Designing a Smart-Contract Application Layer for Transacting Decentralized Autonomous Organizations

Conference Paper *in* Communications in Computer and Information Science · November 2016

DOI: 10.1007/978-981-10-5427-3_61

1 author:

Alex Norta
Tallinn University of Technology
76 PUBLICATIONS **408** CITATIONS

SEE PROFILE

Designing a Smart-Contract Application Layer for Transacting Decentralized Autonomous Organizations

Alex Norta

Department of Informatics, Tallinn University of Technology,
Akadeemia tee 15A, 12816, Tallinn, Estonia
alex.norta.phd@ieee.org

Abstract. This keynote paper addresses existing problems with traditional non-machine readable contracts that are based on trust. Such contracts have mostly a ceremonial purpose between transacting business parties and when conflicts occur, traditional contracts are often not enforcible. On the other hand, so called smart contracts that are machine readable and supported by blockchain-technology transactionalities do not require qualitative trust between contracting parties as blockchain establish instead a quantitative notion of trust. However, currently existing smart-contract solutions that equip the protocol layer on top of blockchains with Turing-complete programming languages lead to the false claim by industry practitioners they can manage smart contracts successfully. Instead, it is important to start considering the currently missing application layer for smart contracts.

Key words: decentralized autonomous organization, smart contract, blockchain, e-governance, lifecycle management

1 Introduction

The traditional understanding of a contract is a written or spoken agreement enforceable by law. An important prerequisite for a contract is that the parties involved voluntarily engage to establish a consensus [10]. In most business cases, contracts are documents [22] that identify the contracting parties uniquely, a service that is offered for some form of compensation that is usually monetary, and a set of additional clauses such as service-delivery dates, penalties for delivery failure, compensation clauses, and so on. Subsequent transactions are trust-based and contracting parties usually consider contracts as a symbol for an existing business deal. Another problem with the traditional form of setting up and managing contracts is that they are often underspecified. Most importantly, traditional contracts do not provide sufficient details about the actual transaction process and consequently, frictions between the contracting parties are very likely, e.g., one party assumes a specific product certificate before delivering a partial compensation and the other party assumes the opposite. The resulting

deadlocks lead to costly conflict resolutions, or even a collapse of the entire contract transaction. Also the enforcement of traditional contracts [14] proves to be either too complicated, time consuming, or impossible, certainly in international circumstances.

A solution for the listed issues pertaining to managing traditional contracts is to aim for an automation in the form of smart electronic contracts [26] that govern business transactions between decentralized autonomous organizations (DAO) [4]. A smart contract is a computerized transaction protocol [25] to execute contract terms and for achieving non-repudiation and fact-tracking of a consensual smart-contract agreement, blockchain technology [15] is suitable. The blockchain is a distributed database for independently verifying the chain of ownership of artefacts [21] in hash values that result from cryptographic digests.

Recently, an experiment with a DAO [1] has been developed with the smart-contract technology of Ethereum[1]. This DAO served as a crowdfunding project[2] and was hacked because of security flaws, resulting in a loss of $50 million. This incident shows it is not enough to merely equip the protocol layer on top of a blockchain with a Turing-complete language such as Solidity[3] to realize smart-contract management. Instead, we propose in this keynote paper that it is crucial to address a gap for secure smart-contract management pertaining to a currently ignored application-layer development.

The remainder of this keynote paper is structured as follows. Section 2 gives a brief overview of the sociotechnical implications of smart contracts for DAOs. Section 3 commences with the setup phase of a lifecycle for a smart-contract management application. Next, Section 4 discusses the ad-hoc establishment of a distributed governance process for smart-contract enactment, semantic rollback and termination that Section 5 shows. Section 6 briefly discusses latest scholarly publications and blockchain-technology innovations that enable smart-contract management. Finally, Section 7 concludes this keynote paper and proposes future research directions.

2 Sociotechnical Implications

It is important to briefly point out the profound implications of blockchain-technology enabled machine-readable smart contracts. Traditional contract managed can be described as one of soft and fraudulent, qualitative decision making that requires enormous centralized-planning coercion for stateism-powered enforcement. On the other hand, smart contracts promise to achieve a transition of society [2, 29] towards hard and objective, quantitative decision making based on trustless [24], disintermediated, decentralized and distributed organizational structures.

