Governance Transformation: Establishing Scalable and Adaptable Decentralized Network on EVM-Compatible Blockchain

Larysa Katerynych[1], Maksym Veres[1], Kyrylo Riabov[1] and Kostiantyn Zhereb[1]

[1] *Taras Shevchenko National University of Kyiv, Academician Glushkov Avenue 4d, Kyiv, 03680, Ukraine*

Abstract

This article explores scalable and adaptable governance through decentralized networks, enabling collective decision-making and evolution without necessitating a complete system overhaul when original functionalities become obsolete. It delves into the utilization of interconnected Smart Contracts on the EVM-based blockchain to resolve foundational governance issues. However, the principal advantages are compromised if the system requires continual redeployment to adapt to environmental changes.

We introduce a solution that combines a role-based access system and a modular system contracts architecture to enhance the system's scalability and adaptability. This approach allows for modifications and scaling at any time through community proposals and voting, catering to the specific needs of its members and eliminating the need for manual configuration by a centralized entity for new modules or functionalities. Members can propose configuration parameters for a new module, and with community majority approval, the system can adapt and scale, safeguarding the interests of its members, provided there is cooperation within the majority of the community.

Keywords

Decentralized Autonomous Organization, Meta Governance, WEB3, Role-Based Access System, Ethereum, Blockchain, Solidity

1. Introduction

Navigating decision-making on the Internet poses a significant challenge in establishing a robust system where members cannot manipulate outcomes and only eligible parties can participate. On another side, it was important to prevent 'double-spending' problem, so users cannot reply the same action again and again in order to get a benefit. The solution to the problem was proposed by the adoption of the Bitcoin [1]. However, it did not allow creating a complex governance structures with custom rules that cannot be omitted.

The adaptation of Smart Contracts [2, 3] on the Ethereum network has addressed this issue, operating under the principle of 'code is law' [4]. While this approach serves smaller communities and projects focusing on straightforward financial management effectively, it struggles with continuously evolving systems that need to adapt to real-world trends and changes.

Ethereum, inherently a perpetually evolving protocol, enables the development of a diverse ecosystem of commercial products, allowing community interaction and participation in decision-making processes. A prevalent instance is the creation of platforms, known as Decentralized Autonomous Organizations (DAOs), where community members or investors can influence the project's trajectory in the WEB3 sphere. Given the irreversible nature of user actions and Smart Contracts in Ethereum, significant modifications to the contract logic are challenging, and manual

14th International Scientific and Practical Conference from Programming UkrPROGP'2024, May 14-15, 2024, Kyiv, Ukraine
* Corresponding author.
† These authors contributed equally.

katerynych@gmail.com (L. Katerynych); veres@ukr.net (M. Veres); kyryl.ryabov@gmail.com (K. Riabov); zhereb@gmail.com (K. Zhereb)
0000-0001-7837-764X (L. Katerynych); 0000-0002-8512-5560 (M. Veres); 0009-0003-4118-8492 (K. Riabov); 0000-0003-0881-2284 (K. Zhereb)

upgrades through a Proxy Upgrade pattern [5] can be risky. Contracts also face a size limitation, hindering scalability when the available space is exhausted. Additionally, the inability to adapt to new products and protocols in the Ethereum ecosystem can render older DAOs obsolete.

To address these limitations, a modular architecture incorporating a role-based access system is essential, allowing seamless integration with new protocols and functionalities without redeploying the entire system. This approach not only eliminates the need for system redeployment and reconfiguration but also empowers the community to modify the system's functionality through voting, ensuring security and sustainability as long as the majority actively cooperate.

2. Role-Based Access System

This article introduces a Role-Based Access System (RBAS) serving as a pivotal connector between system components, establishing a set of rules that delineate access to system resources. These rules are designed to be transparent and comprehensible to community members, offering enhanced flexibility to the system.

Where each community member or the system itself can be considered an Entity that performs specific Actions on resources.

In the prevalent Ethereum ecosystem, most contracts depend on account addresses to determine resource access, a method that restricts the system to specific addresses. This limitation becomes problematic in systems with numerous contracts, confining them to precise implementations that cannot be easily and securely modified.

RBAS, in contrast, abstracts resource access to parties with specific permissions, granted through community voting, simplifying the management of system component relationships. It centralizes all access rules, unlike traditional approaches where access to protected functions requires extensive contract code analysis. Despite the additional gas usage, RBAS compensates by facilitating a more manageable and adaptable system.

By leveraging RBAS, which grants permissions through community voting, management of system components is simplified and adaptability is enhanced. For example, in Corporate Governance [6], implementing RBAS in DAOs illustrates how it can supplement traditional business structures, enabling dynamic, decentralized regulatory solutions and showcasing a compliant approach to managing corporate entities.

The core capability of the RBAS is its ability to assimilate external modules while maintaining tight control over them and ensuring that the integrated components conform to the control protocols established in the system. This integration is key to extending and diversifying the functionality of the system, allowing the inclusion of various components such as DAO bridges, Uniswap, Treasuries and other elements that require the influence of DAO management.

