
BGRA: A REFERENCE ARCHITECTURE FOR BLOCKCHAIN GOVERNANCE

Yue Liu^{1,2}, Qinghua Lu^{1,2}, Guangsheng Yu¹, Hye-Young Paik², Liming Zhu^{1,2}

¹Data61, CSIRO, Australia

²University of New South Wales, Australia

yue.liu@data61.csiro.au, qinghua.lu@data61.csiro.au, saber.yu@data61.csiro.au

h.paik@unsw.edu.au, liming.zhu@data61.csiro.au

November 14, 2022

ABSTRACT

Blockchain technology has been integrated into diverse software applications by enabling a decentralised architecture design. However, the defects of on-chain algorithmic mechanisms, and tedious disputes and debates in off-chain communities may affect the operation of blockchain systems. Accordingly, blockchain governance has received great interest for supporting the design, use, and maintenance of blockchain systems, hence improving the overall trustworthiness. Although much effort has been put into this research topic, there is a distinct lack of consideration for blockchain governance from the perspective of software architecture design. In this study, we propose a pattern-oriented reference architecture for governance-driven blockchain systems, which can provide guidance for future blockchain architecture design. We design the reference architecture based on an extensive review of architecture patterns for blockchain governance in academic literature and industry implementation. The reference architecture consists of four layers. We demonstrate the components in each layer, annotating with the identified patterns. A qualitative analysis of mapping two concrete blockchain architectures, Polkadot and Quorum, on the reference architecture is conducted, to evaluate the correctness and utility of proposed reference architecture.

Software engineering, reference architecture, blockchain governance, decision rights, incentive, accountability, pattern.

1 Introduction

Blockchain can provide both distributed data storage and computing platform, where a large network of untrusted participants need to reach agreements on transactional data states [1]. As an innovative distributed ledger technology, blockchain has improved certain software attributes and brought its distinctive features, e.g., transparency, immutability, and on-chain autonomy, into various application scenarios. In recent years, blockchain has been leveraged as a software component in application systems in a substantial number of projects by enabling a decentralised infrastructure [2], for instance, energy supply [3], industrial IoT [4], etc.

Despite being considered a viable solution to re-architect application systems, there are significantly increased concerns that blockchain systems may suffer from the defects of on-chain algorithmic mechanisms, and tedious disputes and debates in off-chain communities. The negative crises in two world-renowned blockchain systems, Ethereum and Bitcoin, have severely affected the trustworthiness of blockchain. In 2016, the “DAO” (Decentralised Autonomous Organisation) attack in Ethereum was caused by flaws in smart contract code and resulted in the loss of over 60 million US dollars. This was remedied by conducting a hard fork to reverse the impacted transactions [5]. While in Bitcoin, the debate of whether to increase block size caused the split of the whole ecosystem [6]. After these events, the blockchain community started to explore a more trustworthy governance process for both on-chain and off-chain businesses.

Blockchain governance refers to the structures and processes that are designed to ensure the development and use of blockchain are compliant with legal regulations and ethical responsibilities [7]. It determines the allocation of decision

rights, incentives, and accountability based on the blockchain decentralisation level, which further regulates stakeholders' behaviour throughout the whole blockchain development lifecycle, and the overall blockchain ecosystem. Blockchain governance can refer to existing governance frameworks (e.g., IT governance [8, 9], data governance [10, 11]), while further investigation is also necessary considering the absence of a clear source of authority in blockchain systems. In recent years, there are continuously increasing attention focusing on this research topic, including the customised governance methods in permissioned blockchains [12, 13], regulations for blockchain-based decentralised finance (e.g., cryptocurrencies) [14, 15], etc.

Nevertheless, it is found that there is a lack of consideration for software architecture design in this area, which may hinder the design and implementation of blockchain with proper governance solutions, resulting in conflicts between stakeholders and failures of blockchain systems. In this regard, this paper presents a pattern-oriented reference architecture, which can serve as a guideline to assist system architects and developers in the development of governance-driven blockchain systems, with reusable patterns as architecture components.

The contributions of this paper are as follows:

- We propose a reference architecture to guide and facilitate the design and development of governance-driven blockchain systems. To the best of our knowledge, this is the first study learning blockchain governance from the perspective of architecture design.
- We associate multiple architectural patterns with the different components in the proposed reference architecture, to address the recurring governance-related issues in blockchain systems. The architectural patterns are gathered and analysed via a systematic literature review and further review of multiple blockchain systems.
- We evaluate the correctness and utility of our proposed reference architecture, by mapping two blockchain system architectures on the proposed reference architecture.

The remainder of this paper is organised as follows. Section 2 introduces background knowledge and related work. Section 3 explains our research methodology. Section 4 presents the overall reference architecture with annotated patterns. Section 5 evaluates our architecture. Section 6 concludes the paper and outlines future work.

2 Background and Related Work

2.1 Blockchain

Blockchain was popularised by Bitcoin [16] and the subsequent cryptocurrencies. The concept of blockchain was then generalised to distributed ledger technology, and considered an emerging paradigm for building next-generation applications in a decentralised way. In a software system, the blockchain component can provide two core elements: (i) a distributed ledger, and (ii) a decentralised “compute” infrastructure.

Blockchain can verify and store digital transactions via the underlying distributed ledger, without relying on any central authority to establish trust between the interoperating entities [1]. In permissionless blockchains, trust is preserved via game theoretic incentives to maintain a majority of honest nodes [16]. While in permissioned blockchains, trust is achieved through the compulsory identity verification of participating entities. On-chain transactions carry the changing states of data, and blocks are containers for storing transactions. Except for the genesis block, all the blocks are linked to the previous block and thus form a chain.

Blockchain can be leveraged as a “compute” infrastructure via the on-chain programmability (i.e., smart contracts). Smart contracts are user-defined programs that can be deployed and executed in a blockchain system to enable complex business logic such as triggers and conditions [17]. For instance, Ethereum provides a built-in Turing-complete scripting language, Solidity, for developing smart contracts.

2.2 Blockchain Governance

Existing studies have analysed the governance frameworks for blockchain [13, 18, 19, 20]. Please note that in this study, the term “blockchain governance” refers to “governance of blockchain”, where we focus on how governance fits in the development and use of blockchain.

