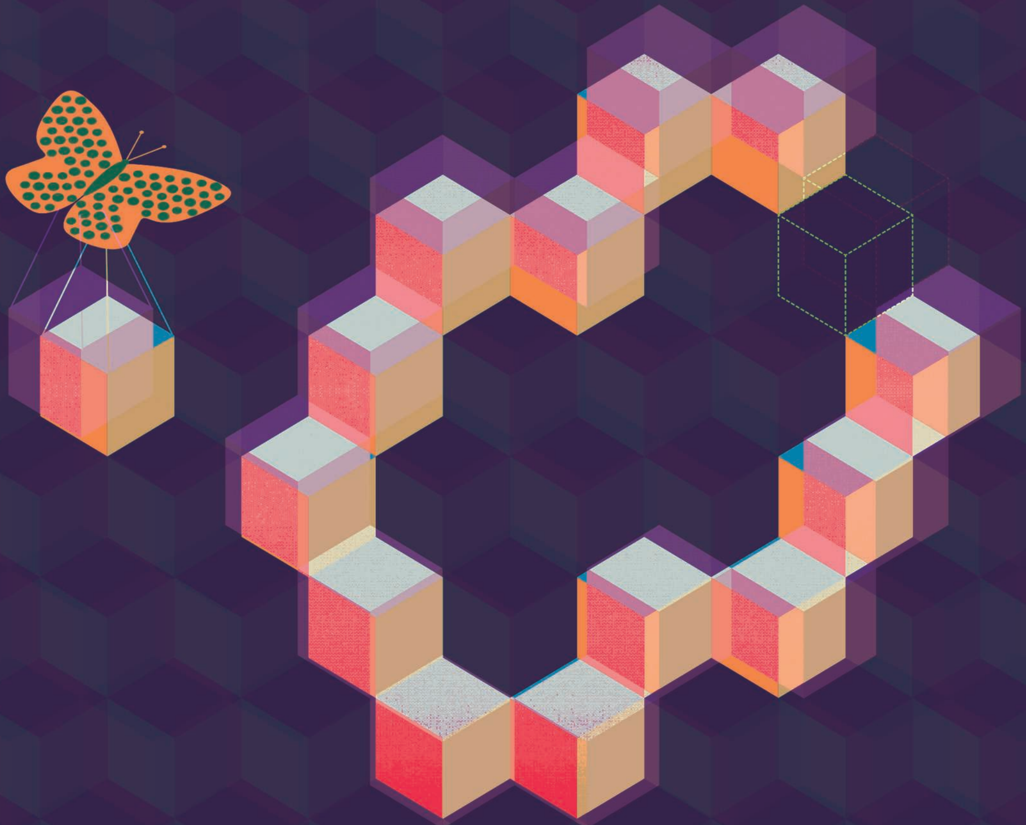
Blockchain and Smart Contract Engineering

Xabier Larrucea, TECNALIA, Basque Research and Technology Alliance

Cesare Pautasso, University of Lugano



Digital Object Identifier 10.1109/MS.2020.3000354
Date of current version: 20 August 2020

BLOCKCHAINS HELP TO build trust among a decentralized network of unknown and untrusted peers who need to agree on a common protocol and trust the correctness and compatibility of the corresponding software implementations. The software engineering discipline cannot ignore this trend, as it fundamentally affects the way software is designed, developed, deployed, and delivered.[1] As with the emergence of the Internet, software evolved from a closed to an open[2] and programmable[3] world, where every "thing" can be and eventually will be interconnected. Smart contracts and blockchains further push the boundary toward fully decentralized architectures of distributed applications. The goal of this *IEEE Software* special issue is to bring to light the fundamental connection between blockchains and software engineering, where the latest advances in software engineering may potentially find a natural application to the rigorous construction of safe and verifiable blockchain systems, while recognizing that blockchain and smart contract development also poses novel challenges for further software engineering research.

