Architectural Design Decisions for Blockchain-Based Applications

1st Maximilian Wöhrer
University of Vienna, Faculty of Computer Science
Research Group Software Architecture
Vienna, Austria
maximilian.woehrer@univie.ac.at

2nd Uwe Zdun
University of Vienna, Faculty of Computer Science
Research Group Software Architecture
Vienna, Austria
uwe.zdun@univie.ac.at

***Abstract*—Designing blockchain-based applications is a challenging task and requires a number of coordinated architecture decisions. To guide decision making in this regard, we systematically explore this architectural design space and possible solution strategies. More precisely, we provide architectural design decisions and decision options in terms of patterns and practices. Our research shows that most design decisions are influenced by the need to offset current blockchain drawbacks such as scalability, privacy, and usability by using centralized elements. This suggests that a hybrid architecture is beneficial in many design situations.**

***Index Terms*—blockchain, software architecture, decentralized application, DApp, smart contract, design pattern**

I. Introduction

Blockchains are distributed peer-to-peer systems which implement a trustless shared public append-only transaction ledger [1]. They are being recognized as a useful technology in a wide variety of business applications to increase operational efficiency and enable new business models. However, considering the degree of maturity of blockchain technologies in practical applications, the acceptance and adoption of the technology is still in an early stage. Accordingly, there is currently a lack of a systematic and holistic approach to system design of blockchain-based applications [2][3]. While there is academic literature addressing the applicability [4], selection [5], and configuration of blockchains [6], there is currently limited work on architectural design decisions covering the implementation and integration of blockchain-based solutions. Betzwieser *et al.* [7] provide a decision model for the implementation of blockchain solutions. Their work highlights prerequisites, business and technical considerations, as well as design decisions, but integration aspects are only sparsely addressed. Wessling *et al.* [8] propose blockchain tactics as a means to support the process of integrating decentralized elements. However, the authors focus on lower level design patterns and do not provide architectural guidance. Blum *et al.* [9] adopt the former tactics idea and propose a design approach using existing architectural concepts such as strategies, tactics and design patterns. While relationships between these concepts are outlined, a discussion of the concepts themselves is lacking.

978-0-7381-1420-0/21/$31.00 ©2021 IEEE

So far none of these works provide systematic architecture guidance in terms of design decisions. To close this gap, we investigate architectural design options for blockchain-based software solutions by gathering data from different sources and applying Grounded Theory (GT) techniques to extract and identify common practices.

In order to concretize the research objectives, we ask the following research questions: *RQ1)* What are the key architectural design decisions for blockchain-based applications? *RQ2)* What are possible design options regarding these decisions and the associated (best) practices? This paper focuses on public permissionless blockchains and refers for illustrative purposes to the Ethereum blockchain, today's most popular ecosystem. Please note that the presented concepts are independent from a particular blockchain implementation.

The paper is structured as follows: First, we elaborate architectural design decisions and decision options of blockchain-based solutions as main contribution in Section II. Then, we discuss our findings and draw conclusions in Section III.

II. Blockchain Oriented Software Engineering

Blockchain-Oriented Software Engineering (BOSE) is a growing discipline focused on defining and applying software engineering principles for blockchain-based system design, development, and deployment. Activities in this context still represent a challenging endeavor, and the degree to which blockchain is used today is significantly influenced by characteristics such as performance, usability, and user experience. A well designed architecture helps to better align and meet desired requirements including the above criteria. To this end, this section discusses design guidance for blockchain integration that we found and coded in our study. For better illustration, we begin with a feature model that provides a general outline, and then discuss individual design issues.

A. Feature Model

An overview of elaborated architectural decisions as well as decision options is given in a feature model representation in Figure 1. The feature model models possible relationships (or/alternative) between design options (modeled as features) via affiliated tags (mandatory/optional) representing various design aspects or concerns. In addition, drawn relationships

(requires) indicate necessary design decisions to achieve a fully decentralized architecture. Note that the design options have been also evaluated for their positive impact on privacy, usability, and scalability, where applicable.

