| Mdo | 84.37/S | 100/S | 85/S | 50/B | 25/C | 70.4555/A |
| mrdoob | 86.07/S | 100/S | 95/S | 10/D | 14/D | 61.1805/A |

1) github-zh.com

由表 11 的结果可知，热门项目以及其中贡献排名第一的开发者在项目管理能力、编程能力和敬业度上都表现出色，价值评估总分都超过了 50 分，等级都达到了 B 级或更高。按照本文的定义，这些开发者都属于优秀开发者，他们的项目管理能力以及自身的技术能力都很强。由此可以看出这些项目能够受到大众的关注和支持与这些开发者所做的贡献密不可分。但从本文方法的角度来看，他们在团队协作能力和学习能力上的表现相对较弱。可以认为，这些优秀开发者可能更加专注某个技术领域，做了大量的原始代码贡献[21]，极可能是开源软件项目创始人之一。同时，由于这些优秀项目在发布开源时已经经过了严谨的测试，相对比较稳定和成熟，在开源社区中被发现的问题较少，因此开发者的协作交互行为也较少。

由表 12 的结果可知，这些知名网站中被大众所认可的开发者在基于本文的模型进行评分后，依然能够达到本文所定义的优秀开发者的标准，且由于本文模型指标众多，考虑范围较广，相比传统网站上的评选方式，例如按照粉丝数量排名，更加具有说服力，且更加精准。

本节的分析证明，通过开发者价值评估模型计算出的排名不仅与大众认可的排行榜相吻合，且由于模型里包含的指标和计算的权重方法更优秀，因此能够更加准确地评选出能力更强的开发者。

### 5.4　指标相关性分析

在对开发者价值进行分析时发现，这些优秀开发者的 5 项一级指标并不都是名列前茅。因此，本节将会对价值评估体系的每个一级指标与综合目标的相关性进行分析[22]。确定哪些指标和综合目标的相关性较高，更能影响开发者的价值排名。

本文采用 Spearman 相关系数来计算和分析同级指标之间以及不同级指标之间的相关性。Spearman 系数检验能够检测两个变量之间的依赖性，利用单调方程来评价变量之间的相关性。要想更精确地描述变量间的相关关系，就要计算相关关系的相关系数。计算相关系数一般需要大样本，样本容量最好大于 30 个，这样才能比较准确反映两个变量之间的关系。相关系数 $r$ 的取值一般介于 $-1\sim1$ 之间。表 14 列出了相关系数 $r$ 取不同值时对应的相关性。

表 14　相关系数 $r$ 取不同值时对应的相关性

Table 14　Correlation corresponding to correlation coefficient $r$

| 相关系数 | $0\leqslant\lvert r\rvert<0.3$ | $0.3\leqslant\lvert r\rvert\leqslant0.7$ | $0.7<\lvert r\rvert<1$ | $\lvert r\rvert=1$ |
|---|---|---|---|---|
| 相关性 | 弱相关 | 中等相关 | 强相关 | 完全相关 |

为了评估 Spearman 系数的显著性，计算 $p$ 值来分析结果是否显著有效。使用不同个数的 * 来表示显著性值的大小等级，如表 15 所列。

表 15　显著性值的大小等级分布

Table 15　Distribution of significance levels

| * 数量 | ** | * | * =0 |
|---|---|---|---|
| $p$ 值 | $p<0.01$ | $p<0.5$ | $p>0.5$ |
| 显著性 | 有显著统计学差异 | 有统计学的差异 | 缺乏统计学差异 |

得到价值评估体系后需要基于这个体系对开发者进行评分，分析的具体情况有助于找到优秀开发者之间的共同之处，同时也能检验价值评估体系的合理性。

1）指标编程能力的相关性分析

由表 16 可以看出，编程能力与项目管理能力、敬业度、总分数都呈现出中等正相关，而与团队协作能力、学习能力都呈现出弱相关。这意味着一个开发者的编程能力越强，他的项目管理能力和敬业度也越强，而他的团队协作能力和学习能力并不一定强。另一方面，这也说明价值评估总分较高的开发者往往具有较强的编程能力。

表 16　编程能力的相关性分析

Table 16　Correlation analysis of programming ability

| 指标 | 相关系数 | 相关性 |
|---|---|---|
| 项目管理能力 | 0.487 ** | 中等正相关 |
| 学习能力 | 0.241 ** | 弱相关 |
| 团队协作能力 | 0.121 ** | 弱相关 |
| 敬业度 | 0.578 ** | 中等正相关 |
| 价值评估总分 | 0.659 ** | 中等正相关 |

