COMP 6710: Software Quality in DLT/Blockchain

Frank McFadden, Alex Lewin

10/14/20

1 Executive Summary

This paper aims to present an overview of Software Quality Assurance (SQA) within the context of Distributed Ledger Technology (DLT) enabled applications. Through this paper, we address the definitions and fundamental architecture of DLT and blockchain, and we discuss several candidate approaches for addressing software quality as they pertain to a DLT ecosystem. Further, we will focus on some of the DLT-related project details that most affect Software Quality efforts.

To conclude, we discuss the challenges of upholding Software Quality Standards in a rapidly emerging field, especially with respect to DLT's position in existing industries. We conclude that DLT is an emerging field of technology which invites a new set of challenges, distinguishing the industry from the traditional software landscape.

1.1 What is DLT?

Distributed Ledger Technologies refer to applications that implement **decentralized**, **transactional** databases to exchange information and prove authenticity.

- **Decentralization** refers to a major departure from traditional database architecture. Traditionally, databases are typically **centralized**, where all data is centralized in a single location. This creates virtual libraries of data, which pose security issues analogous to physical libraries. The centralized database model is sometimes referred to as the **honeypot model**, as it organizes assets into a single, attackable interface for cyber criminals. On the other hand, **decentralized** databases spread information across a network of machines that work together to maintain security.
- **Transactional** databases refer to a special method of storing data. Traditional databases represent information by descriptively characterizing the properties of given entities.

For example, a database on a banking website could describe the content of an account by stating the following:

$$Customer\ X \rightarrow balance = \$100 \qquad \text{Traditional Example}$$

. This format represent information by a series of transactions. A bank implementing a

transactional database would represent accounts by stating the following:

$$Customer\ X paid\ Customer\ Y \rightarrow \$20 \qquad \text{Transactional Example}$$

This leaves the onus of tracking running balances to the users.

The decentralized and transactional nature of DLT is enabled by the use of **cryptographic signatures**. Whenever a transaction is written to the ledger, the author cryptographically signs the entry. This process proves the authenticity of the entry and prevents tampering. This allows the ledgers to maintain integrity over its contents.

1.2 What is Blockchain?

While DLT has not yet infiltrated the vernacular of the greater public, the term *blockchain* has absolutely garnered the attention of the masses. While blockchain and DLT are closely related, they have notably different meanings. Blockchain exists as a *subcategory* of Distributed Ledger Technologies. DLT refers to any decentralized, transactional database.

Blockchain, on the other hand, refers to a DLT system that organizes transactions into "blocks" and stores the cryptographic hash of the previous block onto the current block, creating a *chain* of *blocks* (ie. *blockchain*). This is an important distinction in order to understand the technology, but for the purposes of this paper, we will use the two terms interchangeably.

---

2 Candidate Approaches for Achieving Software Quality

Blockchain represents the culmination of many topics in software development - including enterprise software, open source projects and organizational cooperation. Because of this, when analyzing SQA in the context of blockchain, there are several perspectives that we can approach from.

2.1 Generic Software (Each of these should be brief, comes directly from section 3 of this paper: link)

2.2 Open Source

2.3 Process Maturity

---

3 Blockchain-Specific SQA

3.1 Scope (What are they? Pros and Cons)

3.1.1 Public Blockchains

3.1.2 Private Blockchains

3.1.3 Software Quality Concerns

3.2 Verification Protocol (Definition, Description from our presentation)

3.2.1 Single Verifier

3.2.2 M-of-N

3.2.3 Ad-Hoc

3.2.4 Verification: SQA Concerns

3.3 Consensus Protocol (Definition)

3.3.1 Proof of Work

3.3.2 Proof of Stake

3.3.3 Consensus Protocols: SQA Concerns

3.4 Use Cases

3.4.1 Cryptocurrency

3.4.2 Zero Knowledge Proofs

3.4.3 Self-Sovereign Identity

3.4.4 Solution Complexity and SQA Complexity (Just talk about the relationship)

---

4 Concluding Thoughts

On October 31st 2008, an author under the pseudonym Satoshi Nakamoto published a white paper that described a implementation for a technology that implemented cryptography to create a completely decentralized currency. Practically overnight, Bitcoin and other cryptocurrencies alike rocked the financial industry and inspired a massive surge of innovative projects that leveraged the same technology.

Today's society has entrusted some of its most mission-critical industries into blockchain-enabled systems, which has placed a multi-billion dollar incentive to develop the products.

Together, blockchain's immaturity and its undeniable value create the perfect conditions for a massive software failure. However, given its place in society today, we hope engineers and

organizations prioritize the SQA concerns that plague new technologies to hopefully avoid a meltdown.