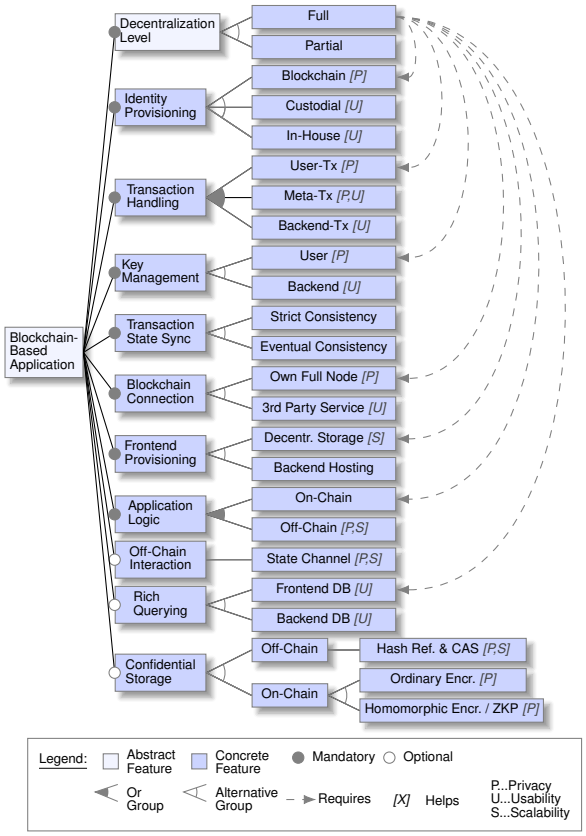

Fig. 1. Feature model for a blockchain-based application.

B. Architectural Design Decisions and Decision Options

Software architects are faced with a number of design issues for which there are usually one or more alternative solutions (design options). The following is a discussion of some design issues that arise in the course of developing a blockchain-based solution. We focus on operational and integration aspects rather than matters around the blockchain technology itself (e.g., blockchain type, consesus mechanism, blockchain configuration, platform selection), as this has already been investigated in previous work (see Xu *et al.* [6]).

1) Decentralization Level: One can use blockchain as a stand-alone platform capable of implementing a complete application logic (on top of smart contracts) or as an auxiliary tool in enterprise solutions to meaningfully complement business aspects (e.g., auditable history, asset tracking, etc.). These are the major decentralization styles discussed below.

a) Fully Decentralized Applications: A decentralized application (DApp) is a software solution built on top of a distributed peer-to-peer network. A DApp typically consists of a Web frontend that makes direct calls to a decentralized backend infrastructure (i.e., the blockchain executing smart contracts incorporating the entire application logic). This structure is similar to a two-tier client-server architecture, with no intermediate support required for operation. Benefits of DApps include an increased trust level and resistance to censorship, as the execution is not relying on a central provider which makes computation more transparent and further lowers the risk for a single point of failure. The disadvantages include low transaction throughput, high response times, difficult updating, incurring transaction costs (to be paid by the user), fluctuating transaction costs, and in general an immature technology stack accompanied by a vendor lock-in. For a comprehensive empirical study of blockchain-based DApps we refer to [10] and for an intra-architectural performance comparison to [11].

b) Hybrid (Semi-Decentralized) Applications: Building fully decentralized applications is a difficult undertaking. DApps based solely on distributed components quickly reach their limits due to current technical limitations and usability challenges. As a result, the current approach in building such applications is more nuanced. Instead of relying exclusively on decentralized components, often a hybrid architecture is realized and centralized components are added where appropriate. In this context, a traditional backend is still relevant and several reasons speak for its use, although it lowers the trust compared to purely decentralized applications.

2) Identity Provisioning: Blockchain users have a decentralized identity based on asymmetric encryption, also called public key cryptography. Here, the identity is represented by a pair of keys. The public key (in a shorter representation) serves as the account identifier (address) and is derived from the private key that grants ownership of that account. An important decision is whether this decentralized identity concept is appropriate for an application scenario, or whether a typical password-protected centralized account is preferred, where blockchain operations take place under a designated application account. A middle ground in the form of custodial identity management is also possible, i.e., the blockchain identity is managed for users within an application and linked to a password-protected centralized account. Overall, this decision must also be made with regard to transaction handling and key management, which are explained in more detail below.

3) Transaction Handling: Transactions provide the means to interact with a blockchain. Essentially, a transaction is a cryptographically signed instruction that is generated by an account, serialized and then transmitted to the blockchain for processing. There are three options how transactions can be initiated [10][12] which are discussed below.

a) User Signed Transaction: The traditional way to initiate a transaction is from the user. In this case, the user interacts directly with a smart contract by signing a transaction with his private key and then sending it to the blockchain network along with a payment to cover the execution costs. This procedure allows the user the highest sovereignty of his identity, but requires that he has software that supports client-side interactions (e.g. a wallet) and tokens to pay for the transaction.