2）指标项目管理能力的相关性分析

由表 17 可以看出，项目管理能力与编程能力呈现出弱相关，与敬业度、价值评估总分呈现出中等正相关，而与学习能力、团队协作能力呈现出弱相关。这意味着一个开发者的项目管理能力越强，他的敬业度和总分数也越高，而他的编程能力、学习能力和团队协作能力并不一定强。另一方面，这也说明一个好的项目管理者不一定是最擅长编写代码的那一个，但是他的工作量一定较大，同时价值评估总分较高的开发者往往具有较强的项目管理能力。

表 17　项目管理能力的相关性分析

Table 17　Correlation analysis of project management capability

| 指标 | 相关系数 | 相关性 |
|---|---|---|
| 编程能力 | 0.225 ** | 弱相关 |
| 学习能力 | 0.086 ** | 弱相关 |
| 团队协作能力 | 0.076 ** | 弱相关 |
| 敬业度 | 0.532 ** | 中等正相关 |
| 价值评估总分 | 0.559 ** | 中等正相关 |

3）指标敬业度的相关性分析

由表 18 可以看出，敬业度与项目管理能力呈现出中等正相关，这可能与这两个一级指标中某些二级指标比较相似有关，例如活跃天数和更新版本数量等。同时，这也很符合现实的情况，一个开发者在开源社区做的工作越多，他管理或参与的仓库的质量也会越高。此外，敬业度与价值评估总分也呈现出中等正相关，表明价值评估总分较高的开发者往往具有较高的敬业度。

表 18　敬业度的相关性分析

Table 18　Correlation analysis of dedication

| 指标 | 相关系数 | 相关性 |
|---|---|---|
| 项目管理能力 | 0.338 ** | 中等正相关 |
| 学习能力 | 0.089 ** | 弱相关 |
| 团队协作能力 | 0.051 ** | 弱相关 |
| 编程能力 | 0.150 ** | 弱相关 |
| 价值评估总分 | 0.403 ** | 中等正相关 |

4）指标学习能力的相关性分析

由表 19 可以看出，学习能力与编程能力、项目管理能力

呈现出弱相关，而与团队协作能力、敬业度呈现出极低相关。这意味着一个开发者的学习能力越强，他的编程能力和项目管理能力并不一定强，而他的团队协作能力和敬业度之间也没有明显的关系。另一方面，学习能力与价值评估总分呈现出中等正相关，但是相关系数不高，表明价值评估总分较高的开发者往往具有较强的学习能力，但是这并不是决定性因素。

表 19 学习能力的相关性分析

Table 19 Correlation analysis of learning ability

| 指标 | 相关系数 | 相关性 |
|---|---|---|
| 项目管理能力 | 0.208 ** | 弱相关 |
| 编程能力 | 0.189 ** | 弱相关 |
| 团队协作能力 | −0.098 * | 弱相关 |
| 敬业度 | −0.095 * | 弱相关 |
| 价值评估总分 | 0.378 ** | 中等正相关 |

5)指标团队协作能力的相关性分析

由表 20 可以看出，团队协作能力与其他 4 项指标以及价值评估总分都呈现出极低相关。这意味着一个开发者的团队协作能力与他的其他能力和价值评估总分没有明显的关系，不能作为评判优秀开发者的标准，几乎不会对价值评估总分产生较大的影响。

表 20 团队协作能力的相关性分析

Table 20 Correlation analysis of teamwork ability

| 指标 | 相关系数 | 相关性 |
|---|---|---|
| 项目管理能力 | −0.014 | 弱相关 |
| 学习能力 | −0.016 | 弱相关 |
| 编程能力 | −0.016 | 弱相关 |
| 敬业度 | −0.030 | 弱相关 |
| 价值评估总分 | 0.136 ** | 弱相关 |

6)指标价值评估的相关性分析

由表 21 可以看出，价值评估总分与项目管理能力、学习能力、编程能力、敬业度都呈中等正相关，而与团队协作能力呈现出弱相关。这与表 15－表 19 的实验结果相符。为了检验相关性结果的显著性，本文还计算了 $p$ 值。从表 20 中可以看出，除了团队协作能力以外，其他指标的 $p$ 值都小于 0.01，说明结果具有显著的统计学意义。

