DAOSYS: An Autonomous Service Engine for Decentralized Finance

CYOTEE DOGE[1], IAN C. MOORE, PHD[2], RYLAND ARBOUR[3], and JAGDEEP SIDHU, MSC[4]

[1]DAO Advisor, Syscoin Platform (e-mail: cyotee@syscoin.org)
[2]Syscoin Researcher, Syscoin Platform (e-mail: imoore@syscoin.org)
[3]L2 Advisor, Syscoin Platform (e-mail: rylandarbour@syscoin.com)
[4]Syscoin Lead Developer, (e-mail: sidhujag@syscoin.org)

ABSTRACT Despite popular perception, treasuries of Dentralized Autonomous Organizations (DAOs) tend to be centrally controlled and do not reflect the true ethos of cryptocurrency (i.e., *not your keys, not your coins*). DAOSYS solves this problem with its new Autonomous Service Engine (ASE) technology by deploying a reference platform for self-sovereign capital coordination. This is made possible by innovating on the multi-faceted proxy standard defined in EIP-2535. The ASE will serve as the cornerstone of all SYS Labs decentralized finance (DeFi) products, as supported by the Syscoin Platform.

INDEX TERMS DAOSYS, Decentralized Autonomous Organization, Decentralized Finance, SYS Labs, EIP-2535

I. INTRODUCTION

The objective of a decentralized autonomous organization (DAO) is to solve the principal-agent dilemma. This dilemma is a result of misaligned incentives where agents acting in a system are incentivized towards their own benefit over the benefit of a principle or other agents acting within the system [1]. Typically, these are found in centralized systems where the central acting authority is the main compromised agent. The DAO solves this by decentralizing the governance process by utilizing smart contracts running on open source blockchains.

The first inception of the DAO concept happened in May 2016 out of the Ethereum community, which was known as Genesis DAO, and was built as a smart contract on the Ethereum blockchain. However, this resulted in the well known DAO Hack which resulted in the draining of $60M USD worth of funds from its treasury [2]. Today, there are many DAOs in operation with Uniswap, Aave and Maker DAO being amongst the most popular. However, these DAOs still fundamentally violate the core value proposition of self sovereignty that crypto currency promises, where DAOs currently take ownership of capital managed in a treasury controlled by a few individuals. This is the problem that DAOSYS intends to address.

DAOSYS vision is to operate like a pure automated market maker (AMM) and be implemented in a manner that does not require external controls. In 2018, Uniswap became the first decentralized platform to successfully utilize an AMM [4]. However, like Uniswap and many AMMs, control of the capital in their respective liquidity pools is still centralized. DAOSYS intends to address this core issue via its new Autonomous Service Engine (ASE) technology [3], hence allowing DAOs to be more autonomous and fully decentralized. One of the interesting by-products of this technology will allow users to test, implement and realize countless decentralized finance (DeFi) usecase designs through the ASE without having to continually redeploy new builds.

II. GOVERNANCE

DOAs are governance structures for groups of people to come together and make decisions, which is one of its main distinguishing features. Unlike traditional organizations (e.g., private companies, non-governmental organizations, charities, etc.) which have a centralized structure, these decisions are coordinated and enforced on a blockchain in a decentralized manner. Since DAOSYS proposes to be autonomous and fully decentralized, it has no top level governance.

AMMs are an essential part of the DeFi ecosystem. They allow digital assets to be traded in a permissionless and automatic way by using liquidity pools rather than a traditional market of buyers and sellers. Applying the AMM model to DAOs means that users create their own treasuries for specific ventures. These treasuries may apply a variety of governance solutions along with their treasuries. This allows for a compartmentalization of the politics that arise with any governance solution from the actual treasury management.

Under this model, the Syscoin Foundation behaves more like the software vendor. The factory makes open-source reference implementations of DeFi components available to compose into treasuries. Updates to these smart-contracts are available for deploying new pools that may be added to a DAO. Hence, this removes the need for top-level governance

solutions that decides whether to include an update because users are free to create new pools.

A user creates a DAO by selecting which vaults and bond markets they would like to include. These vaults may come from one of four sources.

A. REUSE AN EXISTING VAULT

This works best for when users wish to maintain their position in one DAO, but want to add more pools to form a new DAO.

B. RECREATE AN EXISTING VAULT

As the adage goes,*if it's not broke, don't fix it*. The investment strategy implemented in a vault can be used across several instances of vault pools. This works well for new DAOs that wish to replicate the financial strategies of an existing DAO. Also, for when a new DAO would like to invest in other DAOs using the same strategy.

C. NEW POOL WITH NEW INVESTMENT STRATEGY

A user may wish to create a new DAO reusing functionality available from the factory, but configured in a novel manner. The flexibility available in the ASE means that even a simple strategy has several configuration options. This is useful for when a DAO wishes to adopt a novel investment strategy that might not have been previously viable.