In practical terms, this means that any component, once integrated, automatically adapts to the rules and protocols of system management, inheriting established norms and operational frameworks. This integration is critical for consistency and uniformity across the system, ensuring that all components, regardless of their origin and nature, operate within a common governance structure, reducing the risks associated with inconsistencies and non-compliance.

In essence, integrating external modules using RBAS not only enriches the ecosystem with diverse functionality, but also strengthens the governance structure by ensuring uniformity and compliance across all components. This is an example of a balanced synergy between extension and governance, allowing the system to evolve and adapt while maintaining its underlying principles and integrity.

2.1. Roles, Resources, Entities, Actions, and Permissions

Our proposed solution initiates with the delineation of roles, resources, entities, actions, and permissions within the system, as illustrated in Figure 1.

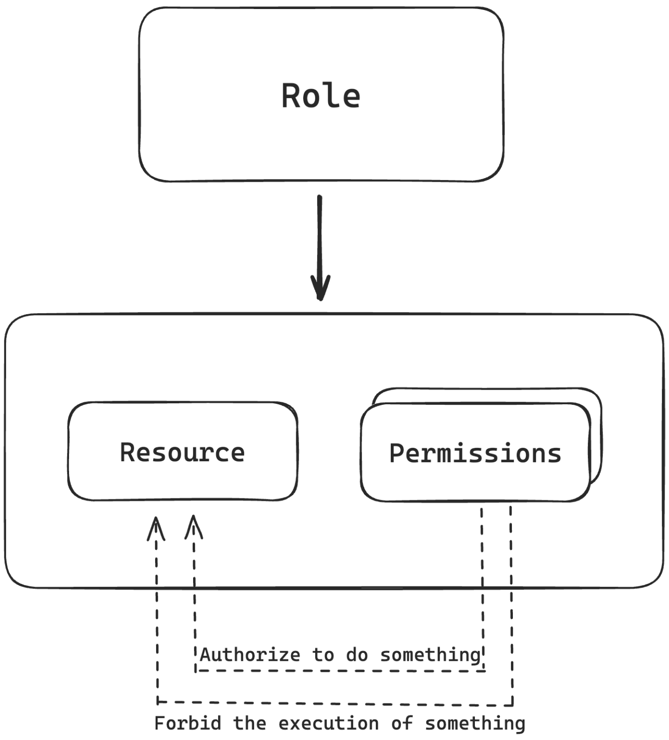

Figure 1: Role, Resource, Permissions Model.

Entity (E): An individual or component that interacts with the system.

$$\mathrm{E} = \{e_1, e_2, \ldots, e_n\}$$

Action (A): An operation performed by an entity on a resource.

$$\mathrm{A} = \{a_1, a_2, \ldots, a_n\}$$

Permission (Perm): A set of rules determining access to the system's resources, either allowing or denying such access.

$$\mathrm{Perm} = \{\mathrm{accept}, \mathrm{reject}\}$$

Resources (Res): A set of functions grouped either as a Smart Contract (SC) or as functions within an SC.

$$\mathrm{Res} = \{\mathcal{SC}_1, \mathcal{SC}_2, \ldots, \mathcal{SC}_n\}$$

Role (Role): An amalgamation of one resource and one or more permissions, structuring relationships within the system in a simple yet effective manner.

$$\mathrm{Role} = \mathrm{Res} \times \mathrm{Perm}$$

To enhance efficiency and optimization, we used the concept of a group abstraction.
Group (Grp): A compilation of members sharing identical roles.

$$\mathrm{Grp} = \{g_1, g_2, \ldots, g_k\}$$

Instead of granting roles directly to members (M), they are role assigned to groups.

$$\forall g \in \text{Grp}, \exists \text{Role}_g \subseteq \text{Role}$$

$$\mathcal{M} \in \text{Grp} \Rightarrow \mathcal{M} \text{ has Role(Grp)}$$

Consequently, a member's inclusion in a group bestows upon them all the permissions allocated to that group. Essentially, a group operates as an array to which a specific role can be granted. Consequently, each member within this array inherits the permissions associated with the group, allowing for a streamlined allocation of roles and permissions, as illustrated in Figure 2.

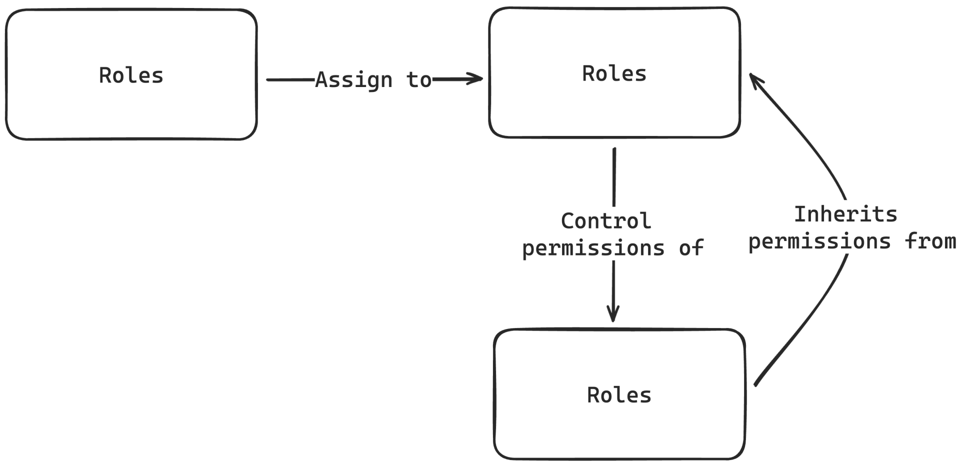

Figure 2: Group-Based Role Allocation.