Software engineers need guidance for matching application domain requirements with the specific characteristics of blockchain solutions. This enables them to take advantage of smart contracts for solving new classes of real-world problems, as opposed to introducing blockchains everywhere, where they may be unnecessary, or provide an inefficient and environmentally unsound solution.[4]

As witnessed by a surge of dedicated standardization bodies, scientific workshops, conferences, and journal special issues, blockchain is making an impact well beyond computer science research. There are many potential industrial applications beyond online currencies, digital asset management, and distributed ledger technologies (DLTs), including product traceability along supply chains,[5] energy smart grids, healthcare, customer relationship management, programmable money, verifiable qualifications, business choreographies,[6] and e-voting,[7] to name a few.

Software evolved from a closed to an open and programmable world, where every "thing" can be and eventually will be interconnected.

Software Versus Smart Contract Engineering

To frame the articles in this special issue, here we give a brief overview of the most important differences between traditional software engineering practices and the assumptions that blockchain developers make when writing smart contracts. For more background information, see "The Origin of Blockchain."

Buggy smart contracts can have a serious financial impact, when, for example, funds leak or get locked forever inside a smart contract. Since smart contracts are immutable, by default they cannot be fixed. Upgrading them requires maintaining a delegation chain between smart contract versions or a registry so that the latest version can be looked up and dynamically bound. Code verification and static analysis have found a promising niche to help with early detection of bugs thus preventing incorrect smart contracts from being released.[8]

The original vision of utility computing[9] introduced the pay-per-use model which turned software into a service.[10] Cloud service providers would charge by the hour; now with serverless, the meter is ticking every second. Blockchain virtual machines running smart contracts have figured out how to charge for every microinstruction they execute.

Software engineers had to learn how to deal with limited resources by, for example, recycling allocated memory. On the blockchain, code execution may fail also because it has run out of funds. Developers have a big incentive to minimize the consumed storage, bandwidth, and the amount of computation performed, so that the cost of running smart contracts is kept within budget; the cost of running failed smart contracts which have reached the limit is not refunded.

While software as a service needs to be protected from malicious clients sending malformed input or performing denial of service attacks, smart contracts are an easier target because attackers have full visibility into their source code and their internal private state. Also, they can control the order in which transactions are processed.

Rate limits have been used in the past to keep over eager Web application programming interface

THE ORIGIN OF BLOCKCHAIN

What turns bits into money? It is the same force that turns paper cash bills into money: faith. Most human cultures believe in money as a store of value and as a unit of accounting, but mostly as a medium of exchange to facilitate payment transactions. Traditionally, money was based on scarce commodities (e.g., precious metals), which had to be divisible (to represent arbitrary amounts), durable over long periods of time, and easy to transfer (with minimal transaction costs). Incidentally, cash payments also happened to be anonymous: the parties exchanging cash could do so without revealing their identity to one another or to the central bank which printed the paper bills.

Long gone are the days of gold-backed currencies; now most of the money in circulation is digital, carefully preserved in your bank's data centers. There runs the software watching over the consistency of your account's balance, whose amount is a secret shared between you and your bank.

Every payment takes time and costs money. This is why financial institutions charge more for delivering express payments into your account. These costs also impact what is the smallest amount worth getting paid for. The vision of electronic money, digital cash, or cryptocurrencies like bitcoin was to disrupt the financial industry by lowering both the fees and the latency of payments over the Internet.[S1]

What turns bitcoins into money? The protocol everyone agrees to follow, whose goal is to allow everyone to exchange payment transactions directly while preventing double spending.[S2] As a consequence, not only does everyone need to be aware of everyone's payment history, but everyone has to agree about the order in which every payment transaction occurs. Since everyone is tracking everyone else's payments, anonymity is possible only if you remember to digitally sign every payment using a different pseudo-identity.

From a software design perspective, to keep track of the state (i.e., the account balance), we are using event sourcing to log the entire history of all state transitions (i.e., the payment transactions). Instead of entrusting your bank's database with this log, we fully replicate the log across an entire peer-to-peer network.

While forgery where money is concerned has always been attempted, it has never been easier to create money from nothing by flipping a few bits and then submitting fake payment transactions with money already spent. How does one validate payments and prevent untrusted peers from tampering with the log? Cryptography.

In the same way memory chips periodically refresh the bits they have stored to prevent data loss, all network participants continuously verify their copy of the log. To ensure that the order of transaction is preserved, each block of transactions refers to the previous one, identified by its

(*Continued on next page*)

consumers in check or to encourage them to upgrade their subscription plan. In the blockchain, rate limits are supposed to be used when transferring funds just in case the transfer has been triggered by mistake. They also offer a safety net to prevent attackers to instantaneously drain the balance of the smart contract they have taken over.