D. NEW POOL WITH CUSTOM CODE

The Syscoin Foundation makes internal decisions regarding what smart-contracts are available through the factory similar to open source software development. Because this only concerns the software available from the foundation, this does not need to be open to public governance. When the community at large wishes to release custom code outside the foundation, a user may use the factory to deploy their own factory offering their custom code. This new factory inherits the offerings of the parent factory and may add their own modules.

These pools form the foundation of the DAO. Autonomous and permissionless liquidity pools that act as the agreed upon foundation for DAO treasury management. From there users may launch further liquidity pools that may accept the DAOs Treasury Token for deposit. These form the Roundtables for managing ventures within the DAO. The Roundtables compartmentalize management teams, Councilors, of the various ventures being executed under a DAO's mission statement. A Roundtable typically does not have it's own governance token, instead using a Council Token used to resolve disputes by executing buyout options.

From the Roundtables, any Councilor may use their contribution to the Roundtable to launch a bond offering for a Quest. Quests define the bounty award and terms for completing a task. The Councilor that issues the quest puts their share of the treasury in escrow to fund the Quest. The interest being earned from that underlying position is then split to fund the bounty, compound into that position, and to sell on the bond market. This ensures that Questors know the payment for work they deliver is secured. And protects the Councilor from failure to deliver; see Fig. 1 for outline.

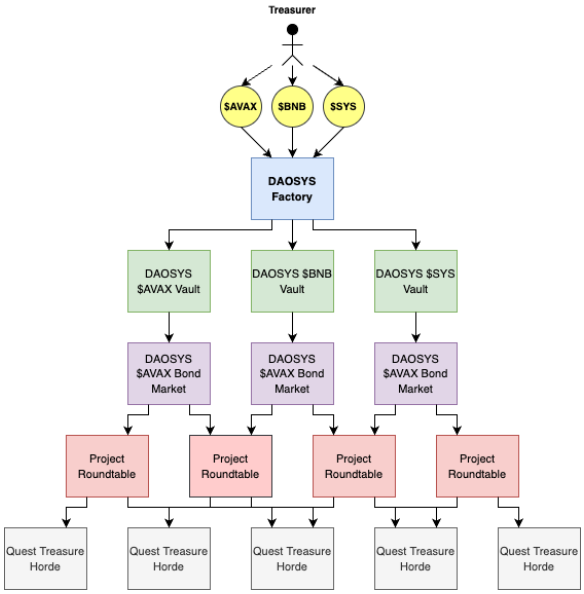

FIGURE 1: DAOSYS Governance Structure: new pool with custom code (ie, Quests)

III. ARCHITECTURE

The innovative software architecture that allows DAOSYS to pioneer a revolution in DAOs is the ASE. This introduces a revolution allowing DAOs to become more autonomous and fully decentralized. The ASE iterates on the Diamond proxy standard from EIP-2535 (see Appendix A) to apply AMM functionality for more flexible capabilities.

A. ICREATE2METADATA

The `CREATE2` opcode is a low level human readable instruction used within the stack-based architecture of the Ethereum Virtual Machine (EVM) that was introduced in 2019 with EIP-1014 [6]. This is the technology that enables AMMs in EVM implementations which allows a smart-contract to deploy another smart contract, and is typically called the Factory design pattern. Protocols like Balancer and Uniswap provide the ability to create permissionless liquidity pools. However, the limit of these implementations is the immutable nature of smart-contract bytecode as they can only deploy a single type of contract.

All contracts deployed through the ASE, new service proxies and delegate service, store the metadata for their deployment. This includes the factory address they were deployed from, and the value used to salt their deployment. Combined with the contract's `codehash` this can be used to recalculate the contract's address from the assertion of it's `ICreate2Metadata` interface hook.

B. IDELEGATESERVICE

Protocols like Aavegotchi have pioneered smart-contract architecture by iterating on the proxy capabilities of the EVM to advance the Factory design pattern. Smart-contract proxies take advantage of the `DELEGATECALL` opcode to allow a smart-contract to reuse the logic implemented in other smart-contracts. The Autonomous Service Engine advances this innovation with an infinitely flexible Diamond Factory design. The Diamond Factory design factory combines the Factory and Diamond design patterns to deploy configurable proxies. This allows for infinitely composable proxies.

Delegate Services replace the Facets defined in ERC-2535. Delegate Services define a strict storage allocation and access standard beyond the theory presented in ERC-2535. A Service is a smart-contract or library implemented following the Deterministically Dynamic Storage Allocation standard. A Service also reports the factory that deployed that contract and the salt used during deployment. This way the recalculation of the address from the factory init code hash and salt can be used to verify new Services as an implicit ACL.

