Smart Contract as a Service: A Paradigm of Reusing Smart Contract in Web3 Ecosystem

Jinghan Sun
The Chinese University of Hong Kong, Shenzhen
Mohamed Bin Zayed University of Artificial Intelligence

Wei Cai
The Chinese University of Hong Kong, Shenzhen

Abdulmotaleb El Saddik
Mohamed Bin Zayed University of Artificial Intelligence
University of Ottawa

***Abstract*—Recently, smart contract, a code script that autonomously runs on blockchains, has facilitated numerous decentralized applications and accelerated the boost of the Web3 ecosystem. However, the current underutilization of smart contracts and a large number of duplicate smart contracts have caused significant challenges to blockchains. Considering the reusability and composability of smart contracts, we can innovatively address these issues, promote diversity for decentralized projects, and increase efficiency for Web3 developers. In this tutorial paper, we first propose a novel service-oriented paradigm: Smart Contracts as a Service (SCaaS), then analyze the feasibility of SCaaS, delve into its application scenarios and finally explore the challenges and limitations and propose the potential solutions accordingly.**

■ **WEB3, THE NEXT** generation of the Internet, provides a user-centric on-chain ecosystem. It emerges from blockchain technology, which endows the on-chain data and transaction records with transparency through decentralized ledger. Blockchain technology utilizes consensus algorithms to ensure that the data of each decentralized node is unified and tamper-proof. This characteristic has attracted a

Digital Object Identifier 10.1109/MCE.2024.3353178
Date of publication 12 January 2024; date of current version 5 December 2024.

large number of developers to participate in the blockchain ecosystem construction. Take Ethereum, the public blockchain with one of the most thriving developer ecosystems, as an example: the number of on-chain smart contracts has increased cumulatively from 5.6 million in September 2018 to 61 million in September 2023.[a] While the rapid emergence of blockchain projects attracts users to join Web3 markets, the drawbacks of the increasing number of smart contracts are becoming nonnegligible.

First, on-chain storage pressure. On-chain data's accumulation in the distributed ledger leads to a continuous expansion of on-chain data. Currently, the size of Ethereum's full node has surged from 2.0 terabytes in January 2019 to 15.7 terabytes.[b] The excessive demand for data storage undoubtedly escalates the threshold for full nodes to join the blockchain network, thereby reducing blockchain decentralization.[1]

Second, increasing development costs. Development of advanced functionalities for blockchain projects increases development costs.[2] Besides, the influx of decentralized projects intensifies the competition in the Web3 market. Consequently, project developers resort to strategies like token airdrops, whitelists, advertising promotions, and community-building, which have greatly raised the cost of user acquisition in the Web3 market.[3,4]

Third, project ecosystems barriers. The competition in the Web3 market is not only reflected in attracting users but also in encouraging developers to take part in the project building and exploring the functional diversity. However, the competitive relationships between projects broaden the ecosystem barriers and curtail cross-project interoperability.

Last, smart contract security threats. In the current decentralized market, numerous unaudited or risky smart contracts still exist. Consequently, participants in the Web3 ecosystem may face potential threats to their asset security due to technical vulnerabilities within these smart contracts.

Given the reusability of smart contract, it is necessary to promote the reuse of pre-existing smart contracts with desired functionalities during new smart contract development. This approach not only addresses the drawbacks mentioned above but offers several intuitive advantages shown as follows:

Alleviating storage pressure. By reusing the deployed smart contracts, developers can minimize the duplication of the smart contracts on the blockchain, and thus ease the storage pressure of full nodes, and prevent state bloat[c] of blockchain.,[5,6]

Reducing marginal costs. Reusing smart contracts can reduce marginal costs in project development and user acquisition. Developers can save time and effort by reusing pre-existing smart contracts when developing novel ones.[7] Meanwhile, users who trust the reused smart contracts are more likely to migrate to the new project, reducing user acquisition costs and increasing the potential user base.[8]

Building an interconnected ecosystem. Open source smart contracts enable permissionless collaborative development and secondary reconstruction. This paves the way for interoperability between derivative projects from the same original projects, creating an interconnected ecosystem among these decentralized projects. The interaction of functionalities and asset interoperability between these derivative projects further enhances the diversity of the financial system and delivers a richer user experience.

Bolstering project security. Reusing time-tested and reliable smart contracts in new projects, developers can reduce the security vulnerabilities in new contracts, thereby bolstering the overall security of the novel projects,[7,9] and enhancing network effects[d] by migrating user trust of the reused smart contracts.