[1] https://ethereum.org/
[2] https://www.wired.com/2016/06/50-million-hack-just-showed-dao-human/
[3] https://solidity.readthedocs.io/en/develop/

We give some examples of those novel solutions that blockchain-technology enables beyond the initial crypto-currency inception of bitcoin [15]. These examples are free-market innovations that are superior to stateism-rooted established solutions. One example is a blockchain-based notary service[4] that cryptographically digests multimedia files and returns a uniquely identifying number at a specific timestamp. If only a single bit of the multimedia file changes, the number changes resulting the cryptographic digest.

Another notable example is a blockchain-technology based passport[5] that is more secure than traditional government-issued passports. The former can be instrumental for trading services on a Governance 2.0 platform termed Bitnation[6] that also serves as an e-notary for the Estonian e-Residency program[7].

Also state-of-the-art industrial applications employ blockchain technology. For example, cyber-physical systems use blockchains [5] to solve critical e-governance problems. Blockchains also enable new business models [32] in Internet-of-Things (IoT) applications. Even potential data-privacy violations in e-healthcare [31] can be tackled with the help of blockchains and there are many more sociotechnical application domains we can not list here due to page limitations.

For all the examples above, smart contracts are a means to enable e-governance for novel business model. Consequently, we next conceptually describe the application-level lifecycle management that industrial smart-contract solutions do not yet consider. Note that the respective lifecycle stages are conceptual summaries from published research papers that we reference in the sequel.

3 The Setup Phase

A peer-to-peer (P2P) smart-contract collaboration model Section 3.1 presents. Next, Section 3.2 gives the high-level structure of the academic smart-contract language termed eSourcing Markup Language (eSML) that we use as a proof of concept for an earlier ontology study [19]. Finally, Section 3.3 shows the actual setup lifecycle.

3.1 P2P-collaboration model

Pertaining to DAO-collaborations, Figure 1(a) conceptually depicts a configuration. The blueprint for an electronic-community formation is a so-called business-network model (BNM) [23]. The latter captures choreographies that are relevant for a business scenario and it contains legally valid template contracts that are service types with affiliated organizational roles. The BNMs are available in a

[4] https://proofofexistence.com/
[5] https://www.youtube.com/watch?v=1iAg6BITPdc
[6] https://bitnation.co/
[7] https://cointelegraph.com/news/estonian-e-residency-and-bitnation-launch-new-public-notary-in-blockchain-jurisdiction

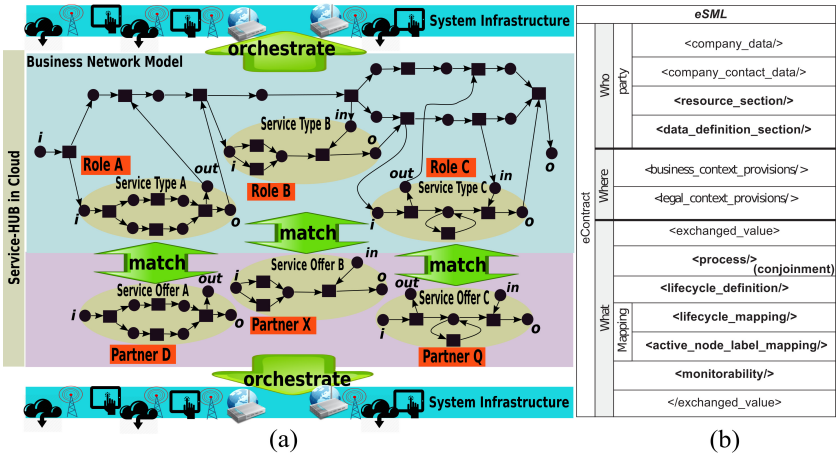

Fig. 1. P2P-collaboration using the eSourcing framework [17].

collaboration hub that houses business processes as a service (BPaaS-HUB) [9] in the form of subset process views [9]. The latter enable a fast and semi-automatic discovery of collaboration parties for learning about their identity, services, and reputation.

On the external layer of Figure 1(a), service offers identically match with service types contained in the BNM with the contractual sphere of collaborating parties. Additionally, a collaborating partner must match into the specific partner roles associated with a respective service type. We refer the reader to [8, 9] for details about the the tree-based process-view matching to establish a DAO-configuration into a contract-based collaborations.

3.2 Smart-contract structure

Figure 1(b) shows the top-level structure a smart-contract language termed eSourcing Markup Language (eSML) [19]. The core structure of a smart contract we organize according to the interrogatives *Who* for defining the contracting parties together with their resources and data definitions, *Where* to specify the business- and legal context, and *What* for specifying the exchanged business values. For achieving a consensus, we assume the *What*-interrogative employs matching process views that require cross-organizational alignment for monitorability. We refer to [19] for more information about the smart-contract ontology.