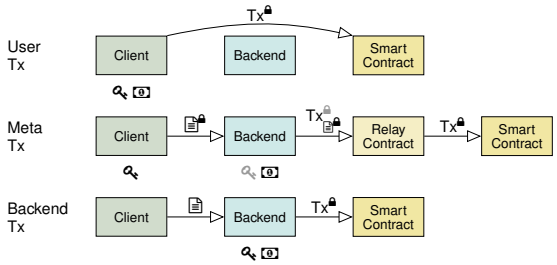

Fig. 2. An overview of transaction handling options.

b) Meta-Transaction: Transaction costs and the acquisition of tokens to pay for them are a major hindrance to a mainstream adoption of DApps. Meta-transactions aim to solve this onboarding issue so that first time users can execute decentralized transactions without a wallet. The simple idea is that a third party sends another user's transaction and pays for the execution. In this arrangement, the user signs a message containing information about a transaction the user wants to execute. This message is then sent free of charge to the off-chain third party, who subsequently wraps this information in a transaction and sends it to the blockchain network. This transaction is usually sent to an intermediate contract that verifies the user's signature of the attached transaction payload, before forwarding a subsequent transaction that executes a user intended method on the target contract. The advantage of this method is that the user is in control of his private key and does not need to bother with transaction fees. On the downside, the total costs are higher due to the additional costs for the relay contract and the fact that more transactions are required. Another issues is that the third party is centralized and could turn rogue, censoring transactions.

c) Backend Signed Transaction: Another approach to solving problems related to the accessibility of decentralized applications is to take the entire transaction signing and payment process away from the user and handle these matters in the backend. Although this method offers a high degree of comfort for the user and is relatively easy to implement, it destroys the fundamental concept of sovereignty and decentralized trusted execution, which is a basic principle of blockchains. Another downside is lacking transparency as the application may take unknown or unauthorized actions on behalf of the user and there is no way to challenge or reverse misaligned transactions. Further, since users do not own private keys, they cannot own tokens or perform operations directly with other smart contracts. The only way to achieve this is to manage user keys in the backend, which requires a sophisticated security concept to avoid any attacks.

4) Key Management: There are various ways to manage and store private keys, and the solutions are always caught between convenience and security, as the two are difficult to reconcile. Approaches can be broadly categorized whether key management is left to the user or to a backend respectively an application. In addition, a distinction is often made between hot or cold storage, which refers to whether or not a key management solution has network (Internet) connectivity. Users typically use wallets for key management, which can be either a hardware device, a physical medium, a program, or service that stores a user's private and public keys. In addition, wallets may provide the functionality of encrypting, signing, and forwarding information (transactions) to the blockchain. For the backend, there are several complex strategies and different software solutions that allow private keys to be stored quite securely. Some solutions are based on geographically distributed databases, while others are built on specially designed hardware. A service-based approach in the form of a key management system (KMS) or key vault is used to abstract the management, control, audit, and execution of certain actions involving secret keys. This allows for example a transaction manager (e.g. EthSigner) to sign transactions via a key vault (e.g., HashiCorp Vault, Azure Key Vault) without exposing secret keys.

5) Transaction State Synchronization: Blockchain has an asynchronous character. This is due to the latency in the execution and confirmation of transactions. An application that relies on blockchain transactions can deal with this aspect in two ways. Either the application flow is halted until transaction finality is reached (synchronous), or the application flow continues without waiting for transaction finality (asynchronous). The former implies strict consistency for transactions, which facilitates state management between the application and the blockchain, but comes at the cost of lengthy wait times. The latter assumes an eventual consistency for transactions, whereby the application expects that any blockchain transaction waiting on, will eventually confirm and continues on as usual. This approach leaves the application in a state which is ahead of the blockchain, allowing for example an improved (more immediate) user experience. However, having two instances of state (i.e. blockchain and application) can be problematic if state management is not handled carefully including rollback scenarios; namely, in case a transaction fails or confirmation takes longer than expected.