This structured approach sets the stage for defining the core properties and formal guarantees of the RBAS system.

2.2. Core Properties and Formal Guarantees

The systematization above not only streamlines internal relationships but also ensures a structured approach to resource and permission allocation. This allows us to extract the following properties:

Role Assignment: Each entity e is assigned one or more roles, which collectively determine the actions they are authorized to perform on resources:

$$\forall e \in \text{E}, \exists \text{Role}_e \subseteq \text{Role}$$

Permission Enforcement (Perform): For an entity e to perform an action a on a resource r, the entity must possess a role that includes the corresponding permission:

$$\text{Perform}(e, a, r) \Rightarrow \exists \text{Role}_e \ni (r, a)$$

Access Control Function (Access): An access control function *Access* verifies whether an entity *e* has the necessary permissions to perform an action on a resource *r*:

$$\text{Access}(e, a, r) = \begin{cases} \text{true}, & \text{if } \exists \text{Role}_e \ni (r, a) \\ \text{false}, & \text{otherwise} \end{cases}$$

By combining the core properties and system primitives, we achieve the following formal guarantees:

Correctness and Consistency: The RBAS ensures that only authorized actions are performed, preventing unauthorized access and modifications, and ensuring that role assignments and permissions are uniformly enforced across all actions:

$$\forall(e, a, r) \text{ if } \mathrm{Access}(e, a, r) = \text{true } \textbf{then } \mathrm{Perform}(e, a, r)$$

The RBAS, as formalized above, provides a robust framework for managing access and permissions within the decentralized network. It ensures that every action is authorized, maintaining the integrity and security of the system while allowing for scalability and adaptability. The subsequent sections will elucidate its application within the context of a DAO.

3. Modular Smart Contracts Architecture

To construct a system capable of scaling indefinitely, a modular System Contract architecture is imperative, aligning with the standards proposed by Nick Mudge in ERC-2535 [7]. Given the restrictive 24KB [4] contract size limitation, this architecture organizes a collection of Smart Contracts under the ERC-2535 standard to accommodate extensive systems.

Each system component is a distinct module or contract, enabling the DAO to integrate with a myriad of protocols within the Ethereum ecosystem and incorporate new functionalities with static addresses. This modularity negates the need for users to navigate through multiple System Contracts to locate specific functionalities provided by individual Smart Contracts within the system.

The DAO is essentially defined as:

$$\mathrm{DAO} = \{\mathrm{Storage},\ \{\mathrm{mod}_1, \mathrm{mod}_2, \ldots, \mathrm{mod}_p\}\}$$

where the Storage represents the state of the DAO and modi, is a distinct component or contract within the system.

The adaptability inherent in Modular Smart Contracts Architecture allows DAOs to seamlessly integrate or modify modules, ensuring continuous compliance with evolving disclosure regulations and aligning with the transparency and consumer protection needs of the decentralized finance ecosystem [8].

As visualized in Figure 3, the actual data is housed in the main entity, which, in this instance, is a DAO. This main entity uses specific delegate calls to interact with and utilize the functionalities provided by separate entities, referred to as facets.

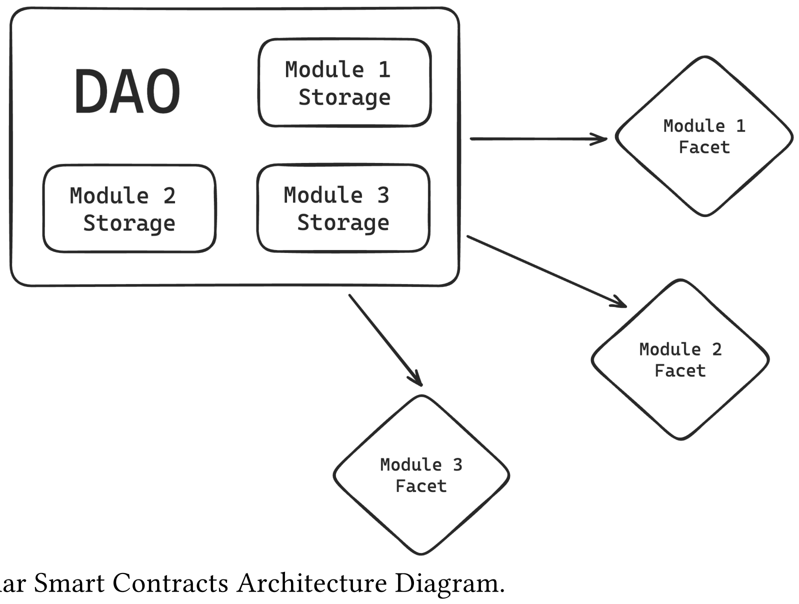

Figure 3: Modular Smart Contracts Architecture Diagram.