Essentially, blockchain can be classified into three types to meet different requirements. Different types of blockchain reflect the different levels of decentralisation, and affect the governance structure regarding the allocation of decision rights, accountability, and incentives. In a blockchain system, a transparent decision-making process can help oversee whether decisions are reasonable, hence, to gain the trust of all stakeholders. Accountability can be established via both institutional and technical manners, to ensure the identifiability and answerability of stakeholders for their decisions. In

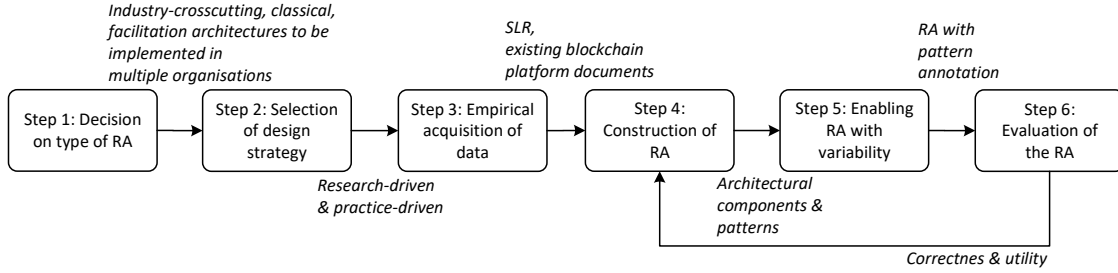


Figure 1: Methodology.

blockchain governance, incentives are considered factors that may influence stakeholders’ behaviours. The governance structure can provide incentives to motivate desirable behaviours and resolve conflicts between stakeholders.

In addition, blockchain governance should be realised throughout the overall ecosystem. For on-chain transactions, the governance emphasises accountable access control. The capabilities of sending, validating, and reading transactions are assigned to different stakeholders considering the selected blockchain type. Regarding the blockchain platforms, they need to undergo a series of formalised procedures to finalise improvement proposals. Further, blockchain-based applications need to comply with industry regulations and specifications, any changes may lead to upgrades of the underlying blockchain platform. For the off-chain community governance, the stakeholders are gradually divided into different groups regarding their roles and decision rights, e.g., stakeholders may have different communication channels for certain issues.

Furthermore, blockchain governance should ensure that the related decisions and processes conform to legal regulations and ethical responsibilities. Specifically, managing legal compliance relies on local and international policies regarding where to deploy a blockchain, while promoting ethical guidelines can preserve human values in blockchain governance.

In recent years, many researchers explore the topic of blockchain governance from diverse perspectives. For instance, Katina et al. [21] propose and analyse seven interrelated elements of blockchain governance, including philosophy, theory, axiology, methodology, axiomatic, method and applications. Beck et al. [13] adopt the three major dimensions of IT governance (i.e., decision rights, incentives, and accountability), and discuss their allocations in blockchain governance. Allen and Berg [22] focus on the exogenous and endogenous governance methods for blockchain platforms, while John and Pam [23] and Pelt et al. [18] both investigate this topic regarding on-chain and off-chain development processes. Hofman et al. [19] propose a high-level analytic framework for blockchain governance, covering six different aspects (i.e., why, who, when, what, where, and how). However, the existing studies only provide general discussion and high-level principles to realise governance, while practitioners require more detailed solutions to face the learning curve and facilitate the architecture design of governance-driven blockchain systems.

2.3 Reference Architecture

A reference architecture can be regarded as “a reference model mapped onto software elements (that cooperatively implement the functionality defined in the reference model) and the data flow between them. Whereas a reference model divides the functionality, a reference architecture is the mapping of that functionality onto a system decomposition” [24]. A reference architecture can support system development by addressing the inclusive business rules, architectural styles, best practices of software development, and software elements [25]. There are existing studies about blockchain reference model and reference architecture. For instance, Yuan and Wang propose a 6-layer reference model of blockchain [26], while Ellervee et al. [27] present a blockchain reference model from the perspectives of actors, services, processes, and data models. Homoliak et al. [28] present a security reference architecture for blockchain based on the blockchain network implementation stacks proposed by Wang et al. [29]. In addition, there are reference architectures for diverse blockchain-based applications, for instance, crowdsourcing [30], healthcare [31], and government services [32], etc. Nonetheless, we found a lack of consideration of software architecture design for blockchain governance. Hence, this study illustrates a pattern-oriented reference architecture for governance-driven blockchain systems.

3 Methodology

This section introduces the methodology of this study. We adopted an empirically-grounded design methodology for reference architecture [33], and the overall research process is illustrated in Figure 1. There are six steps, and each step has an output to the following step.

The first step is to determine the type of our reference architecture. Based on Galster and Avgeriou’s proposal [33], the reference architecture is an industry-crosscutting (*usage context*), classical (*when*), facilitation (*why*) architecture to be implemented in multiple organisations (*where*). Hereby, “industry-crosscutting” means that the reference architecture can cover more than one industry, and “classical” refers to that this reference architecture is developed based on existing blockchain systems. “Facilitation” indicates that the reference architecture aims to provide guidance for the future design of blockchain systems, while “multiple organisations” is determined by the decentralised nature of blockchain.

The second step is to select our design strategy. In this study, our design strategy is a combination of “research-driven” and “practice-driven”. “Research-driven” means that the design of this reference architecture is based on state-of-the-art research from a systematic literature review, while “practice-driven” is also applicable since this study is based on the review results of multiple extant blockchain systems and summary of the best-practices for blockchain governance.

The third step is the empirical acquisition of data. This study adopts and extends several existing studies as data acquisition. Specifically, Liu et al. performed a systematic literature review, in which 37 primary studies were selected and analysed regarding six research questions [7]. The extracted results covered the definition, motivations, objects, process, stakeholders, and mechanisms of blockchain governance. Afterwards, the researchers reviewed the existing governance frameworks and standards (i.e., IT governance [8, 9], data governance [10, 11], OSS governance [34, 35], platform ecosystem governance [36]), to understand the characteristics of blockchain governance. They also scrutinised the open websites and documents of five blockchain platforms (i.e., Bitcoin, Ethereum, Dash, Tezos, and Hyperledger Fabric), to understand how blockchain governance is implemented in a real-world context [20]. In addition, they presented a pattern language for blockchain governance [37].

Based on the acquired data, in the next step, we constructed a reference architecture for governance-driven blockchain systems by integrating the architectural patterns into a widely-accepted reference model of blockchain [38]. Meanwhile, the variability of our reference architecture design in step five was enabled by annotating that applying different patterns can lead to the instantiation of various concrete architectures for blockchain systems.

In the final step, the evaluation of our proposed reference architecture was carried out by reviewing two other blockchain systems, Polkadot and Quorum. We map the architectural components of Polkadot and Quorum on the proposed reference architecture, to prove that the reference architecture can be transformed into meaningful concrete architectures. Further, the evaluation results can help refine the reference architecture.