Considering these advantages, existing literature extensively explores smart contract reuse. Chen et al.[10] implemented an empirical study about the reuse of open source smart

[a][Online]. Available: https://dune.com/sharptraderx/scdot
[b][Online]. Available: https://etherscan.io/chartsync/chainarchive

[c]State bloat refers to the situation where a state data grows so large that it is difficult for the full nodes to handle, store, and share its data efficiently.
[d]Network effects denotes the increase of projects value as their user base expands.

contracts on Ethereum, revealing that most reused subcontracts pertain to token-issuing, and developers tend not to revise the original smart contracts during the reuse process. Guida et al.[11] proposed a platform to store smart contract descriptions to expedite the existing smart contracts' discovery and reuse, thereby accelerating smart contract development. Gec et al.[12] introduced a system offering smart contract templates adaptable for merging, extension, or modification, aiding developers in reutilizing these templates. Despite highlighting the significance of reusing smart contracts and aiming to simplify the process, none of these studies propose a paradigm for volume-producing new smart contracts through reusability and composability or analyze the influence of reuse smart contracts on the Web3 ecosystem. From this point, we propose the Smart Contract as a Service (SCaaS) paradigm, leveraging smart contracts as computational components to achieve the smart contracts' reusability and composability in service-oriented computing in blockchain. Through the SCaaS developers can not only streamline the development of Web3 projects by reusing and composing existing smart contracts akin to "Lego bricks" but also capitalize on the established user base of these smart contracts. To the best of our knowledge, this is the first work to propose the SCaaS paradigm based on the Web3 ecosystem.

THE INNOVATION OF SCaaS PARADIGM

As a cloud computing service, the "as-a-service" paradigm has garnered significant attention.[13,14,15,16,17] It can be defined as a pay-as-you-go model that offers users web-based value-added services, enhancing resource utilization and cost-effectiveness by using cloud computing resources.[13,14,15] The widely used "as-a-service" models encompass Infrastructure-as-a-Service (IaaS), offering fundamental cloud computing infrastructure to developers, Platform-as-a-service (PaaS), providing developers with a cloud platform for software development, and Software-as-a-service (SaaS), delivering software applications to end-users.

In addition to PaaS and SaaS, Blockchain-as-a-Service (BaaS), combining cloud computing service and blockchain technology has emerged as a well-known approach in blockchain development.[16,17,18] Operates similarly to PaaS, BaaS is a cloud-based service tailored to help users build new blockchains[e] and offers blockchain-related application development. However, BaaS primarily focuses on building new blockchains, rather than developing new smart contracts. Some providers, such as Microsoft ABW BaaS and IBM BaaS enable efficient smart contract development by offering templates or visualized smart contract modules, rather than reusing existing smart contracts, which does not address the mentioned issues arising from the smart contracts' duplication.

Different from traditional service computing models,[f] blockchain divides service provider into two separate roles: algorithm provider and computing resource provider, as shown in Figure 1. In the SCaaS paradigm, blockchain operates as the platform for executing smart contracts and providing algorithm services to Web3 project developers. More specifically, Web3 developers propose their project requirements to the service provider. Subsequently, the computing resource provider, usually the miner, provides computational resources to execute the smart contracts. Additionally, the algorithm provider is responsible for analyzing the existing smart contracts that satisfy developers' demands and then provides the corresponding smart contracts to complete the desired new smart contracts. Furthermore, an algorithm provider may present multiple smart contracts for different algorithms.

The comparison of IaaS, BaaS, PaaS, SaaS, and SCaaS is illustrated in Table 1. The SCaaS we proposed is a novel service paradigm targeting smart contract developers. It offers a distinct advantage by smart contract service provision,

[e]Including public chains, private chains, and mostly consortium chains based on technologies like Hyperledger.

[f]In traditional service computing (e.g., cloud computing service), the service provider provides both algorithm services and computing resources to run the algorithm in their server.

