Towards a Reference Architecture for Blockchain-based Document Management Systems

1st Hauke Precht
Department of Computing Science / VLBA
Carl von Ossietzky Universität Oldenburg
Oldenburg, Germany
0000-0002-6308-5924

2nd Joschka Andreas Hüllmann
Behavioural, Management and Social sciences (BMS)
University of Twente
Enschede, The Netherlands
0000-0001-5704-8644

3rd Jorge Marx Gómez
Department of Computing Science / VLBA
Carl von Ossietzky Universität Oldenburg
Oldenburg, Germany
0000-0002-7833-7549

***Abstract*—Despite existing approaches such as Blockchain-oriented software engineering (BOSE), blockchain-based systems lack architectural rigor and traceable decisions. This paper aims to address this by proposing a reference architecture that emphasizes rigor and traceability. The domain of blockchain-based document management systems serves as an experimental environment providing a variety of materials. Based on the ProSA-RA method, architectural knowledge from an analysis of 113 publications in the domain of blockchain-based document management systems is derived. Following a structured requirements engineering process utilizing qualitative content analysis, the paper elicits 68 functional and 43 non-functional requirements. The requirements are modeled in SysML to investigate relationships between requirements while ensuring consistency. From the modeling, clusters emerged from which 24 reference architecture requirements are derived. The reference architecture requirements were defined using a grammar-based schema with embedded traceability links. In the ProSA-RA synthesis step, seven architectural design decisions are developed that address the derived reference architecture requirements. The design decisions are supported by a mapping to known blockchain-oriented patterns. Two core design decisions are a proxy-based multi-tier architecture, and a hybrid on-/off-chain storage strategy for document integrity and authenticity. This first iteration of a reference architecture makes key trade-offs such as *decentralization versus performance* explicit and justifiable, providing a method-driven foundation for disciplined blockchain system design.**

***Index Terms*—blockchain, reference architecture, requirements, design decisions, ProSA-RA.**

I. Introduction

Blockchain, as a Distributed Ledger Technology (DLT), is evolving into a mature technology that is increasingly deployed in enterprise applications. Concomitantly, the demand for rigorous enterprise integrations and well-designed architectures continues to intensify, as blockchain software developers emphasize the importance of rigorous development approaches [1]. The need for blockchain development approaches was addressed first through the emergence of *blockchain-oriented software engineering (BOSE)* [2]–[4]. Approaches such as *ABCDE* emphasize smart contract design and testability, with smart contracts acting as key elements in blockchain-based systems [5], [6]. Other studies discuss architectural patterns for blockchain-oriented applications [7]–[9], and the general role blockchain can take in software architectures [10]. To this end, "decisions on both the architectural and implementation level have to be justified" ([11, p. 237]).

Despite the blockchain's growth in enterprise deployments, recent studies show that research-based blockchain projects continue to lack rigorous justification and methodological foundations [12], with an in-depth discussion concerning architectural patterns and methods in the domain of blockchain-based document management systems [13].

To address these shortcomings, this paper proposes a *reference architecture* that supports blockchain developers in justifying their designs in a rigorous, traceable manner, addressing the domain of blockchain-based document management systems. The reference architecture is developed via the *ProSA-RA* method [14], emphasizing traceability and rigor. The method comprises four steps (Fig. 1). The reference architecture is derived from an analysis of existing blockchain-based document management systems, a literature stream offering a variety of research papers and materials [13]. From this literature stream, cross-domain knowledge and informed connections between requirements, patterns, and architectural design are extracted and synthesized. The proposed reference architecture can serve as a blueprint, including patterns and structures that can be used for particular architectures and applications [15]–[17].

This paper's contribution is twofold. First, it discusses the applicability of the ProSA-RA method in the blockchain domain, with particular focus on the traceability aspect in developing reference architecture requirements. Second, it presents a first iteration of a reference architecture for the domain of blockchain-based document management systems.

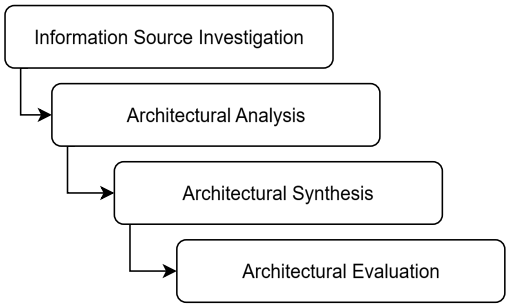

Fig. 1. Methodology for the Development of Reference Architecture, adapted from [14]

II. Information Source Investigation

In the first step, the base set of necessary information for the to-be-developed reference architecture is gathered and curated. The authors conducted two systematic literature reviews in prior studies that investigated existing systems and architectures in the domain. The first systematic literature review focuses on the state of the art in blockchain-based document management systems [13], which serves as the selected domain for the development of a reference architecture due to the availability of source materials. The second systematic literature review presents insights into the usage and development of reference architectures in the blockchain domain [18]. These two studies serve as the main input and foundation for the second step in ProSA-RA, the architectural analysis. Their key results are briefly outlined below.