Whereas traditional containers protect the underlying environment from the untrusted code they execute, on the blockchain the smart contract code itself needs to be protected from its sandbox. The sandbox should not be trusted as it may attempt to steal the funds carried by the smart contract.

Software engineering promotes modular design and it has never been easier to rapidly assemble reusable components.[11] While the blind trust software developers put in the transitive closure of their dependencies not to contain any malicious code is remarkable, smart contracts also inherit the same security issues. One the one hand, smart contracts should be kept small so they can be understood, properly verified, and fully tested. On the other hand, their small size implies a modular architecture in which calling external contracts may execute malicious code and external callers may exploit and misuse smart contracts by arbitrarily invoking their public interface.

THE ORIGIN OF BLOCKCHAIN (CONT.)

cryptographic hash value. As long as malicious peers do not computationally overpower the network, this forms an immutable chain, to which new blocks of transactions can only be appended.

Why should everyone volunteer to store a copy of the blockchain and pay for the electricity needed to keep the blockchain software running? And how do we put money into the system? In the same way miners would dig the earth looking for gold; there is an incentive: freshly minted bitcoins are awarded to the peer who first computes a valid hash for a new block and adds it to the longest chain. Computing such cryptographic proof requires a significant processing time and effort,[S3] hence the term *proof of work*; bitcoin miners race to be the first to solve an inverse hashing problem whose difficulty increases over time to compensate for Moore's law.

A side effect of this race is the huge energy consumption to work on redundant computations.[S4] More sustainable consensus mechanisms are being worked out, with what is termed *proof of stake* being a promising one. As opposed to the miner controlling the most computational power, the likelihood for a validator to issue the next block depends on the amount of owned cryptocurrency.

While first-generation blockchains focused on cryptocurrency applications, supporting only monetary transactions, second-generation blockchains are much more flexible, as they can execute arbitrary transactions programmed using so-called smart contracts.[S5] Since smart contracts control the ownership of digital assets, they have been used to encode, execute, and enforce the terms of legal contracts established between business or institutional parties.[S6] The blockchain thus evolved from a replicated, immutable, append-only data structure into a virtual machine for open execution.[S7]

References

S1. P. Franco, *Understanding Bitcoin: Cryptography, Engineering and Economics*. Hoboken, NJ: Wiley, 2015.

S2. S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Bitcoin, 2008. Accessed on: June 16, 2020. [Online]. Available: https://bitcoin.org/bitcoin.pdf

S3. C. Dwork, and M. Naor, "Pricing via processing or combatting junk mail," in *Advances in Cryptology—CRYPTO 1992*, E. F. Brickell, Ed. Berlin, Heidelberg: Springer, 1992, pp. 139–147.

S4. Digiconomist, "Bitcoin energy consumption index." Accessed on: June 16, 2020. [Online]. Available: https://digiconomist.net/bitcoin-energy-consumption.

S5. V. Buterin, "A next-generation smart contract and decentralized application platform," Ethereum.org white paper. Accessed on: June 16, 2020. [Online]. Available: https://ethereum.org/whitepaper/

S6. N. Szabo, "Smart contracts: Building blocks for digital markets," in *EXTROPY J. Transhumanist Thought,* no. 16, 1996.

S7. K Salah, E. Damiani, A. Al-Fuqaha, T. Martin, K. Taha, and M. K. Khan, "Open execution—The blockchain model," *IEEE Blockchain Technical Briefs,* Dec. 2018. [Online]. Available: https://blockchain.ieee.org/technicalbriefs/december-2018/open-execution-the-blockchain-model

Software engineers are great at managing the history of their software projects; they have invented version control branches after all. The blockchain applies the concept of "hard forks" to capture the moment in which there is no longer a single, agreed-upon perspective over the history of how the collective state of a system has evolved.

Electrical engineers introduced circuit breakers. Microservice architects adopted them to prevent cascading failures. The same metaphor is used with smart contracts, whose circuit breakers can automatically or manually suspend their operation if a bug is discovered.