4 Reference Architecture

In this section, we present a pattern-oriented reference architecture for governance-driven blockchain systems. Figure 2 illustrates the overview of the architecture, which consists of: 1) infrastructure layer, 2) platform layer, 3) API layer, 4) user layer, and 5) cross-layer functions. Specifically, the platform layer and API layer should be implemented in each participating node, and all nodes in a blockchain system share the same infrastructure layer and cross-layer functions. Moreover, this figure includes non-blockchain systems and other blockchain systems to illustrate the interactions of API layer. We apply a set of patterns as architectural components to realise governance in the reference architecture, which are annotated in the figure. In addition, we summarise these components in Table 1, to explain the applicable decentralised level, type, and responsibility of each annotated component.

4.1 Infrastructure Layer

First, the infrastructure layer of a blockchain system consists of the functional components for building a fundamental operating environment, including *data storage*, *communication network*, and *computation*. For *data storage*, the blockchain system architects or developers need to determine the physical and logical location of on-chain data, and corresponding CRUD (i.e., create, read, update and delete) operations. *Communication network* includes the peer-to-peer network for blockchain nodes, connection between stakeholders and the blockchain system, and interactions between a blockchain system with other systems. In the *communication network*, **network freezer** can be leveraged by the system administrators or governors to suspend all on-chain business. This pattern can disconnect the nodes or block data traffic in a short time, to avoid the negative impact in an emergency situation. A frozen blockchain system requires human interventions for reactivation. *Computation* enables the runtime environment for each node, and on-chain programmability with complex business logic. In this layer, the structure of blockchain may influence the other three main components. Specifically, a blockchain can have either a single or multiple shards. A **sharded chain** refers to that a blockchain is partitioned into different shards, consequently, the data storage, communication network, and computation are accordingly split regarding the shards. Blockchain nodes only need to process the transactions in their own shards. Compared with single-shard blockchains, a **sharded chain** is more scalable and has better performance.

Table 1: Pattern-oriented components in the reference architecture.

Component	Decentralisation level	Type	Responsibility
Network freezer	Permissioned & permissionless	Optional	Suspending blockchain transactions at the network level, to stop the broadcast of malicious transactions.
Sharded chain	Permissioned & permissionless	Optional	Splitting blockchain into multiple shards, where the data storage, computation, and communication are accordingly split, to improve the scalability of blockchain systems.
Incentive distributor	Permissioned & permissionless	Optional	Providing on-chain tokens to drive the motivation and behaviour of stakeholders in decision-making process.
Protocol upgrade	Permissioned & permissionless	Mandatory	Implementing the software upgrades to a blockchain system.
Data migrator	Permissioned & permissionless	Optional	Migrating data from a source blockchain system to target blockchain system(s) based on data governance and management requirements.
Participation permission	Permissioned	Optional	Managing the participation to a blockchain system, requiring real-world identity verification and the approval of authorities.
Accountability tracer	Permissioned & permissionless	Mandatory	Identifying the source of a blockchain transaction, to ensure the accountability of transaction senders.
Benevolent dictator	Permissioned & permissionless	Mandatory	Specific stakeholders possess additional decision rights for certain governance-related issues.
Transaction filter	Permissioned & permissionless	Optional	Examining submitted transactions to ensure the validity of transaction format/content, rejecting and discarding invalid transactions.
Validator selection	Permissioned & permissionless	Mandatory	Selecting the node operator who is eligible to validate and append the candidate block to the blockchain.
Block finality decider	Permissioned & permissionless	Optional	Waiting for a certain number of subsequent blocks to confirm that a previous block and its contained data is finalised and immutable.
Log extractor	Permissioned & permissionless	Optional	Extracting logged event information from the blockchain system for further analysis and audit.
Contract freezer	Permissioned & permissionless	Optional	Suspending all the operations to a particular smart contract.
Social contract	Permissioned & permissionless	Optional	Specifying the future maintainer or qualification of maintainers for a blockchain system.
Scam list	Permissionless	Optional	Listing the malicious blockchain addresses to warn all stakeholders of risky interactions.
Token locker	Permissionless	Optional	Locking a certain amount of on-chain tokens for a specified time period, to restrict the token holder's behaviour in a decision-making process.
Carbonvote	Permissionless	Optional	Counting votes for improvement proposals according to the tokens held by blockchain addresses, to prevent Sybil attack.
Quadratic voting	Permissionless	Optional	Consuming n^2 number of tokens when a blockchain address submits n votes for an improvement proposal, to capture the preference of stakeholders' decisions.
Cross-chain token voting	Permissionless	Optional	Issuing tokens and holding votes in other blockchain systems, for the improvement proposals in the original blockchain system.
Liquid democracy	Permissionless	Optional	Delegating the decision rights and revoking the delegation for improvement proposals to/from other stakeholders.

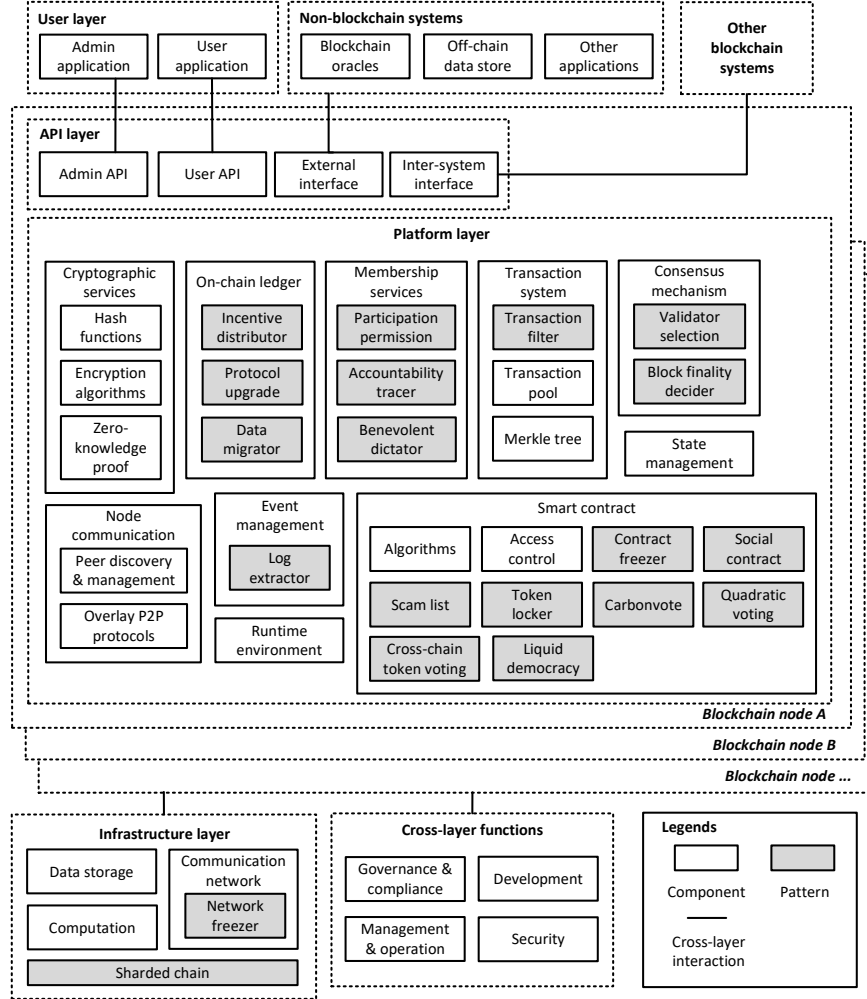


Figure 2: A pattern-oriented reference architecture for governance-driven blockchain systems.