Both systematic literature reviews follow the methodology outlined by [19], aiming for a comprehensive and exhaustive review [20], and use PRISMA guidelines [21] for communicating the process. In the first review, a set of 113 papers discussing blockchain-based document management systems was identified. The first author assigned descriptive codes to all papers, using verbatim statements or inferred meanings during full-text screening, as guided by [22]. In the second round, the first two authors reviewed the codes together and agreed on the final set to resolve disagreements and validate the codes [23], [24]. The key findings highlight that the majority of analyzed papers preferred a proxy-based approach (45%) over a pure dApp-based approach (38%), with a preference for off-chain storage (68%) [13]. This set of 113 papers forms the source material and serves as input for the following step 2 of ProSA-RA, architectural analysis.

In addition to the domain-specific information set, we extend the corpus with a set of design patterns and methods to aid in the development and decision-making of blockchain-oriented software. The design patterns and methods serve as preparation for the architectural synthesis, the third step of ProSA-RA. When it comes to design patterns, [25] acknowledge the potential impact of design patterns in smart contract development in their comparative analysis of smart contract languages. Their findings are underlined by [11], who show that known design patterns from object-oriented programming [26] can have a negative impact on blockchain software development projects, for example, higher transaction costs. As a response, new smart contract and blockchain specific patterns emerged and were analyzed. Specifically for smart contracts, the pattern categories security, data, structural, creational, and behavioral emerged [27]. Similar patterns have been derived from a systematic mapping study conducted by [28], which outlines eight patterns, including encryption for on-chain data, off-chain storage, tokenization, off-chain secrets, state channels, multiple smart contract authorizations, oracles, and contract registries. Another comprehensive review identified 36 design patterns and 48 best practices, focusing on security across the phases of design, development, and testing [29].

III. Architectural Analysis

The second step of the ProSA-RA method is the architectural analysis. The result of this step is a set of reference architecture requirements that the reference architecture must address. As input material, the curated information from step 1 is used. The following subsections outline the development towards the reference architecture requirements, with special emphasis on traceability and the justification of design decisions.

A. Functional and Non-functional Requirements

We conduct a bottom-up approach, i.e., we first analyze the existing architecture and its underlying functional and non-functional requirements. We follow established guidelines for requirement engineering [30], [31]. "Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior and to their evolution over time and across software families" ([32, p. 214]). Typically, the requirements engineering process consists of the condensed main phases *(1) requirements elicitation*, and *(2) requirements specification and validation* [31].

1) Requirements Elicitation: The curated information from step 1 was subjected to a qualitative content analysis to identify functional and non-functional requirements. Like the previous systematic literature review, the qualitative content analysis followed [24], [33] and used consensual coding [23] to reach agreement on codes. In the first round, the papers were analyzed for explicitly stated functional and non-functional requirements using the common clue words *must, could, should*. Only 29 papers explicitly mentioned functional or non-functional requirements; only one paper used the term *functional requirements*, while the other papers simply used the term requirement. This finding echoes the observations from previous research, criticizing a lack of rigor in justifying design decisions, and the failure to acknowledge existing methods and terminology [12], [13], [18]. In the second coding round, we looked for implicit requirements, i.e., those that can be derived from the given description of the systems and their environment. Based on both coding rounds, 68 functional and 43 non-functional requirements were elicited.

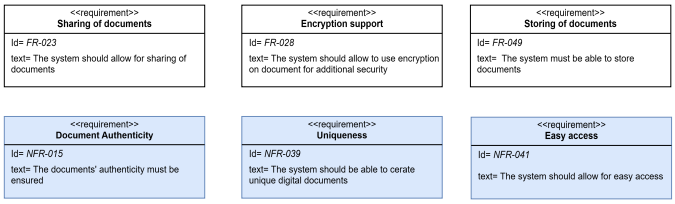

Fig. 2. Excerpt of Functional and Non-Functional Requirements in SysML Notation

2) Requirements Specification and Validation: Each requirement was subject to the quality criteria: necessity, understandability, unambiguity, coordination, and completeness [30]. The requirements are transformed into a graphical representation to analyze the interconnections among functional and non-functional requirements. To this end, SysML, an extension of UML that supports modeling requirements and their dependencies, was used. Each requirement is represented as a «`requirement`», a headline, and properties `ID` and `text`. The ID acts as a unique identifier for the respective requirement and is defined as FR-001 . . . FR-n for functional requirements and NFR-001 . . . FR-n for non-functional requirements. The text is a single string using the previously outlined structured natural language. An excerpt of the graphical modeling of requirements is presented in Fig. 2. We used an additional color coding to visually distinguish between functional (white) and non-functional (blue) requirements. Following up on this overview, we modeled the relationships among functional and non-functional requirements. In this process, we used the SysML-defined relationships «`trace`» and `refine`. This modeling enables us to identify the central functional and non-functional requirements based on their dependencies and position in the SysML diagram, validating their consistency, completeness, and validity [30]. Thus, the modeling process serves also as a review method [34]. In the functional requirements, the trace relationships indicate that storage and tamper resistance are central concerns. In the non-functional requirements, the trace relationships indicate that integrity and authenticity are the main concerns. These dependencies are visualized in Fig. 3. The modeling shows 'Data Integrity' as a necessary condition for authenticity and security.

B. Deriving Reference Architecture Requirements

Based on the elicited functional and non-functional requirements and the relationship modeling, we identified emerging clusters of similar and related requirements. These clusters serve as the starting point for deriving reference architecture requirements, which are aggregated from the previously identified functional and non-functional requirements. The reference architecture requirements operate at a higher level of abstraction [14]. We defined a grammar to describe the reference architecture requirements (Fig. 4).