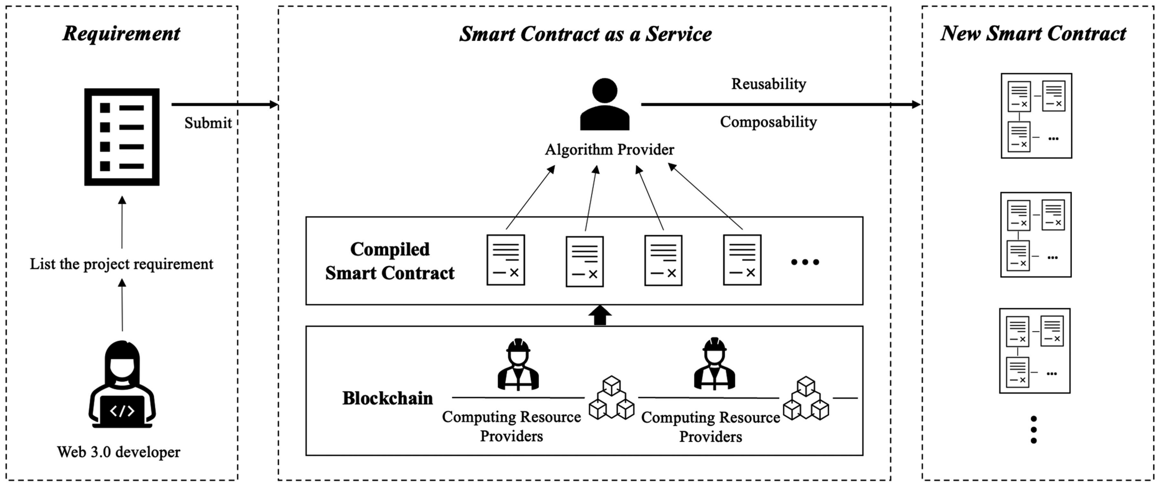

FIGURE 1. Framework of Smart Contract as a Service.

effectively resolving the problems associated with smart contract duplication.

This tutorial paper summarizes the drawbacks posed by the proliferation of smart contracts and outlines recent practices of smart contracts' reusability and composability in the Web3 ecosystem. The contributions of this article can be summarized as follows:

- We analyze the potential of addressing problems introduced by smart contract duplication through smart contract's reusability and composability, summarizing the advantages, types, and industrial practices of these two characteristics.
- We propose the SCaaS, a novel service-oriented paradigm for developers, to reduce blockchain's storage pressure while accelerating smart contract development and explicate its novelty and feasibility in the Web3 ecosystem. Then, we discuss the future research directions in this field.

TECHNICAL INFRASTRUCTURE OF SCaaS

To implement the SCaaS paradigm, it is necessary to fully utilize two important features of open source smart contracts: reusability and composability.

- *Reusability:* The property that the previously deployed smart contracts can be invoked in new smart contracts to avoid repeated writing.[19]
- *Composability:* The property that deployed smart contracts with different functions can be interconnected through their interface. Developers can construct more complex smart contracts by reusing deployed smart contracts as "Lego bricks."

According to the literature, composability can be divided into three categories as follows:

- Syntactic composability: Syntactic composability denotes the smart contract and

TABLE 1. Comparison of IaaS, BaaS, PaaS, SaaS, and SCaaS.

	IaaS	BaaS	PaaS	SaaS	SCaaS
Primary Function	Computing resources provision	Blockchain infrastructure construction	Development platform construction	Software delivery	Smart contract development
Target Audience	System administrator	Blockchain developer	General developer	End-user	Smart contract developer
Service Method	Infrastructure management	Infrastructure management	Infrastructure management	Service provision	Service provision
Use Case	OpenStack	AWB	OpenShift	iCloud	-

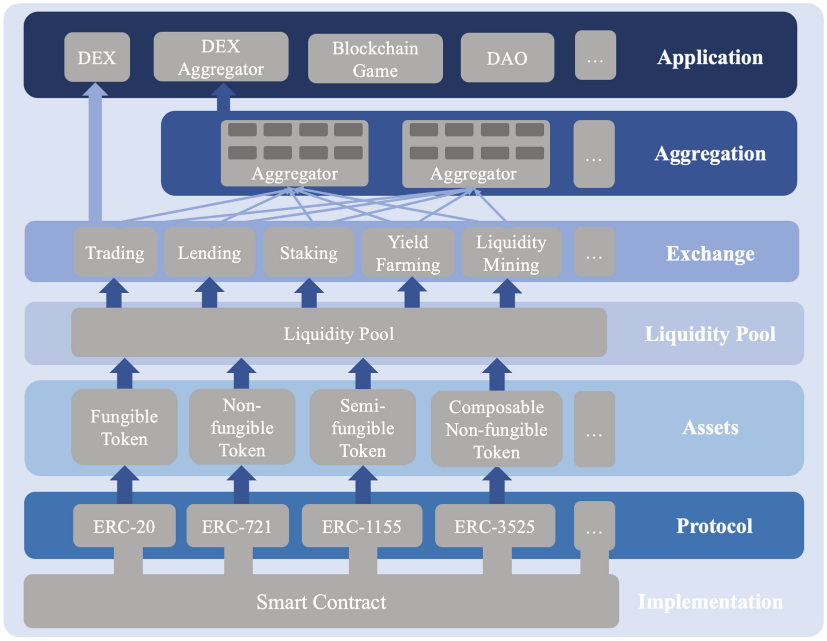

FIGURE 2. Reusability and composability of smart contract in Web3 framework.