4. Governance Structure

Traditional governance modules predominantly rely on the voting power of community members, represented through various means such as the quantity of ERC-20 [9], NFT [10], ERC-1155 [11] tokens or native currency locked within the system. For a proposal to gain acceptance, it must achieve a requisite quorum and a majority of votes.

However, as governance expands with increasing user participation, the likelihood of suboptimal decisions escalates. To mitigate this, we introduce an Expert Group, comprised of members with the authority to veto specific community-selected proposals and to initiate expert-specific proposals, necessitating an in-depth understanding of the system.

Formally, Expert Group (ExpGrp) is a set of entities with special permissions:

$$\text{ExpGrp} = \{e_1, e_2, \ldots, e_k\} \text{ where } e_i \in \text{E}$$

This additional protective layer aims to reduce the risk of system stagnation by ensuring that decisions are meticulously scrutinized and are reflective of informed and expert opinions, thereby enhancing the robustness and reliability of the governance structure.

At every level of the governance structure, from the creation to the execution of proposals, the community diligently works to uphold the security and stability of the system, as illustrated in Figure 4. This visualization depicts the community's commitment to preventing any disruptions and ensuring the continuous, smooth operation of the system.

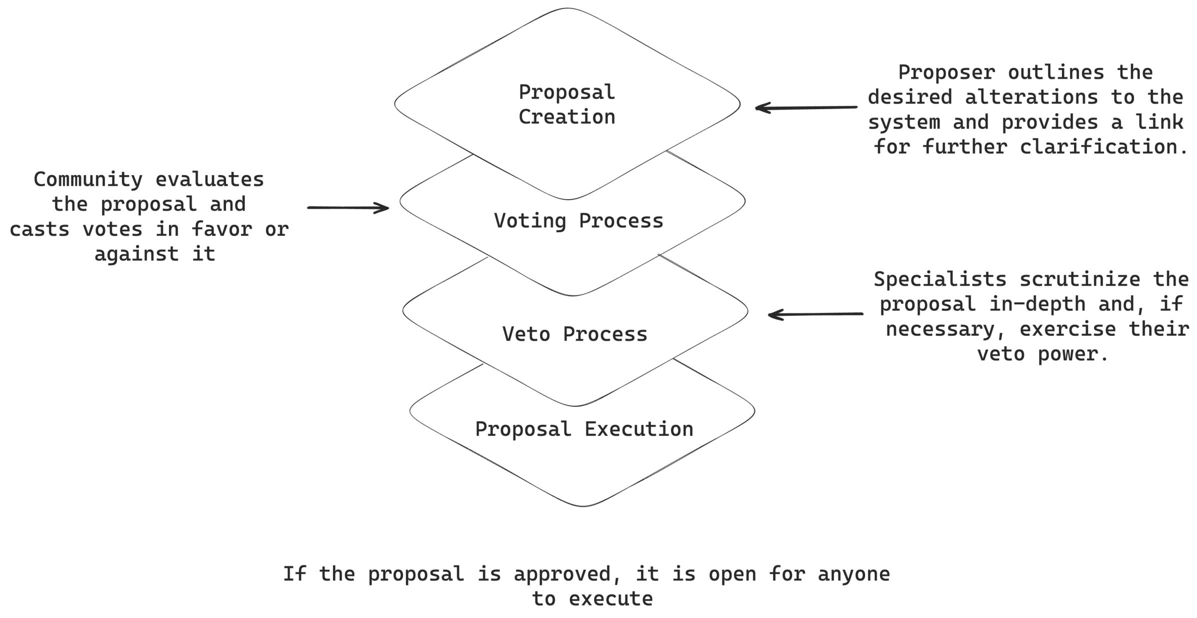
Proposal Creation
Proposer outlines the desired alterations to the system and provides a link for further clarification.
Community evaluates the proposal and casts votes in favor or against it
Voting Process
Specialists scrutinize the proposal in-depth and, if necessary, exercise their veto power.
Veto Process
Proposal Execution
If the proposal is approved, it is open for anyone to execute

Figure 4: Layers of Governance.

Employing a multi-layer strategy is a prudent governance practice adopted by various protocols within the Ethereum ecosystem to bolster system security [12, 13].

4.1. Insights into the Voting Mechanism

The voting process (VP), in its most comprehensive configuration, is bifurcated into two pivotal phases: voting and vetoing. The uniqueness of this process is attributed to the concept of the 'voting situation,' a set of parameters defining the target module of the proposal and specifying the entities endowed with the authority to veto the proposal.

4.1.1. The Role of Veto in Governance

The veto process is integral to the governance of the DAO, acting as a protective mechanism to halt proposals that may diverge from the DAO's foundational principles and constitution. This process is reserved for Experts, appointed during the DAO governance process, and serves as a safeguard to ensure the alignment of all proposals with the DAO's values and objectives.

Experts during the voting phase, assess the proposals based on their adherence to the DAO. If a proposal is deemed detrimental or misaligned with the DAO's interests, they can exercise their veto power to prevent its implementation, thereby preserving the integrity and values of the organization. The veto right can be described as following:

$$\mathrm{VetoRight}(\mathrm{VP}, \mathrm{e}) = \begin{cases} 1, & \text{if } e \in \mathrm{ExpGrp} \\ 0, & \text{otherwise} \end{cases}$$

4.1.2. Voting and Veto Configuration

Veto configurations are customizable for each contract within the DAO, allowing Experts to veto proposals targeting specific modules. The duration for exercising a veto is also adjustable, ensuring flexibility in alignment with the unique needs of each DAO.