The grammar includes a traceability property that identifies the underlying functional and non-functional requirement (or clusters) in the SysML diagram, from which the reference

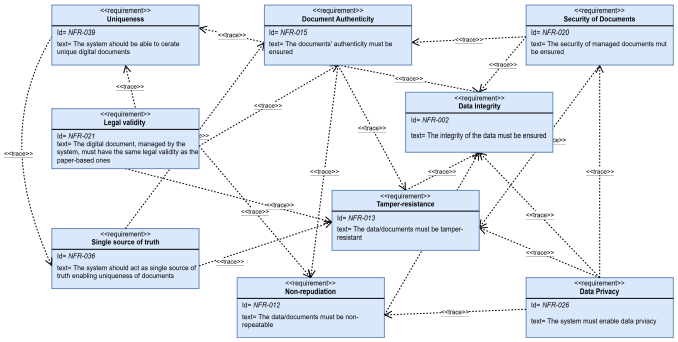

Fig. 3. Modeled Relationships between Non-functional Requirements SysML Notation

```
RAR ::= <architectural_concern>
        <architectural_intent>
        [<quality_attribute>]
        [<design_constraint>]
        <traceability>

<traceability> ::= derived_from { FR-id }+
                   [ derived_from { NFR-id }+ ]

Example:
RAR-03 ::=
  <architectural_concern> ::= Data Management
  <architectural_intent>  ::= Enable persistent
                              document storage
  <traceability>          ::= derived_from { FR-49 }
```

Fig. 4. Grammar-style schemas for reference architecture requirements with explicit traceability

architecture requirement is derived. The inclusion of the traceability property and the use of the trace relationships in the SysML requirement diagrams ensure the traceable and justified derivation and origin of reference architecture requirements. Fig. 5 shows an example in which the derivation for the reference architecture requirement 'Document Integrity and Authenticity' is shown. The reference architecture requirement was derived from the previously identified cluster shown in Fig. 3. For the sake of readability, we omitted the relationship modeling between the underlying non-functional requirements in Fig. 5. We added the color green to the modeling to highlight reference architecture requirements and distinguish them from functional and non-functional requirements. A total of 24 reference architecture requirements was derived. The complete list is presented in Tab. I with the respective ID and description for future referencing in the following step architecture synthesis. Reference architecture requirements that primarily address quality attributes of the architecture are prefixed with `RAR-Q`. For better readability, we stated the reference architecture requirements as a single sentence using structured natural language derived from the previously outlined grammar schema. An end-to-end example that highlights the identification and derivation of requirements, followed by design decisions, is presented in Fig. 6.

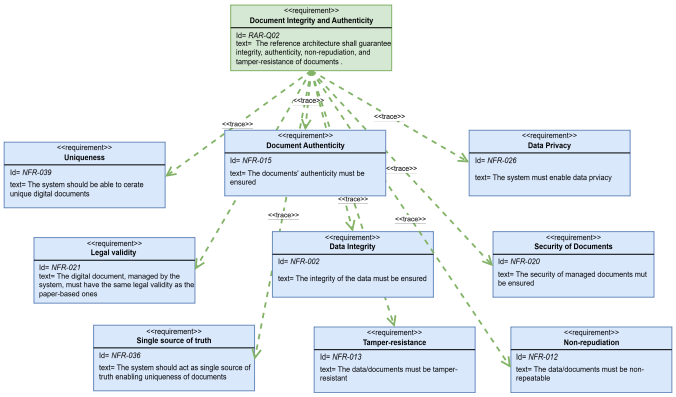

Fig. 5. Modeling of the reference architecture requirement 'Document Integrity and Authenticity

IV. Architecture Synthesis

In the third step of the ProSA-RA process, we map the identified reference architecture requirement to design decisions and supporting patterns identified in step 1 [14]. As a baseline for the design decisions serves the work of [36], in which a set of fundamental design decisions and their impact on blockchain-specific properties are presented. In total, we defined seven design decisions that address the derived reference architecture requirements. This is a many-to-many relationship, meaning that a design decision potentially addresses multiple reference architecture requirements and vice versa. The design decisions, supporting patterns, and affected modules, along with the reference architecture requirements, are presented in Tab. II.

In the following subsections, we discuss the first three design decisions, as they are particularly consequential for the derived reference architecture.

A. Design Decision 1 - Proxy-based multi-tier architecture

This design decision separates the user-facing front-end module and the blockchain module by introducing a proxy module. The proxy partially reintroduces a trusted intermediary, improves throughput while mitigating transaction costs, and additionally allows for transaction buffers and batch processing. Therefore, an architectural trade-off between performance and decentralization is made in favor of performance.

Another trade-off emerges between user experience and decentralization. By moving blockchain-specific complexity (e.g., wallet management, key management, or transaction signing) from the front-end (i.e., the user) to the proxy, we enhance the overall user experience and lower the adoption barrier.

This concurs with design decision 08. However, shifting responsibilities decreases the decentralization goals, affecting design decision 04.

The proxy module enables integration with existing systems via common APIs and connectors. Interoperability with other systems supports the identity and access management outlined in design decision 05.