This means that the deployment process for new Services deviates from industry standard. New Services are deployed as compiled bytecode passed to the Service Proxy Factory as the argument for the deployment function. The Service Proxy Factory then instantiates that bytecode as a new contract. ASE compliant Services must include the ASEServiceBootstrapper library to retrieve the address salt to initialize the Service. This should be done by delegating to the canonical external library deployment. ASE compliant external libraries may precalculate their address salt and store it as a constant. The standard Service initialization functions must still be implemented, but may hard code the values and return values since they can not store state.

All Services are required to implement ERC-165 including the Service extension that enumerates the functions. The Service extension to ERC-165 includes a per interface enumeration of the function selectors that define the interface ID. Additionally, there is an enumeration of all the function selectors across all interface IDs, and a ServiceDef struct that includes the information for initializing a Service Proxy to consume the Service.

IV. TOKENOMICS

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

A. SIMULATOR

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

V. ROADMAP

A. SYSLABS

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

B. L2: NEVM

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

C. FIRST USE CASE: MASTERNODE YIELD FARMING

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

VI. SUMMARY

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

.

APPENDIX A EIP-2535 (DIAMOND FACETS)

EIP-2535 was an Ethereum proposal launched in 2020 to allow for the creation of modular contract systems that can be extended after deployment [5]. To solve this problem, this new proposal introduced a concept known as Diamonds. A Diamond refers to a smart contract system where functionality and storage is split up into separate contracts, and is an extension of a proxy contract.

A proxy contract is the the immutable part a user interacts with which holds data. It contains a fallback function which will catch any function call and use `delegatecall` to forward it to a second logic contract. As a reminder, `delegatecall` will execute a function defined in the called contract (logic), but within the context of the calling contract (proxy). Thus, we have our logic defined separately from our data. This allows users to change the logic contract without changing the immutable underlying data.

There are several limitations to proxy contracts which include: (a) implementing minor upgrades; (b) 24kb max contract size limit; (c) can only create identical proxy instances using one logic contract; and (d) cannot have a modular permission system. Diamonds solve these issues by allowing for multiple logic contracts which talks to the original proxy contract via the `delegatecall`; for comparison between these two systems see Fig. 2 .

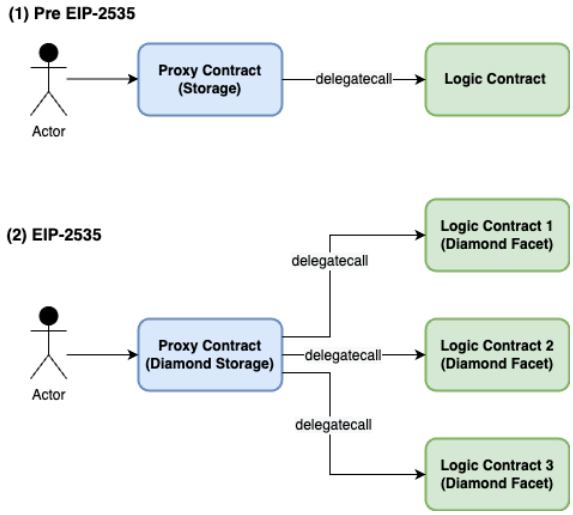

FIGURE 2: Multi-facet proxy; (TOP) prior to EIP-2535, proxy contracts could only make delegates calls to another contract only; (BOTTOM) EIP-2535 introduced the capability of performing delegates calls to multiple contracts

REFERENCES

[1] S.J. Grossman, O.D. Hart, *An Analysis of the Principal-Agent Problem*, Econometrica, Volume 51, Issue 1 (Jan., 1983), 7-46.
[2] D. Siegel, *Understanding The DAO Attack*, Coindesk, Aug. 2022. Accessed on: Sept 2022. [Online]. Available: https://www.coindesk.com/learn/2016/06/25/understanding-the-dao-attack/
[3] Syscoin Dev Team, *DAOSYS Lite Paper*, Accessed on: Sept 2022. [Online]. Available: https://github.com/syscoin/daosys
[4] Uniswap Blog, *A short history of Uniswap*, Sept. 2019. Accessed on: Sept 2022. [Online]. Available: https://uniswap.org/blog/uniswap-history
[5] N. Mudge, *EIP-2535: Diamonds, Multi-Facet Proxy*, Aug. 2020. Accessed on: Sept 2022. [Online]. Available: https://eips.ethereum.org/EIPS/eip-2535

[6] V. Buterin, *EIP-1014: Skinny CREATE2*, Apr. 2018. Accessed on: Sept 2022. [Online]. Available: https://eips.ethereum.org/EIPS/eip-1014

[7] J. Sidhu and I.C. Moore, *Syscoin 4.0: A Peer-to-Peer Electronic Cash System Built For Business Applications*, Dec 2021, Accessed on: Sep 2022. [Online]. Available: https://syscoin.org/file/syscoin4-whitepaper.pdf

• • •