4.2 Platform Layer

The core services and features of a blockchain system are implemented and embodied in the platform layer. The main components in this layer include *cryptographic services*, *on-chain ledger*, *membership services*, *transaction system*, *consensus mechanism*, *node communication*, *event management*, *runtime environment*, and *smart contract*.

A blockchain system incorporates a series of *cryptographic services* to preserve data confidentiality and integrity. For instance, *hash functions* can map arbitrary-size data to fixed-size data. The above-mentioned *Merkle tree* is supported by *hash functions*. *Encryption algorithms* can generate and decrypt ciphertext via secret keys. *Zero-knowledge proof* can preserve privacy in verification issues by only confirming that an entity knows or possess certain data without revealing the actual data.

Essentially, the *on-chain ledger* is the implementation of *data storage* in the infrastructure layer. Each participating node maintains a local replica of the blockchain ledger to preserve data integrity, availability, and consistency. A full node keeps all historical transaction information while a light node can only store the block headers, but a light node needs to rely on full nodes for data enquiry. To align the distinct objectives of stakeholders, especially the nodes, the *incentive distributor* rewards tokens (i.e., programmable digital assets) to stakeholders who obey the codified rules and contribute to the operation of a blockchain system. More generally, the *incentive distributor* can drive stakeholders' motivation and behaviour in a decision-making process. In addition, new functionalities of the blockchain system are implemented via *protocol upgrade*, which may affect the records of on-chain ledger if forking is required. Please note that there are two types of forking: i) backward-compatible upgrades as soft forks, and ii) backward-incompatible upgrades as hard forks. For on-chain ledger data, a *data migrator* is responsible for interacting with external migration

tools, which can help migrate the ledger data from a source blockchain system to a target blockchain system, enabling more comprehensive on-chain data management and governance services. Migrating on-chain data can be realised via multiple ways [39], for instance, generating a snapshot of the source blockchain (including the entire states, smart contracts, and transactions), cloning a node from the source blockchain system, etc.

Membership services implemented in blockchain systems are related to the decentralisation level of the deployed blockchain. Specifically, *participation permission* refers to the identity verification of stakeholders, and approval of authorities (e.g., system administrator) in permissioned blockchain systems. *Accountability tracer* is enabled by the digital signature of transaction senders. Every transaction needs the sender's signature, which is generated via two steps: i) hashing the original data, and ii) encrypting the hash value. Transactions need to be verified by block validators before they are officially recorded. If a transaction contains malicious information, the decrypted hash value can be used to check whether the transaction data is altered during transmission, and the signature can ensure the traceability and identifiability of transaction senders. Please note that in permissioned blockchain systems, accountability can be realised in terms of identifying the real-world stakeholders, while in permissionless blockchain systems, only accountable blockchain addresses can be located due to the inherent anonymity. In permissionless blockchain systems, *benevolent dictator* is arranged throughout the blockchain development and operation stages, referring to stakeholders who have more decision rights than others. For instance, stakeholders may trust the decisions of core developers, who are considered the benevolent dictators based on their expertise of technical meritocracy and the collective benefits of update decisions.

The *transaction system* is closely connected to the *on-chain ledger*. This component can be regarded as the data entry of blockchain. All varying data states on blockchain are carried by the transactions. When a transaction is generated and sent to the blockchain system, a deployed *transaction filter* can examine whether the submitted transactions meet the format or content requirements predefined by the blockchain project team or administrators, to avoid unauthorised or harmful information being fed to blockchain. The valid transactions are temporarily collected in the *transaction pool*, which is a local memory in each node. Nodes can select transactions from the pool and generate candidate blocks, in which the transaction information is compressed in the form of *Merkle tree*. A *Merkle tree* is created via hashing the transactions, and then iteratively summarising and hashing the hash values until a Merkle root is generated. This data structure can preserve data integrity, facilitate the verification of historical transactions, and save the local space of nodes.

Blockchain in practice is a distributed ledger technology where each participating node holds a local replica of the whole ledger contents. A critical issue is how the multiple nodes can agree with the states of blockchain. Conflicts about the blockchain states will impact the security and availability of the overall platform: i) The ledger contents may be compromised if they are not synchronised across the nodes. Attackers can easily modify historical transactions and claim to be the valid version. ii) Requests for the same data at the same time are replied with identical responses. To address the above problems, *consensus mechanisms* are leveraged as a governance method to align the agreement of different nodes. When node operators join a blockchain system, they should all synchronise the ledger contents. During the blockchain operation, each node collects pending transactions and generates a candidate block. *Validator selection* decides a block validator each round, who is allowed to append its block to the blockchain, while other nodes all need to synchronise this block in their local replicas. The block validator can be selected according to different criteria, which are implemented as diverse consensus mechanisms (e.g., computation capability in Proof-of-Work, possessed stakes in Proof-of-Stake, appointed by the system administrator in Proof-of-Authority, etc.). *Block finality decider* reinforces the immutability of blocks and their contained transactions in the way that, after a block is appended to blockchain, certain numbers of subsequent blocks can be regarded as the confirmation to ensure that the previous block is recorded and finalised with high probability.

In a blockchain system, each node needs to keep listening to the network to collect broadcast transactions and synchronise appended blocks, which is accomplished via *peer discovery and management* and *overlay P2P protocols*. These two components compose the *node communication* component. The *state management* component can be exploited to update the on-chain digital assets (e.g., tokens) based on the new transactions, while the *event management* component logs the required information when a transaction triggers particular event(s). Stakeholders can analyse the logged information via *log extractor*. In addition, *runtime environment* supports the execution of smart contracts.

