Software Engineering Group
Department of Computer Science
Nanjing University
http://seg.nju.edu.cn

Technical Report No. NJU-SEG-2026-CJ-003

2026-CJ-003

基于区块链的开放协作开发供应链可信管理平台

张硕骁，汤恩义，张浩枫，单一啸，潘佳滨，刘子豪，成浩亮，赵建华，

陈鑫，李宣东

Technical Report 2026

软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.000000]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于区块链的开放协作开发供应链可信管理平台*

张硕骁[1,2]， 汤恩义[1,2]， 张浩枫[1,2]， 单一啸[1,2]， 潘佳滨[1,2]， 刘子豪[1,2]， 成浩亮[1,2]， 赵建华[1,3]， 陈鑫[1,3]， 李宣东[1,3]

[1](计算机软件新技术全国重点实验室(南京大学)，江苏 南京 210023)

[2](南京大学软件学院，江苏 南京 210023)

[3](南京大学计算机学院，江苏 南京 210023)

通讯作者: 汤恩义，E-mail: eytang@nju.edu.cn

摘 要: 开放协作开发是重要的软件开发模式，在智能化基础软件的开发过程中得到了广泛应用。然而，该模式在软件供应链安全性方面存在显著隐患：一方面，协作开发过程中的收益分配可能遭受操纵篡改；另一方面，代码资产在被用户合法获取后，仍可能被非法转售或泄露。针对这些问题，本文提出了一种开放协作开发供应链可信管理平台，以区块链作为可信执行与可追溯的技术支撑，在保障过程可验证性的同时进一步提升安全性。该平台引入贡献点来量化用户在协作开发中的贡献，并通过智能合约依据贡献点的价值公平分配项目收益。此外，本文实现了基于智能合约的押金与信用积分安全机制，有效降低了协作开发过程的软件信息泄露与恶意行为风险。实验结果表明，本文平台能够有效保障开发者权益，显著提升开发者的开发积极性与响应速度，同时保持合理的开销水平，为软件供应链安全提供了有力保障。

关键词: 供应链安全;协作开发;贡献;激励;区块链;智能合约

中图法分类号: TP311

Blockchain-Based Trusted Management Platform for Open Collaborative Development Supply Chains

ZHANG Shuo-Xiao[1,2], TANG En-Yi[1,2], ZHANG Hao-Feng[1,2], SHAN Yi-Xiao[1,2], PAN Jia-Bin[1,2], LIU Zi-Hao[1,2], CHENG Hao-Liang[1,3], ZHAO Jian-Hua[1,3], CHEN Xin[1,3], LI Xuan-Dong[1,3]

[1](National Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210023, China)

[2](Software Institute, Nanjing University, Nanjing 210093, China)

[3](School of Computer Science and Technology, Nanjing University, Nanjing 210023, China)

Abstract: Open collaborative development is a prominent software engineering model that has been extensively applied in the development of intelligent foundational software. While this model offers substantial benefits in terms of efficiency, innovation, and resource integration, it also introduces notable risks to software supply chain security. On the one hand, the revenue distribution process in collaborative development may be subject to manipulation and tampering; on the other hand, code assets, even after being legitimately obtained by users, may still be illegally resold or leaked. To address these challenges, this paper proposes a trusted supply chain management platform for open collaborative development, supporting trusted execution and end-to-end traceability to ensure process verifiability and strengthen security. The platform introduces a contribution point mechanism to quantitatively assess individual user contributions and utilizes smart contracts to fairly allocate project revenues based on the value of these contributions. Furthermore, a security mechanism integrating deposits and credit scores is implemented via smart contracts to effectively mitigate the risks of information leakage and malicious behavior during the development process. Experimental results demonstrate that the proposed platform effectively protects developer rights, significantly enhances development motivation and responsiveness, and maintains a reasonable level of overhead, thereby providing a robust safeguard for software supply chain security.

Key words: supply chain security; collaborative development; contribution; incentive; blockchain; smart contract

* 基金项目: 国家重点研发计划(2024YFB2505604)

1 引言

开放协作开发作为一种重要的软件开发模式，已广泛应用于智能化基础软件的开发，尤其在提升开发效率、促进创新及资源共享方面展现了显著优势[1-2]。这一模式通过汇聚全球开发者的力量，推动了技术进步与产品迭代。然而，随着协作规模扩大与协作关系复杂化，开放协作开发日益面临协作可信性不足这一关键挑战。实践表明，当协作过程缺乏可信支撑、贡献回报关系不清晰或难以验证时，开发者难以确认贡献是否被正确记录、权益是否会被兑现，协作信任随之下降并削弱持续参与意愿，进而影响项目长期活跃度与可持续发展，已有大量潜力项目因此难以维系甚至中止[3]。从协作机制视角看，开放协作开发通常依赖捐赠与收益分配等方式激励持续投入，但在多主体参与场景中，若贡献与回报的映射关系缺乏透明性与可验证性，易引发对协作公平性的质疑并抑制参与积极性[4-5]。同时，开放协作开发涉及多角色与多阶段协作过程，协作决策与资源分配往往依赖分散的信息共享与主观判断，若缺乏统一且可信的协作规则支撑，将放大不确定性并降低对协作过程可预期性的认知[6-10]。开放协作开发中的恶意风险主要体现为两类：其一是收益分配层面的操纵行为，例如伪造贡献记录、操纵投票结果或干预收益结算，以非法获取项目收益；其二是代码资产层面的泄露行为，即用户在获得合法访问权限后对代码进行非法转售或泄露。前者直接破坏协作公平性与贡献回报映射关系，后者则损害项目产权。因此，如何同时抑制这两类恶意行为，构建兼顾公平分配与产权保护的可信协作机制，成为开放协作开发中必须解决的关键问题。

基于上述问题，本文提出了一种基于区块链的开放协作开发供应链可信管理平台[1]，旨在利用区块链与智能合约的可追溯、不可篡改、可验证执行特性，将贡献点、渐进式开放模型、押金与信用积分安全机制及仲裁模块等关键规则固化为可核验的协作协议，以降低争议与不确定性，并提升协作与收益分配的透明性与可执行性；相应地，本文在实验部分以开发积极性与响应效率作为协作可信提升后的行为响应指标，并通过更高的完成率与更快的响应增强审查与纠错能力，从而对供应链安全形成持续性反哺。

相比之下，传统中心化平台即便结合数据库与审计机制，平台方仍掌握贡献点计算、权限授予/回收、收益分配与惩戒触发等关键环节的控制权；而审计多为事后证明，难以从机制上彻底消除选择性执行、内部修改及争议难举证等问题。本文平台则依托链上数据的公开可验证性，使任何参与方均可独立核验关键执行结果，从而将协作与激励由“依赖平台信用”转化为“依赖协议与共识的可验证执行”。因此，区块链在本文平台中具有不可代替性。

本文定义终端用户为真正认可软件内在价值并愿意为之投资的用户。本文平台通过所提出的渐进式开放模型，建立了终端用户与开发者之间直接关联的收益关系，双方基于明确的收益划分形成互惠的交易关系。具体的渐进式开放模型将在后续章节详细阐述。项目贡献者依据其贡献程度获得市场收益，并通过区块链技术保障收益分配过程的透明与可信。本文平台的渐进式开放模型在开放协作开发过程中保护了贡献者的贡献产权。该模型根据开发者的贡献点，逐步授予其对项目工件的访问权限。已有研究表明，开发者在项目中持有更高贡献点份额时，侵犯版权的意图会显著降低[11]。因此，本文平台根据贡献动态授予访问权限，而未取得足够贡献点的开发者则需通过租用访问权限完成任务，并在后续通过贡献赚取贡献点偿还。本文平台提出了一种新颖的方法，通过根据贡献比例将市场收益分配给项目贡献者，从而激励其持续参与开放协作软件项目。为衡量每位贡献者在项目中所拥有的产权份额，本文引入了贡献点的概念。贡献点用于表示贡献者在项目中获得的产权比例，且依据各项目预设的贡献点获取规则进行确定。项目所获得的市场收益将按照贡献点比例在贡献者之间进行分配。需要注意的是，不同项目彼此独立，一个项目中的贡献点不可在其他项目中通用或互换。此外，本文通过采用区块链技术，建立了透明可信的模型与安全机制。通过部署智能合约，本文平台能够依据贡献点自动计算并分配收益，消除了传统人工分配可能引入的偏差问题。同时，基于区块链的安全机制有效保护了贡献者的产权，进一步增强了整个开发过程的可信性。 表 1 对 BountySource[12]、Gitcoin[13]、OpenCollective[13]以及本文平台进行了对比分析，重点突出各平台在功能设计与特性方面的差异。

1 https://blocksop.com

BountySource 与 Gitcoin 作为软件开发众筹领域的先驱平台，分别代表了不同的发展方向：BountySource 主要通过对完成特定任务或目标的奖励来激励开发者；而 Gitcoin 则利用区块链技术，促进对开源项目的资金支持。OpenCollective 则致力于为开源项目及社区驱动型项目提供透明、协作式的资助机制。

本文平台在收益分配方面相较于 BountySource、Gitcoin 与 OpenCollective 的主要优势体现在其基于智能合约实现的公平性。智能合约能够依据实际贡献自动分配奖励，确保收益分配过程的透明与公正。而 BountySource、Gitcoin 与 OpenCollective 则依赖人工判断，易受主观决策或外部因素影响，存在潜在的不公平性。此外，本文平台基于链上可追溯记录与智能合约自动执行构建了恶意行为检测机制，能够实现对恶意活动的及时识别与追踪；相比之下，Gitcoin 与 OpenCollective 主要依赖社区反馈与人工审核，容易引发延迟与安全漏洞，而 BountySource 则缺乏有效的恶意行为检测系统。因此，本文平台在防范恶意行为方面展现出更高的效率。本文平台采用区块链作为可追溯、可验证的记录载体以强化项目的版权保护，并基于链上数据对代码所有权与版权归属进行验证与追踪；而 BountySource、Gitcoin 与 OpenCollective 在版权保护方面缺乏系统性机制。通过集成区块链智能合约，本文平台实现了资金流向的透明管理，投资者资金能够直接投入项目，并根据项目进展自动生成回报；相较之下，BountySource、Gitcoin 与 OpenCollective 主要依赖资助机制，投资回报路径更加间接且透明度不足。因此，本文平台为投资者提供了更加清晰、直接的回报路径。在访问控制方面，本文平台采用渐进式开放模型，确保敏感信息仅对授权用户开放，实现了细粒度的权限管理，而这一能力在 BountySource、Gitcoin 与 OpenCollective 中难以实现。此外，本文平台基于区块链构建了公开透明的工作流程，涵盖项目管理、资金流转与代码提交，整体流程的透明度远高于仍依赖后端管理、缺乏公开机制的 BountySource 与 OpenCollective。

表 1　BountySource、Gitcoin、OpenCollective 与本文平台功能对比分析

特点描述	BountySource	Gitcoin	OpenCollective	本文平台
公正分配收益	●	●	●	○
恶意行为检测	●	◑	◑	○
基于版权保护的安全机制	●	●	●	○
直接投资收益	●	●	●	○
渐进式开放访问权限	●	●	●	○
基于区块链的透明工作流程	●	◑	●	○

其中○表示该平台完整支持对应功能，●表示该平台尚不具备该功能，◑表示该平台仅部分支持该功能

本文的主要贡献如下：

- 本文提出了一种基于区块链的开放协作开发供应链可信管理平台，创新性地引入贡献点作为量化度量开发者协作价值的新指标，并基于贡献点设计了自动化、可追溯且防篡改的收益分配机制，我们在 RQ2 中进一步验证了该机制的有效性。
- 针对开放协作环境中多方参与、角色异构和安全性需求，本文在技术实现上解决了多个难点：(1) 设计了可扩展的多角色支持的智能合约结构，实现跨角色的自动化收益结算与争议仲裁；(2) 提出了基于代码水印的版权防护机制，将代码水印技术与区块链不可篡改特性相结合，实现代码版权的有效追溯；(3) 构建了押金与信用积分的联动惩戒策略，并由智能合约自动触发，提高了恶意行为检测与处置的实时性与公正性，我们在 RQ1 中进一步验证了上述技术的有效性。
- 通过与 BountySource 和 OpenCollective 的实验数据对比，证明了本文平台在提升开发积极性与响应速度方面优于 BountySource，且在吸引投资者关注方面优于 OpenCollective，我们在 RQ3 中进一步验证了上述本文平台的优势。
- 对比了本文平台与 Gitcoin 在性能开销方面的实验结果，结果表明本文平台在吞吐量与延迟方面略优于 Gitcoin，同时开销控制在合理范围内，我们在 RQ4 中进一步验证了本文平台相对 Gitcoin 的开销。

2 背景

区块链本质上是一种去中心化、分布式的点对点(P2P)网络结构，其中各节点地位平等。这种结构有效缓解了针对中心节点的网络攻击风险，并解决了单点故障问题。区块链以交易作为数据通信与资源交换的媒介，不仅作为数据存储的特殊数据库存在，同时也保障了数据的可信传输[14]。在实际应用中，区块链可实现对有价值资产的数字化存储与安全管理[15]。依托其防篡改、去中心化与可追溯等特性，区块链可为多种应用场景提供低风险、低成本的安全服务解决方案。

区块链的篡改防护能力来源于其独特的数据结构：每个区块包含一定数量的交易数据，按时间顺序依次连接成链[16]。若欲篡改某一区块数据，需同时重构其后所有区块。共识机制的引入使得修改多个区块的成本极为高昂，从而使数据篡改几乎不可行[17]。在基于工作量证明(Proof of Work, PoW)的区块链网络中，攻击者需掌握至少 51%的全网算力方能篡改数据[18]，但这种投入对持有大量算力的节点自身并无利益，因此进一步增强了区块链数据的可靠性。智能合约是一种以数字形式编码的承诺集合，用以描述参与方在特定条件下的自动执行约定[19]。智能合约支持可信、可追溯且不可篡改的交易，消除了对第三方中介机构的依赖，极大提升了交易效率与安全性[20]。

智能合约本质上是在区块链上运行的条件式代码，如图 1 所示，合约条款、触发条件与响应规则均在代码中事先定义。智能合约作为一份多方共同认可与签署的协作协议，随用户发起的交易一并提交，经由 P2P 网络传播，由矿工节点进行验证后存储于区块链的特定区块中。用户收到返回的合约地址及相关信息后，即可通过再次发起交易调用合约，系统内置的激励机制驱动矿工持续投入算力以完成交易验证工作。