In essence, the veto process is a crucial governance tool, ensuring the alignment of proposals with the DAO's principles and protecting the interests of its members by preventing the execution of potentially harmful proposals. The veto process is integral to maintaining the integrity and alignment of a DAO with its foundational principles. Expert Groups, designated during the DAO setup, are entrusted with the authority to veto proposals that may not align with the DAO's fundamentals. This mechanism is designed to act as a safeguard, ensuring that all decisions made within the DAO adhere to its core values and objectives.

The veto process provides a structured method for Experts to assess the merits and alignment of proposals. By scrutinizing proposals against the DAO's principles, they can halt the implementation of any proposal deemed potentially detrimental to the organization or its members. This protective measure is crucial for preventing the enactment of misaligned or harmful decisions, thereby upholding the DAO's integrity.

Moreover, the veto process is adaptable, allowing each DAO to configure veto settings according to its unique requirements. This includes the ability to customize the veto period, ensuring that the duration within which a veto can be exercised aligns with the specific governance needs of the DAO. By incorporating these customizable elements, the veto process enhances the flexibility and responsiveness of the DAO's governance structure.

4.1.3. Voting Configuration Insights

The voting process within a DAO is a fundamental mechanism allowing Members or Experts to participate in decision-making by voting on various proposals. This process is highly configurable, allowing for modifications to proposal types and associated settings, such as quorum and majority, through the parameter voting.

Voting power ($\mathrm{V_p}$) is determined by the number of locked tokens in the DAO.

$$\mathrm{V_p}(e) = \text{tokens locked by } e$$

Therefore,

$$\mathrm{VoteRight}(VP, e) = \begin{cases} \text{true}, & \text{if } \mathrm{V_p}(e) > 0 \\ \text{false}, & \text{otherwise} \end{cases}$$

To ensure a proposal's validity, it must meet a specified quorum (Q), which is the minimum number of votes required.

$$Q = \sum_{e \in E} \mathrm{V_p}(e) \times q_e \quad \text{where } q_e \in [0, 1]$$

Additionally, it should not be vetoed by more than *n* experts in the ExpGrp. The veto is true if more than *n* experts decide to veto the proposal:

$$\text{Veto}(VP, e) = \begin{cases} \text{true}, & \text{if } e \in \text{ExpGrp and VetoRight}(VP, e) > n \\ \text{false}, & \text{otherwise} \end{cases}$$

The system offers three types of voting: Partially Restricted, Restricted, and Non-Restricted Voting, each serving different purposes and allowing various levels of participation from experts and the community. When a proposal is submitted, entities cast their votes. The validation of a proposal depends on meeting the quorum and veto criteria. If the condition is satisfied, the proposal passes and is executed; otherwise, it fails.

$$\text{Execute}(VP) = \begin{cases} \text{true}, & \text{if } \sum_{e \in E} \mathrm{V_p}(e) \geq Q \text{ and } \text{Veto}(VP) \neq \text{true} \\ \text{false}, & \text{otherwise} \end{cases}$$

The voting process is transparent, with users unable to retract or revote, and voting is not anonymous in its basic configuration. The process is subject to various statuses based on user activity and decisions, ranging from Pending to Executed, each representing a different stage in the proposal's lifecycle.

Voting parameters are outlined in the Parameter Storage module, allowing for customization of essential parameters that determine the structure and execution of voting within a DAO. These parameters include voting and veto periods, required quorum and majority, and voting type, among others, enabling flexibility and adaptability as the organization evolves.

Ultimately, due to its modular architecture, any default functionality can be expanded, exemplifying the system's scalability and adaptability. For instance, features such as the ability to retract a vote can be seamlessly integrated.

5. DAO Architecture

This article explores a governance system designed to adapt and scale within an ever-evolving environment, referred to as a DAO within the Ethereum community. For effective DAO operation, the integration of key components such as Permission Manager, Vault, Voting, Member Storage, and Parameters Storage is essential. The integration of these components, as outlined in Figure 5, forms the fundamental structure of the DAO, enabling optimal performance and efficient scalability while maintaining core functionality. Within the Ethereum community, such systems are encapsulated by the term Decentralized Autonomous Organization (DAO), which will henceforth be used to describe our system.

For the effective operation of a DAO, the definition and integration of the following components are essential:

Permission Manager: Administers and regulates access permissions within the system, ensuring structured and secure interactions.

Vault: Serves as a fortified repository for assets and valuables within the DAO, safeguarding against unauthorized access and manipulation.

Voting: Orchestrates the decision-making paradigm, enabling members to propose initiatives and cast votes, thereby driving collective consensus.

Member Storage: Acts as a comprehensive database, cataloging members along with their corresponding roles and permissions, facilitating efficient management and reference.

Parameters Storage: Houses the adjustable parameters and configurations of the DAO, allowing for tailored system settings.