Smart contract denotes the programs run in a blockchain, which enables the decentralised applications built on a blockchain system. A smart contract function is triggered by transactions, and the outputs can be stored on-chain. Smart contracts can be codified with various algorithms to provide different functionalities, or access control mechanisms to restrict users' behaviour [40]. Developers can implement *contract freezer* in smart contracts, and define stakeholders who are eligible to trigger the freezer. *Contract freezer* can pause or terminate all operations to a smart contract, when an attack is made to this contract. In addition, the blockchain project team or system administrators can deploy a special kind of smart contract, *social contract*, to announce the future maintainers, or qualification of future maintainers

for a blockchain system. If malicious operations are identified, the related blockchain addresses, both stakeholders and smart contracts, can be recorded and listed in the *scam list*, which is also in the form of smart contract. Other stakeholders can stop interacting with these scams referring to the list. *Token locker* can be deployed in permissionless blockchain systems, which can grant stakeholders the decision rights for specific governance issues (e.g., the approval of improvement proposals) when stakeholders lock a particular number of tokens for a certain time period as the security deposit. If stakeholders do not obey the rules during a decision-making process, the decision rights will be revoked, and the locked tokens may be destructed. In addition, a series of patterns for voting can be implemented via smart contracts to resolve conflicts and reach consensus. For instance, to update a blockchain system, stakeholders can submit, broadcast, and discuss improvement proposals via off-chain means, while the final decisions are usually made by virtue of voting. In *carbonvote*, the votes are counted in terms of the number of tokens possessed by stakeholders, to prevent the Sybil attack where a stakeholder can register multiple blockchain addresses to compromise the vote. *Quadratic voting* can express stakeholders' preferences when finalising improvement proposals. In this voting scheme, voting for improvement proposals consumes tokens as funds, and the preference is indicated via the exponential increase of consumed tokens that submitting n votes will cost n^2 number of tokens. *Cross-chain token voting* requires the interactions between blockchain systems. The blockchain project team or system administrator needs to issue tokens and deploy smart contracts for voting in source blockchain systems, and the token holders are eligible to vote for improvement proposals in the original blockchain system. *Liquid democracy* allows stakeholders to delegate the decision rights to other trusted stakeholders, and revoke the delegation anytime. Finally, all approved improvement proposals are implemented via *protocol upgrade*.

4.3 API Layer, User Layer, and Other Systems

The API layer consists of four main types of functions for a blockchain node to provide services to different applications and systems. Specifically, *admin API* and *user API* can provide access to the platform layer components for system administrators and users via *admin application* and *user application* in the user layer respectively. The *external interface* can connect a blockchain system to non-blockchain systems such as oracles, and off-chain databases, while the *inter-system interface* is for the communication between nodes in different blockchain systems.

4.4 Cross-Layer Functions

Cross-layer functions include *governance and compliance*, *development*, *management and operation*, and *security*, to provide auxiliary services to components in all other layers. In particular, *governance and compliance* highlights the allocation of decision rights, incentives, and accountability within a blockchain system. This component specifies the high-level guidelines of how a blockchain system is operated and maintained, to meet the legal regulations, industry specifications, and broader ethical requirements, while other cross-layer functions need to refer to its guidance. The *development* component is exclusively for developers' activities, for instance, codifying and testing the updated rules, building software packages, etc. This component needs to record developers' contributions for further incentive distribution and accountability process. *Management and operation* can be regarded as the execution of *governance and compliance* by system administrators, who need to monitor, manage, and control the blockchain system to ensure normal operation and risk management. The *security* component can preserve data confidentiality, integrity and availability in a blockchain system, via predefined decision rights and fine-grained access control of certain stakeholders (e.g., node operators) to restrict their behaviour, positive or negative incentives to drive their motivations, and identity verification and management to establish a complete accountability process.

5 Evaluation

In this section, we evaluate the correctness and utility of our proposed reference architecture by mapping existing blockchain system architectures on the reference architecture. Regarding that our reference architecture is adapted from a widely-accepted reference model, the evaluation focuses on the validation of pattern-oriented architecture components for governance. We selected two blockchain systems: Polkadot¹ and Quorum². Polkadot maintains a permissionless multi-chain ecosystem well-known for the cross-chain interoperability, while Quorum provides permissioned blockchain systems for enterprises and individuals. We collected and scrutinised the available documents provided by these two blockchain systems, and mapped the pattern-oriented components to the reference architecture. Figure 3 and 4 demonstrate the simplified architecture mapping of Polkadot and Quorum respectively, and Table 2 present an intuitive comparison of these two blockchain systems regarding the use of components for governance.

¹<https://polkadot.network/>

²<https://consensys.net/quorum/>

Table 2: Comparison of the use of pattern-oriented components in Polkadot and Quorum architectures.

Component	Polkadot	Quorum
Network freezer	Block validators can vote to suspend the validation system of a certain parachain.	System administrators can suspend the operations of blockchain nodes or accounts to freeze the system.
Sharded chain	Parachains are operated as shards, while the relay chain is regarded as the coordinator between different parachains.	N/A
Incentive distributor	Stakeholders are rewarded for their contributions to Polkadot operation.	Incentives are inactivated by default, and can be triggered during deployment.
Protocol upgrade	Polkadot can upgrade the on-chain protocol without forking.	All participants need to approve the upgrade, and update the configuration file.
Data migrator	Polkadot can interact with external blockchain systems via cross-consensus messages.	N/A
Participation permission	N/A	The deployer of a Quorum system needs to directly send invitations to other participants.
Accountability tracer	Polkadot participants are identified via their on-chain addresses.	The accountability includes stakeholders' real-world identities based on <i>participation permission</i> .
Benevolent dictator	The council and technical committee have additional decision rights for improvement proposals.	The Quorum project team can provide additional support to a Quorum system.
Transaction filter	Polkadot defines a universal transaction format across the system.	Quorum offers different transaction types, transactions not meeting specific type requirements cannot be executed.
Validator selection	Block validators are selected according to the staked tokens of candidates and nominators.	Block validators are determined and assigned by the system administrators.
Block finality decider	Block validators vote to decide the valid chain, where the blocks are finalised.	In Quorum, certain consensus protocols can achieve immediate finality after a new block is appended to the blockchain.
Log extractor	N/A	System administrators can monitor and analyse all activities within a Quorum system.
Token locker	Acquiring certain decision rights requires the locking of Polkadot tokens, e.g., becoming a block validator.	N/A
Carbonvote	Votes are counted regarding the number and locking period of Polkadot tokens.	N/A

5.1 Architecture mapping of Polkadot

Polkadot consists of multiple parachains which can process transactions independently, and a relay chain to collect and confirm all blocks generated by each parachain and enable communication between them. In general, Polkadot can be regarded as a replicated sharded state machine (*sharded chain*) where all parachains operate as shards, and the relay chain is responsible to preserve the consensus among all shards [41]. While inter-shard communication is facilitated by the relay chain, *data migrator* is realised via the cross-consensus message format and protocols, through which Polkadot can send, receive and process data to/from external blockchain systems [42].