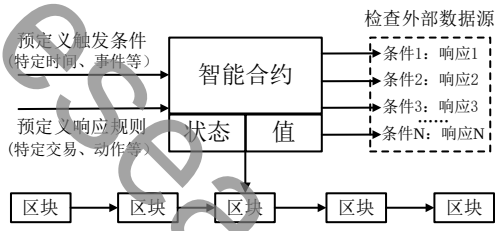


图 1 典型以太坊智能合约的结构

智能合约的可信性与不可篡改性主要源于以下两方面：首先，一旦智能合约部署到区块链网络，其代码与执行结果将被广播至整个网络，由所有节点共同存储与验证[21]。智能合约的执行过程在区块链上有序记录并附带时间戳，确保了信息的时间顺序与一致性。由于交易记录公开透明，任何人均可审阅与验证合约执行情况，从而进一步增强了智能合约的可信度。其次，在 PoW 等共识机制下，篡改已部署智能合约需掌控超过 50%的全网算力。获取如此大规模的计算资源成本极其高昂，几乎不可实现。因此，一旦智能合约部署至区块链，即天然具备不可篡改性，保障了合约内容与执行结果的完整性与可信性。

智能合约作为部署于区块链之上的逻辑执行层，以可编程代码形式承载平台的核心业务规则，涵盖贡献点获取、贡献点使用、收益分配与仲裁触发等流程；区块链网络则作为底层可信支撑层，依托分布式账本、共识机制与不可篡改存储，为上述规则执行提供公开可验证的运行环境以及全流程可追溯的数据保障。 二者的区别在于，前者侧重于“规则定义与自动执行”，后者侧重于“结果记录、共识确认与可信保”； 二者的联系在于，智能合约依赖区块链网络完成部署、执行与状态上链，而区块链网络通过对合约代码及执行结果的共同存储与验证，为平台规则执行、状态记录与结果验证提供可信支撑。

3 方法

3.1 本文平台的主要工作流程

本平台的主要参与方包括：用户、投资者、获得贡献点模块、贡献点使用模块、仲裁模块以及区块链网络。在安全性假设方面，我们设定如下前提：

(1) 大部分用户与投资者在完成平台身份认证后被视为诚实参与者，即不会主动实施恶意攻击；同时，假设少部分用户与投资者可能具备潜在的恶意意图，存在蓄意发起攻击的可能性；

(2) 获得贡献点模块与贡献点使用模块基于智能合约实现，其逻辑执行结果公开透明，并受区块链共识机制保障，任何单一方均无法篡改运行结果；

(3) 仲裁模块具备独立性，裁决过程依托链上证据与多方签名机制，确保结果公正、可追溯且不可抵赖；

(4) 区块链网络假设大多数节点为诚实节点，通过加密机制与共识协议防止数据被篡改、伪造或拒绝服务攻击，并保障交易与存储的机密性与完整性。

如图 2 所示，本文平台的主要工作流程由三个核心模块组成，分别为获得贡献点模块、贡献点使用模块与仲裁模块。本平台旨在确保所有软件项目的贡献者，无论是开发者还是投资者，均能够以透明、公平的方式共享项目的所有权与收益。

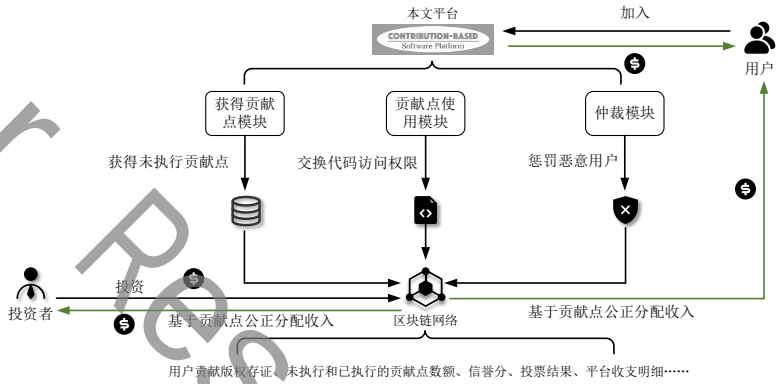


图 2 本文平台的工作流程

在本文提出的机制中，潜在用户可通过投资或完成技术任务申请加入项目。申请成功后，系统激活获得贡献点模块，要求用户通过对项目的持续投入获得未执行贡献点。贡献点既可以通过资金投资获得，也可以通过技术贡献获取。当用户成功积累了一定数量的未执行贡献点后，贡献点使用模块随之激活，用户可使用这些未执行贡献点兑换更高权限的代码访问权限，进一步将其转化为正式的贡献点。

若在开发过程中发生恶意行为，项目所有者可调用仲裁模块。仲裁模块将从高信用用户数据库中随机抽取若干用户组成评审团，并通过投票方式判定恶意行为的真实性。若评审团确认存在恶意行为，系统将对恶意用户进行惩罚，例如扣减其信用积分等措施。

此外，平台将用户的贡献版权信息、未执行与已执行贡献点数量、信用积分、投票结果、平台收益与支出等关键信息全部记录在区块链网络中，确保整个开发与收益分配流程的公平性与透明性。最终，区块链网络将依据用户所持贡献点的比例，透明、公平地完成收益分配。

若不使用区块链，上述机制将产生可明确界定的损失。首先，贡献点、权限变更、收益分配与惩戒记录等关键数据由平台单点维护，平台内部人员或入侵者一旦获得数据库权限，便可能对历史记录进行事后修改、回滚或选择性执行；而审计通常为事后追溯，难以从机制上消除内部篡改空间，争议举证亦更为困难。相比之下，区块链账本在共识机制约束下具备不可篡改性，篡改成本显著抬升；以典型 PoW 为例，攻击者需掌握 51% 算力方可能重写链上历史，从而有效提高恶意行为门槛。其次，在中心化执行模式下，收益结算与惩戒触发不可避免存在平台自由裁量与人为偏差空间；本文通过智能合约依据贡献点自动计算并分配收益、自动执行惩戒规则，减少人工干预与分配偏差，提升过程透明性与结果公正性，并将关键信息链上记录以保障流程公平与透明。最后，中心化平台天然面临单点故障与单点被攻破风险；智能合约部署后，其代码与执行结果由全网节点共同存储与验证，任何参与方均可基于链上数据独立核验，且事后篡改代价极高，从而增强系统鲁棒性与可验证性。

3.2 渐进式开放模型

本文提出的基于区块链的开放协作开发供应链可信管理平台，旨在通过促进开发过程的透明性与可持续性，改善当前的软件开发生态。该创新范式基于这样一种基本理念：有价值的开源软件尤其需要终端用户的直接资助[22]。传统开源融资模式中，贡献记录与收益回报之间往往缺乏清晰、可验证的对应关系，导致开发者难以准确判断自身的贡献投入能够获得何种回报，进而削弱其持续参与意愿[23]。鉴于软件开发行业本质上以创新与客户需求为驱动[24]，本文认为，构建以用户为中心的资助模式能够显著提升用户满意度，促进积极口碑传播，推动产品采用，从而形成开发生产率持续提升的良性循环。

然而，实现开放性与资金可持续性的双重目标仍然面临严峻挑战[25]。为此，本文提出了渐进式开放模型。该模型通过在开放协作与融资需求之间寻求平衡，优化协作开发过程，建立了一套基于权限控制的可行融资框架。具体而言，渐进式开放模型在执行层面由智能合约模块与仲裁模块共同负责。智能合约模块通过预定义规则对贡献者角色进行动态识别和权限分配，确保访问权限严格遵循“最小必要”原则，仅开放完成当前贡献所需的组件，从而在机制上降低项目敏感信息的外泄风险。同时，仲裁模块在权限授予和回收过程中承担监督与裁决职能，对存在越权或违规访问行为的贡献者实施限制或惩罚，保证执行过程的透明性与可追溯性。在具体方式上，访问权限与贡献点的绑定关系通过链上记录加以确权，贡献点的增减由贡献行为自动触发，且与项目收益挂钩，实现了从激励源头约束高权限角色的滥用行为。

为确保划分方式的合理性，系统采用智能合约驱动的访问控制机制，将权限分配规则以链上代码形式固化，从根本上避免人为干预带来的偏差。与此同时，权限调整通过贡献点量化与历史交互记录的动态校正实现，以确保访问范围与实际贡献能力的精确匹配。所有权限的授予与变更均在社区仲裁机制的透明监督下完成，并在区块链上形成可追溯与可验证的记录，从而有效保障其公正性与公信力。

如图 3 所示，在本文平台中，贡献者根据贡献点数量可分为高贡献点贡献者与低贡献点贡献者两类。渐进式开放访问模型不仅依据贡献点数对用户进行差异化访问控制，还引入了租赁型权限机制。对于贡献点数较低的参与者，平台允许其通过租赁方式临时获取部分项目细节的访问权限，以便其能够在理解项目整体架构的基础上完成后续贡献。该租赁权限在任务完成后需自动归还，从而避免长期暴露敏感信息的风险。相比之下，高贡献点数的贡献者可通过直接兑换贡献点的方式，获得项目整体内容的全面访问权限。该机制在安全性与灵活性之间实现平衡，既保障了低贡献者的学习与参与机会，又确保了高贡献者在项目中的主导性和激励效果。

本文平台的租赁访问通过多重约束协同抬升恶意获取与逃逸代价。一方面，平台由智能合约遵循最小必要原则，仅开放完成当前贡献所需组件，并由仲裁模块对权限授予与回收进行监督，所有权限变更均链上留痕、可追溯，从而降低敏感信息的长期暴露风险。另一方面，低贡献者需以实际贡献价值证明访问请求的合理性，项目所有者将结合申请权限范围与潜在风险进行集体投票审批，且租赁权限在任务完成后自动归还。与此同时，平台通过押金与信用积分安全机制明确设定租赁代价：访问者需缴纳押金并维持有效期，若发生泄露且经水印验证成立，押金将被扣除并同步扣减信用积分，从而避免恶意用户通过极低成本获取核心代码后逃逸。

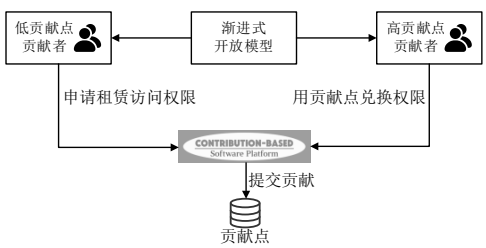


图 3 不同贡献者通过渐进开放模型参与项目的过程

该模型特别强调依据个体贡献分配项目所有权，贡献越大，所拥有的项目产权份额越高。贡献点份额赋

予贡献者三项核心权益：项目收益分成、在重要项目决策中的投票权以及对更多核心与外围模块的访问权。通过将收益与决策权与贡献紧密绑定，高贡献点贡献者由于与项目收益紧密挂钩，不会轻易泄露项目资源，从而有效遏制了高贡献点贡献者泄露敏感项目信息的动机，缓解了开放协作中普遍存在的信任问题。

在涉及低贡献点贡献者的开放协作中，低贡献点贡献者仅需访问其完成任务所必需的部分组件。为继承开源精神，渐进式开放模型引入了“租赁访问”机制，低贡献点贡献者需通过实际贡献价值向项目决策者证明其访问请求的合理性。项目决策者由项目创建者、优秀开发者与杰出投资人组成，他们将综合考量贡献的有效性、所申请访问权限的范围及潜在风险，通过集体投票方式决定是否授予访问权限。该流程强化了项目所有者在协作过程中的关键作用，确保权限分配的科学性与公正性。此外，模型规定贡献者可通过完成任务后获得的贡献点归还租赁访问权限。具体而言，贡献点在初始阶段为“未执行状态”，可用于兑换具体代码访问权限；完成访问兑换后转为“已执行状态”，仅可用于参与项目收益分成及决策过程。低贡献点贡献者通过完成任务，将未执行贡献点转化为已执行状态，完成租赁访问权限的偿还；高贡献点贡献者则可通过积累的未执行贡献点兑换新的访问权限。该机制在保障开放协作访问灵活性的同时，有效维护了项目安全性与资金可持续性，促进了协作开发的健康有序发展。

3.3 获得贡献点模块

图 4 展示了用户可通过两种方式参与项目。第一种方式是投资。用户可通过购买交易合约投资于平台上的软件项目。购买成功后，用户将正式加入项目并获得相应数量的未执行贡献点。投资过程不仅使用户能够获得更多关于软件项目的代码访问权限，也使得来自开放世界、虽无法直接进行技术贡献但对项目感兴趣的用户，能够通过支付投资费用获取更多信息。所投资金亦将依据各贡献者在项目中的贡献点比例进行收入分配，且该过程通过智能合约自动执行，以确保收益分配的公平性与透明性。

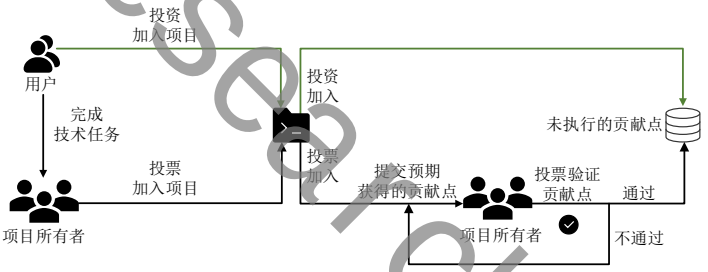


图 4　获得贡献点模块的工作流程

第二种方式是完成技术任务。当用户提交已完成的技术任务后，平台将通过投票合约由项目所有者对技术任务进行审核，并评估该用户是否具备正式加入项目的资格。一旦用户成功加入项目，需继续进行新的贡献，并提交预期获得的贡献点。若用户获得超过 80%的项目所有者支持票数，则可直接获得预期的贡献点；若仅获得 60%以上但不足 80%的支持票数，则原定的贡献点申请作废，但所有项目所有者将依据各自评估为用户打分，最终取平均值作为用户的实际贡献点；若获得不足 60%的支持票数，则用户无法获得任何贡献点，需继续进行新的贡献行为以争取通过审核。用户获得贡献点的算法过程如算法 1 所示。贡献点获取算法由智能合约直接执行，用于对贡献者的任务完成情况进行可信记录与实时计算，并与获得贡献点模块相结合，从源头上保证贡献点分配的公平性与不可篡改性。