In the Cloud, open source code gets deployed out of sight, and one may only wonder whether the original "open" source code is the one actually running. The blockchain enables open execution; everyone can verify that your smart contract has not been tampered with by reexecuting it themselves and comparing the outcome. Agreeing on the outcome of replicated executions requires smart contracts to be deterministic. Any external interference (e.g., time, randomness, local configuration details) is forbidden.

Most applications have been built and can still be built using their own centralized database. To ensure the

FOR THE BLOCKCHAIN SKEPTIC

Just like any other overhyped technology, will the blockchain bubble burst and leave no trace behind? Or is there a fundamental breakthrough, which has the potential to change once again the way we design, develop, and deliver our software? Here, we discuss the former.

Blockchain has been characterized as a "solution looking for a problem."[S8] At the same time, it also fits the "golden hammer" antipattern,[S9] as it has been proposed as the means to solve every problem (including the coronavirus pandemic).[S10] A still unsolved problem, micropayments, did not work out with blockchain as originally intended due to the growth and volatility of transaction fees, also affected by the speculative bubble affecting the cryptocurrency exchange rates. From an innovation perspective, blockchain smart contracts have been compared with database-stored procedures.[S11] Smart contracts programming languages are Turing-complete, but programs should be kept deterministic.

From a security perspective, complex systems are as secure as their weakest link: if the core blockchain protocols are too difficult to conquer, the target of attackers shifts to the exchanges at the edge of the network[S12] or the private keys in your wallet. The performance—in terms of the transaction confirmation time—of centralized payment solutions is also hard to beat with a peer-to-peer network of untrusted but also churning and highly heterogeneous nodes.[S13] Blockchain seems to have found a niche as a payment channel of last resort,[S14] available to those whose access to traditional payment mechanisms has been denied (e.g., refugees or financial crisis victims, but also cybercriminals, such as ransomware authors).

References

S8. J. Bloomberg, "Eight reasons to be skeptical about blockchain," *Forbes*, May 31, 2017. [Online]. Available: https://www.forbes.com/sites/jasonbloomberg/2017/05/31/eight-reasons-to-be-skeptical-about-blockchain/#40f1cfb85eb1.

S9. W. H. Brown, R. C. Malveau, H. W. McCormick, and T. J. Mowbray, *AntiPatterns: Refactoring Software, Architectures, and Projects In Crisis*. New York: Wiley, 1998.

S10. D. C. Nguyen, M. Dinh, P. N. Pathirana, and A. Seneviratne, "Blockchain and AI-based solutions to combat coronavirus (COVID-19)-like epidemics: A survey," *Preprints*, vol. 2020, pp. 2020040325, doi: 10.20944/preprints202004.0325.v1.

S11. S. Nathan, C. Govindarajan, A. Saraf, M. Sethi, and P. Jayachandran, "Blockchain meets database: Design and implementation of a blockchain relational database," *Proc. VLDB Endowment*, vol. 12, no. 11, pp. 1539–1552, July 2019. doi: 10.14778/3342263.3342632

S12. C. Decker, and and R. Wattenhofer, "Bitcoin transaction malleability and MtGox," in *Computer Security-ESORICS 2014*, M. Kutyłowski, J. Vaidya, Eds. Cham, Switzerland: Springer, 2014, pp. 313–326.

S13. M. Swan, *Blockchain: Blueprint for a New Economy.* Sebastopol, CA: O'Reilly Media, 2015.

S14. C. M. Christopher, "Why on earth do people use bitcoin?" *Bus. Bankruptcy Law J.*, vol. 2, no. 1, pp. 1–10, 2014.

durability of the stored information the database needs a backup, from time to time. To make its data highly available, the database can be replicated.[12] Read-only data sets can be mirrored along a content-delivery network. With the blockchain, the append-only data structure shared by all applications gets fully replicated everywhere. For more information, see "For the Blockchain Skeptic" and "Research Challenges."

In This Theme Issue

In this special issue of *IEEE Software*, we invited submissions to explore the intersection between blockchain and software engineering. We received 65 submissions, out of which we selected six articles with the help of many reviewers, to whom we are grateful for their timely and constructive feedback. The small collection of articles you are about to read presents the latest advances and experiences in exploring the implications of blockchain on software engineering techniques, methods, and tools such as service-orientation, design patterns, model-driven engineering and fault tolerance.