表 21 价值评估总分的相关性分析

Table 21 Correlation analysis of total score valuation

| 指标 | 相关系数 | 相关性 |
|---|---|---|
| 项目管理能力 | 0.358 ** | 中等正相关 |
| 学习能力 | 0.434* * | 中等正相关 |
| 编程能力 | 0.475 ** | 中等正相关 |
| 敬业度 | 0.364 ** | 中等正相关 |
| 团队协作能力 | 0.099 ** | 弱相关 |

根据以上一级指标和价值评估总分的相关性分析，本文得出以下结论：项目管理能力、编程能力、敬业度这 3 项一级指标相互相关，并且对价值评估总分影响较大。一个优秀的开发者在这 3 项指标上相对会表现出色。学习能力与价值评估总分的相关性也较高，说明学习能力也会影响开发者的价值评估。但是通过比较学习能力的相关性结果发现，学习能力对价值评估总分的影响不如项目管理能力、编程能力和敬业度。而团队协作能力与其他所有指标一级价值评估总分的相关性都很低，因此团队协作能力的高低可能无法代表一个开发者是否优秀，这一点与本文前面的实验结果相符，价值评估得分最高的开发者并不一定是团队协作能力最强的那一个。

因此，本文提出的开发者价值评估模型能够较为合理地区分开发者群体的能力水平，能够为公司猎头和开发者提供一个较为有效的模型，帮助其寻找符合条件的开发者。

### 5.5 有效性分析

为了保证实验结果的有效性和普适性，本文将从以下 3 个方面进行分析：原始数据的获取与预处理、核心算法的正确性、研究方法与结论的普适性。

首先，在数据处理阶段，对 GitHub2020 年全域日志数据进行了合理的筛选和汇总，以保证数据的完整性和准确性，避免在数据的选择上存在偏见。

其次，在算法方面，采用的 Critic 方法是一种常用的确定权重的方法，该方法通过计算各个指标的标准差和相关系数，来反映各个指标在决策中的重要性。该方法的有效性已经被众多实验所证明，也适用于本文的指标权重计算。因此，本文通过 Critic 权重分析方法得到的计算结果，能够有效地降低分析结果的主观性。

最后，在普适性方面，在 GitHub 上选取了一年的数据作为数据集。GitHub 是世界上最大的开源社区之一，其开发者群体极具代表性，因此本文结果能够推广到其他的开源社区。但是，由于选取的数据仅为一年的数据，对于部分开发者的评价可能会有些许偏差，未来可以考虑统计更多年份的开发者事件数据，以增强结论的普适性和可信度。

**结束语** 本文旨在提出一种开源社区开发者价值评估体系，以科学、合理、客观地评估开源软件开发者的价值和能力。通过对开发者的项目管理能力、编程能力、学习能力、团队协作能力和敬业度 5 个维度的细分度量评价，本文方法可以较客观、全面地评估开发者的价值和能力，为开源软件人才的培养、发现和管理提供一种数据性、科学性和操作性较强的衡量方法，从而为新一代“开源协作”软件开发模式的应用和普及奠定基础。

然而，本文研究也存在一些局限性。例如，评估指标的选择和定义可能因项目特性和开源社区的相异性而有所变化[23]，需要在具体应用中进行调整和优化。未来的研究将探索更多的指标和影响因素，以适应不同类型的开源项目和开发者。此外，还可以结合机器学习和数据挖掘技术，挖掘更多的隐含信息，并提供个性化的价值评估和推荐服务。

总之，开源软件开发者的价值评估对于促进开放式软件协作和开源生态发展至关重要。随着开源社区的不断发展和演进，相信这个领域还有许多值得探索和研究的问题。本文工作只是一个开端，期待更多的研究者和开发者共同努力，为开源社区的繁荣做出更大的贡献。

## 参 考 文 献

[1] LI R N, TANG C. Analysis of influencing factors of open source hardware patent value and construction of evaluation index system[J]. China Invention & Patent(Journal of Intellectual Property Information Science), 2022, 8: 15-24.

[2] WU Z F, ZHU T T, XUAN Q, et al. Evaluation of Core Developers in Open Source Software by Contribution Allocation[J]. Journal of Software, 2018, 29(8): 2272-2282.

[3] TANG J J, CAO Y Z, ZHU J W, et al. Human resource value prediction of open source community software developers based on hybrid neural network[J]. Computer Applications and Software, 2021, 38(8): 64-77.