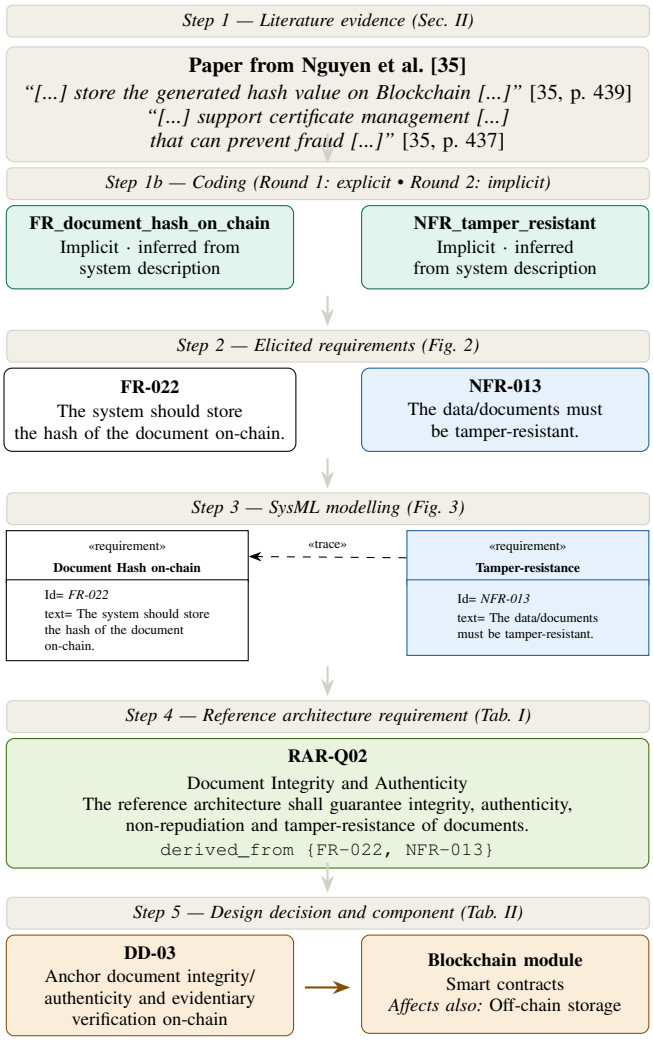

Fig. 6. Traceability example for RAR-Q02. From a literature source (Step 1), two codes are derived (Step 1b), one functional and one non-functional requirement are elicited based on the codes (Step 2), their relationship is modelled in SysML (Step 3), and RAR-Q02 is specified with explicit traceability links (Step 4), and mapped to a design decision and architectural component (Step 5). teal = Codings; white = FR; blue = NFR; green = RAR; amber = DD.

The proxy module, therefore, allows for an easier-to-audit and implement policy management, authentication, and role enforcement.

The consequences of these design decisions affect implementation, requiring blockchain-specific SDKs as dependencies. This dependency increases the architectural complexity but simplifies user interaction with the system while enabling fine-grained access control, for example, via role-based or attribute-based access control methods. Finally, the proxy provides a natural starting point for integrating additional systems, logics, or compliance features. In summary, design decision 01 opted for a compromise, favoring enterprise-grade usability and performance over pure decentralization.

TABLE I
COMPLETE SET OF DERIVED REFERENCE ARCHITECTURE REQUIREMENTS (RARS)

RAR ID	Reference Architecture Requirement
RAR-001	The reference architecture shall support databases, repositories, wallets, IPFS, sidechains, and hybrid on-/off-chain storage mechanisms.
RAR-002	The reference architecture shall support metadata management, indexing, compression, archiving, and versioning of documents.
RAR-003	The reference architecture shall provide capabilities for creation, upload, editing, publishing, retrieval, distribution, sharing, deletion, and archiving of documents.
RAR-004	The reference architecture shall enable workflows such as approval, request for signing, and collaborative editing.
RAR-005	The reference architecture shall support document authenticity via signatures, seals, watermarking, timestamps, and cryptographic proofs.
RAR-006	The reference architecture shall provide encryption, password protection, anonymisation, zero-knowledge proofs, and policy concealment mechanisms.
RAR-007	The reference architecture shall support authorization, registration, and revocation of entities and issuers.
RAR-008	The reference architecture shall provide verification through on-chain hash validation, QR codes, issuer verification, and dashboards.
RAR-009	The reference architecture shall enable revocation and transfer of documents or credentials, including ownership management.
RAR-010	The reference architecture shall provide document and information access via web UIs, dashboards, visualizations, printing, and notifications.
RAR-011	The reference architecture shall support OCR, search (full-text and hash-based), and multilingual translation.
RAR-012	The reference architecture shall enable hybrid blockchain integration patterns such as hash anchoring, IPFS/sidechain storage, and smart contract supervision.
RAR-013	The reference architecture shall provide programmable workflows and automation via smart contracts, including conditional issuance and auto-creation.
RAR-014	The reference architecture shall support tokenization mechanisms such as non-fungible tokens for uniquely identifying and transferring documents.
RAR-Q01	The reference architecture shall ensure compliance with regulations and support legal validity of digital documents.
RAR-Q02	The reference architecture shall guarantee integrity, authenticity, non-repudiation, and tamper-resistance of documents.
RAR-Q03	The reference architecture shall ensure confidentiality and privacy of data and users.
RAR-Q04	The reference architecture shall reduce reliance on trusted third parties, ensuring decentralization and governance.
RAR-Q05	The reference architecture shall ensure high availability, resilience, and robustness of services, avoiding single points of failure.
RAR-Q06	The reference architecture shall enable traceability of document changes, verifications, and auditability.
RAR-Q07	The reference architecture shall provide scalability, high throughput, concurrency, and cost/resource efficiency.
RAR-Q08	The reference architecture shall support interoperability, portability of deployments, and flexible handling of diverse document types.
RAR-Q09	The reference architecture shall define holistic security principles covering documents, users, and processes.
RAR-Q10	The reference architecture shall support process optimization, fraud prevention, and digital uniqueness of documents.