在本文平台中，首批开发者主要依靠项目创建者去吸引，而平台主要负责给项目创建者提供创建项目的机会。如果早期项目贡献点价值尚不明确，项目创建者可以通过如下方式吸引首批开发者：第一，强化宣传引导。平台可向潜在开发者清晰展示项目的发展前景、核心功能及待完成任务，帮助其形成明确预期，从而促进早期参与意愿。第二，增强规则可信性。平台通过智能合约将资金流向、贡献点获取规则和收益分配流程固化上链，并支持公开验证，使潜在开发者能够确信后续收益分配将严格按照既定规则执行。因此，开发者即使在项目早期进入，也能够基于规则可信性形成稳定预期，并认识到越早参与、贡献越多，未来可获得的收益也越高。第三，突出个人成长价值。部分开源开发者可能更关注技术积累与能力提升，而非短期经济

回报。为此，平台采用渐进式开放模型，允许低贡献点用户通过“租赁”方式接触高质量项目代码，并在完成任务后逐步积累贡献点、解锁更多访问权限。该机制形成了学习、贡献、成长的正向循环，有助于降低早期参与的技术门槛和心理门槛，从而吸引更多开发者在项目初期加入。

算法 1 获得贡献点算法

Input:

项目所有者数量 N;
贡献点数值 V_i;
期望贡献点数值 V_e;
来自项目所有者的同意票数量 A; $A \le N$

Output: 最终贡献点数值 V_f

提交技术任务和期望贡献点数值 V_e
项目所有者对 V_e 进行投票
if $A \ge 0.8N$ **then**
    $V_f = V_e$
else if $A \ge 0.6N$ **then**
    项目的 N 个所有者分别设定 V_i,并计算平均值
    $V_f = \dfrac{SUM(V_i)}{N}$
else
    $V_f = 0$
end if
Return V_f

首次投票环节“投票加入项目”旨在筛选适合参与项目的用户。通过允许现有项目成员进行投票，系统确保仅具备相应技术能力或良好道德标准的个体被选中，从而有效避免低质量贡献或恶意行为对项目整体质量造成负面影响。第二次投票环节“投票验证贡献点”主要用于评估用户应获得的贡献点数量。若仅依赖单方决策进行贡献点分配，易导致分配不公或操作操控等问题。投票机制通过去中心化决策过程，确保贡献点的分配获得集体认可，从而降低了系统被滥用或操纵的风险。上述两种参与方式的主要区别在于用户获得未执行贡献点的速度不同。通过投资方式加入项目的用户，可根据其投资金额，立即按比例获得未执行贡献点；而通过完成技术任务加入的用户，则需进行新的贡献提交，并通过现有项目所有者的投票审核后，方可获得相应的未执行贡献点。需要特别指出的是，不同项目之间相互独立，一个项目的贡献点不可用于其他项目，贡献点不可跨项目通用。

3.4 贡献点使用模块

图 5 展示了用户使用贡献点获取权限的完整流程。当用户希望获取更多代码访问权限时，需要通过智能合约将其持有的贡献点进行兑换。系统首先检验用户贡献点的状态，判断其为已执行还是未执行。若为已执行贡献点，则无法再次用于兑换更多代码访问权限；若为未执行贡献点，则可依据对应软件项目中预设的兑换比例进行兑换。兑换成功后，相关的未执行贡献点将被转化为已执行贡献点。已执行贡献点一旦生成，将不能再次用于获取新的代码访问权限。尽管如此，无论是未执行还是已执行的贡献点，都赋予贡献者平等的软件收益分成权。此处的软件收益指的是当有用户对本软件项目进行投资时所产生的收益分配。智能合约会根据每位贡献者在项目中所持有的贡献点占比，自动进行收益的分配。贡献点使用模块的核心逻辑亦依托智能合约执行，智能合约在此过程中负责自动校验贡献点余额，并根据预设规则触发访问权限开放与收益分配，确保贡献点与项目资源及收益之间的绑定关系安全透明。

在实验周期内，贡献点能够兑换访问代码的比例（“汇率”）在项目中是固定的，用户通过贡献点使用模块自动兑换对应比例的访问代码，因此贡献点的汇率具有稳定性；而贡献点给用户带来的每次投资对应的分

红额度（“价值感”）则会出现波动，这是因为虽然用户可依据平台固定的贡献点收益分配规则，自动获得每次投资对应的分红收益，但单次分红金额仍取决于单次投资金额大小、个人贡献点占比以及外部市场环境等因素。

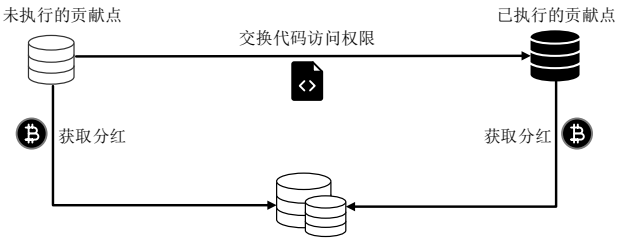


图 5　贡献点使用模块的工作流程

3.5 仲裁模块

如图 6 所示，仲裁模块的工作流程旨在保障整个开发过程的安全性与可靠性，是平台降低恶意行为风险的重要机制。当本文平台需要当平台需要组建评审团时，将首先依据争议项目的标签、技术栈及候选用户的历史贡献记录，对高信用用户库进行领域相关性筛选，在此基础上，再向匹配用户随机发送邀请。只有在用户明确同意参与后，其才会被纳入候选评审团成员范围。如果高信用用户认为自身缺乏相关领域知识、可能影响裁决准确性，则可主动拒绝此次邀请。平台最终从领域匹配且自愿参与的高信用用户中随机抽取评审团成员。该机制能够在保留随机性的同时，降低跨领域知识不匹配带来的裁决风险。

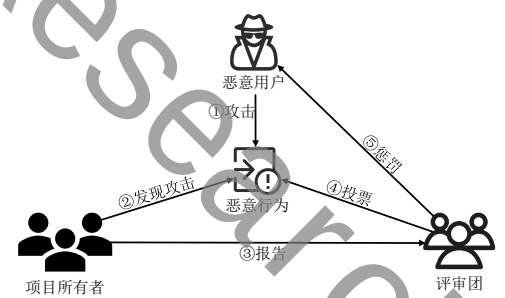


图 6　仲裁模块的工作流程

当项目所有者发现潜在的恶意行为时，可启动仲裁流程。仲裁评审团由从平台中随机选取的 M 名用户中抽取的 K 名高信用用户组成。评审团的任务是调用投票合约，对被举报用户的恶意行为的真实性与严重性进行投票表决。若恶意行为被确认，平台将依据行为的严重程度对用户实施相应惩罚措施。例如，扣减用户的信用积分，而信用积分直接影响用户可获得的贡献点及收益金额，因此，信用积分的下降将导致用户潜在收益减少。在恶劣情况下，恶意用户还可能被列入黑名单，永久禁止参与后续项目。关于信用积分的详细定义与计算方法将在第 5.2 节中进一步阐述。仲裁模块的执行机制同样通过智能合约实现，其预定义裁决规则能够在争议发生时自动触发，从而提升仲裁过程的透明度与可信度。

对于误操作或被恶意举报后的申诉与信用恢复，本文平台也开放了对应的通道。当用户认为存在误操作或遭遇恶意举报时，可在规定时限内发起申诉请求，并提交可验证证据如水印验证结果、链上交互记录、任务提交记录等，以触发申诉通道，使事件重新进入仲裁投票流程进行复核；同时，系统对申诉过程中的关键证据与投票结果进行链上记录，以保证复核过程可审查、不可篡改。若复核结果否定恶意结论，则对已被扣减的信用分进行等额的补偿。相反，若复核确认恶意结论属实，则维持原裁决并继续执行相应惩罚措施同时将复核结果链上记录，从而增强裁决的可追溯性并抑制滥用举报、强化对恶意行为的震慑效应。

3.6 模块交互

如图 7 所示，我们在平台系统设计中进一步细化了分层结构及模块间的交互逻辑，以确保系统可实现性与可扩展性。系统整体分为用户层、业务逻辑层、智能合约层和数据存储层，各层通过标准化接口与统一数据格式进行交互，从而确保数据一致性与高效性。具体而言：(1)用户层通过 API 以 JSON 格式提交用户请求与贡献数据，用户管理模块完成身份验证并生成 JWT 令牌传递至贡献点管理模块，实现安全的访问控制；(2)贡献点管理模块在接收验证信息后，依据预设参数(如贡献点类型、贡献点数量)生成贡献点更新请求，并以 JSON 格式发送至智能合约模块以完成链上状态更新，同时向仲裁模块推送争议信息以保障传输完整性与可追溯性；(3)仲裁模块基于收到的争议信息进行裁决，并将仲裁结果以 JSON 格式回传至智能合约模块以触发后续处理；(4)数据存储层采用链上与链下混合存储策略，链上以 JSON 格式记录交易与状态数据以确保不可篡改性，链下以 SQL/JSON 存储辅助数据以优化查询性能，并通过数据同步机制保持链上链下数据一致。该分层设计不仅明确了各模块的功能职责与接口定义，还确保了数据格式的标准化与交互流程的可追溯性，从而提升了系统的可维护性与跨模块协同能力。

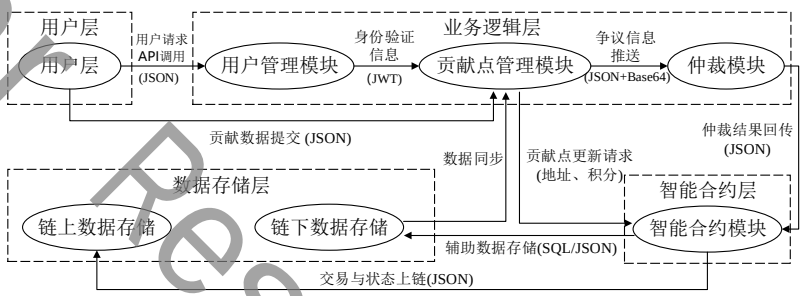


图 7 不同模块之间的交互示意图

3.7 区块链网络

本节重点讨论第二类恶意行为，即代码资产层面的泄露行为。具体而言，恶意用户可能通过投资或合法访问获得平台上的软件代码，并进一步非法转售或泄露以谋取个人利益。此类代码泄露不仅会导致项目遭受重大经济损失，还将严重侵害平台用户的整体利益。非法转售的全过程如图 8 所示。

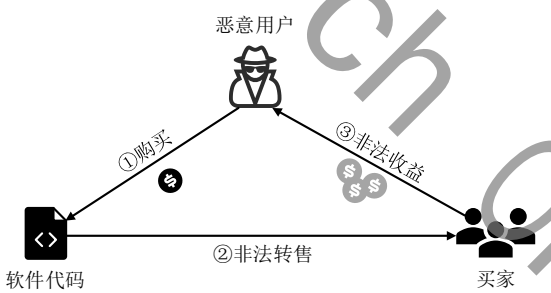


图 8 恶意用户的代码泄露行为

为切实保障平台全体用户的权益，本文设计并实施了一套完善的安全机制，以防止恶意用户泄露代码信息。具体措施包括基于智能合约的押金机制、代码水印技术以及基于区块链的信用积分系统，通过对用户施加相应的奖励与惩罚，实现有效的威慑与激励。鉴于大多数恶意用户以利润驱动为主要动机，本文旨在最大化恶意用户泄露信息的成本，同时对举报人给予及时奖励，构建竞争性利润格局，从而迫使潜在的恶意用户放弃泄露代码的行为。从技术角度来看，本文平台中的奖励与惩罚机制均通过智能合约即时执行，确保了执行过程的公正与高效。基于智能合约的押金机制抵御代码泄露的具体步骤将在下文展开论述，详细的交互过程如图 9 所示：

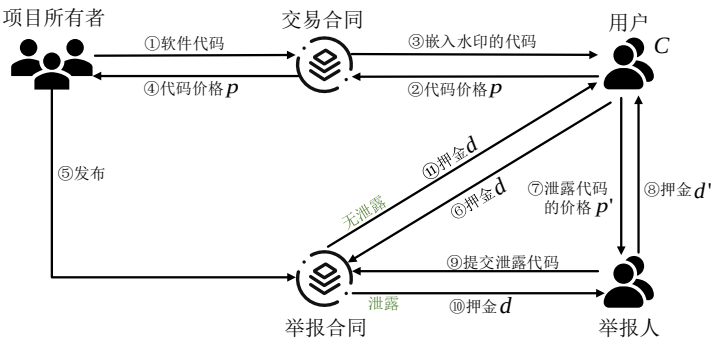


图 9　区块链安全机制的工作流程

步骤 1：项目所有者发布交易合约，并将待售的软件代码存储于交易合约中。

步骤 2：希望购买软件代码的用户 C 需向合约账户转账金额 p 。

步骤 3：交易合约将在软件代码中嵌入数字水印后，将代码交付给用户 C 。同时，每位贡献者将按比例获得分红，金额为 xp，其中 x 表示该贡献者在软件项目整体贡献中的占比。

步骤 4：交易合约将金额 p 从合约账户转账至各项目所有者的账户。

步骤 5：在发布交易合约的同时，项目所有者同步发布举报合约。所有购买软件代码的用户需向举报合约账户缴纳押金 d ，并维持押金有效期 t 。

步骤 6：用户将押金 d 存入举报合约账户。

步骤 7：若存在泄露者，其可能以价格 p' 对所购买的软件代码进行泄露交易。值得注意的是，在此交易中，泄露者可与购买者签订保密协议，以防止自身被举报，并要求购买者缴纳押金 d'，押金有效期为 t' 。