protocol are compatible and interoperable, which standardize smart contracts' communication methods and interfaces, thereby facilitating the reusability of the smart contract or protocol.[20,21] It means it will be easier for developers to develop smart contracts and accelerate the development process by reusing the existing smart contracts.

- Atomic composability: Atomic composability denotes the capacity that various reusable smart contracts can be bundled to construct a novel smart contract that satisfies "multiple transactions in one operation."[22] The representative application is "flash loans," which bundles multiple decentralized exchanges (DEXs) and decentralized applications (Dapps) to help users swap, borrow, and transfer tokens in a singular transaction.[23]
- Morphological composability: Morphological composability refers to the characteristic of utilizing reusable token protocols to facilitate token circulation among different smart contracts.[24,25] For example, the non-fungible tokens (NFTs), standardized by the ERC-721 protocol, can be applied in different blockchain games and used in multiple decentralized autonomous organizations (DAOs) for identity verification.

Composability and reusability often intersect in the realm of open source smart contracts, enabling them to be used as Lego bricks, thereby facilitating the creation of an assortment of innovative smart contracts. However, it is crucial to underscore that the reusability of smart contracts does not inherently accompany composability. Particularly in token issuance, developers may exclusively apply the reusability of the smart contract protocol to compile tokens.

THE FEASIBILITY OF SCaaS IN WEB3 ECOSYSTEM

Compared to the centralized programs in Web2, the open source smart contracts in Web3 make it easier to implement the SCaaS paradigm. We summarize the current Web3 ecosystem as the framework with multilayer illustrated in Figure 2, where each layer provides essential support to the subsequent layers above it. The Web3 ecosystem takes smart contracts on the blockchain as the principal implementation layer. At the pinnacle of this framework, all layers ultimately converge into decentralized applications on the top. The composability and reusability of smart contracts are either manifest or have potential applicability throughout the framework. Such attributes substantiate the feasibility of implementing SCaaS within the expansive Web3 ecosystem. More details are shown as follows.

Protocol: The Ethereum Request for Comment (ERC) standard refers to standardized smart contracts for token development on Ethereum. It provides guidelines for token issuance, streamlines token development, and ensures smart contract reusability and composability, which in turn enhances token liquidity and interoperability across diverse decentralized platforms.

Assets: ERC token is classified by the corresponding ERC protocol. For example, tokens developed according to the ERC-20, ERC-721, ERC-1155, and ERC-998 standards are known as fungible tokens, non-fungible tokens, semi-fungible tokens, and composable tokens, respectively. Standardized token protocols make it possible for the interoperability among different types of tokens.

Liquidity Pool: Liquidity pool is a fundamental and integral component of Decentralized Exchange (DEX), which is typically a fund pool locked in a smart contract to provide liquidity for DEX and facilitate decentralized transactions such as trading, lending, staking, and liquidity mining.

Exchange: Decentralized Exchanges (DEX) enable asset trading by reusing different types of token smart contracts (usually ERC-20), catering to user needs for various decentralized financial activity introduced in the last *"Liquidity Pool"* section.

Aggregation: Aggregation in the Web3 ecosystem typically refers to the aggregator that combines multiple DEXs by leveraging smart contract's reusability, composability, and interoperability. It breaks down barriers between different DEXs. Therefore, aggregators can achieve all the functions of a DEX and provide services enabled by "atomic composability" that allow for transactions across different DEXs within the same platform and one single operation.

Application: Decentralized applications (Dapps) are located at the top of the framework, satisfying users' trading needs in DeFi through interactions with protocols, assets, and DEX, thereby bolstering the development of the DeFi stack. The typical types of Dapps are summarized as follows.