B. Design Decision 2 - Hybrid on-/off-chain document storage

The second design decision led to a hybrid storage mechanism in which document data is primarily stored off-chain, while only cryptographic proofs (hashes) are stored on-chain. Alternatives are full on-chain and full off-chain storage. A full-on-chain storage solution would prioritize maximum immutability and transparency, but at the cost of scalability and cost efficiency due to transaction fees and block sizes. Full off-chain solutions, on the other hand, weaken document authenticity and integrity by increasing reliance on centralized storage providers. Therefore, the chosen hybrid approach serves as a strategy to balance the trade-offs between integrity and scalability, costs and decentralization, and complexity and performance.

Regarding the trade-off between integrity and scalability, storing hashes on-chain preserves tamper evidence of related documents, while off-chain storage of the documents themselves scales. Regarding costs versus decentralization, off-chain storage of documents reduces potential costs but requires additional infrastructure beyond a blockchain. Lastly, the additional need for coordination between on-chain and off-chain components increases architectural complexity but improves performance and storage efficiency.

As a consequence of this hybrid approach, the reliance on oracles that facilitate interactions between on-chain (smart contract) and off-chain (storage) components increases. This dependency requires governance and trust concerning oracle operators, which is connected to design decision 01. Since the system must manage both blockchain modules and conventional storage infrastructure, the deployment and maintenance complexity increases. In conclusion, the hybrid approach for document storage in design decision 2 preserves the essential blockchain properties of integrity, authenticity, and auditability while providing cost efficiency, regulatory compliance, and scalability.

C. Design Decision 3 - Anchor document integrity/authenticity on-chain

Design decision 03 manifests the document integrity and authenticity on-chain and is therefore tightly coupled with design decision 02. Document hashes, related lifecycle events, and ownership information are recorded on the blockchain utilizing tokenization.

Recording lifecycle events and proofs on-chain enhances immutability and non-repudiation, but increases the number of

TABLE II
DESIGN-DECISION-CENTERED MAPPING FROM REFERENCE ARCHITECTURE REQUIREMENTS (RARS) TO PATTERNS AND AFFECTED MODULES (EXCERPTED RARS IN THE PAPER; FULL RAR SET FROM THE REQUIREMENTS DIAGRAM).

Design Decision	Addressed RARs	Patterns / Practices	Affected Modules
DD-01: Adopt a proxy-based multi-tier architecture (introduce an intermediary service layer for batching, buffering, policy enforcement, and integration; discourage direct dApp-to-ledger access)	RAR-010, RAR-Q01, RAR-Q05, RAR-Q07, RAR-Q09	Proxy pattern; transaction batching/buffering; integration gateway	Front-end module; Proxy module; (interfaces to) Storage module
DD-02: Use a hybrid on-/off-chain document storage strategy with pluggable backends (store minimal proofs on-chain; persist full documents and rich metadata off-chain; support multiple storage technologies)	RAR-001, RAR-002, RAR-011, RAR-012, RAR-Q06, RAR-Q08	Off-chain storage pattern; hash anchoring; adapter pattern for storage providers; oracle pattern (controlled on/off-chain interaction); content-addressable storage (e.g., IPFS)	Storage module (off-chain storage sub-module + Blockchain sub-module); Proxy module
DD-03: Anchor document integrity/authenticity and evidentiary verification on-chain (commit document proofs on-chain; enable verification and audit trails; prevent fraud and ensure uniqueness)	RAR-005, RAR-008, RAR-009, RAR-Q02, RAR-Q06, RAR-Q10	Tokenization (e.g., NFT/credential-as-token); embedded permission; signature/timestamp anchoring; revocation registry;	Blockchain module (smart contracts); Verification service / dashboards; (interfaces to) Off-chain storage
DD-04: Establish a decentralized trust and governance model (reduce reliance on trusted third parties; define participation, consensus, and governance rules)	RAR-Q04, RAR-Q05, RAR-Q07, RAR-Q08	Decentralization design (permissioned/permissionless, partial decentralization); consensus protocol selection; fault-tolerant deployment practices	Blockchain module (blockchain network / consensus component); governance processes
DD-05: Provide a dedicated identity and access management capability (register/revoke entities and issuers; enforce authorization consistently across proxy and contracts)	RAR-007, RAR-Q09, (supports RAR-Q03)	role-based access control/ attribute-based access control; on-chain identity/issuer registry; contract registry for trusted issuers; key management practices	Proxy module (identity management component); Blockchain module (smart contracts, authorization checks)
DD-06: Implement confidentiality and privacy controls as first-class mechanisms (protect document content, metadata, and user privacy while preserving verifiability)	RAR-006, RAR-Q03, RAR-Q09	Encryption; selective disclosure; anonymization/pseudonymization; policy concealment; secure key handling	Proxy module (identity management component); Storage module (off-chain encryption + access mediation)
DD-07: Encode document lifecycle workflows and automation via smart contracts (workflow states, approvals, signing requests, conditional issuance, automation)	RAR-004, RAR-013, RAR-014, (supports RAR-Q10)	Smart contract workflow/state machine patterns; event-driven architecture; contract modularization; automation rules	Blockchain module (workflow contracts); Proxy module (workflow orchestration); Front-end (workflow UI)

transactions and, therefore, potentially the costs. As a result, a trade-off between decentralization and costs is observed. Given that document signing and management are facilitated by the proxy module (design decision 01), trust in the proxy module is established. Alternatively, users could sign and submit transactions directly for documents, preserving decentralization but increasing complexity and user responsibilities.