步骤 8：购买者将押金 d' 转账给泄露者，并获得泄露的软件代码。

步骤 9：举报人将购买获得的泄露软件代码提交至举报合约，从而发起泄露举报。

步骤 10：系统对泄露的软件代码提取并验证水印信息，若确认存在代码泄露行为，则泄露者所缴押金 d 将转移至举报人账户，同时泄露者的信用积分将被扣减。

步骤 11：若用户在押金有效期 t 内未发生任何代码泄露行为，则押金 d 将全额退还给用户。

3.7.1 区块链水印追踪模块

代码水印技术[26]是一种在软件或代码中嵌入特定标记或信息的手段，类似于数字水印，其主要用于代码的跟踪、识别与保护。该技术常被应用于知识产权保护、盗版防止、源代码使用追踪及代码合法性验证等场景中[27]。代码重排为水印信息的隐蔽嵌入提供了有效途径，通过调整代码段中语句的执行顺序，不仅提高了水印的隐蔽性，难以被泄露者察觉，同时也便于后续泄露者的追踪定位。本文通过在代码段 S 内不断交换任意两行代码的执行顺序，且保证程序执行逻辑正确，来实现代码重排。如算法 2 所示，具体流程为：首先检测代码段中所有可以交换顺序且不影响程序正确执行的代码行号对，并将其存储在集合 L 中。每一对行号 $<L_i,L_j>$ 均需满足交换后程序逻辑不变的条件。随后，从集合 L 中随机且不重复地选取一个子集 L'，并对其中每一对行号 $<L_i,L_j>$ 对应的代码行进行交换，生成新的代码段 L'。同时，将选取的 L' 作为水印信息保存在服务器的水印数据库 P 中。当发生代码泄露事件时，可依据泄露代码中携带的水印信息，准确追溯到对应的用户身份，从而实现有效的责任追究。算法 2 的复杂度主要取决于代码行数 L，时间复杂度约为 $O(L^2)$ 。实际应用中，由于代码重排水印算法仅在必要的水印嵌入或验证阶段触发，其时间开销对平台整体性能影响相对有限。

算法 2　代码重排水印算法

Input: 代码段 S

Output: S'

1: 检测所有可以交换位置的代码行对，并生成集合 $L=<L_1,L_2>,...,<L_{n-1},L_n>$

2: 从集合 L 中随机且不重复地选取子集 L'

3: **for** $<L_i,L_{i+1}>\in L'$，**do**：

4: $<L_i,L_{i+1}>\rightarrow<L_{i+1},L_i>$

5: **end for**

6: 生成新的代码段 S'

7: 将 L' 添加到集合 P 中

8: **Return** S'

在通过代码重排生成新的代码片段 S' 后，若发生代码泄露事件，可通过与代码水印对应的水印信息库中保存的记录来追溯泄露用户。由于每位用户对应的水印信息 L' 具有唯一性，因此可准确识别泄露者的身份。算法 3 详细描述了具体的识别过程。首先，算法从水印信息 L' 库 P 中提取出所有水印信息。对于每一条水印信息 L'，算法依次提取其中包含的所有行号对 $<L_i,L_j>$，并按照行号对交换相应的代码行，同时将泄露的代码块 M 据此转化为 M'。随后，算法分别计算每个变换后的代码块 M' 与原始代码片段 S 之间的相似度，并将每个相似度值添加至集合 N 中。完成所有相似度计算后，算法在集合 N 中找到最大值 N_i，并确定对应的水印信息 L'，从而最终锁定代码泄露用户的身份。算法 2 主要适用于存在可重排空间的代码段，当检测到代码段整体存在严格依赖、无法安全重排时，平台将采用动态代码水印方法[28-30]，在运行期或交付阶段引入不影响程序语义的水印标记，以实现版权追踪与责任定位。此外，算法 2 在嵌入水印前对代码段进行控制依赖与数据依赖分析，仅将不存在严格顺序依赖、交换后不改变程序语义的语句对纳入候选集合，并以语义保持不变作为可交换的必要条件。本文采用的是一种保守的水印算法，适用于静态源代码中顺序语义明确、可静态分析的局部语句块；而对于复杂异步调用、反射机制、关键硬件指令序列等语义敏感代码，本文不进行水印嵌入，以防止因语句交换而影响程序语义正确性。

算法 3　水印检测算法

Input:

1: 代码片段 S

2: 泄露的代码段 M

3: 算法 2 中的集合 P

Output: 最大相似度

4: **for** $L'\in P$ **do**

5: **for** $<L_i,L_{i+1}>\in L'$ **do**

6: $<L_i,L_{i+1}>\rightarrow<L_{i+1},L_i>$

7: 提取生成一份水印处理后的代码副本 M'

8: **end for**

9: 计算 M' 与 S 之间的相似度，并将结果加入集合 N

10: **end for**

11: **Return** 返回集合 N 中最大值对应的 N_i

12: 获取与该 N_i 对应的水印用户

3.7.2 信用分模块

在本节中，我们对区块链环境下的用户信用分数(CS)机制进行了详细介绍。信用分数直接反映了用户开发活动的质量，通过区块链技术进行记录，确保过程的透明性与公平性。CS 分为两个部分：一是用户所累积的正向得分，二是因不当行为被扣除的负向得分。为了更精准地衡量用户的信用水平，平台对用户在所有项目中累积的正向与负向得分进行汇总计算。每位用户的 CS 均具有唯一性，其最终得分由基础得分与正向

得分之和减去负向得分确定，用户的累积正向信用分数 A 与负向信用分数 B 分别定义如下：

$$A=\sum_{u=1}^{x}\sum_{v=1}^{y}AP_{uv}$$

$$\mathrm{B}=\sum_{u=1}^{x}\sum_{v=1}^{y}BP_{uv}$$

其中， x 表示用户参与的项目数量， y 表示每个项目中用户可能进行的最大开发活动次数， AP_{uv} 与 BP_{uv} 分别表示第 u 个项目中第 v 次开发活动获得的正向得分与被扣除的负向得分。平台设定用户初始信用分数为 100 分。当信用分数降至 60 分以下时，用户将被禁止继续参与项目。设 I_{final} 为用户最终可获得的贡献点，$I_{initial}$ 为项目所有者投票评定的初始贡献点， I_{credit} 为用户当前的信用分数，则最终贡献点计算公式如下：

$$I_{final}=I_{initial}\times\frac{I_{credit}}{100}\quad(I_{credit}\geq 60)$$

上述公式表明，用户最终获得的贡献点数与其信用分数成正比。在收益分配方面，设 G_{final} 为用户在项目中可获得的最终奖励， G_{sum} 为项目总体收益， T_{sum} 为项目中所有贡献点的总和，则最终奖励的计算公式为：

$$G_{final}=G_{sum}\times\frac{G_{final}}{T_{sum}}$$

需要指出的是，不同项目之间的贡献点与收益分配互不通用，保持独立性。

此外，用户信用分数 C 的增长受到两大因素影响：一是成功贡献的次数 s ，二是连续贡献的时间 t 。其增长函数定义如下：

$$C=e^{-e^{-(t-365)}}+e^{-e^{-(s-200)}}\quad(0\leq t,0\leq s\leq 10^5)$$

本机制鼓励用户频繁且持续地进行贡献，以提升平台项目的可持续发展水平。尽管兼职开发者持续贡献存在一定难度，但他们仍可通过成功提交任务逐步累积信用分数。高频率、长期贡献的用户将享有更快的信用分数增长速度。需要注意的是，贡献次数 s 的上限设定为 100,000 次，超过此数后贡献次数不再对信用分数增长产生影响，而连续贡献时间 t 则不设上限。然而，仅依赖贡献次数与持续时间仍不足以完全约束职业仲裁黑产，即恶意用户通过在多个小项目中刷高信用分的行为。职业仲裁黑产的核心风险在于恶意用户可能通过低成本参与大量低成熟度的小项目快速累积信用分，并渗透进入仲裁候选池。为抵御该风险，本文引入了项目成熟度约束，将信用分有效增量与项目成熟度绑定，降低低成熟度小规模项目所能带来的信用分有效增量，提高恶意用户通过多个小项目快速刷分的时间成本、贡献成本、组织成本，进而提升其进入仲裁候选池的难度，缓解此类风险。

具体而言，设用户 u 在项目 j 中获得的原始信用分增量为 $C_{u,j}$，项目 j 的成熟度得分为 $M_j\in[0,1]$，本文将根据项目 j 的成熟度 M_j 重新计算信用分的有效增量 $C_{u,j}^{weighted}$ ，以实现不同成熟度项目间信用分积累速率的差异化调控。

$$C_{u,j}^{weighted}=C_{u,j}*M_j$$

该式表明，成熟的大规模项目通常能够带来更高的信用分增量，而不成熟的小规模项目带来的增量则相对有限。项目成熟度 M_j 由贡献者规模 S_j 、贡献者多样性 D_j 、项目活跃度 A_j 、项目成立时间 T_j 、Issue 解决效率 E_j 等五个不同的维度组成，具体定义如下：

贡献者规模 S_j：设 N_d 为项目 j 中至少完成 10 次有效贡献的独立用户总数，其中有效贡献指提交任务并通过投票获得贡献点或者直接投资获取贡献点的行为。

$$S_j = \min\left(1, \frac{N_d}{1000}\right)$$

该指标反映了项目的贡献者规模，成熟的大型项目通常拥有较大规模的贡献者群体，而恶意用户攻击的小项目往往贡献者规模较小。

贡献者多样性 D_j：设 P 为项目 j 中贡献者的组织类别数（如企业、高校、科研机构等），$n_{j,k}$ 为来自 k 类组织的贡献者人数，则采用辛普森多样性指数定义为：

$$D_j = 1 - \sum_{k=1}^{P}\left(\frac{n_{j,k}}{N_d}\right)^2$$

该指标用于衡量贡献者在不同组织类别之间的分布均衡程度，该值越接近 1 表示贡献者组织来源越分散、参与结构越均衡。该指标可作为识别异常集中参与模式的辅助特征，有助于降低单一组织集中参与所带来的异常信用累积风险，因为真实项目通常汇聚来自不同组织的贡献者。

项目活跃度 A_j：设 a_j 为项目 j 在近 6 个月内的日均代码提交次数，则定义：

$$A_j = \min\left(1, \frac{a_j}{20}\right)$$

该指标反映项目在观察期内是否保持持续开发活动。活跃度越高，说明项目在近期仍具有持续开发特征。

项目成立时间 T_j：设 d_j 为项目 j 自创建以来的累计存续天数，则定义：

$$T_j = \min\left(1, \frac{d_j}{730}\right)$$

该指标用于衡量项目的存续时长与时间稳定性，730 天对应两年的持续运营周期。相比短期新建项目，长期存续项目经过了更充分的时间验证，因此该指标有助于提高通过临时构造项目快速累积信用分的时间成本。

Issue 解决效率 E_j：设 $\bar{t}_j$ 为项目 j 中 Issue 从提出到关闭的平均时长，则定义：

$$E_j = \frac{1}{1 + 0.1\bar{t}_j}$$

该指标主要反映项目的持续维护状态，平均关闭时间越短，说明项目维护响应越及时，该指标得分越高。

在此基础上，项目成熟度 M_j 定义为：

$$M_j = 0.2S_j + 0.2D_j + 0.1A_j + 0.4T_j + 0.1E_j$$

其中，各维度权重之和为 1。本文对项目成立时间赋予更高权重，原因在于长期持续运营所体现的稳定性更难以通过短期行为操纵[31]；贡献者规模与贡献者多样性分别反映项目的参与广度与贡献者类型的丰富程度，对识别低成本异常累积行为具有重要作用[32]；项目活跃度与 Issue 解决效率则作为辅助性指标，用于刻

画项目的活跃与维护状态[33]。基于上述设计，信用分增长不再仅取决于单次贡献行为，而是受上述五个维度的共同约束。因此，恶意用户若试图通过低成熟度项目快速累积信用分，不仅将面临更高的时间成本、组织成本与贡献成本，还需持续维持项目活跃度、贡献者多样性和长期运行状态，这些都显著增加了职业仲裁黑产的成本和难度。

3.7.3 有效性证明

为验证平台的安全性，本研究聚焦于以下关键研究问题：

RQ1：本文平台所采用的安全机制在降低软件项目信息泄露风险方面的有效性如何？

鉴于安全领域攻击场景多样且复杂，直接通过实证数据回答此研究问题存在较大挑战。为此，本文引入博弈论方法对平台机制进行理论分析，并据此推导相关结论。博弈论作为一门融合数学与经济学的方法论体系，旨在系统分析与建模不同理性个体在多种情境下的策略交互行为[34]。本研究采用博弈论方法，针对本文平台中用户可能存在的软件信息泄露行为进行理论分析。

在本研究中，我们将通过泄露软件代码以获取非法利益的恶意用户统称为盗版者。假设所有用户均为理性个体，均以收益最大化为决策目标，并且彼此独立作出行为选择。基于此，本文构建了本文平台下代码泄露过程的博弈模型，旨在验证以下命题：对于任何在本文平台中购买软件代码的用户而言，无论其在信息泄露过程中是否与接收方签订保密协议，其通过泄露或盗用软件代码所能获得的利润均不超过其选择不泄露时所能获得的利润。

本文分析以一名用户 C 为起点，C 通过本文平台获取软件。基于区块链机制，平台要求用户 C 在购买许可软件(费用为 p)的同时，缴纳一笔金额为 n 、维持期限为 t 的押金，以防止盗版行为的发生。此外，每当一位真实用户产生时，C 将获得一笔奖励，金额为 xp ，其中 x 表示 C 对该软件整体贡献的比例。与此同时， x 也决定了 C 能访问的软件信息比例，即 C 有权泄露的软件信息范围。仅当 x 值较大时， C 所泄露的信息量才具有足够价值，使盗版行为具备可行性。

如图 10 所示，游戏树模型描述了在具备盗版条件下，用户 C 的决策过程。C 面临三种选择路径： T_1 ，即既不泄露也不从事盗版； T_2 ，泄露软件信息但未与盗版购买者签订保密协议； T_3 ，在泄露软件信息的同时，与盗版购买者签订保密协议。保密协议要求每位盗版购买者缴纳押金 d' ，押金维持期限 t' 必须长于 t ，以确保 C 能顺利赎回其最初缴纳的押金 d 。由于正版软件的售价通常高于盗版软件，因此少有盗版购买者愿意支付高于正版软件价格的押金，故而 d' 的设置通常低于 d 。