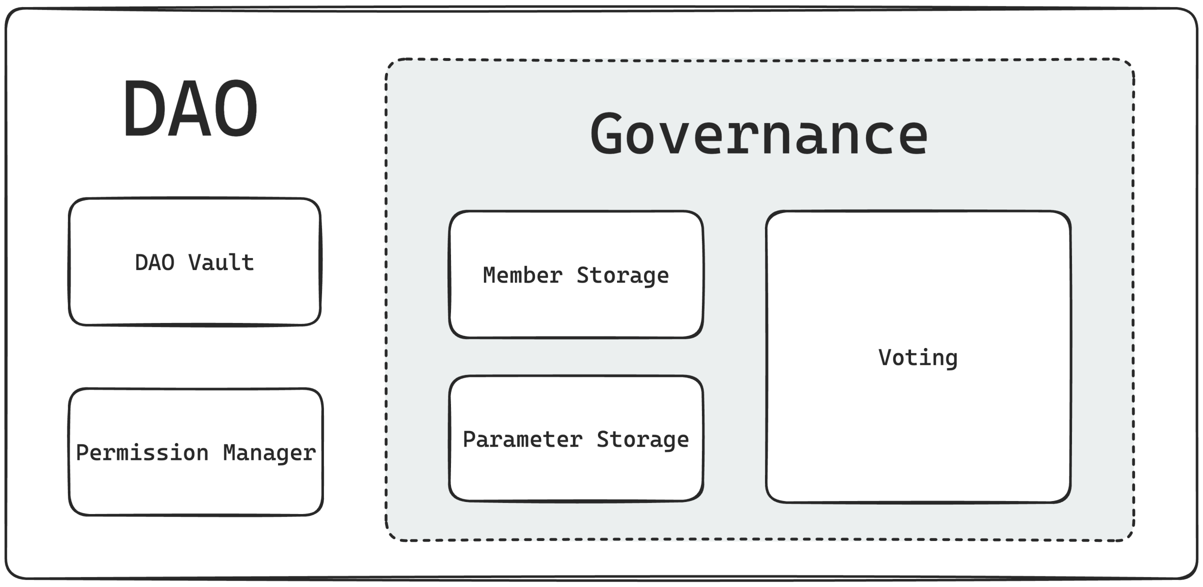

Figure 5: Core DAO Framework.

These components will be discussed in detail in the following subchapters. The integration of these components, as outlined in Figure 5, lays the basis for the fundamental structure. This approach allows the DAO to reach its optimal performance and scale efficiently, while maintaining the core functionality that was originally designed.

5.1. Permission Manager

The Permission Manager Module is a foundational component of the DAO, grounded in RBAS theory. It serves as the nucleus of the DAO, recording and storing permissions, and groups, and defining the essential functionality to uphold the governance structure previously outlined.

This module is pivotal for integrations and upgrades, acting as the gateway for any interactions with the DAO. Every component or entity that interacts with the DAO does so through the Permission Manager Module, ensuring a structured and secure interface for all integrations and modifications.

Given that every segment of the system interacts with the Permission Manager Module, efficiency is crucial, as users incur additional costs for each extra action. The modular architecture and unified shared storage for each core module significantly reduce costs when the module is not engaging with an external resource but merely accessing this module's storage, subsequently lowering the gas usage for users.

In the current landscape, this module can assist in meeting regulation requirements, as it can be effectively used to pause some components without harming the whole project at once. In addition to the modular architecture, every module within the product can be assigned a separate role, and even more than one, allowing system maintainers to act quickly in case of emergency.

In contrast, when a company or other entity deploys their DApps or DAO, they could face significant opposition from the state in which their company is registered. By its nature, all of the contracts that are deployed to the Ethereum network are meant to be immutable. Therefore, only well-designed architecture can solve the problem of regulatory compliance.

5.2. Vault

The Vault component acts as the entry point to the DAO Governance System, serving as a platform where users deposit tokens to engage in DAO governance activities. It is pivotal in governance processes, securing tokens during voting periods and releasing them post-voting to maintain the integrity of the voting process and to thwart governance attacks such as 'double-voting' or vote manipulation through token borrowing.

By delegating the locking mechanism to the Vault rather than embedding it within the token contract, the system accommodates a broad spectrum of existing tokens for DAO governance, enhancing versatility and inclusivity. This approach ensures a secure and reliable voting environment, reinforcing the robustness of the overall governance structure.

Given its modular architecture, this module can effortlessly accommodate new tokens, each representing a new functionality that can be integrated into the vault module. Thus, if new standards emerge post-deployment, or if integration with a previously overlooked standard, such as ERC-5484 [14], becomes commercially significant, it only requires a single proposal to incorporate it into the system.

5.3. Voting

The Voting component orchestrates the voting process, serving as the hub where community members can initiate proposals and cast their votes, and where Experts can exercise their veto power. Given the system’s capability to incorporate new components or extend functionalities to existing ones, the voting mechanism is designed to be adaptable to such modifications.

This component is intricately linked to the Permission Manager contract to validate the eligibility of the proposer to initiate voting, to cast a vote, or to exercise a veto. A generic interface is established to outline the 'voting situation', allowing the configuration of key voting parameters such as quorum, majority, and voting duration.