Consequently, this design decision couples the blockchain module with the proxy module, especially for transaction submission and identity management, which are orchestrated at the proxy module level. This tight coupling enhances operational control and compliance by accepting a constraint in architectural flexibility.

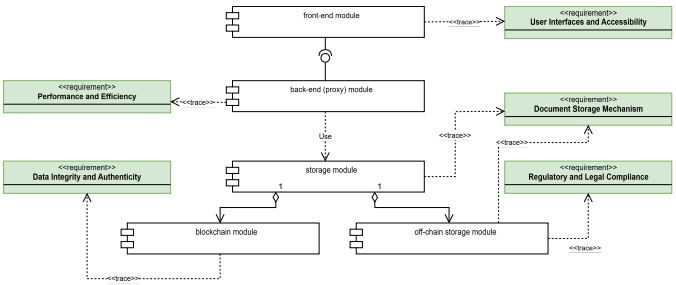

Fig. 7. Overview of the high-level reference architecture with modules traced back to reference architecture requirements

D. High-level Architectural Overview

Drawing from the design decisions presented in Table II and discussed in the prior subsections, we present in Fig. 7 the first iteration of a high-level overview of our reference architecture for blockchain-based document management systems, rigorously and transparently derived. This first overview utilizes a block diagram with SysML elements, addressing the logical view of the source code viewpoint [14]. We used the «`trace`» relationship in the architecture overview, enabling instant traceability between modules and reference architecture requirements. For example, the proxy module fulfills the reference architecture requirement for performance and efficiency.

In turn, the proxy module uses the storage module, which consists of a blockchain module and an off-chain storage module. The blockchain module addresses the reference architecture requirements for data integrity and authenticity. Similarly, the off-chain storage module addresses the reference architecture requirements for storage mechanisms and regulatory and legal compliance.

V. Preliminary Architectural Evaluation

While we do not exhaustively evaluate the architecture, we demonstrate how to reason about it using quality-attribute scenarios inspired by the Software Architecture Analysis Method (SAAM) [37]. As an example, we discuss the quality attributes of performance and scalability, along with the integrity and authenticity of documents, as both pose central concerns for the architecture.

1) Example Scenario: A Surge of Document Issuance: Below, we outline a scenario to reason about the performance and scalability of the system.

Scenario At the end of the final semester, an academic institution must issue 800 digital certificates within a short time window.

Affected quality attributes The scenario primarily addresses scalability and throughput (RAR-Q07), integrity and authenticity (RAR-Q02), and availability and robustness (RAR-Q05).

Architectural elements involved The scenario involves the front-end (interface for issuance), the proxy module, the off-chain storage module, and the blockchain module. In particular, design decisions 01, 02, and 03 are affected.

Scenario analysis The issuance surge is handled as a direct scenario by the reference architecture. The proxy module allows buffering and bundling of issuance requests in batches, reducing the number of required transactions and smoothing load peaks at the blockchain level. Full certificate data is stored off-chain while the cryptographic proofs (document hashes) are anchored on the blockchain. This provides tamper-evident certificates and verifiability without 800 dedicated on-chain storage operations. Identity and authorization checks ensure that only registered issuers with proper authorization can issue certificates in bulk.

Trade-offs The scenario underscores the trade-off between performance and decentralization discussed earlier. By shifting the proxy module's processing and transaction orchestration, we improve throughput and cost efficiency, but reintroduce a partially trusted intermediary.

This SAAM inspired reasoning style demonstrates that the reference architecture can manage surge issuance while making architectural trade-offs explicit.

VI. Conclusion, Limitations and Outlook: Architectural Evaluation and Instantiation

In this paper, we address the observed lack of architectural rigor and traceability in blockchain-based architectures by developing a reference architecture utilizing the ProSA-RA method in the domain of blockchain-based document management systems.

Building on two prior systematic literature reviews, we conducted an in-depth qualitative content analysis as part of the requirements engineering process in the ProSA-RA step architecture analysis. We identified 68 functional and 43 non-functional requirements, which were modeled in SysML to ensure consistency, traceability, and structured validation. Based on the modeling of relationships between requirements, clusters emerged. From these clusters, we derived 24 reference architecture requirements using a grammar-based specification scheme, with an emphasis on traceability and linking to their originating requirements. In the subsequent ProSA-RA step (architecture synthesis), we developed seven design decisions that address these reference architecture requirements, along with supporting patterns. This leads to the first iteration of a reference architecture for blockchain-based document management, in which the key elements (proxy-based multi-tier architecture, hybrid on- and off-chain storage, and on-chain anchoring of document integrity and authenticity) are rigorously and traceably derived.

We captured central trade-offs, including the blockchain trilemma, i.e., decentralization versus performance, integrity versus scalability, along with cost versus transparency. By accounting for these trade-offs and their explicit, traceable development, we contribute to a more disciplined, justifiable architectural design process in the blockchain domain.