6) Blockchain Connection: In order to connect to the blockchain a blockchain endpoint is needed. A blockchain endpoint is a device or data point running a piece of software that implements the blockchain protocol to participate in the blockchain network. One can either run their own blockchain endpoint or rely on a service provider (e.g., Infura, QuikNode), which run node clusters to allow users to interact with the blockchain without setting up their own node. While the latter is a hassle-free option used by many, the performance is not on par with a sovereign private node, which is a necessity if the blockchain is to be used in a truly private, self-sufficient, and trustless manner.

7) Frontend Provisioning: The frontend code of a blockchain-based application is no different from that of a traditional web application and can therefore be written in any language. Specially tailored frontend libraries that fusion blockchain node interaction with popular JavaScript frontend technologies (e.g., React, Vue) exist to make writing DApp user interfaces easier (e.g. Drizzle). Regarding hosting, the

frontend code can be hosted either on a dedicated backend or a decentralized storage like IPFS for which several frameworks are available to build DApps (e.g., Dappkit, Fission, Fleek, or Textile). The former is more in line with today's web development practices while the latter achieves complete decentralization of the application , but can be more cumbersome to setup also with domain name integration.

As a side note, there are some peculiarities in terms of UI and UX design for blockchain-based applications. These include the presentation of hashes and keys that seem "strange" to users, the lack of an "undo" functionality due to irreversible blockchain transactions, and dealing with prolonged interaction delays due to the asynchronous nature of blockchain.

8) Application Logic: The logic required to achieve business goals can be executed either on or off-chain, whereby a fundamental design philosophy is to use blockchains sparingly because they are slow and expensive. The difference between what is processed on-chain versus off-chain depends largely on the level of trust and performance required. On-chain processing is designed to be trustless, meaning it is suited when the goal is to perform actions independently and verifiably in the absence of trust between parties. Off-chain processing is suitable for cases where no immutable (trans)actions need to be independently validated and authenticated (i.e., the parties trust each other), complex computations exceed the blockgas limit, or a scheme exists to verify off-chain processing results on-chain. Overall, off-chain transactions may bring lower fees, instant settlement, and greater anonymity, but they lack the trust level that on-chain transactions establish.

9) Off-Chain Interaction: As mentioned above, interactions can be taken off-chain. Instead of using the blockchain as the primary processing layer, the key idea is to use it as a settlement layer. State updates occur outside and are only propagated to the blockchain when necessary (e.g., on dispute or for final settlement). This approach, also known as a state channel, allows for faster transaction flow and increased privacy as participants interact directly. However, there are still some issues, such as the state channel transparency, transaction traceability, and the inability to transfer off-chain state back to the blockchain on an ad-hoc basis [13].

10) Rich Querying: If data is stored on the blockchain, it is likely that it will also need to be queried, but querying blockchain data directly is ineffective and there is no built-in query language. A common solution to this problem is to create a local replication of relevant on-chain events and data in a backend that supports caching and indexing to enable search, filter, sort, and pagination functions. Alternatively, if a decentralized solution without a dedicated backend is desired, a frontend database can be used. In this case, a browser database in JavaScript (e.g., PouchDB, GunDB) synchronizes all relevant on-chain events or data, but this approach is not suitable for applications with a high data or event load.

11) Confidential Storage: There are various approaches to storing confidential data on the blockchain. First, a distinction can be made between storing data on-chain or off-chain. For on-chain storage, the most obvious solution is to use ordinary encryption and then share the decryption keys over another secure channel. For off-chain storage, a common approach is to keep the raw data in external storage and only store the respective hashes on-chain. Here, the external storage allows a more controlled management of confidential data and access privileges and may also serve as an exchange channel when a shared storage is used. Its implementation can take many forms depending on the type of data, such as a database (e.g., SQL, NoSQL) or a decentralized content addressable storage (CAS) (e.g., IPFS, Swarm). In addition to the above, there are also more advanced techniques to ensure data confidentiality. These include zero-knowledge proof (ZKP), which allows users to prove their knowledge of a value without revealing the value itself, and homomorphic encryption (HE), in which data is encrypted before being shared on the blockchain in such a way that it can be analyzed without decryption. These techniques are promising, but the approaches mentioned earlier may be better equipped to provide value today.