Polkadot implements *incentive distributor* via inherent token issuance. Stakeholders contributing to the relay chain and parachain operation are rewarded with tokens, e.g., validators and nominators can obtain tokens according to their staked tokens after block inclusion in the relay chain, and fishermen can get rewards by reporting illegal actions in parachains [41]. In Polkadot, transaction fees are split into two parts: one fraction is paid to the validator, while the other fraction is saved to support the implementation of future improvement proposals. Meanwhile, *token locker* is enforced in different activities in Polkadot to assign certain decision rights to stakeholders while also restraint their behaviors [41, 43]. For instance, the selection of block validators, auction of parachain slots, and voting of improvement proposals all require stakeholders to deposit a certain number of tokens during the event. Any malicious operations may cause the loss of Polkadot tokens.

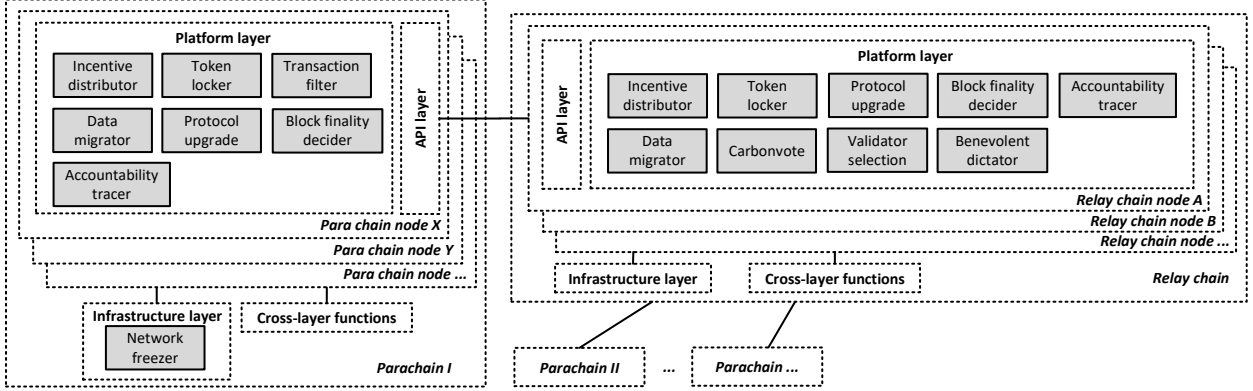


Figure 3: Architecture mapping of Polkadot.

In Polkadot, all on-chain *protocol upgrades* need to undergo referendum before implementation. A variant of *carbonvote* is found applied in Polkadot that votes are counted regarding the number of staked tokens, and also the staked period [41]. A voluntary extended locking of tokens can increase the voting power of stakeholders, since a long-term locking can express the preference of stakeholders' decisions to some extent. Accepted proposals are implemented via upgrading Polkadot's WebAssembly execution host without the need of forking [44].

Polkadot employs the Nominated Proof-of-Stake consensus mechanism, where *validator selection* is according to the staked tokens of validator candidates themselves or nominators [41]. Finally, a set of validators are selected and randomly assigned to each parachain at the beginning of every era (i.e., roughly one day). In each parachain, collators are responsible for the collection and execution of transactions, and generate blocks for the assigned validators. Note that Polkadot leverages *transaction filter* by defining a universal transaction format [45]. Validators need to examine the parachain blocks, while a relay chain block is produced via the Blind Assignment for Blockchain Extension protocol [46]. For each parachain, the *block finality decider* would be the relay chain block, which means that a parachain block is finalised when it is included in a relay chain block. Whilst, the *block finality decider* for Polkadot's relay chain is the GRANDPA protocol [47], in which validators need to vote for the longest relay chain. When more than two third of validators affirm the same chain containing a particular same block, this block and all its predecessors are finalised.

In Polkadot, the council and technical committee can be considered the *benevolent dictator*, who have the rights to trigger fast-tracked referenda, or cancel an improvement proposal or referendum via internal voting. Within Polkadot, the *accountability tracer* is realised via participants' on-chain addresses, and a common penalty is to destroy the staked tokens of malicious participants. Finally, validators can vote to activate the *network freezer* of a parachain to suspend its validation system, and the recovery can be decided via either a validator-voting or referendum.

5.2 Architecture mapping of Quorum

Blockchain application providers can deploy Quorum blockchain systems via Blockchain as a service. Basically, Quorum blockchain systems can be considered permissioned Ethereum systems for enterprises or individuals. Consequently, *participation permission* is realised that the deployer (i.e., blockchain application provider) can directly send invitations to other participants, and only the entities with valid invitation code can join a particular Quorum blockchain system [48]. In terms of *validator selection*, Quorum supports Proof-of-Authority where the block validators are defined and managed by the deployer. Specifically, Quorum supports alternative consensus protocols, including QBFT, IBFT, Raft, and Clique [49]. It is noted that *block finality decider* may be different according to the employed consensus protocol. In particular, QBFT, IBFT, and Raft can achieve immediate finality when a new block is appended, whilst forks might occur when Clique is selected.

In addition to the deployer, Consensus (i.e., the project team of Quorum) is also regarded as the *benevolent dictator* in the circumstance that when the deployer or appointed administrator leaves the Quorum system without claiming a new administrator, Consensus can provide support based on the deployment agreement [48]. Quorum does not implement inherent token issuance or *incentive distributor*, since permissioned blockchain system stakeholders need to adhere to off-chain organisational hierarchy or business agreements where the incentives and decision rights are ascertained. However, the deployer can allocate Ether tokens in a Quorum system if needed [49].

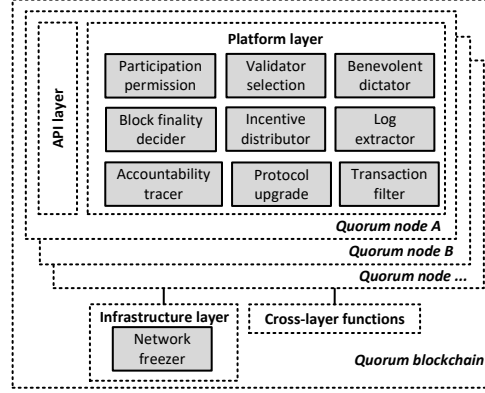


Figure 4: Architecture mapping of Quorum.

In a Quorum system, all activities are recorded and can be used for further analysis and audit via *log extractor* [49]. The *accountability tracer* is enabled by the generated blockchain account and a public and private key pair [48]. Considering the above-mentioned *participation permission*, accountability in Quorum can be extended to stakeholders’ real-world identity. *Protocol upgrade* in a Quorum blockchain system requires the approval of all participants, then updating the configuration file and restarting the nodes [49, 50]. Quorum realises *network freezer* by managing the node and account permissioning [51], suspending the operation of all accounts can freeze a Quorum system. Further, Quorum can specify the types of transactions that a blockchain account is permitted to send [51]. *Transaction filter* is deployed to examine the transaction type.