Our limitations are threefold. First, in our current setup, we rely on academic literature to investigate requirements and design patterns. While this provides a solid foundation, industry viewpoints and practical experiences need to be integrated into future development. Second, the evaluation of the proposed reference architecture is based on a single scenario. To allow a detailed evaluation of key quality characteristics, we plan to conduct systematic architectural evaluation using established methods such as Architecture Tradeoff Analysis (ATAM) [38] in future work. At the same time, instantiation in at least two disjunct domains for blockchain-based document management systems (e.g., academic certificates and logistics) is needed to reason about the generalizability of the reference architecture. Third, the current reference architecture primarily addresses the logical view of the source code viewpoint. But the variability viewpoint as being specific for reference architectures that captures which architectural elements are fixed and which are variable across instantiations, needs to be addressed in future work. For example, the blockchain module exhibits variability in the choice between public and private blockchain systems, a decision that has cascading effects on the trust, governance, and performance trade-offs discussed in DD-04. Additional architectural viewpoints, for example, the crosscutting, runtime, or deployment viewpoint, need to be addressed [14], [39] as well. Through these next steps, the developed reference architecture can evolve into a robust, well-grounded foundation for future blockchain-based document management systems.

References

[1] S. Khalid and C. Brown, "Software Engineering Approaches Adopted By Blockchain Developers," in *2023 Tenth International Conference on Software Defined Systems (SDS)*. IEEE, Jan. 2023, pp. 1–6.

[2] S. Porru, A. Pinna, M. Marchesi, and R. Tonelli, "Blockchain-Oriented Software Engineering: Challenges and New Directions," in *2017 IEEE/ACM 39th International Conference on Software Engineering Companion: ICSE-C 2017 : 20-28 May 2017, Buenos Aires, Argentina : Proceedings*. Piscataway, NJ: IEEE, Jan. 2017, pp. 169–171.

[3] G. Destefanis, M. Marchesi, M. Ortu, R. Tonelli, A. Bracciali, and R. Hierons, "Smart contracts vulnerabilities: A call for blockchain software engineering?" in *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*. IEEE, 2018, pp. 19–25.

[4] Z. Z. Li, H. Wang, D. Gasevic, J. Yu, and J. K. Liu, "Enhancing Blockchain Adoption through Tailored Software Engineering: An Industrial-grounded Study in Education Credentialing," *Distributed Ledger Technologies: Research and Practice*, vol. 2, no. 4, pp. 1–24, Jan. 2023.

[5] L. Marchesi, M. Marchesi, and R. Tonelli, "ABCDE –agile block chain DApp engineering," *Blockchain: Research and Applications*, vol. 1, no. 1, p. 100002, Jan. 2020.

[6] N. Kannengieser, S. Lins, C. Sander, K. Winter, H. Frey, and A. Sunyaev, "Challenges and Common Solutions in Smart Contract Development," *IEEE Transactions on Software Engineering*, vol. 48, no. 11, pp. 4291–4318, Jan. 2022.

[7] F. Wessling and V. Gruhn, "Engineering Software Architectures of Blockchain-Oriented Applications," in *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*. IEEE, Apr. 2018, pp. 45–46.

[8] X. Xu, C. Pautasso, L. Zhu, Q. Lu, and I. Weber, "A Pattern Collection for Blockchain-based Applications," in *Proceedings of the 23rd European Conference on Pattern Languages of Programs*. New York, NY, USA: ACM, Jan. 2018, pp. 1–20.

[9] 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," in *2017 IEEE International Conference on Software Architecture (ICSA)*. Gothenburg, Sweden: IEEE, Apr. 2017, pp. 243–252.

[10] X. Xu, I. Weber, and M. Staples, *Architecture for Blockchain Applications*. Cham: Springer, Jan. 2019.

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

[12] H. Treiblmaier, "Toward More Rigorous Blockchain Research: Recommendations for Writing Blockchain Case Studies," in *Blockchain and Distributed Ledger Technology Use Cases*, ser. Progress in IS, H. Treiblmaier and T. Clohessy, Eds. Cham: Springer International Publishing, Jan. 2020, pp. 1–31.

[13] H. Precht, J. A. Hüllmann, and J. Marx Gómez, "Paperless Everything: A Systematic Literature Review for the Design of Blockchain-based Document Management Systems," *Distributed Ledger Technologies: Research and Practice*, vol. 5, no. 2, pp. 1–48, Jun. 2026.

[14] E. Y. Nakagawa, M. Guessi, J. C. Maldonado, D. Feitosa, and F. Oquendo, "Consolidating a Process for the Design, Representation, and Evaluation of Reference Architectures," in *2014 IEEE/IFIP Conference on Software Architecture*. IEEE, Jan. 2014, pp. 143–152.

[15] S. Angelov, P. Grefen, and D. Greefhorst, "A framework for analysis and design of software reference architectures," *Information and Software Technology*, vol. 54, no. 4, pp. 417–431, Jan. 2012.

[16] H. Cervantes and R. Kazman, *Designing Software Architectures: A Practical Approach*, 2nd ed., ser. SEI Series in Software Engineering. Hoboken, NJ: Addison-Wesley Professional, Jan. 2024.

[17] E. Y. Nakagawa and P. O. Antonino, "An Overview of Reference Architectures," in *Reference Architectures for Critical Domains*, E. Y. Nakagawa and P. Oliveira Antonino, Eds. Cham: Springer International Publishing, Jan. 2023, pp. 5–15.