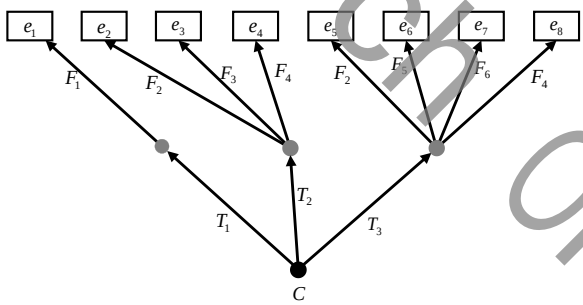


图 10　用户 C 决策过程的博弈树

如图 10 所示，当用户 C 做出决策 T_1 、 T_2 或 T_3 时，潜在的盗版购买者可分别采取六种不同的反应策略 F_1-F_6 ，最终形成八种独特的情境 e_1-e_8 。用户 C 及潜在盗版购买者的收益与成本关系在表 2 中进行了详细列示。在该背景下，若 C 选择 T_1 决策，对应唯一的响应为 F_1 ，即未发生软件信息泄露及盗版行为。此时，潜在购买者只能选择购买正版软件， C 获得的收益为 $m\times xp$ ，其中 m 表示购买者数量， x 表示 C 在软件整体贡献中的比例， p 表示正版软件价格。

若 C 选择在无保密协议的 T_2 情况下进行盗版，则盗版购买者可选择 F_2 、 F_3 或 F_4 中的任一策略，对应情境 e_2 、 e_3 或 e_4 。盗版购买者将选择使其收益最大化的响应策略，如表 2 所示，最优选择为 F_2 。在该情境下，

购买者可在押金维持期 t 内通过举报盗版行为而获得押金 d 作为奖励。与此同时，C 失去押金 d，并以 $m\times p'$ 的非法收入替代原本可获的 $m\times xp$ 正版收益，其中 p' 为盗版价格。此外，C 与本文平台的长期合作关系将受到损害，其信用积分与贡献点亦将被扣减。无论盗版行为是否被举报，平台均可通过水印技术识别泄露的软件版本，并对 C 施加信用及贡献点的惩罚。

类似地，当 P 选择在签订保密协议的 T_3 前提下进行盗版时，盗版购买者可选择 F_2、F_5、F_6 或 F_4 中的任一策略，对应情境 e_5、e_6、e_7 或 e_8。此时，购买者依然倾向选择收益最大的响应 F_2，即在押金维持期 t 内举报盗版行为。在此情境下，购买者虽失去押金 d'，但通过举报获得押金 d，由于 $d'<d$，故总体上仍可获利。与此同时，C 虽可通过保密协议获得购买者押金 d'，并以 $m\times p'$ 的盗版收入替代 $m\times xp$ 的正版收益，但其失去押金 d，且 C 最终仍将被本文平台检测出，并面临信用积分及贡献点的扣减。

基于表 2 中的分析，若要使盗版行为在短期内具有利润空间，C 需满足 $mxp<mp'-d+d'$ 的条件。然而，在实际盗版需求的约束下，此条件难以成立，因为盗版价格 p' 必须远低于 p，且泄露信息完整且有价值要求 x 不宜过小，同时盗版购买者通常不愿意支付高于正版软件价格的押金，因此 $d'<d$。更重要的是，C 在长期将面临更为严重的损失，信用积分和贡献点的下降将导致其在平台上的长期收益显著减少，且未来将被禁止再次进行盗版行为。因此，在本平台中，C 的最优策略是避免从事盗版行为。

表 2　用户 C 与盗版购买者各自的收益与成本

决策	描述	C 的收益	C 的损失	购买者利润
T_1	(F_1, e_1) 无盗版	mxp	-	-
T_2	(F_2, e_2) 在 t 期间盗版被举报	$mp'-d$	损失信用分与贡献点	d
	(F_3, e_3) 在 t 之后盗版被举报	mp'	损失信用分与贡献点	-
	(F_4, e_4) 从未被举报	mp'	损失信用分与贡献点	-
T_3	(F_2, e_5) 在 t 期间盗版被举报	$mp'-d+d'$	损失信用分与贡献点	$d-d'$
	(F_5, e_6) 在 t 与 t' 之间盗版被举报	$mp'+d'$	损失信用分与贡献点	$-d'$
	(F_6, e_7) 在 t' 之后盗版被举报	mp'	损失信用分与贡献点	-
	(F_4, e_8) 从未被举报	mp'	损失信用分与贡献点	-

此外，本文提出的平台安全机制在面对非理性攻击与多方共谋等复杂威胁场景时同样具有适用性。具体而言，对于单一的非理性攻击者，即使其偏离理性最优策略，平台仍可依托仲裁模块的多方验证机制快速识别异常行为并触发仲裁流程，并通过扣除押金等措施惩罚其恶意行为，从而有效降低单点破坏的风险。同时，对于可能发生的多方共谋，信用分机制在长期博弈过程中引入累积效应，使合谋者因恶意行为而信用分显著下降，从而失去更多潜在收益。仲裁模块与信用分机制形成的双重约束，使参与者即便在非理性与合谋条件下，也需权衡短期收益与长期信用损失，最终维持系统的整体安全性与稳健性。而针对软件供应链中的信息篡改与伪造风险，本文平台的应对建立在区块链的底层可信原理之上：一方面平台将贡献点变更、权限授予回收、收益分配、投票与仲裁结果等关键状态以交易形式写入链上，由区块的哈希链式结构与时间戳将当前状态与历史记录强绑定，使任一历史字段被修改都会导致哈希不一致并被全网快速识别。另一方面区块链通过分布式共识机制在多数诚实节点假设下对新区块进行一致性确认，从而将写入正确从单点判断转化为多方共同验证，攻击者若要篡改已确认记录，必须重构目标区块及其后的链上历史并获得多数共识，在 PoW 情形下通常需掌控超过全网半数算力，其所需付出的计算/权益成本与持续追赶主链的成本代价极高，现实中几乎不可能达成，并且交易级公私钥签名为每次状态更新提供身份绑定与不可否认性，攻击者若要伪造行为身份，需窃取私钥或突破签名体系，成本与难度同样显著。

4　实验与分析

本研究基于 Solidity 0.4.25 版本实现了与本文平台相关的智能合约，并在以太坊测试网络上进行了合约部

署与性能评估。实验环境为搭载 AMD 3.00GHz 八核 CPU 与 16GB 内存的联想 R7000p 笔记本电脑。围绕本文平台的有效性与性能开销，本文进一步设计并开展了系列实验，旨在回答如下研究问题：

RQ2：本文平台在多大程度上提升了软件开发的积极性？

RQ3：本文平台是否有助于加速软件开发的完成进程？

RQ4：本文平台引入的区块链开销规模如何？

RQ5：本文平台各核心模块的必要性与有效性如何？

4.1 回答RQ2

为回答研究问题 RQ2，我们对本文平台与 BountySource 平台在任务完成率方面进行了对比分析，这两者均为完成指定任务提供经济激励。由于 Gitcoin 平台的财务数据不可获取，故本文仅在 6.4 节后续部分中对其链上数据进行单独比较。对于 BountySource 平台，本文参考 Zhou 等人方法[35]，收集了 2019 年 1 月 1 日至 2022 年 12 月 31 日期间的 669 份悬赏任务状态记录。需要指出的是，BountySource 平台随后一度停止运营。若悬赏猎人成功完成任务并获得奖励，则该任务被标记为完成；如果任务被领取但在规定时间内未完成，则标记为失败；未被任何悬赏猎人接取并到期的任务则归为被忽略。为确保平台任务类型与难度分布的一致性，本文在表 3 中对 BountySource 与本文平台上的任务进行了分类，划分为功能、改进、文档、错误、设计和代码评审六类。

表 3　BountySource 平台与本文平台各自问题与任务的类型

类型	难度	描述	BountySource(%)	本文平台(%)
功能	一般	创建新功能	42.4%	41.9%
改进	一般	改进已有功能	16.8%	17.6%
文档	简单	创建文档	7.4%	7.6%
错误	一般	修复 bug	21.3%	20.9%
设计	困难	设计系统	10.8%	10.3%
代码评审	简单	评审代码	1.3%	1.7%

如图 11 所示，我们对本文平台与 BountySource 两个平台的任务完成率进行了对比分析。BountySource 平台通过提供经济激励，使得 20%的悬赏任务成功完成，但仍有 67.3%的任务在被领取后未能完成，另有 12.7%的任务在到期前无人接取而被归为被忽略任务。值得注意的是，在这些被忽略的任务中，超过一半的任务设置了超过一年的到期时间，表明存在较长时间内无人关注的问题。相比之下，本文平台的任务完成率达到了 32.4%，失败率为 56.8%，被忽略任务的比例仅为 10.8%。这一结果表明，与 BountySource 相比，本文平台在激励开发者积极性方面展现出更高的有效性，其激励机制能够更有效地促进开发者完成任务。

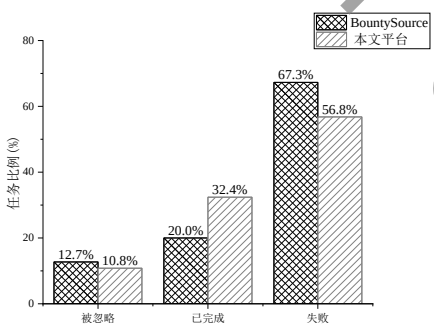
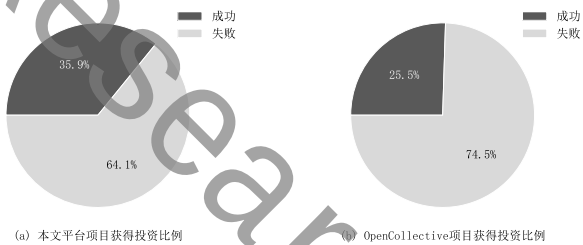


图 11　BountySource 平台和本文平台上任务的完成比例

对平台特性的深入分析表明，本文平台在激励能力方面表现优越，其原因可归结为以下两点。首先，本文平台引入了基于贡献点的收益分配机制，能够根据每位开发者的实际贡献分配收益，而 BountySource 仅对成功完成悬赏任务的开发者提供奖励，缺乏此类机制。更加公平的分配方式有效激发了开发者积极性，使

本文平台任务完成率达到 32.4%，显著高于 BountySource 的 20%。其次，本文平台在经济激励之外进一步引入了技术激励机制：针对更重视技术成长而非经济回报的开源开发者，渐进开放模型随开发者完成任务的累积逐步赋予其访问更多技术细节与高质量项目的权限，从而激发其内在动机。这一激励方式难以在以金钱回报为唯一驱动的 BountySource 中奏效，因而本文平台在激励多样性与开发者持续参与方面更具优势。值得注意的是，BountySource 对于悬赏金额不足 100 美元的任务，即使任务未完成或到期失效，资金亦不返还赞助者，致使赞助者发布任务时趋于审慎。然而其任务失败率仍偏高，根本原因在于多开发者协作支持不足，复杂软件开发任务通常需要互不相识的多名开发者协同完成，而 BountySource 的单一悬赏得主奖励机制难以适应此类需求。相比之下，本文平台有效缓解了复杂任务的协作与收益分配难题，降低了任务失败率。

此外，本文对 OpenCollective 与本文平台上获得资助或投资的项目比例进行了对比分析，如图 12 所示。通过调用 OpenCollective 的开放接口，本文收集了 2019 年 1 月 1 日至 2022 年 12 月 31 日期间，该平台上 838 个集体的捐赠数据。我们将成功获得捐赠的项目定义为成功项目，未获得捐赠的项目定义为失败项目。对应地，在本文平台上，成功获得投资的项目亦被定义为成功项目，未获得投资的项目则定义为失败项目。分析结果显示，本文平台上项目获得投资的比例为 35.9%，略高于 OpenCollective 平台的 25.5%。这一差异可能源于本文平台特有的投资激励机制，该机制允许投资者直接从其投资行为中获益。此外，本文平台基于区块链的去中心化设计实现了高透明性与可追溯性，有效降低了投资风险，提升了项目的可信度，从而增强了投资者对平台的信任感。相比之下，OpenCollective 平台缺乏类似的投资回报机制，捐赠者主要基于利他动机进行捐赠，对机构投资者的吸引力相对较弱。同时，OpenCollective 在资金使用的透明度方面，尤其是在处理复杂或长期项目时，存在一定问题，可能进一步削弱捐赠者的信任。



(a) 本文平台项目获得投资比例　　(b) OpenCollective项目获得投资比例

图 12　本文平台获得投资项目和 OpenCollective 平台获得捐赠项目的比例对比

此外，我们进一步收集了 2019-2022 年不同类型项目的平均最大贡献比例与获捐项目比例统计数据，并结合表 4 对渐进式开放模型下个体贡献比例是否会影响赞助方意愿进行了进一步分析。

表 4　2019-2022 渐进式开放模型下不同类型开源项目的平均最大贡献比例与获捐项目比例

项目类型	2019 (平均最大贡献比例 / 获捐项目比例)	2020(平均最大贡献比例 / 获捐项目比例)	2021(平均最大贡献比例 / 获捐项目比例)	2022(平均最大贡献比例 / 获捐项目比例)
开发框架	7.7% / 31.2%	8.4% / 33.0%	6.9% / 35.1%	9.2% / 38.6%
开发工具	6.7% / 30.1%	7.2% / 32.4%	6.5% / 34.6%	8.4% / 37.4%
数据处理	8.9% / 27.4%	6.4% / 30.2%	8.2% / 33.0%	7.8% / 35.4%
安全隐私	5.9% / 28.9%	8.1% / 31.5%	7.7% / 32.8%	8.3% / 34.8%
操作系统	7.3% / 34.1%	7.6% / 35.6%	9.0% / 37.4%	8.1% / 39.4%
平均	7.3% / 30.4%	7.5% / 32.5%	7.7% / 34.6%	8.4% / 37.1%

表 4 显示，2019-2022 年间，渐进式开放模型下各类开源项目的获捐项目比例整体保持在较高水平，并呈持续上升趋势，平均值由 2019 年的 30.4%提升至 2022 年的 37.1%。这表明，按贡献分配产权并未削弱赞助方的参与积极性，赞助方依然维持了较高的赞助积极性。分项目类型来看，开发框架、开发工具、数据处

理、安全隐私和操作系统五类项目的获捐项目比例均稳步增长，至 2022 年分别达到 38.6%、37.4%、35.4%、34.8%和 39.4%，说明渐进式开放模型对不同类型项目均具有较好的资金吸引能力。进一步结合平均最大贡献比例可以发现，个体贡献集中度与获捐项目比例之间并未表现出显著一致的对应关系。例如，2019 年数据处理类项目的平均最大贡献比例为 8.9%，高于多数项目类型，但其获捐项目比例仅为 27.4%；而操作系统类项目同期的平均最大贡献比例仅为 7.3%，获捐项目比例却达到 34.1%。