In "Design Pattern as a Service for Blockchain-Based Self-Sovereign Identity," the authors present a reference architecture for self-sovereign identity management using blockchain. The article describes how to

RESEARCH CHALLENGES

The intersection of software engineering with blockchain is an area rich in research challenges,[S15] some of which are represented by the articles featured in this special issue. We highlight two main topics, focusing on how to engineer 1) software for running the blockchain itself and 2) applications running within (i.e., smart contracts) or just outside the blockchain.

Blockchain software needs to be reliable[S16] and secure, at scale, with a fully decentralized architecture. How does one significantly increase the performance of the current generation of blockchain virtual machines within these constraints? How does one define the right mix of incentives to ensure the long term sustainability of the blockchain?

Given the rapid expansion of competing blockchain networks, with different consensus protocols or new smart contract programming languages, how likely is it that only one will prevail? To deal with such ecosystem fragmentation, interoperability and portability solutions and standards will need to emerge once the technology matures.[S17] Will smart contracts be able to dynamically migrate between different blockchains, or will it be enough to run some form of distributed transactions across multiple blockchains?

Smart contracts that own and control valuable digital assets need to be correct and secure.[S18] There is a growing research niche of software analysis, measurement, verification, testing, and other quality assurance techniques being applied to smart contracts.[S19] This is critical, because smart contracts deployed in production are immutable. Iterative and incremental methods are difficult to apply.

Regarding applications, should every application use the blockchain? Or more precisely, which part of an application should run on-chain and which part off-chain?[S20] Is existing knowledge and experience gained for similar decisions (e.g., client versus server, or mobile versus cloud) transferable to this scenario?

References

S15. S. Porru, A. Pinna, M. Marchesi, and R. Tonelli, "Blockchain-oriented software engineering: Challenges and new directions," in *Proc. of the 39th Int. Conf. Software Engineering Companion*, Buenos Aires, Argentina, May 2017, pp. 169–171.

S16. I. Weber et al., "On availability for blockchain-based systems," in *2017 IEEE 36th Symp. Reliable Distributed Systems (SRDS)*, pp. 64–73. doi: 10.1109/SRDS.2017.15.

S17. S. Schulte, M. Sigwart, P. Frauenthaler, and M. Borkowski, "Towards blockchain interoperability," *Proc. BPM Blockchain Forum*, 2019, pp. 3–10.

S18. S. Sayeed, H. Marco-Gisbert, and T. Caira, "Smart contract: Attacks and protections," *IEEE Access*, vol. 8, pp. 24416–24427, 2020. doi:10.1109/ACCESS.2020.2970495

S19. E. Viglianisi, M. Ceccato, and P. Tonella, "A federated society of bots for smart contract testing," *J. Syst. Softw.* , vol. 168, Oct. 2020. doi:10.1016/j.jss.2020.110647

S20. J. Eberhardt and S. Tai, "On or off the blockchain? Insights on off-chaining computation and data," in *Proc. European Conf. Service-Oriented and Cloud Computing (ESOCC)*, Oslo, Norway, 2017, pp. 3–15. doi: 10.1007/978-3-319-67262-5_1.

introduce blockchain to provide a decentralized solution for the critical problem of identity management.

In "From Domain-Specific Language to Code: Smart Contracts and the Application of Design Patterns," the authors present how they applied model-driven engineering to generate smart contracts. They propose a domain-specific language known as *contract modeling language* based on a set of smart contracts design patterns and show that it is possible to transform high-level smart contract specifications into code to be executed on the blockchain.

The article "Neural Distributed Ledger" presents a novel approach for building and interconnecting blockchains. Ledger-of-ledgers developments approaches require a shift of developers' mindsets and a clear software design strategy and method. As stated by the Software Engineering Body of Knowledge, a software design strategy and method is useful as a common framework for teams of software engineers. This article provides a method and a tool suite for developing complex blockchains as ledgers of ledgers. In addition, authors present a set of assumptions that lead to a rapid and business-oriented DLT applicability assessment.