5.3 Discussion

From the evaluation results, it can be found that our reference architecture is correct and usable, as two blockchain system architectures can be mapped on the proposed reference architecture. Comparing Polkadot and Quorum, it is observed that permissionless blockchain systems may implement additional components for voting. The decision rights are allocated to all stakeholders to engage their participation, and increase their trust to the system upgrades, since the results are finalised according to their own choices. Relatively, permissioned blockchain systems like Quorum would centralise the decision rights to the system deployer/administrator, and highlight the permissioning of stakeholders to replicate the real-world positions.

We noticed that several patterns are applied in the off-chain environment, instead of blockchain architectural components. For instance, Polkadot provides a *scam list* introducing several common types of scams to raise stakeholders’ awareness [52]. Quorum posts an official press release claiming that Consensus acquired Quorum from J.P. Morgan [53], which can be regarded as an off-chain *social contract*. Besides, since Quorum deploys Ethereum blockchain instances, *contract freezer* is supported by the internal Ethereum virtual machine by default [37]. In addition, we observe more novel governance mechanisms in the evaluation process. For example, Polkadot proposes a concept of “*adaptive turnout biasing*”, where the threshold in a voting process is related to the turnout rate [41]. Finally, we remark that the proposed reference architecture is adaptive, so that future research can explore more governance patterns and integrate them into this reference architecture.

6 Conclusion

Governance is a significant factor throughout the lifecycle of a blockchain system, to ensure normal operation and continuous evolution. Nevertheless, it is found that most existing studies only provide high-level guidelines, while there is a lack of consideration from the respective of architecture design. In this article, we presented a reference architecture to help architects operationalise governance approaches in the future design and development of governance-driven blockchain systems. Specifically, we adopt a widely-accepted blockchain reference model, and apply a set of architectural patterns for governance to the components. We explain the responsibility of each component, and evaluate the correctness and utility of our proposed reference architecture via mapping two existing blockchain system architectures. In future work, we plan to develop decision models for the governance-driven blockchain system design.

References

- [1] F. Tschorsch and B. Scheuermann, “Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, p. 464, 2016.
- [2] A. Bratanova *et al.*, “Blockchain 2030: A look at the future of blockchain in Australia,” Data61, CSIRO, Brisbane, Australia, Tech. Rep., Apr. 2019. [Online]. Available: https://www.researchgate.net/publication/332298704_Blockchain_2030_A_Look_at_the_Future_of_Blockchain_in_Australia
- [3] N. Z. Aitzhan and D. Svetinovic, “Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams,” *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 840–852, 2018.
- [4] D. Miller, “Blockchain and the internet of things in the industrial sector,” *IT Professional*, vol. 20, no. 3, pp. 15–18, 2018.
- [5] N. Atzei, M. Bartoletti, and T. Cimoli, “A survey of attacks on ethereum smart contracts (sok),” in *Principles of Security and Trust*, M. Maffei and M. Ryan, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, pp. 164–186.
- [6] P. De Filippi and B. Loveluck, “The invisible politics of bitcoin: governance crisis of a decentralized infrastructure,” *Internet Policy Review*, vol. 5, no. 4, 2016.
- [7] Y. Liu, Q. Lu, L. Zhu, H.-Y. Paik, and M. Staples, “A systematic literature review on blockchain governance,” 2021. [Online]. Available: <https://arxiv.org/abs/2105.05460>
- [8] P. Weill and J. W. Ross, “IT governance on one page,” *Available at SSRN 664612*, 2004.
- [9] S. Cobit, “A business framework for the governance and management of enterprise IT,” *Rolling Meadows*, 2012.
- [10] C. Ballard, J. Baldwin, A. Baryudin, G. Brunell, C. Giardina, M. Haber, E. A. O’neill, S. Shah *et al.*, *IBM information governance solutions*. IBM Redbooks, 2014.
- [11] “Information technology – Governance of IT – Governance of data – Part 1: Application of ISO/IEC 38500 to the governance of data,” International Organization for Standardization, Standard ISO/IEC 38505-1:2017, 2017. [Online]. Available: <https://www.iso.org/standard/56639.html>
- [12] Z. Bao, K. Wang, and W. Zhang, “An auditable and secure model for permissioned blockchain,” in *Proceedings of the 2019 International Electronics Communication Conference*, ser. IECC ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 139–145. [Online]. Available: <https://doi.org/10.1145/3343147.3343170>
- [13] R. Beck, C. Müller-Bloch, and J. L. King, “Governance in the blockchain economy: A framework and research agenda,” *Journal of the Association for Information Systems*, vol. 19, no. 10, p. 1, 2018.
- [14] H. Nabilou, “Bitcoin governance as a decentralized financial market infrastructure,” *Stanford Journal of Blockchain Law & Policy*, vol. 4, p. 1, 2020.
- [15] P. Paech, “The governance of blockchain financial networks,” *The Modern Law Review*, vol. 80, no. 6, pp. 1073–1110, 2017.
- [16] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” <https://bitcoin.org/bitcoin.pdf>, 2008, accessed 26-May-2022.
- [17] S. Omohundro, “Cryptocurrencies, Smart Contracts, and Artificial Intelligence,” *AI Matters*, vol. 1, no. 2, pp. 19–21, Dec. 2014. [Online]. Available: <http://doi.acm.org/10.1145/2685328.2685334>
- [18] R. van Pelt, S. Jansen, D. Baars, and S. Overbeek, “Defining blockchain governance: A framework for analysis and comparison,” *Information Systems Management*, vol. 38, no. 1, pp. 21–41, 2021.
- [19] D. Hofman, Q. DuPont, A. Walch, and I. Beschastnikh, “Blockchain Governance: De Facto (x) or Designed?” in *Building Decentralized Trust*. Springer, 2021, pp. 21–33.
- [20] Y. Liu, Q. Lu, G. Yu, H.-Y. Paik, and L. Zhu, “Defining blockchain governance principles: A comprehensive framework,” *Information Systems*, vol. 109, p. 102090, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0306437922000758>
- [21] P. F. Katina, C. B. Keating, J. A. Sisti, and A. V. Gheorghe, “Blockchain governance,” *International Journal of Critical Infrastructures*, vol. 15, no. 2, pp. 121–135, 2019.
- [22] D. W. Allen and C. Berg, “Blockchain Governance: What we can Learn from the Economics of Corporate Governance,” *The Journal of The British Blockchain Association*, p. 12455, 2020.
- [23] T. John and M. Pam, “Complex adaptive blockchain governance,” in *MATEC Web of Conferences*, vol. 223. EDP Sciences, 2018, p. 01010.