- *Decentralized Exchange (DEX):* Beside the previous introduction, DEX does not rely on centralized institutions for asset trading and storage. Instead, it offers users the on-chain wallet that ensures users have complete control over their assets. Each transaction within DEX can be regarded as a reuse or invocation of the token's corresponding contract.
- *DEX Aggregator:* As a typical aggregator, Furucombo[g] allows users to choose DeFi protocols and transaction types to optimize their benefits without any restriction. It intuitively demonstrates the interoperability of DEX, such as Uniswap,[h] Aave,[i] and Maker[j] in Ethereum and Polygon.

  In Furucombo's "creation mode," users can choose the token type to deposit and token type to receive, respectively, then choose from various DeFi protocols and execute their strategy across multiple DeFi contracts in a single transaction to acquire their desired token. A standout advantage is the support for "flash loans," where users use the reusability and composability of smart contracts to complete borrowing and repayment in one single transaction and maximize their profit.
- *Fully On-chain Game (FOG):* As a game and token ecosystem collective that fully runs on the blockchain network, Full On-Chain Games (FOG) has gained more recognition from the native blockchain community since 2023. Its compilable game mode, together with the smart contract's permissionless reusability, has boosted the diversity of gameplay and significantly reduced the marginal cost of development. All the state variables, computational storage, and code interfaces of FOG are implemented through smart contracts, which can provide exponential growth of gameplay and improve the gaming experience.

  Taking the "Loot,[k]" a popular NFT Fully On-chain Game, as an example, its

[g][Online]. Available: https://furucombo.app/
[h][Online]. Available: https://uniswap.org/
[i][Online]. Available: https://aave.com/
[j][Online]. Available: https://makerdao.com/en/
[k][Online]. Available: https://www.lootproject.com/

composability and interoperability have attracted many developers and gamers to build various derivative projects. “Realms: ETERNUM game,[l]” as the most representative land derivative game, inherits the composability of non-fungible tokens in “Loot,” enabling users to combine different non-fungible tokens to enrich the game’s playability. Compared with the land game of “Realms: ETERNUM game,” “Crypts and Caverns[m]” also uses non-fungible tokens as props provided by “Loot,” but changes the game assets from cities to smaller maps named “Crypts and Caverns.” The game is an infinite game, and as long as Ethereum keeps running, it will continue to affect changes in the game pattern. Recently, the developing teams of “Realms: ETERNUM game” and “Crypts and Caverns” are also exploring the interoperability between these two games to stimulate more diverse FOG modes.

– *Decentralized Autonomous Organization (DAO):* Managing a DAO encompasses multifaceted requirements. It should satisfy diverse community needs, including facilitating communication and collaboration among DAO members, overseeing the voting and decision-making process, and distributing finance and contribution rewards. As one of the most widely used DAO management tools, the Dapp Aragon[n] has supported over 5000 DAOs and provides a series of Dapps and smart contracts to satisfy the abovementioned needs.

  During the voting and decision-making stage, data transparency and tamper resistance of blockchain bolster the reliability and trustworthiness of voting results. Therefore, Aragon introduces on-chain voting smart contracts for DAO members. In addition, Aragon implements the on-chain POAPs protocol for member identity verification, allowing users to mint NFTs that represent membership. Aragon makes various Dapps and smart contracts into “programmable Legos” through the reusability and composability of smart contracts and makes it possible for DAO organizers to invoke smart contracts with almost zero coding through a user-friendly interface.

[l][Online]. Available: https://www.realmseternum.com/
[m][Online]. Available: https://threepwave.com/cryptsandcaverns
[n][Online]. Available: https://aragon.org/

THE MARKET FUNDAMENTALS OF SCaaS

To attract developers to participate in the Web3 ecosystem construction, various smart contract development frameworks have been established for Web3 projects. These frameworks primarily aid developers by simplifying the development process, enhancing project security, and ensuring cost-effectiveness. Here are some widely used frameworks in this domain.

Hardhat Framework: In order to reduce the cost of smart contract deployment, the Hardhat framework builds a local Ethereum network, which provides developers with the environment for test running at a lower cost. It accelerates the Solidity debugging process with features such as Solidity stack traces, console.log, and explicit error messages.

Truffle Framework: As one of the most popular development frameworks for Ethereum, Truffle transforms raw requests to the Ethereum Network into more manageable contract abstractions, thereby streamlining the process of smart contract development. It also provides a personal blockchain called Ganache for rapid scenario testing.