Two articles focus on smart contract failures. In "On the Need of

ABOUT THE AUTHORS

XABIER LARRUCEA is a project leader at TECNALIA, Basque Research and Technology Alliance. He is a Senior Member of the IEEE. Contact him at xabier.larrucea@tecnalia.com.

CESARE PAUTASSO is a full professor at the Software Institute, USI Faculty of Informatics, Lugano, Switzerland. He is a Senior Member of the IEEE. Contact him at c.pautasso@ieee.org.

Understanding the Failures of Smart Contracts," the authors introduce a smart contract debugger, which helps developers track failed smart contract executions by means of well-known software engineering techniques: code instrumentation, fuzzing, and failure location through binary search. In "Reducing Smart Contract Runtime Errors on Ethereum," the authors present both a taxonomy of different types of smart contract failures and empirically measure which are the most frequently occurring errors. The authors propose different error-avoidance heuristics, which can potentially save wasted computational efforts by using miners as canaries.

In "Unified Integration of Smart Contracts Through Service Orientation," the authors transfer service-oriented computing abstractions to the blockchain domain. Their smart contract description language, locator, and invocation protocol will foster the reusability, discoverability, and interoperability of smart contracts deployed within heterogeneous blockchain platforms.

Two of the columns in this issue also tie in with our theme. In the "Insights" column, you will find an interview with experts reflecting on the state of the blockchain and its relationship with software engineering. The "Practitioners' Digest" is also planning to summarize recent publications at the intersection between software engineering and blockchain research.

References

1. X. Xu, I. Weber, and M. Staples, *Architecture for Blockchain Applications*. New York: Springer, 2019.
2. L. Baresi, E. Di Nitto, and C. Ghezzi, "Toward open-world software: Issues and challenges," *Comput.*, vol. 39, no. 10, pp. 36–43, 2006. doi: 10.1109/MC.2006.362.
3. A. Taivalsaari and T. Mikkonen, "A roadmap to the programmable world: Software challenges in the IoT era," *IEEE Softw.*, vol. 34, no. 1, pp. 72–80, 2017. doi:10.1109/MS.2017.26.
4. B. A. Scriber, "A framework for determining blockchain applicability," *IEEE Softw.*, vol. 35, no. 4, pp. 70–77, July/Aug. 2018. doi: 10.1109/MS.2018.2801552
5. Q. Lu and X. Xu, "Adaptable blockchain-based systems: a case study for product traceability," *IEEE Softw.*, vol. 34, no. 6, pp. 21–27, Nov. 2017. doi: 10.1109/MS.2017.4121227.
6. J. Ladleif, M. Weske, and I. Weber, "Modeling and enforcing blockchain-based choreographies," in *Business Process Management (BPM 2019)*, vol 11675. New York: Springer, 2019, pp. 69–85.
7. N. Kshetri and J. Voas, "Blockchain-enabled e-voting," *IEEE Softw.*, vol. 35, no. 4, pp. 95–99, Jul/Aug 2018. doi: 10.1109/MS.2018.2801546
8. D. Magazzeni, P. McBurney, and W. Nash, "Validation and verification of smart contracts: A research agenda," *Comput.*, vol. 50, no. 9, pp. 50–57, 2017. doi:10.1109/MC.2017.3571045
9. G J. Feeney, R D. Hilton, R L. Johnson, T J. O'Rourke, and T E. Kurtz. "Utility computing: A superior alternative?" in *Proc. Nat. Computer Conf. and Expo. (AFIPS)*, 1974, pp. 1003–1004. doi: 10.1145/1500175.1500370.
10. C. Szyperski. "Component technology—what, where, and how?," in *Proc. 25th Int. Conf. Software Engineering, (ICSE 2003)*, pp. 684–693.
11. T. Mikkonen and A. Taivalsaari, "Software reuse in the era of opportunistic design," *IEEE Softw.*, vol. 36, no. 3, pp. 105–111, 2019. doi: 10.1109/MS.2018.2884883.
12. B. Kemme and G. Alonso, "A new approach to developing and implementing eager database replication protocols," *ACM Trans. Database Syst. (TODS)*, vol. 25, no. 3, pp. 333–379, 2000. doi:10.1145/363951.363955.