[4] YANG B, YU Q, ZHANG W, et al. Influence factors correlation analysis in GitHub open source software development process [J]. Ruan Jian Xue Bao/Journal of Software, 2017, 28(6): 1330-1342.

[5] OLIVA G. Characterizing key developers: A case study with apache ant[C]// Proceedings of the International Conference on Collaboration and Technology. Springer-Verlag, 2012: 97-112.

[6] LI Z X, YU Y, WANG T, et al. Empirical Study on Pull-request Revisions in Open Source Software Community of TensorFlow [J]. Journal of Software, 2023, 34(9): 1-13.

[7] YUAN S, TANG J, GU X T. A Survey on Scholar Profiling Techniques in the Open Internet[J]. Journal of Computer Research and Development, 2018, 55(9): 1903-1919.

[8] LIAO Z F, YANG H Y, SONG T H, et al. Developer Project Recommendation Model Based on CNN-LSTM in GitHub[J]. Acta Electonica Sinica, 2020, 48(11): 2202-2207.

[9] JIANG J, WU Q D, ZHANG L. Open Source Community Review Process Measurement System and Its Empirical Research [J]. Journal of Software, 2021, 32(12): 3698-3709.

[10] LEI J, YE H J, WU Z S, et al. Big-Data Platform Based on Open Source Ecosystem[J]. Journalof Computer Research and Development, 2017, 54(1): 80-93.

[11] SOWE S K, STAMELOS I, ANGELIS L. Understanding knowledge sharing activities in free/open source software projects: Anempirical study[J]. Journal of Systemsand Software, 2008, 81(3): 431-446.

[12] NAKAKOJI K, YAMAMOTO Y, NISHINAKA Y, et al. Evolution patterns of open-source software systems and communities [C]// Proceedings of the 14th International Workshop on Principles of Software Evolution. 2002: 76-85.

[13] HUNTER P, WALLI S. The rise and evolution of the open source software foundation[J]. IFOSS L. Rev., 2013, 5: 31.

[14] CASALNUOVO C, VASILESCU B, DEVANBU P, et al. Developer onboarding in GitHub: the role of prior social links and language experience[C]// Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering. 2015: 817-828.

[15] YU Y, WANG H M, YIN G, et al. Reviewer recommendation for pull-requests in GitHub: What can we learn from code review and bug assignment[J]. Information and Software Technology, 2016, 74: 204-218.

[16] LENARDUZZI V, TAIBI D, TOSI D, et al. Open source software evaluation, selection, and adoption: a systematic literature review[C]// 2020 46th Euromicro Conference on Software Engineering and Advanced Applications(SEAA). IEEE, 2020: 437-444.

[17] MOCKUS A, FIELDING R T, HERBSLEB J D. Two case studies of open source software development: Apache and Mozilla [J]. ACM Trans. on Software Engineering and Methodology, 2002, 11(3): 309-346.

[18] YE Y, KISHIDA K. Toward an understanding of the motivation open source software developers[C]// Proceedings of the 25th International Conference on Software Engineering. Portland, 2003: 419-429.

[19] SEN R, SINGH S S, BORLE S. Open source software success: Measure and analysis [J]. Decision Support Systems, 2012, 52(2): 364-372.

[20] WANG T, YIN G, WANG H M, et al. Linking stack overflow to issue tracker for issue resolution[C]// Proceedings of the 6th Asia-Pacific Symposium on Internetware on Internetware. ACM Press, 2014: 11-14.

[21] CROWSTON K, WEI K, HOWISON J, et al. Free/Libre opensource software development: What we know and what we do not know[J]. ACM Computing Surveys(CSUR), 2012, 44(2): 7.

[22] WANG, Z D, YANG F, YI W, et al. Unveiling Elite Developers' Activities in Open Source Projects[J]. ACM Transactions on Software Engineering and Methodology (TOSEM), 2019, 29: 1-35.

[23] BOC K, THOMA S, NILS A, et al. Automatic Core-Developer Identification on GitHub: A Validation Study[J]. ACM Transactions on Software Engineering and Methodology, 2023, 32: 1-29.

**YOU Lan**, born in 1978, Ph.D, professor, is a senior member of CCF (No. H8967M). Her main research interests include spatiotemporal big data, natural language processing, and social computing.

**CHEN Zhijun**, born in 1977, professor, master's supervisor. His main research interests include artificial intelligence and databases.

（责任编辑：杨雪敏）