III. Conclusion

Blockchain is considered a disruptive technology that enables new business models and technological solutions. Consequently, new types of architectures and designs are required to utilize the technology at its best, while addressing currently associated inefficiencies. To this end, we studied architectural design solutions from which we inferred architectural design decisions along with decision options (patterns and practices). Our research highlights various important design issues that arise in the course of developing a blockchain-based solution and points to possible resolution strategies. Thus, our findings can serve as a guideline for practitioners to asses and design potential blockchain implementations.

A typical software architecture design requires various trade-off decisions to balance desired quality attributes. In the case of blockchain integration, this boils down to striking a balance between decentralization and scalability, privacy, and usability. It can be said that the more decentralized a solution is, the more difficult it is to ensure the above quality attributes. To tackle this challenge, a hybrid architecture approach currently offers a good compromise. Decentralized and centralized components are combined, allowing the advantages of both to be used. In this light, many of the presented design options can be understood as a way to circumvent current blockchain disadvantages by using centralized elements. Thus, through cleverly tuned design decisions, it is possible to enable otherwise infeasible tasks, such as performing complex computations, storing large amounts of data, keeping data private, or querying blockchain-related data.

In the future, ongoing developments in the area of blockchains could lead to blockchains becoming more powerful and mainstream. Architectures embedding the technology will likely evolve and provide a promising foundation for diverse applications. In this context, future research could investigate generally applicable architectural patterns as well as migration patterns to transfer existing functionality or architectural components to blockchain technology.

REFERENCES

[1] S. Tai, J. Eberhardt, and M. Klems, "Not ACID, not BASE, but SALT: A transaction processing perspective on blockchains," in *CLOSER 2017 - Proceedings of the 7th International Conference on Cloud Computing and Services Science*, 2017.

[2] S. Porru, A. Pinna, M. Marchesi, and R. Tonelli, "Blockchain-oriented software engineering: Challenges and new directions," *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering Companion, ICSE-C 2017*, no. February, pp. 169–171, 2017.

[3] G. Wood, "Ethereum: a secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, 2014.

[4] B. A. Scriber, "A Framework for Determining Blockchain Applicability," *IEEE Software*, vol. 35, no. 4, pp. 70–77, 2018.

[5] S. Farshidi, S. Jansen, S. Espana, and J. Verkleij, "Decision Support for Blockchain Platform Selection: Three Industry Case Studies," *IEEE Transactions on Engineering Management*, vol. PP, pp. 1–20, 2020.

[6] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba, "A Taxonomy of Blockchain-Based Systems for Architecture Design," *Proceedings - 2017 IEEE International Conference on Software Architecture, ICSA 2017*, pp. 243–252, 2017.

[7] B. Betzwieser, S. Franzbonenkamp, T. Riasanow, M. Böhm, H. Kienegger, and H. Krcmar, "A decision model for the implementation of blockchain solutions," *25th Americas Conference on Information Systems, AMCIS 2019*, no. Dm, pp. 1–10, 2019.

[8] F. Wessling, C. Ehmke, O. Meyer, and V. Gruhn, "Towards Blockchain Tactics: Building Hybrid Decentralized Software Architectures," *Proceedings - 2019 IEEE International Conference on Software Architecture - Companion, ICSA-C 2019*, pp. 234–237, 2019.

[9] F. Blum, B. Severin, M. Hettmer, P. Huckinghaus, and V. Gruhn, "Building Hybrid DApps using Blockchain Tactics -The Meta-Transaction Example," *IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2020*, 2020.

[10] K. Wu, Y. Ma, G. Huang, and X. Liu, "A first look at blockchain-based decentralized applications," *Software: Practice and Experience*, no. April, pp. 1–18, 2019.

[11] K. M. Kina-Kina, H. E. Cutipa-Arias, and P. Shiguihara-Juarez, "A comparison of performance between fully and partially decentralized applications," in *Proceedings of the 2019 IEEE 26th International Conference on Electronics, Electrical Engineering and Computing, INTERCON 2019*, 2019.

[12] F. Wessling and V. Gruhn, "Engineering Software Architectures of Blockchain-Oriented Applications," *Proceedings - 2018 IEEE 15th International Conference on Software Architecture Companion, ICSA-C 2018*, pp. 45–46, 2018.

[13] B. Podgorelec, M. Herieko, and M. Turkanovic, "State Channel as a Service Based on a Distributed and Decentralized Web," *IEEE Access*, vol. 8, pp. 64 678–64 691, 2020.