Embark Framework: The Embark framework can reduce the project maintenance time by tracking the smart contracts’ state automatically and redeploying the contract if there are any changes. The framework intergrades with IPFS, a decentralized storage network, to storage files across multiple nodes so as ensure the security of the files.

Brownie Framework: The most significant character of Brownie Framework is its Python-based development and test framework targeting the smart contracts on the EVM. The integration with Python allows developers who are already familiar with this programming language to more easily engage in the construction of the Web3 ecosystem.

In essence, these development frameworks primarily assist developers in three ways: reducing the test execution time, simplifying the development language, and supplying repositories of smart contract templates. However, they overlook the advantages of reusing existing on-chain smart contracts. In contrast, the SCaaS paradigm we propose not only bolsters the development efficiency of Web3 projects but also enables developers to inherit the financial characteristics of the reused projects. This approach spares developers from establishing the user base from scratch. Furthermore, ensuring the security of a project does not necessitate an additional repository of smart contract templates. Instead, it only needs to ensure that the reused smart contracts have a substantial number of interactions and are reliable to users in the Web3 market.

APPLICATION OF SCaaS IN WEB3

Some of the most popular public blockchains, such as Ethereum, Binance Smart Chain, and NEAR, have highlighted the significance of reusing smart contracts. Their white papers advocate for developers to reuse deployed smart contracts when they contribute to the Web3 ecosystem. Such advocacy establishes a robust base for the application of SCaaS in Web3. Given the current methods for encouraging the reuse of smart contracts, SCaaS has the following potential applications to optimize these practices.

Standardized Module and Automated Execution

Projects like OpenZeppelin[o] and Gnosis V2[p] offer open source smart contract repositories on GitHub to provide developers with standardized and reusable smart contracts. This approach empowers developers to develop new smart contracts by utilizing the pre-existing smart contracts. However, screening suitable smart contracts within the repositories still takes a lot of time. To address this issue, the SCaaS builds patterns based on its repositories where smart contracts can be automatically arranged and executed. Specifically, developers propose project requirements, and the algorithm provider automatically provides a series of standardized smart contracts that can satisfy the requirements and have a standardized interface from the repository. After that, these provided smart contracts are combined to build a new smart contract and run the corresponding results automatically. The SCaaS has the potential to make the development of decentralized projects easier and more efficient.

[o][Online]. Available: https://github.com/OpenZeppelin/openzeppelin-contracts
[p][Online]. Available: https://github.com/gnosis/gp-v2-contracts

Enhanced Transparency of Smart Contract

Decentralized projects have attracted numerous users and developers because of the data transparency, and many public blockchains encourage developers to reveal their codes. However, there are still many decentralized projects that refuse to disclose their code due to copyright breaches and malicious attack concerns.[26] This situation not only diminishes users' trust in the project but also increases developers' difficulty in reusing smart contracts. Fortunately, since new projects developed through SCaaS are composed of standardized open source smart contracts, this enhances the transparency of new projects and avoids issues caused by the undisclosed smart contracts.

Community-Driven Feedback System

Platforms like the Ethereum Stack Exchange and Solidity Forum allow developers to discuss and seek assistance on challenges in developing smart contracts. While these platforms have garnered some engagement, issues still exist, including limited participant numbers, delayed responses, and insufficient feedback, leading to decreased user activity. Leveraging the repository characteristic of SCaaS, there exists potential to build a SCaaS-centric, community-driven platform that encourages participants to review, vote on, and share advice on smart contracts within the SCaaS and solve problems in the process of using SCaaS. By focusing on participant feedback, it becomes possible to constantly refine the smart contracts. Such an approach not only amplifies the participants' engagement but also bolsters trust in the reused smart contracts and community, potentially broadening the scope to encompass a wider range of technical discussions.

In conclusion, SCaaS has the potential to improve and expand Web3 from multiple dimensions, such as elevating project development efficiency, bolstering project credibility, fostering community participation, and further attracting more developers and users to make contributions to Web3.

CHALLENGE AND LIMITATION OF SCaaS

While the SCaaS offers numerous advantages, as previously mentioned, certain challenges and limitations have remained to overcome. In this section, we list some common concerns and propose a primary solution from the perspective of blockchain foundation and Web3 project operators.

Project Risk: If a reused contract has a vulnerability, every project that reuses this contract will inherit the vulnerability. Besides, projects incorporating numerous smart contracts can complicate the execution process, elevating the risk of attacks. For instance, users in Furucombo suffered a loss of over 15 million dollars because of the attack.