实验结果表明，在渐进式开放模型下，个体贡献大小不会影响赞助方的赞助意愿，赞助方同样保持了较高的参与积极性。这一现象可能与赞助方更关注收益分配规则是否清晰、透明、可验证有关，而不必然取决于个体产权份额本身。在渐进式开放模型下，贡献点、权限授予与收益分配均通过智能合约自动执行并上链存证，资金流向与回报机制完全透明、可追溯，任何参与方均可独立核验分配结果的公正性。收益分配严格遵循链上贡献点比例自动执行，项目决策亦通过贡献点投票机制实现多方制衡，从而降低了赞助方对个别高贡献者主导收益分配或扭曲项目方向的担忧，维持了赞助方持续参与的积极性。

4.2 回答RQ3

本文比较了 BountySource 平台从资助者发布问题、设定悬赏金额，到用户最终解决问题并成功领取悬赏的整个时间区间，与本文平台从任务发布到用户成功贡献并获得贡献点的完整时间区间。本文之所以仅选择 BountySource 作为对比平台是因为 BountySource 在任务悬赏模式和社区驱动机制上与本文研究平台具有高度可比性：两者均通过悬赏驱动开发者完成特定任务，从而在方法论上形成严格对应。相较之下，其他平台(如 GitHub Sponsors 与 OpenCollective)在运行逻辑上与本文研究问题存在显著差异：GitHub Sponsors 侧重于持续性捐赠与赞助机制，OpenCollective 则主要服务于透明化的资金管理与支出跟踪，而并非以任务驱动的悬赏机制为核心。

根据图 13 的结果可以看出，在首次响应时间方面，BountySource(77.3%)与本文平台(81.5%)的首次响应主要集中在 0-10 区间，但本文平台在该区间内的占比比 BountySource 高出 4.2%。在完整响应时间方面，本文平台的完整响应时间(34.6%)主要集中在 0-10 区间，而 BountySource 的完整响应时间(26.1%)虽也主要集中在 0-10 区间，但完整响应耗时明显长于本文平台。

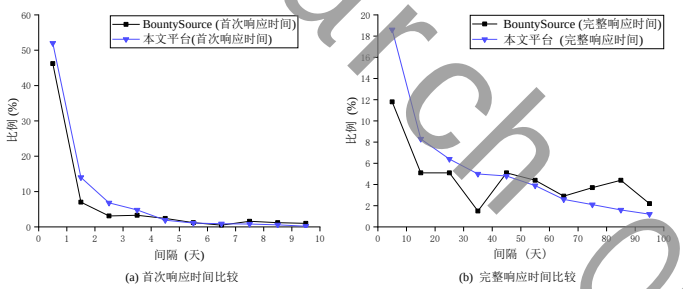


图 13 BountySource 平台与本文平台的首次响应时间与完整响应时间对比

造成这一差异的主要原因有两点：首先，在 BountySource 平台上，资助者在提出问题后并不会立即添加悬赏金额，通常会等待一段时间，观察问题是否能够在无悬赏的情况下被自然解决。若在等待期间问题未能得到解决，资助者才会开始设立悬赏。这一等待过程额外消耗了较多时间。而在本文平台上，任务一经发布，用户即可直接参与解决，无需经历类似 BountySource 的等待过程，从而缩短了响应时间。其次，BountySource 上存在部分难度较高、悬赏金额较大的问题。由于该平台仅支持单一开发者领取悬赏，因此当遇到复杂问题时，单个开发者往往需要较长时间才能完成任务。相比之下，本文平台采用多用户协同开发机制，能够通过多名开发者的合作加速复杂问题的解决进程，有效缩短了完整响应时间。

此外为了进一步了解本文贡献点分配机制相较于既有收益分配与贡献评估方式的有效性与优势。本文将贡献点分配机制与单一悬赏机制、平均分配机制进行系统比较，本文为这三种机制均设定了同等总额的激励预算，同时在任务规模、难度、类型、时间周期等其它属性上均保持一致，从而尽可能排除因预算差异带来

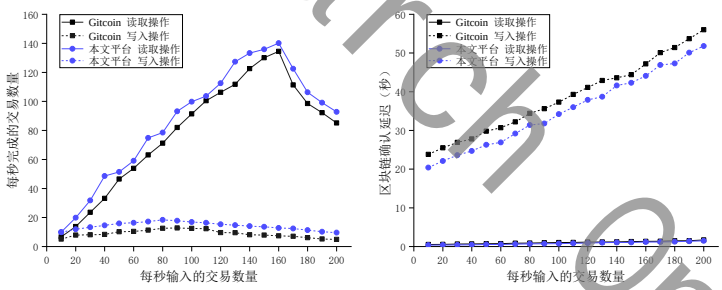
的干扰。表 5 中的实验结果表明，采用贡献点分配机制时，任务完成率提升至 32.4%，显著高于单一悬赏机制(20.4%)与平均分配机制(26.8%)，同时任务被忽略率降至 10.8%，明显提高了任务被完成的可能性，降低了任务被忽视的概率；在任务效率方面，其首次响应中位数与完整响应中位数分别缩短至 0.6 天与 48.6 天，均优于对照机制。上述结果表明，相较于基于固定悬赏或平均分配的机制，贡献点分配机制更能够有效地调动开发者的积极性，且其通过贡献与收益的动态绑定有效引导持续、高质量协作，提高了任务的完成率。

表 5 不同贡献计算方法收益对比

收益方式	任务完成率	任务被忽略率	首次响应中位数(天)	完整响应中位数(天)
单一悬赏机制	20.4%	25.3%	1.5	61.5
平均分配机制	26.8%	19.7%	1.2	53.7
(本文)贡献点分配机制	32.4%	10.8%	0.6	48.6

4.3 回答RQ4

本研究基于区块链技术设计实现了本文平台，以保障平台的透明性与可信性。然而，区块链智能合约带来的开销是不可避免的扩展性限制因素之一。为此，本文采用主流区块链技术 Ethereum 进行平台实现。Ethereum 具备约 40 笔交易每秒(transactions per second，tps)和 15 秒出块时间的性能表现，相较于 Gitcoin 仅支持约 7 笔交易每秒、出块时间约为 10 分钟的性能指标，Ethereum 展现出明显优势。为系统评估所提出技术的有效性，本文采用业界广泛认可的区块链性能评估工具 Hyperledger Caliper 进行测试[36]。实验同时收集并对比分析了 Gitcoin 平台的区块链数据，所使用的 Gitcoin 版本为 1.0 版。Gitcoin 是一个利用区块链技术促进开源软件开发的典型平台[37]。本研究重点考察了不同区块链操作的吞吐量与延迟，相关结果如图 14 所示。实验涉及读操作(如数据查询)与写操作(如数据更新)，向两个平台的智能合约提交了速率在 10 至 200 笔每秒的交易请求，随后使用 Hyperledger Caliper 工具收集交易完成数量及延迟数据，以评估系统性能。图 14 显示本文平台在区块链操作性能上略优于 Gitcoin。在相同的环境配置与输入参数下，本文平台平均每秒比 Gitcoin 多完成 8.21 笔读操作和 5.27 笔写操作。同时，在读操作方面，本文平台区块确认延迟比 Gitcoin 缩短了 0.21 秒；在写操作方面，延迟缩短了 3.33 秒。



(a) 区块链吞吐量 (b) 区块链延迟

图 14 Gitcoin 平台与本文平台的区块链性能分析

除上述主要结果外，我们还发现以下几点：首先，在读操作中，当输入速率达到约 160 笔每秒时，吞吐量逐渐趋于饱和；在写操作中，该峰值出现在约 90 笔每秒。这表明，无论是本文平台还是 Gitcoin，当负载过高时，均可能出现拥塞现象，从而限制平台的扩展性。针对这一问题，本文平台引入了事务缓冲机制，以应对高并发用户提交操作的场景，未来亦计划探索更先进的区块链扩展技术以进一步提升处理能力。其次，实验结果表明区块链延迟与负载呈线性增长关系，整体趋势可接受。鉴于写操作延迟普遍在数十秒左右，本文平台通过合并写操作以减少区块数量，进一步优化写入延迟。

总体上本文平台在操作成本方面相较于 Gitcoin 展现出明显优势，主要体现在吞吐量与延迟两个核心指标上。本文平台在相同负载条件下的吞吐量显著高于 Gitcoin，能够支持更大规模的并发交易处理，从而减少

了因交易堆积导致的额外资源消耗和确认成本。同时，在延迟方面，本文平台整体表现优于 Gitcoin，其平均交易确认延迟更低，系统能够更快速完成状态更新与数据同步，有效降低了开发者与用户在使用过程中的时间成本。

本文平台具备平衡安全性和负载效率的能力，以太坊主链的安全性相对副链 Polygon 较高，适合执行低负载的操作。在高负载的情况下对以太坊主链负荷较大，因此会迁移到以太坊副链 Polygon 上进行高负载操作。表 6 中的数据显示, 随着负载的增加，本文平台侧链的读写吞吐量快速上升，读写延迟逐步增加，写操作的延迟略高于读操作。这反映了在高负载环境下，侧链的性能略微受到资源瓶颈、网络拥堵等因素的影响，尽管在高负载下延迟表现略有下降，侧链仍能保持较高的读写吞吐量表现，系统仍能满足高并发请求，从而保障本文平台在高负载情况下的正常运行。

基于侧链的低成本特性，使得本文平台不仅适用于成熟的大型项目，也能够支撑微型协作任务和初创开源项目等资金受限场景，这一经济可行性拓宽了本文平台的应用前景。

表 6 本文平台侧链在高负载下的性能表现

负载(TPS)	读吞吐量 (TPS)	写吞吐量(TPS)	读延迟(ms)	写延迟(ms)
500	487.23	493.68	38.12	48.56
600	573.45	578.92	43.23	53.78
700	660.89	668.27	49.56	58.91
800	738.12	746.84	54.68	66.79
900	818.64	826.09	59.12	77.23
1000	896.47	905.73	64.45	88.34

此外，我们还对本文平台上的操作 Gas 消耗进行了评估。表 7 列出了不同类型交易在 Ethereum 网络上执行所需的平均 Gas 消耗量，并结合 2024 年 4 月 Ethereum 市场价格(2957.22 美元/ETH)及 Gas 价格(12Gwei)换算得出对应的美元费用。结果表明，使用 Ethereum 智能合约的经济成本相对较低。交易费用与操作的计算复杂度与存储复杂度密切相关，例如由于获取贡献点函数的工作流程较为复杂，其 Gas 消耗明显高于其他函数。同时，Ethereum 价格波动与网络拥塞程度亦对智能合约部署成本产生影响，因此，本研究中的费用数据特定于当前时期的市场与网络状况。此外，我们还进行了以太坊主链侧链操作成本的受控实验，我们将相同的智能合约逻辑与交互操作分别部署在以太坊主链与 Polygon 侧链，确保合约代码、调用参数及交互顺序完全一致。同时实验使用相同的节点配置与 Gas 上限设置。通过这一设计，我们最大限度地隔离了非技术性差异，从而确保比较结果能够准确反映链层性能与成本的内在差异。表 7 中的实验结果还显示在相同的合约逻辑和交互机制下，侧链在保持与主链接近的 Gas 消耗量的同时，大幅降低了交易的实际经济成本。例如，仲裁交易在主链的成本为 0.009 ETH(约 27.93 美元)，而在侧链上仅需 0.000231 ETH(约 0.72 美元)，降幅超过 95%；获取贡献点、贡献点使用以及欺诈举报等操作在侧链上的成本也均显著低于主链。该结果表明，当本文平台在高负载场景下运行时，可以通过低成本的侧链实现对主链的有效分流，从而在维持平台的正常运行。

在处理大规模源文件时，代码水印的嵌入与检测开销不会直接增加交易确认延迟。其原因在于本文平台采用主链与侧链分离的架构设计，以太坊主链主要承担交易提交、水印存证与状态记录等轻量级交互，而水印嵌入检测等高负载操作则迁移至 Polygon 侧链执行，因此，与代码水印相关的嵌入与检测等核心计算过程主要在侧链完成，会略微增加侧链的性能延迟，而不会直接转化为主链确认时延。表 6 中的侧链性能数据显示：即使在 500-1000 TPS 的高负载区间内，侧链读延迟仅由 38.12 ms 增至 64.45 ms，写延迟由 48.56 ms 增至 88.34 ms，整体仍保持在毫秒级，说明侧链能够较好承载大规模处理任务，从而为主链提供有效分流，避免高负载计算对主链确认过程造成显著影响。表 7 所统计的 Gas 消耗不包含执行复杂依赖分析和代码重排的计算成本，因为表 7 计算的是主链上的操作成本，而执行复杂依赖分析和代码重排的计算在侧链上。

表 7 本文平台的操作成本

交易类型	Gas(主链/侧链)	以太币(主链/侧链)	美元(主链/侧链)
获取贡献点	1731077/1679145	0.021/0.000504	61.43/1.47
贡献点使用	421985/405106	0.005/0.000122	14.98/0.36
仲裁	786942/771203	0.009/0.000231	27.93/0.72
软件交易	639083/607129	0.008/0.000182	22.68/0.52
欺诈举报	823471/798767	0.01/0.00024	29.22/0.70

此外，为进一步区分区块链机制与贡献点与协作机制单独带来的收益，我们从已参与本文区块链版本平台不同项目的用户中随机邀请 52 名不同背景的用户，同时参与本文非区块链版本平台的协作开发，并承担相同的项目任务。在完成部分任务后，我们邀请其填写对比用户体验的在线调查问卷[2]。招募过程中，开发者均被事先告知研究目的，并同意将开发成果与调查结果用于研究分析。受试者覆盖多类开发者群体，包括华为、阿里巴巴、百度、字节跳动、腾讯等企业员工以及流行开源仓库的贡献者，以保证样本的背景多样性与代表性，调查问卷结果如表 8 所示。