- [24] L. Bass, P. Clements, and R. Kazman, *Software architecture in practice*. Addison-Wesley Professional, 2003.
- [25] E. Y. Nakagawa, P. Oliveira Antonino, and M. Becker, “Reference architecture and product line architecture: A subtle but critical difference,” in *Software Architecture*, I. Crnkovic, V. Gruhn, and M. Book, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 207–211.
- [26] Y. Yuan and F.-Y. Wang, “Blockchain and cryptocurrencies: Model, techniques, and applications,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 9, pp. 1421–1428, 2018.
- [27] A. Ellervee, R. Matulevicius, and N. Mayer, “A comprehensive reference model for blockchain-based distributed ledger technology,” in *ER Forum/Demos*, 2017, pp. 306–319.
- [28] I. Homoliak, S. Venugopalan, D. Reijsbergen, Q. Hum, R. Schumi, and P. Szalachowski, “The security reference architecture for blockchains: Toward a standardized model for studying vulnerabilities, threats, and defenses,” *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 341–390, 2021.
- [29] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim, “A survey on consensus mechanisms and mining strategy management in blockchain networks,” *IEEE Access*, vol. 7, pp. 22 328–22 370, 2019.
- [30] Y. Gong, S. van Engelenburg, and M. Janssen, “A reference architecture for blockchain-based crowdsourcing platforms,” *Journal of Theoretical and Applied Electronic Commerce Research*, vol. 16, no. 4, pp. 937–958, 2021. [Online]. Available: <https://www.mdpi.com/0718-1876/16/4/53>
- [31] G. Leeming, J. Cunningham, and J. Ainsworth, “A ledger of me: Personalizing healthcare using blockchain technology,” *Frontiers in Medicine*, vol. 6, 2019. [Online]. Available: <https://www.frontiersin.org/articles/10.3389/fmed.2019.00171>
- [32] A. Alketbi, Q. Nasir, and M. Abu Talib, “Novel blockchain reference model for government services: Dubai government case study,” *International Journal of System Assurance Engineering and Management*, vol. 11, no. 6, pp. 1170–1191, 2020.
- [33] M. Galster and P. Avgeriou, “Empirically-grounded reference architectures: A proposal,” in *Proceedings of the Joint ACM SIGSOFT Conference – QoSA and ACM SIGSOFT Symposium – ISARCS on Quality of Software Architectures – QoSA and Architecting Critical Systems – ISARCS*, ser. QoSA-ISARCS ’11. New York, NY, USA: Association for Computing Machinery, 2011, p. 153–158. [Online]. Available: <https://doi.org/10.1145/2000259.2000285>
- [34] S. O’mahony and F. Ferraro, “The emergence of governance in an open source community,” *Academy of Management Journal*, vol. 50, no. 5, pp. 1079–1106, 2007.
- [35] P. B. De Laat, “Governance of open source software: state of the art,” *Journal of Management & Governance*, vol. 11, no. 2, pp. 165–177, 2007.
- [36] A. Tiwana, B. Konsynski, and A. Bush, “Platform evolution: coevolution of platform architecture, governance, and environmental dynamics (research commentary),” *Information Systems Research*, vol. 21, no. 4, pp. 675–687, 2010.
- [37] Y. Liu, Q. Lu, G. Yu, H.-Y. Paik, H. Perera, and L. Zhu, “A pattern language for blockchain governance,” 2022. [Online]. Available: <https://arxiv.org/abs/2203.00268>
- [38] “Blockchain and distributed ledger technologies — reference architecture,” International Organization for Standardization, Standard ISO 23257:2022, 2022. [Online]. Available: <https://www.iso.org/standard/75093.html>
- [39] H. D. Bandara, X. Xu, and I. Weber, “Patterns for blockchain data migration,” in *Proceedings of the European Conference on Pattern Languages of Programs 2020*, ser. EuroPLoP ’20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: <https://doi.org/10.1145/3424771.3424796>
- [40] X. Xu, C. Pautasso, L. Zhu, Q. Lu, and I. Weber, “A pattern collection for blockchain-based applications,” in *23rd European Conf. on Pattern Languages of Programs*, 2018, pp. 1–20.
- [41] J. Burdges, A. Cevallos, P. Czaban, R. Habermeier, S. Hosseini, F. Lama, H. K. Alper, X. Luo, F. Shirazi, A. Stewart, and G. Wood, “Overview of Polkadot and its Design Considerations,” 2020. [Online]. Available: <https://arxiv.org/abs/2005.13456>
- [42] Polkadot, “Cross-consensus message format,” <https://wiki.polkadot.network/docs/learn-xcm>, 2022, accessed 19-October-2022.
- [43] G. Wood, “Polkadot: Vision for a heterogeneous multi-chain framework,” *White Paper*, vol. 21, pp. 2327–4662, 2016.
- [44] Polkadot, “Runtime upgrades,” <https://wiki.polkadot.network/docs/learn-runtime-upgrades>, 2022, accessed 19-October-2022.

- [45] —, “Transaction construction and signing,” <https://wiki.polkadot.network/docs/build-transaction-construction>, 2022, accessed 19-October-2022.
- [46] H. K. Alper, “Blind assignment for blockchain extension,” <https://research.web3.foundation/en/latest/polkadot/block-production/Babe.html#>, accessed 19-October-2022.
- [47] A. Stewart and E. Kokoris-Kogia, “GRANDPA: a Byzantine Finality Gadget,” 2020. [Online]. Available: <https://arxiv.org/abs/2007.01560>
- [48] Consensus, “Quorum blockchain service,” <https://consensus.net/docs/qbs/en/latest/>, accessed 19-October-2022.
- [49] —, “GoQuorum Enterprise Ethereum Client,” <https://consensus.net/docs/goquorum/en/latest/>, accessed 19-October-2022.
- [50] Hyperledger BESU, “Network and protocol upgrades,” <https://besu.hyperledger.org/en/stable/private-networks/how-to/upgrade/>, 2022, accessed 19-October-2022.
- [51] C. Nevile, G. Polzer, R. Coote, G. Noble, I. Bashir, C. Beyers, C. Cordi, A. Clarke, J. Ponnappalli, R. Saltini, P. Siemion, A. Toulmé, and T. Ben, “Enterprise Ethereum Alliance Permissioned Blockchains Specification,” <https://entethalliance.github.io/client-spec/chainspec.html>, 2022, accessed 19-October-2022.
- [52] Polkadot, “How to protect yourself from scams,” <https://wiki.polkadot.network/docs/learn-scams>, 2022, accessed 19-October-2022.
- [53] ConsenSys, “ConsenSys Acquires Quorum® Platform from J.P. Morgan,” <https://consensus.net/blog/press-release/consensus-acquires-quorum-platform-from-jp-morgan/>, 2020, accessed 19-October-2022.