Accessibility: The costs of deploying and executing smart contracts on blockchains with higher gas fees can be fluctuant and expensive,[27] especially interacting with the new smart contracts reusing DEXs with liquidity pools. Such unpredictable expenses could potentially restrict the availability of Smart Contract as a Service (SCaaS) to both developers and end-users.

Developer Incentive: Although SCaaS can make the development of smart contracts easier and more efficient, developers can still write smart contracts in other ways. Therefore, direct incentives such as distributing grants to developers will promote the acceptance of the SCaaS paradigm.

Considering the challenges mentioned above, blockchain should enhance the audit of smart contracts, optimize gas fees, and design incentive mechanisms for developers accordingly.

CONCLUSION

In light of an increasing number of redundant smart contracts on the blockchain, we propose the Smart Contract as a Service (SCaaS) paradigm for the Web3 ecosystem. By fully utilizing the reusability and composability of the smart contract, SCaaS holds transformative potential for building a more streamlined, efficient, and secure blockchain landscape, ushering in a new era of Web3. In the scenario based on the SCaaS, how to incentivize developers to use SCaaS through economic motivation will be a valuable topic to be further studied. This will involve crafting optimal strategies for both developers and the foundation within the incentive mechanism. Additionally, the analysis of participant behavior will provide the service provider with insights for broader promotion of the SCaaS.

ACKNOWLEDGMENTS

This work was supported in part by the Shenzhen Science and Technology Program under Grant JCYJ20210324124205016, in part by the Shenzhen Key Lab of Crowd Intelligence Empowered Low-Carbon Energy Network under Grant ZDSYS20220606100601002, and in part by the CUHK(SZ)-White Matrix Joint Metaverse Laboratory.

REFERENCES

1. M. Goint, C. Bertelle, and C. Duvallet, “Secure access control to data in off-chain storage in blockchain-based consent systems,” *Math.*, vol. 11, no. 7, 2023, Art. no. 1592.
2. I. Yaqoob, K. Salah, M. Uddin, R. Jayaraman, M. Omar, and M. Imran, “Blockchain for digital twins: Recent advances and future research challenges,” *IEEE Netw.*, vol. 34, no. 5, pp. 290–298, Sep./Oct. 2020.
3. D. W. Allen, C. Berg, and A. M. Lane, “Why airdrop cryptocurrency tokens?,” *J. Bus. Res.*, vol. 163, 2023, Art. no. 113945.
4. P. P. Momtaz, K. Rennertseder, and H. Schröder, “Token offerings: A revolution in corporate finance?,” 2019.
5. J. Wang, W. Ou, W. Wang, R. S. Sherratt, Y. Ren, and X. Yu, “Data security storage mechanism based on blockchain network,” *Comput., Mater. Continua*, vol. 74, no. 3, pp. 4933–4950, 2023.
6. A. E. Gencer, R. van Renesse, and E. G. Sirer, “Short paper: Service-oriented sharding for blockchains,” in *Proc. 21st Int. Conf. Financial Cryptogr. Data Secur.*, 2017, pp. 393–401.
7. S. Haefliger, G. Von Krogh, and S. Spaeth, “Code reuse in open source software,” *Manage. Sci.*, vol. 54, no. 1, pp. 180–193, 2008.
8. H. Haaskjold, B. Andersen, O. Ldre, and W. Aarseth, “Factors affecting transaction costs and collaboration in projects,” *Int. J. Manag. Projects Bus.*, vol. 13, no. 1, pp. 197–230, 2020.