表 8 区块链和非区块链版本平台用户体验的调查结果

区块链和非区块链体验	增强信任	提升动力	减少信息泄露	额外开销
强烈同意	34.6%(15.4%)	30.8%(13.5%)	32.7%(11.5%)	23.1%(9.6%)
同意	42.3%(26.9%)	40.4%(19.2%)	38.5%(26.9%)	36.5%(19.2%)
中立	15.4%(34.6%)	19.2%(30.8%)	17.3%(32.7%)	21.2%(26.9%)
反对	5.8%(17.3%)	7.7%(28.8%)	7.7%(21.2%)	15.4%(30.8%)
强烈反对	1.9%(5.8%)	1.9%(7.7%)	3.8%(7.7%)	3.8%(13.5%)

在增强信任维度，区块链版本同意及以上为 76.9%，明显高于非区块链版本的 42.3%；同时其反对及以下仅为 7.7%，显著低于非区块链版本的 23.1%，表明区块链版本在信任构建上优于非区块链版本。其次，在提升动力维度，区块链版本同意及以上为 71.2%，而非区块链版本仅为 32.7%；与此同时，区块链版本反对及强烈反对为 9.6%，显著低于非区块链版本的 36.5%，说明区块链版本更能提升开发者的动力。在减少信息泄露维度，区块链版本同意及以上为 71.2%，显著高于非区块链版本的 38.4%，反映出区块链版本在信息安全保护方面更受到用户认可。但在额外开销维度，区块链版本存在额外开销的同意及以上为 59.6%，高于非区块链版本的 28.8%，说明区块链机制在带来信任与安全收益的同时，也引入了额外的开销，如 Gas 费用等。

此外，我们进一步定性论证了在不考虑用户心理因素前提下，区块链的“不可篡改性”可通过降低协作开发中的核验成本和降低协作开发中的争议处置成本两条路径提升开发效率。一方面不可篡改的链上记录能够为贡献归属、权限状态、收益结算等关键事项提供统一、可验证的事实依据，并结合智能合约的确定性执行，减少中心化平台协作开发过程中存在的重复人工确认、交叉核对、多轮沟通以及流程等待所导致的开发效率低下问题，从而促进开发效率的提升。另一方面，依托一致且可追溯的链上证据，平台能够更快界定协作开发过程中的责任边界、缩短仲裁流程，并减少因证据不一致引发的开发任务中断与返工，从而促进开发效率的提升。相比之下，传统中心化平台通常依赖平台侧数据库与日志维护贡献记录、权限变更、收益分配及争议处理结果，容易面临单点修改、追溯链条不完整、责任界定困难及任务阻塞等问题，制约了开发效率的提升。

4.4 回答RQ5

为了更深入地了解本文各个核心模块存在的必要性，本文进一步开展了为期一个月的消融实验：在任务集合、参与者规模和实验周期等保持一致的条件下，分别单独移除渐进式开放模型、获得贡献点模块、贡献

2 https://docs.google.com/forms/d/e/1FAIpQLSffhD2pG0EYRwjODd5wOUi0zbET0GGlJ6f4yUzDp7b-QEtj9Q/viewform

点使用模块、仲裁模块、信用分模块和水印追踪模块，并通过关键指标评估各个模块存在的必要性和有效性。为了更好地进行定量分析，我们定义了如下关键指标：

贡献审核耗时（小时/次）：指单次贡献从提交至得到审核结果的平均耗时，用于衡量贡献审核流程的执行效率。

用户日均活跃度（人/日）：指实验周期内平均每日完成至少一次有效操作（如提交贡献、参与仲裁、使用贡献点交换代码访问权限等）的独立活跃用户数量均值，用于衡量用户参与积极性。

贡献点流通量（次/日）：指通过贡献点完成代码访问权限兑换或激励发放的日均有效交易次数，用于衡量平台激励的活跃程度。

争议未解决率（%）：指平台上的争议事件在实验周期内未能达成任何形式的解决共识的事件占比，用于衡量平台对争议事件的处理能力。

高信用用户流失率（%）：指信用分排名前 20%的用户在实验周期内停止登录或注销账号的比例，用于衡量平台对核心贡献者的留存能力。

追踪泄露次数（次/日）：指平台成功追踪到项目代码未经授权泄露的日均次数，用于衡量版权保护机制的追责与保护能力。

表 9 各核心模块消融实验对比

移除模块	关键指标	完整平台	消融后	变化幅度
渐进式开放模型	贡献审核耗时（小时/次）	0.4	0.6	+50.0%
获得贡献点模块	用户日均活跃度（人/日）	126	48	-61.9%
贡献点使用模块	贡献点流通量（次/日）	84	0	-100.0%
仲裁模块	争议未解决率（%）	2.4%	92.8%	+90.4%
信用分模块	高信用用户流失率（%）	12	37	+208.3%
水印追踪模块	追踪泄露次数（次/日）	6	0	-100.0%

表 9 的实验结果表明，各核心模块在平台运行中均具有不可或缺的作用。具体而言，移除渐进式开放模型后，贡献审核耗时由 0.4 小时/次上升至 0.6 小时/次，增幅为 50.0%，说明该模型有助于提升贡献审核效率并减轻审核负担；移除获得贡献点模块后，用户日均活跃度由 126 人/日降至 48 人/日，降幅为 61.9%，表明缺乏有效的激励入口将显著削弱用户参与动力；移除贡献点使用模块后，贡献点流通量由 84 次/日降至 0 次/日，平台内基于贡献点的代码访问权限兑换与收益分配均无法继续执行，说明该模块是维持激励闭环正常运转的关键支撑；移除仲裁模块后，平台不提供任何官方裁决通道。争议双方只能自行协商或放弃。由于缺乏中立第三方，大量争议将无法达成一致，争议未解决率由 2.4% 增至 92.8%，表明仲裁机制缺失将严重削弱平台的争议处置能力；移除信用分模块后，高信用用户流失率由 12% 升至 37%，说明信用约束对于维持核心用户稳定性和长期参与具有重要作用；移除水印追踪模块后，追踪泄露次数由 3 次/日降至 0 次/日，平台失去对代码泄露事件进行有效追踪与责任定位的能力，表明该模块对版权保护与责任追溯具有关键支撑作用。

基于上述数据，本文进一步讨论了“可信”“清晰的贡献回报”“可验证”三种特征与提升协作意愿之间的关联。对于“可信”特征，信用分模块是其核心组成，移除后高信用用户流失率上升了 25%，表明当可信机制缺失时，核心贡献者的留存意愿显著下降，间接支撑了“可信”对维持协作意愿的重要作用。对于“清晰的贡献回报”特征，获得与使用贡献点是实现该特征的主要环节：移除获得模块后活跃度下降 61.9%，移除使用模块后流通量降至 0，说明当开发者无法清晰获得并兑现贡献回报时，协作开发积极性会明显降低。对于“可验证”特征，水印追踪模块是实现版权可验证追溯的核心手段，移除后追踪泄露次数由 3 次/日降至 0 次/日，开发者将失去对劳动成果被泄露后可追责、可验证的制度保障，这会削弱其对成果安全性的预期信心，并在长期上可能抑制协作参与意愿。

5 有效性威胁

5.1 外部有效性

本研究的实验主要基于以太坊测试网络及特定平台的历史数据，这些环境在网络拓扑、节点规模及交易特性上与真实生产环境可能存在差异，因而限制了结论在其他区块链网络或不同业务场景下的直接推广性。为降低此类威胁，本文通过多次实验取均值、引入不同负载条件和跨平台对比等方法增强结果的稳健性。未来，我们将扩展至多条公链和联盟链的真实部署场景，并采集来自不同时间段与地域的异构数据，以进一步验证结论的普适性与外推能力。当前样本主要来自特定企业背景及部分流行开源仓库贡献者，虽然覆盖了一定量的样本，但总体样本规模仍然有限，可能无法充分代表更广泛开源社区中不同角色参与者的实际情况。未来我们计划进一步扩大样本规模，纳入更多不同背景的用户，开展更大规模的实证研究，以增强结论的普适性与稳健性。

5.2 内部有效性

在问题响应时间的分析中，我们仅纳入了成功领取悬赏奖励的案例，可能导致结果对真实响应效率的反映存在偏差。为降低此类威胁，本文在实验中采用重复测量与均值化处理，并严格控制变量。未来，将引入更丰富的数据样本，包括未领取悬赏奖励但已解决的问题，以更全面地刻画平台运行特征。此外，本文尚缺乏区块链版本平台与中心化版本平台之间的直接客观架构对比实验，现有实验验证的是平台整体方案在开发积极性、响应效率与安全性方面的综合效果，但尚无法严格区分这些效果究竟源于区块链所具备的可追溯、不可篡改、可验证等特性，还是源于贡献点、渐进式开放模型、押金与信用积分等业务机制本身的设计优势。

现有消融实验尚不能完全分离“可信”“清晰的贡献回报”“可验证”三类特征的独立作用，当前实验结果更多反映的是相关模块缺失带来的综合效应。未来我们计划探索通过因子分离实验，在保持其他特征不变的条件下分别移除单一特征，并结合用户归因问卷进一步验证因果关系。

5.3 构造有效性

本文在安全机制设计中重点关注智能合约与代码水印的结合，但现有测量指标仍难以覆盖全部潜在安全风险，未来我们计划补充更多安全性、可扩展性与可维护性等综合评估指标，以更全面反映平台在复杂业务环境下的表现。就水印算法本身而言，本文采用的保守水印算法目前适用于静态源代码中顺序语义明确、可静态分析的局部语句块，对复杂异步调用、反射机制及特定硬件指令序列等代码区域则适用性受限，未来我们将探索基于动态代码水印的扩展机制，在保持程序语义与控制运行开销的前提下，增强对复杂代码结构的版权追踪能力。在模块评估层面，本文通过贡献审核耗时、用户日均活跃度等指标评估各模块的必要性与有效性，但上述行为类指标尚难以全面刻画平台开放性、激励有效性、治理公平性与资源可追溯性等整体特征，未来我们将扩展评估指标体系并结合更长周期实验，对各模块作用进行更系统的检验，以增强结论的可靠性与稳健性。

5.4 结论有效性

在高负载环境下，本文平台侧链的性能略微受到资源瓶颈、网络拥堵等因素的影响，尽管在高负载下延迟表现略有下降，侧链仍能保持较高的读写吞吐量表现，系统仍能满足高并发请求，从而保障本文平台在高负载情况下的正常运行。未来我们将考虑引入分片技术、Rollup 等 Layer-2 扩容机制，以降低交易开销、提升吞吐能力，从而增强平台在高负载环境下的实用性和可扩展性。

6 相关工作

作为典型的虚拟群体开发社区，开源软件项目汇聚了来自不同国家与背景的开发者。开发者通过加入开源社区，与其他开发者协作完成项目开发。目前，知名的开源社区包括 GitHub 与 SourceForge 等[38]。与传统

软件项目相比，开源软件项目开发具有参与门槛低、自由度高等诸多优势。然而，不可否认的是，开源项目在项目质量与可持续性方面仍存在一定的不足[39]。其中最为突出的挑战在于缺乏有效的可持续发展机制。例如，核心开发者长期中断或退出可能导致项目停滞[40]，或者由于缺乏激励机制，项目无法实现持续运行与良性循环[41]。开源项目要实现可持续发展，关键在于维持一定数量的活跃提交者与用户[42]，并建立可靠的管理机制以平衡多方参与者之间的权利、责任与利益[43]。

近年来，捐赠与众筹模式在开源领域日益普及。2019 年 5 月，GitHub 推出了 GitHub Sponsors 服务[44]，允许开源软件开发者直接接受其他 GitHub 用户的资金支持。与以往主要针对具体项目的开源捐赠服务不同，GitHub Sponsors 的独特之处在于支持向个人开发者捐赠。尽管该服务已运行超过三年，但仍有部分开发者反馈未能获得预期的赞助效果。OpenCollective 作为另一个典型平台，通过设定捐赠金额与持续性挂钩的贡献等级，配备预算目标设定、社区支出管理与开放透明账目功能[44-45]，在一定程度上改善了传统捐赠模式存在的透明度不足问题。相较于传统基于区块链的软件管理平台，如 Gitcoin 仅记录项目财务交易与人员信息的做法[46]，本文平台在引导用户贡献和实现激励公平分配方面开创了新的路径。具体而言，本文平台不仅涵盖了基本的财务与人员信息记录，还基于区块链与智能合约机制，明确设定了贡献点获取标准，有效激励了用户的积极参与。平台通过自动化与透明化的激励分配机制，消除了传统中介环节，确保了奖励的公平及时分配。此外，本文平台支持平台功能的灵活扩展与项目的定制开发，能够快速响应市场变化与技术环境的演进，进一步满足开发者多样化与个性化的需求。

7 结论和未来工作

本文提出了一种基于区块链的开放协作开发供应链可信管理平台，创新性地引入贡献点份额概念，依据用户贡献比例公平分配项目收益，并通过区块链技术保障开发者应得权益的兑现。针对开发过程中可能出现的恶意行为与代码泄露问题，本文设计了基于押金与信用分数的安全机制，并结合博弈论对不同决策场景下的用户收益进行了建模与仿真，实验结果表明本文平台有效降低了协作开发过程中软件信息泄露与恶意行为的风险，为软件供应链安全提供了有力保障。需要指出的是，本文平台并不直接面向恶意代码注入、依赖项污染等典型软件供应链安全问题的识别与检测，而是通过抑制贡献信息篡改、收益分配不公、权限滥用与责任追溯困难等治理风险，并增强协作透明性与规则执行一致性，为软件供应链的过程安全提供间接支撑。

在与基线平台的对比实验中，本文平台的任务完成率较 BountySource 高出 12.5%，首次响应时间与完整响应时间在短时间区间内分别降低 4.2% 与 8.5%，显示出在激发开发积极性与缩短响应周期方面的优势；在吞吐量与延迟指标上，本文平台均优于 Gitcoin，且系统开销保持在合理范围之内。在易用性方面，本文平台通过简洁的交互界面对底层复杂机制进行了封装，并辅以技术文档、示例代码与教程，使开发者无需理解技术细节即可参与协作；同时对高负荷操作引入提交频率提示，避免资源过度消耗。未来，我们将进一步优化平台的用户界面与系统架构，降低普通开发者的使用门槛，提升整体用户体验。在应用扩展方面，我们计划将本文平台拓展至知识产权领域：依托区块链不可篡改特性记录与验证创作者的原创性成果版权，并通过智能合约自动执行授权条款与权益分配，以去中心化方式提升知识产权保护效率，降低传统中介环节的成本与风险。