Using the 'voting situation' interface, the Voting component can concurrently manage both community and expert proposals. It interacts with the Member Storage to facilitate the veto function and with the Parameter Storage to adjust voting parameters.

Managing permissions through RBAS allows the voting module to control functionalities in new modules or to confine new functionalities under existing rules. This method enables abstraction from technical specifics like addresses, allowing immediate implementation of high-level structures, focusing solely on overarching organizational frameworks. Further, these structures can be easily replicated in voting situations.

The Voting component is the cornerstone of governance within the system, ensuring that every governance activity is meticulously managed and executed.

5.4. Member and Parameter Storages

The Member and Parameter Storage modules serve as instrumental tools for pivotal DAO components like the Voting and Permission Manager, optimizing organizational efficiency within the DAO.

The Member Storage component organizes Experts within the DAO, essentially acting as a whitelist for community members elected by their peers to assume Expert roles. It streamlines the identification and management of Experts, ensuring a structured approach to expert involvement in governance activities.

The Parameter Storage module facilitates convenient management of DAO parameters, offering diverse functionalities to interact with parameters essential for various DAO components. It simplifies the adjustment and monitoring of system parameters, enhancing the adaptability of the DAO to evolving governance needs.

The modular architecture enhances these two components by allowing the incorporation of specific functionalities post-deployment of the module, with such functionalities being seamlessly integrated into the governance structure through RBAS.

The synergistic integration of these modules reduces the complexity inherent in other DAO components and elevates the overall auditability and transparency of the system, contributing to a more coherent and manageable governance structure.

5.5. Module Integration Flow

Combining the different modules results in a robust system that can be easily scaled up. Thus, including the Regulatory module requires one proposal from the community, as shown in Figure 6.

Due to the modular smart contract architecture, the new regulatory module can be easily integrated into the DAO core. This integration allows the DAO to ensure compliance with the rules required by the state in which it is registered, but it is only one use case. In addition, the RBAS ensures that this module is simultaneously integrated into the DAO governance system. This means that any configuration desired by the DAO can be achieved, such as a configuration where only experts manage the new module. However, the initial adoption of such rules is subject to community approval.

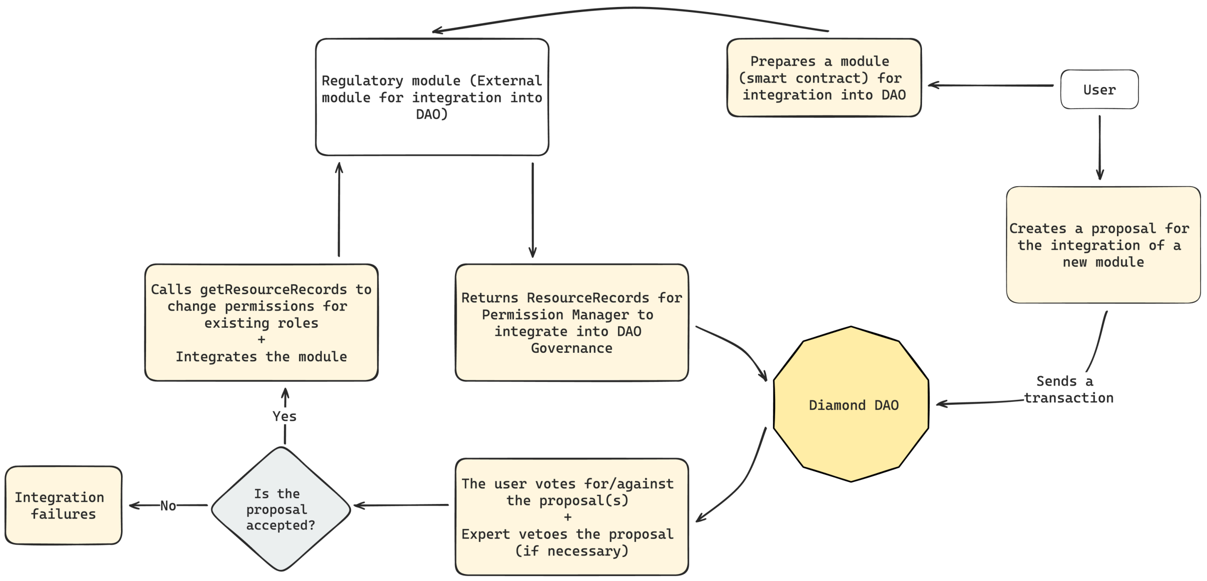
Regulatory module (External module for integration into DAO)
Prepares a module (smart contract) for integration into DAO
User
Creates a proposal for the integration of a new module
Sends a transaction
Diamond DAO
Calls getResourceRecords to change permissions for existing roles + Integrates the module
Returns ResourceRecords for Permission Manager to integrate into DAO Governance
Yes
Integration failures
No
Is the proposal accepted?
The user votes for/against the proposal(s) + Expert vetoes the proposal (if necessary)

Figure 6: Example of Practical Module Integration.

The modular nature of the DAO system allows for the seamless integration of additional functionalities, extending beyond utility modules to encompass various protective and cross-chain capabilities. For instance, specific firewalls can be integrated to safeguard the DAO against malicious actors, enhancing security and stability. Furthermore, cross-chain functionality can be implemented, enabling the DAO to operate not only on its originating blockchain but also on other chains. This cross-chain integration can significantly reduce execution costs by delegating computational tasks, such as proposals or complex validation and verification processes, to different chains.