9. F. Zantalis, G. Koulouras, and S. Karabetsos, “Blockchain technology: A framework for endless applications,” *IEEE Consum. Electron. Mag.*, early access, 2023, doi: 10.1109/MCE.2023.3248872.
10. X. Chen, P. Liao, Y. Zhang, Y. Huang, and Z. Zheng, “Understanding code reuse in smart contracts,” in *Proc. IEEE Int. Conf. Softw. Anal., Evol. Reeng.*, 2021, pp. 470–479.
11. L. Guida and F. Daniel, “Supporting reuse of smart contracts through service orientation and assisted development,” in *Proc. IEEE Int. Conf. Decentralized Appl. Infrastructures*, 2019, pp. 59–68.
12. S. Gec, V. Stankovski, D. Lavbič, and P. Kochovski, “A recommender system for robust smart contract template classification,” *Sensors*, vol. 23, no. 2, p. 639, 2023.
13. A. E. Youssef, “Exploring cloud computing services and applications,” *J. Emerg. Trends Comput. Inf. Sci.*, vol. 3, no. 6, pp. 838–847, 2012.
14. J. Gibson, R. Rondeau, D. Eveleigh, and Q. Tan, “Benefits and challenges of three cloud computing service models,” in *Proc. 4th Int. Conf. Comput. Aspects Social Netw.*, 2012, pp. 198–205.
15. S. K. Garg, S. Versteeg, and R. Buyya, “A framework for ranking of cloud computing services,” *Future Gener. Comput. Syst.*, vol. 29, no. 4, pp. 1012–1023, 2013.
16. M. M. H. Onik and M. H. Miraz, “Performance analytical comparison of blockchain-as-a-service (BAAS) platforms,” in *Proc. 2nd Int. Conf. Emerg. Technol. Comput.*, 2019, pp. 3–18.
17. W. Zheng, Z. Zheng, X. Chen, K. Dai, P. Li, and R. Chen, “Nutbaas: A blockchain-as-a-service platform,” *IEEE Access*, vol. 7, pp. 134422–134433, 2019.
18. J. Singh and J. D. Michels, “Blockchain as a service (baas): Providers and trust,” in *Proc. IEEE Eur. Symp. Secur. Privacy Workshops*, 2018, pp. 67–74.
19. F. Daniel and L. Guida, “A service-oriented perspective on blockchain smart contracts,” *IEEE Internet Comput.*, vol. 23, no. 1, pp. 46–53, Jan./Feb. 2019.
20. A. Tolk, S. Y. Diallo, and C. D. Turnitsa, “Applying the levels of conceptual interoperability model in support of integratability, interoperability, and composability for system-of-systems engineering,” *J. Syst., Cybern., Inform.*, vol. 5, no. 5, pp. 64–74, 2007.
21. K. Babel, P. Daian, M. Kelkar, and A. Juels, “Clockwork finance: Automated analysis of economic security in smart contracts,” in *Proc. IEEE Symp. Secur. Privacy*, 2023, pp. 2499–2516.
22. P. Tolmach, Y. Li, S.-W. Lin, and Y. Liu, “Formal analysis of composable defi protocols,” in *Proc. Financial Cryptography Data Secur. Int. Workshops: CoDecFin, DeFi, VOTING, WTSC, Virtual Event*, 2021, pp. 149–161.
23. R. M. d. Carvalho, H. Inácio, and R. P. Marques, “Money lego auditing: The composability assertion of flash loan transactions,” Available at SSRN 4417760.
24. P. Freni, E. Ferro, and R. Moncada, “Tokenomics and blockchain tokens: A design-oriented morphological framework,” *Blockchain: Res. Appl.*, vol. 3, no. 1, 2022, Art. no. 100069.
25. C. Jacobi, M. Meier, L. Herborn, and K. Furmans, “Maturity model for applying process mining in supply chains: Literature overview and practical implications,” *Logist. J., Proc.*, vol. 2020, no. 12, pp. 1–16, 2020.
26. R. Gupta, S. Tanwar, F. Al-Turjman, P. Italiya, A. Nauman, and S. W. Kim, “Smart contract privacy protection using AI in cyber-physical systems: Tools, techniques and challenges,” *IEEE Access*, vol. 8, pp. 24746–24772, 2020.
27. X. Su, Y. Liu, and C. Choi, “A blockchain-based p2p transaction method and sensitive data encoding for e-commerce transactions,” *IEEE Consum. Electron. Mag.*, vol. 9, no. 4, pp. 56–66, Jul. 2020.

Jinghan Sun is currently working toward the Ph.D. degree with The Chinese University of Hong Kong, Shenzhen, China, and currently works as a research assistant with MBZUAI. Contact her at Jinghansun@link.cuhk.edu.cn.

Abdulmotaleb El Saddik is a distinguished university professor with the University of Ottawa and MBZUAI. His research interests include digital twins for citizen well-being using AI, IoT, AR/VR, and 5G in the Metaverse. He is a IEEE Fellow. Contact him at elsaddik@uottawa.ca.

Wei Cai is an assistant professor with The Chinese University of Hong Kong, Shenzhen, China. His research interests include decentralized systems and interactive multimedia. Cai received the Ph.D. degree in electrical and computer engineering from The University of British Columbia, Vancouver, Canada. He is a senior member of IEEE. He is the corresponding author of this article. Contact him at caiwei@cuhk.edu.cn.