最后，我们计划在未来进一步优化智能合约与区块链平台的性能，以满足大规模交易场景下对系统吞吐量与响应速度的更高要求。

References:

[1] Constantino K, Souza M, Zhou S, Figueiredo E, Kästner C. Perceptions of open-source software developers on collaborations: An interview and survey study. Journal of Software: Evolution and Process, 2023, 35(5): e2393. [doi: 10.1002/smr.2393]

[2] Assavakamhaenghan N, Tanaphantaruk W, Suwanworaboon P, Choetkiertikul M, Tuarob S. Quantifying effectiveness of team recommendation for collaborative software development. Automated Software Engineering, 2022, 29(2): 51. [doi: 10.1007/s10515-022-00357-7]

[3] Avelino G, Constantinou E, Valente M T, Serebrenik A. On the abandonment and survival of open source projects: An empirical investigation. In: Proc. of the 2019 ACM/IEEE Int'l Symp. on Empirical Software Engineering and Measurement. Porto de Galinhas: IEEE, 2019. 1-12. [doi: 10.1109/esem.2019.8870181]

[4] Shimada N, Xiao T, Hata H, Treude C, Matsumoto K. Github sponsors: Exploring a new way to contribute to open source. In: Proc. of the 44th Int'l Conf. on Software Engineering. Pittsburgh: ACM, 2022. 1058-1069. [doi: 10.1145/3510003.3510116]

[5] Zhang Y, Qin M, Stol K-J, Zhou M, Liu H. How are paid and volunteer open source developers different? A study of the Rust project. In: Proc. of the IEEE/ACM 46th Int'l Conf. on Software Engineering. Lisbon: ACM, 2024. 2406-2418. [doi: 10.1145/3597503.3639197]

[6] Kalliamvakou E, Damian D, Blincoe K, Singer L, Germán D M. Open source-style collaborative development practices in commercial projects using GitHub. In: Proc. of the 37th IEEE/ACM Int'l Conf. on Software Engineering. Florence: IEEE, 2015. 574-585. [doi: 10.1109/ICSE.2015.74]

[7] Warner J, Guo P J. Codepilot: Scaffolding end-to-end collaborative software development for novice programmers. In: Proc. of the 2017 CHI Conf. on Human Factors in Computing Systems. Denver: ACM, 2017. 1136-1141. [doi: 10.1145/3025453.3025876]

[8] Accioly P. Understanding conflicts arising from collaborative development. In: Proc. of the 37th IEEE/ACM Int'l Conf. on Software Engineering. Florence: IEEE, 2015. 775-777. [doi: 10.1109/ICSE.2015.246]

[9] Wermke D, Wöhler N, Klemmer J H, Fourné M, Acar Y, Fahl S. Committed to trust: A qualitative study on security & trust in open source software projects. In: Proc. of the 2022 IEEE Symp. on Security and Privacy. San Francisco: IEEE, 2022. 1880-1896. [doi: 10.1109/SP46214.2022.9833686]

[10] Oláh J, Hidayat Y A, Popp J, Lakner Z, Kovács S, Erdei E. The effect of integrative trust and innovation on financial performance in a disruptive era. Economics & Sociology, 2021, 14(2): 111-136. [doi: 10.14254/2071-789X.2021/14-2/6]

[11] Lazear E P, Rosen S. Rank-order tournaments as optimum labor contracts. Journal of Political Economy, 1981, 89(5): 841-864. [doi:10.1086/261010]

[12] Zhou J, Wang S, Kamei Y, Hassan A E, Ubayashi N. Studying donations and their expenses in open source projects: A case study of GitHub projects collecting donations through open collectives. Empirical Software Engineering, 2022, 27(1): 24. [doi: 10.1007/s10664-021-10060-y]

[13] Schmid S, Shestakov D. Invited paper: Blockchain governance and liquid democracy—quantifying decentralization in Gitcoin and Internet Computer. In: Proc. of the 2024 Workshop on Advanced Tools, Programming Languages, and Platforms for Implementing and Evaluating Algorithms for Distributed Systems. Nantes: ACM. 2024. 1-7. [doi: 10.1145/3663338.3663678]

[14] Monrat A A, Schelén O, Andersson K. A survey of blockchain from the perspectives of applications, challenges, and opportunities. IEEE Access, 2019, 7: 117134-117151. [doi: 10.1109/ACCESS.2019.2936094]

[15] Zhou L, Tang C, Bao Z, Hu J, Liu Y, Gao Y. A reputation-based blockchain scheme for sustained carbon emission reduction. Science China Information Sciences, 2024, 67(5): 152104. [doi: 10.1007/s11432-023-3967-0]

[16] Nguyen H-L, Ignat C-L, Perrin O. Trusternity: Auditing transparent log server with blockchain. In: Companion Proc. of The Web Conf. 2018. Lyon: ACM, 2018. 79-80. [doi: 10.1145/3184558.3186938]

[17] Li X. An anti-tampering model of sensitive data in link network based on blockchain technology. Web Intelligence, 2021, 19(3): 227-237. [doi: 10.3233/WEB-210469]

[18] Gervais A, Karame G O, Wüst K, Glykantzis V, Ritzdorf H, Capkun S. On the security and performance of proof of work blockchains. In: Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security. Vienna: ACM, 2016. 3-16. [doi: 10.1145/2976749.2978341]

[19] Zou W, Lo D, Kochhar P S, Le X-B D, Xia X, Feng Y, Chen Z, Xu B. Smart contract development: Challenges and opportunities. IEEE Transactions on Software Engineering, 2021, 47(10): 2084-2106. [doi: 10.1109/TSE.2019.2942301]

[20] Zheng Z, Xie S, Dai H-N, Chen W, Chen X, Weng J, Imran M. An overview on smart contracts: Challenges, advances and platforms. Future Generation Computer Systems, 2020, 105: 475-491. [doi: 10.1016/j.future.2019.12.019]

[21] Luu L, Chu D-H, Olickel H, Saxena P, Hobor A. Making smart contracts smarter. In: Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security. Vienna: ACM, 2016. 254-269. [doi: 10.1145/2976749.2978309]

[22] Bonaccorsi A, Rossi C. Why open source software can succeed. Research Policy, 2003, 32(7): 1243-1258. [doi: 10.1016/S0048-7333(03)00051-9]

[23] Hechtl C, Joblin M, Apel S. Is perceived gender related to contributions and standing in open-source software projects? Empirical Software Engineering, 2025, 30(5): 123. [doi: 10.1007/s10664-025-10653-x]

[24] Jahn L, Engelbutzeder P, Randall D, Bollmann Y, Ntouros V, Michel L K, Wulf V. In between users and developers: Serendipitous connections and intermediaries in volunteer-driven open-source software development. In: Proc. of the CHI Conf. on Human Factors in Computing Systems. Honolulu: ACM, 2024. 1-15. [doi: 10.1145/3613904.3642541]

[25] Yenişen Yavuz E, Riehle D, Mehrotra A. Why do companies create and how do they succeed with a vendor-led open source foundation. Empirical Software Engineering, 2025, 30(1): 40. [doi: 10.1007/s10664-024-10588-9]

[26] Li Z, Wang C, Wang S, Gao C. Protecting intellectual property of large language model-based code generation APIs via watermarks. In: Proc. of the 2023 ACM SIGSAC Conf. on Computer and Communications Security. Copenhagen: ACM, 2023. 2336-2350. [doi: 10.1145/3576915.3623120]

[27] Miller M, Doerr G, Cox I. Applying informed coding and embedding to design a robust high-capacity watermark. IEEE Transactions on Image Processing, 2004, 13(6): 792-807. [doi: 10.1109/TIP.2003.821551]

[28] Ma H, Jia C, Li S, Zheng W, Wu D. Xmark: Dynamic software watermarking using Collatz conjecture. IEEE Transactions on Information Forensics and Security, 2019, 14(11): 2859-2874. [doi: 10.1109/TIFS.2019.2908071]

[29] Collberg C, Carter E, Debray S, Huntwork A, Kececioglu J, Linn C, Stepp M. Dynamic path-based software watermarking. In: Proc. of the ACM SIGPLAN 2004 Conf. on Programming Language Design and Implementation. Washington: ACM, 2004. 107-118. [doi: 10.1145/996841.996856]

[30] Ren C, Chen K, Liu P. Droidmarking: Resilient software watermarking for impeding Android application repackaging. In: Proc. of the 29th ACM/IEEE Int'l Conf. on Automated Software Engineering. Vasteras: ACM, 2014. 635-645. [doi: 10.1145/2642937.2642977]

[31] Li X, Zhang Y, Osborne C, Zhou M, Jin Z, Liu H. Systematic literature review of commercial participation in open source software. ACM Transactions on Software Engineering and Methodology, 2025, 34(2): 1-31. [doi: 10.1145/3690632]

[32] Ye V, Li R, Kerr J, Turkulainen M, Yi B, Pan L, Seiskari O, Ye J, Hu J, Tancik M, Kanazawa A. gsplat: An open-source library for Gaussian splatting. Journal of Machine Learning Research, 2025, 26(34): 1-17.

[33] Miller C, Jahanshahi M, Mockus A, Vasilescu B, Kästner C. Understanding the response to open-source dependency abandonment in the npm ecosystem. In: Proc. of the 47th IEEE/ACM Int'l Conf. on Software Engineering. Ottawa: IEEE, 2025. 2355-2367. [doi: 10.1109/ICSE55347.2025.00004]

[34] Wellman M P, Tuyls K, Greenwald A. Empirical game theoretic analysis: A survey. Journal of Artificial Intelligence Research, 2025, 82: 1017-1076. [doi: 10.1613/jair.1.16146]

[35] Zhou J, Bezemer C-P, Zou Y, Hassan A E. Studying the association between Bountysource bounties and the issue-addressing likelihood of GitHub issue reports. IEEE Transactions on Software Engineering, 2021, 47(12): 2919-2933. [doi: 10.1109/TSE.2020.2974469]

[36] Choi W, Hong J W-K. Performance evaluation of Ethereum private and testnet networks using Hyperledger Caliper. In: Proc. of the 2021 22nd Asia-Pacific Network Operations and Management Symposium. Tainan: IEEE, 2021. 325-329. [doi: 10.23919/APNOMS52696.2021.9562684]

[37] Patrickson B. What do blockchain technologies imply for digital creative industries? Creativity and Innovation Management, 2021, 30(3): 585-595. [doi: 10.1111/caim.12456]

[38] Joo C, Kang H, Lee H. Anatomy of open source software projects: Evolving dynamics of innovation landscape in open source software ecology. In: Proc. of the 5th Int'l Conf. on Communications, Computers and Applications. Istanbul: IEEE, 2012. 96-100.

[39] Alami A. The sustainability of quality in free and open source software. In: Proc. of the ACM/IEEE 42nd Int'l Conf. on Software Engineering: Companion Proceedings. Seoul: ACM, 2020. 222-225. [doi: 10.1145/3377812.3381402]

[40] Foucault M, Palyart M, Blanc X, Murphy G C, Falleri J-R. Impact of developer turnover on quality in open source software. In: Proc. of the 2015 10th Joint Meeting on Foundations of Software Engineering. Bergamo: ACM, 2015. 829-841. [doi: 10.1145/2786805.2786870]

[41] Xu H, Wan J. Innovation in open source software with knowledge: Three challenges for open source competence centres. In: Proc. of the 2008 4th Int'l Conf. on Wireless Communications, Networking and Mobile Computing. Dalian: IEEE, 2008. 1-4. [doi: 10.1109/WiCom.2008.2046]

[42] Iaffaldano G, Steinmacher I, Calefato F, Gerosa M, Lanubile F. Why do developers take breaks from contributing to OSS projects? A preliminary analysis. In: Proc. of the 2nd Int'l Workshop on Software Health. Montreal: IEEE, 2019. 9-16. [doi: 10.1109/SoHeal.2019.00009]

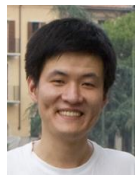
[43] Silva J O, Wiese I S, German D M, Steinmacher I, Gerosa M A. How long and how much: What to expect from Summer of Code participants? In: Lo D, Saraiva J, eds. Proc. of the 2017 IEEE Int'l Conf. on Software Maintenance and Evolution. Shanghai: IEEE, 2017. 69-79. [doi: 10.1109/ICSME.2017.81]

[44] Tsakpinis A, Pretschner A. Analyzing the usage of donation platforms for PyPI libraries. In: Proc. of the 29th Int'l Conf. on Evaluation and Assessment in Software Engineering. Istanbul: ACM, 2025. 628-633. [doi: 10.1145/3756681.3757018]

[45] Overney C. Hanging by the thread: An empirical study of donations in open source. In: Proc. of the ACM/IEEE 42nd Int'l Conf. on Software Engineering: Companion Proceedings. Seoul: ACM, 2020. 131-133. [doi: 10.1145/3377812.3382170]

[46] Tan X, Hou B, Ni X, Zhang Y, Jiang J, Zhou M, Zhang L. Facilitating wise decision-making for bounty backers in open source software communities. IEEE Transactions on Software Engineering, 2025, 52(1): 266-285. [doi: 10.1109/TSE.2025.3633310]

张硕骁(1995－)，男， 南京大学软件学院博士生，主要研究领域为区块链安全技术，实证软件工程，软件测试.



汤恩义(1982－)，男，博士，南京大学软件学院副教授， 博士生导师， CCF 专业会员， 主要研究领域为程序分析， 数值分析， 软件测试.

张浩枫(2004－)，男， 南京大学软件学院本科生，主要研究领域为实证软件工程.

单一啸(2004－)，男， 南京大学软件学院本科生，主要研究领域为可持续开源软件生态.

潘佳滨(2005－)，男， 南京大学软件学院本科生，主要研究领域为软件工程.

刘子豪(2005－)，男， 南京大学软件学院本科生，主要研究领域为软件工程.

成浩亮(1994－)，男， 南京大学软件学院博士生，主要研究领域为软件测试.

赵建华(1971－)，男，博士，南京大学计算机学院教授，博士生导师生，主要研究领域为形式化方法，软件工程，程序设计语言.

陈鑫(1972－)，男，博士，南京大学计算机学院副教授，主要研究方向为形式化方法，可信 AI.

李宣东(1963－)，男，博士，南京大学计算机学院教授，博士生导师生，CCF 会士，主要研究领域为软件工程，系统软件，可信软件，形式化方法.