[18] H. Precht and A. P. Calitz, "A systematic literature review of reference architectures in the blockchain domain," in *Progress in Information Systems*, ser. Progress in Information Systems. Cham, Switzerland: Springer, 2025, to appear. Selected papers from the International Conference on Research in Applied Information Systems (ICRAIS 2025), expanded and revised version.

[19] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature Reviews in Software Engineering," vol. 2, Jan. 2007.

[20] H. M. Cooper, "Organizing knowledge syntheses: A taxonomy of literature reviews," *Knowledge in Society*, vol. 1, no. 1, pp. 104–126, Jan. 1988.

[21] M. J. Page, J. E. McKenzie, P. M. Bossuyt, I. Boutron, T. C. Hoffmann, C. D. Mulrow, L. Shamseer, J. M. Tetzlaff, E. A. Akl, S. E. Brennan, R. Chou, J. Glanville, J. M. Grimshaw, A. Hróbjartsson, M. M. Lalu, T. Li, E. W. Loder, E. Mayo-Wilson, S. McDonald, L. A. McGuinness, L. A. Stewart, J. Thomas, A. C. Tricco, V. A. Welch, P. Whiting, and D. Moher, "The PRISMA 2020 statement: An updated guideline for reporting systematic reviews," *BMJ (Clinical research ed.)*, vol. 372, p. 71, Jan. 2021.

[22] J. Saldana, *The coding manual for qualitative reseachers*. SAGE Publications Ltd, 2009.

[23] U. Kuckartz, *Qualitative Text Analysis: A Guide to Methods, Practice & Using Software*. SAGE Publications Ltd, 2014.

[24] P. Mayring, "Qualitative Content Analysis: Theoretical Background and Procedures," in *Approaches to Qualitative Research in Mathematics Education*, ser. Advances in Mathematics Education, A. Bikner-Ahsbahs, C. Knipping, and N. Presmeg, Eds. Dordrecht: Springer Netherlands, Jan. 2015, pp. 365–380.

[25] M. Bartoletti and L. Pompianu, "An empirical analysis of smart contracts: Platforms, applications, and design patterns," *0302-9743*, vol. 10323, Jan. 2017.

[26] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, ser. Addison-Wesley Professional Computing Series. Reading, Mass: Addison-Wesley, 1995.

[27] V. Rajasekar, S. Sondhi, S. Saad, and S. Mohammed, "Emerging Design Patterns for Blockchain Applications," in *Proceedings of the 15th International Conference on Software Technologies*. SCITEPRESS - Science and Technology Publications, Jul. 2020, pp. 242–249.

[28] A. Carolina Ordonez-Guerrero, J. David Munoz-Garzon, E. Roberto Dulce Villarreal, A. Bandi, and J. Ariel Hurtado, "Blockchain Architectural Concerns: A Systematic Mapping Study," in *2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C)*. Honolulu, HI, USA: IEEE, Mar. 2022, pp. 183–192.

[29] L. Marchesi, L. Pompianu, and R. Tonelli, "Security checklists for Ethereum smart contract development: Patterns and best practices," *Blockchain: Research and Applications*, p. 100367, Aug. 2025.

[30] C. Rupp, *Requirements-Engineering und -Management: Das Handbuch für Anforderungen in jeder Situation*, 7th ed. München: Hanser, 2021.

[31] I. Sommerville, *Software Engineering*, 10th ed., ser. Informatik. Hallbergmoos: Pearson, 2018.

[32] P. Zave, "Classification of research efforts in requirements engineering," in *Proceedings of 1995 IEEE International Symposium on Requirements Engineering (RE'95)*. IEEE Comput. Soc. Press, 1995, pp. 214–216.

[33] J. Saldaña, *The Coding Manual for Qualitative Researchers*, 4th ed. Los Angeles; London; New Delhi; Singapore; Washington DC; Melbourne: SAGE, Jan. 2021.

[34] SUZANNE. ROBERTSON, "Requirements trawling: Techniques for discovering requirements," *International Journal of Human-Computer Studies*, vol. 55, no. 4, pp. 405–421, Jan. 2001.

[35] D.-H. Nguyen, D.-N. Nguyen-Duc, N. Huynh-Tuong, and H.-A. Pham, "CVSS: A Blockchainized Certificate Verifying Support System," in *Proceedings of the Ninth International Symposium on Information and Communication Technology - SoICT 2018*, Unknown, Ed. New York, New York, USA: ACM Press, Jan. 2018, pp. 436–442.

[36] 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," in *ICSA 2017: 2017 IEEE International Conference on Software Architecture : Proceedings : 3-7 April 2017, Gothenburg, Sweden*. Piscataway, NJ: IEEE, Jan. 2017, pp. 243–252.

[37] R. Kazman, L. Bass, G. Abowd, and M. Webb, "SAAM: A method for analyzing the properties of software architectures," in *Proceedings, 16th International Conference on Software Engineering: May 16-21, 1994, Sorrento, Italy*. Los Alamitos, CA: IEEE Computer Society Press, Jan. 1994, pp. 81–90.

[38] R. Kazman, M. Klein, and P. Clements, "ATAM: Method for Architecture Evaluation," Tech. Rep. CMU/SEI-2000-TR-004, Jan. 2000.

[39] P. Kruchten, "The 4+1 View Model of architecture," *IEEE Software*, vol. 12, no. 6, pp. 42–50, Nov. 1995.