Additionally, the integration of new modules introduces functionalities that may not have been available at the DAO's inception. For example, modules that integrate with various KYC providers can be added to facilitate voting schemes like one-person-one-vote, ensuring equitable participation. This feature is particularly pertinent in the contemporary landscape, marked by the rise of Artificial Intelligence and Zero-Knowledge Proof-based systems. Such integration underscore the DAO's ability to adapt and scale in response to evolving technological advancements and community needs.

The architecture described in the article lays a solid foundation for the community to build upon and expand. The inherent flexibility and scalability of the modular approach ensure that the DAO can continually evolve, incorporating new technologies and governance models as they emerge. This adaptability is crucial for maintaining the relevance and efficacy of the DAO in a rapidly changing environment. The proposed architecture guarantees these properties, providing a resilient and scalable framework that can meet the diverse and dynamic requirements of modern projects.

5.6. System Properties

Summarizing all the components described above, our system gains two important guarantees:

Liveness: As soon as the system is updated with new modules, it remains actual and relevant. Formally, for any module mod_i, if mod_i is integrated at time t, then the system state S reflects the integration at $t + \epsilon$:

$$\forall \text{mod}_i, \exists t \text{ such that integrated } (\text{mod}_i, t) \Rightarrow S(t + \epsilon) = \text{ updated}$$

Safety: With a guarantee from the voting process and the RBAS system, the system stays secure, ensuring that only authorized and secure actions are allowed. Formally, for any action a on resource r by entity e, if the action is performed, then e has the necessary permissions:

$$\forall e \in E, \forall a \in A, \forall r \in R,\ \text{Perform}(e, a, r) \Rightarrow \text{Access}(e, a, r) = \text{true}$$

These properties ensure that the DAO remains functional and secure, adapting to changes while maintaining strict access controls.

6. Conclusion

This article has presented a scalable and adaptable governance system on the Ethereum network, designed to circumvent the inherent limitations of the Ethereum protocol. The foundation of this system is the RBAS, which orchestrates the interactions between various components within a DAO. However, RBAS alone is insufficient to overcome the constraints related to Smart Contract size and the immutable nature of contract logic post-deployment inherent in the Ethereum protocol.

To address these challenges, we introduced a modular system architecture, allowing the DAO to expand and integrate a diverse range of modules while maintaining coherent governance. This modular approach, coupled with the Expert governance structure, enhances the security and reliability of the system, ensuring robust protection against potential vulnerabilities.

We have also delineated a core set of components essential for initiating the functionality of the DAO, laying the groundwork for a system that is not only scalable and adaptable but also secure and governed with precision and transparency.

References

[1] F. P. Miller, A. F. Vandome, Information Theory, Alpha Press, 2009.

[2] A. Antonopoulos, G. Wood, Mastering ethereum: building smart contracts and dapps, O'Reilly Media, 2018. URL: https://github.com/ethereumbook/ethereumbook/

[3] E. Foundation, Introduction to smart contracts, Technical Report, 2023. URL: https://ethereum.org/en/smart-contracts/.

[4] Ethereum: a secure decentralised generalised transaction ledger, Technical Report, Ethereum & Parity, 2023. URL: https://ethereum.github.io/yellowpaper/paper.pdf.

[5] OpenZeppelin, Proxy upgrade pattern, Technical Report, 2018 URL: https://docs.openzeppelin.com/upgrades-plugins/1.x/proxies.

[6] W. A. K., Blockchain-based corporate governance, Stanford Journal of Blockchain Law & Policy (2021). URL: https://stanford-jblp.pubpub.org/pub/ blockchain-corporate-governance, professor at University of Saint Thomas School of Law.

[7] N. Mudge, EIP-2535, Diamonds, Multi-Facet Proxy, Technical Report, 2020. URL: https://eips.ethereum.org/EIPS/eip-2535.

[8] C. B. Center, Disclosure, dapps and defi, Stanford Journal of Blockchain Law & Policy (2022). URL: https://stanford-jblp.pubpub.org/pub/disclosure-dapps-defi, agnes N. Williams Professor of Law at Georgetown University Law.

[9] E. Foundation, EIP-20, Token standard, Technical Report, 2015. URL: https://eips.ethereum.org/EIPS/eip-20.

[10] E. Foundation, EIP-721, Non-Fungible token standard, Technical Report, 2018. URL: https://eips.ethereum.org/EIPS/eip-721.

[11] E. Foundation, EIP-1155, Multi token standard, Technical Report, 2018. URL: https://eips.ethereum.org/EIPS/eip-1155.

[12] Synthetix, Governance, Technical Report, 2023. URL: https://synthetix.io/governance.

[13] Q. D. AG, Q whitepaper, v1.0, Technical Report, Q Foundation, 2018. URL: https://q.org/files/Q_Whitepaper_v1.0.pdf.

[14] E. Foundation, EIP-5484, Consensual soulbound tokens, Technical Report, 2022. URL:

https://eips.ethereum.org/EIPS/eip-5484.