3.3 Conceptual setup lifecycle

The lifecycle of Figure 2 commences with breeding collaboration inceptions that produce BNMs comprising service types and roles. The BNMs that emerge from the breeding ecosystem exist permanently for repeated use in the subsequent populating stage. The validation of BNMs matches the available inserted service offers of potential collaboration eCommunity partners against service types.

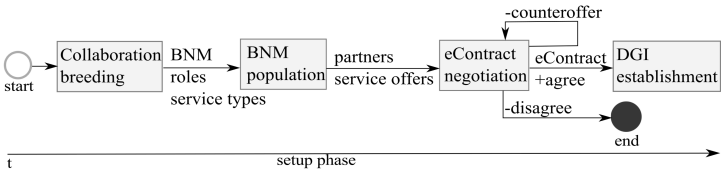

Fig. 2. Conceptual contracting setup-lifeycle for smart-contract establishment [17].

The *populate*-phase in Figure 2, yields a proto-contract for a *negotiate* step that involves the collaborating partners. The negotiation phase has three different outcome options. An agreement of all partners establishes the smart contract for subsequent rollout of a distributed governance infrastructure; a counter-offer from a partner that results in a new contract negotiation; finally, a disagreement of an eCommunity partner results in a complete termination of the setup phase. Note that the setup-lifecycle is formalized and we refer the reader to [16] for further details.

4 Deducing a Distributed Governance Infrastructure

In a time-line depicted in Figure 3, we show the creation sequence for elements of a distributed governance infrastructure (DGI) for enactment. If all eCommunity partners agree during the *negotiate* stage, a smart contract comes into existence that serves as a DGI-coordinating agent. In the *enterprise infrastructure distribution*, local smart-contract copies come into existence for every eCommunity-partner together with business network model agents (BNMA) and monitors. The *extract* stage creates sets of policies from the local smart-contract copies and assigns each a BNMA and monitor. Finally, the *prepare* stage populates the lowest technical DGI-level with matching services and corresponding endpoints for communication channels before *enactment*. For dismantling the DGI, the *termination* stage removes the entire infrastructure step by step.

Once the e-governance infrastructure is set up, technically realizing the behavior in the local copies of the contracts requires concrete local electronic services. After picking these services follows a creation of communication endpoints so that the services of the partners are able to communicate with each other. The final step of the preparation is a liveness check of the channel-connected services. The enactment carries out the tasks contained in the local electronic services by an engine that propels the eCommunity business collaboration technically.

5 Enacting, Rolling Back and Terminating

Three business-semantics rollback scenarios exist [12, 20] that may either be disruptive or calming, and that govern the transition of an eCommunity from one configuration to another. A conceptual depiction of these rollbacks Figure

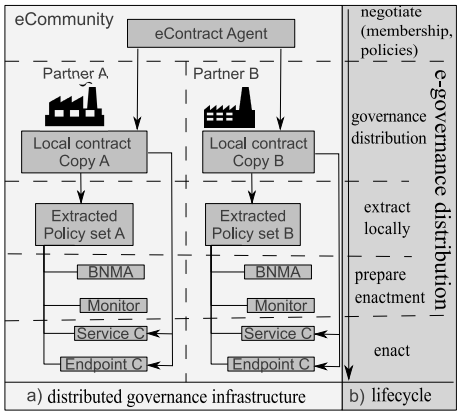

Fig. 3. Properties of a distributed governance infrastructure [12].

4 shows. Briefly, disruptive rollbacks imply a smart-contract renegotiation must start from scratch again. Calming rollbacks imply that the DAOs of an eCommunity see scope to reconcile collaboration issues. In both cases, the eCommunity experiences configuration changes.

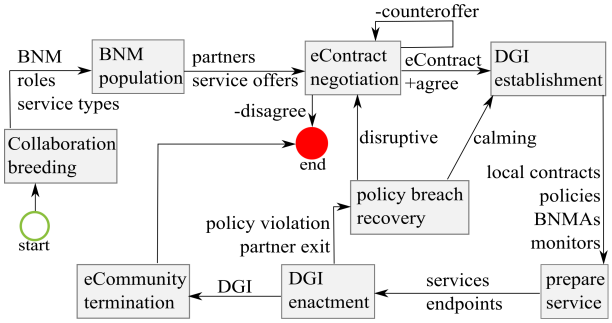

Fig. 4. A conceptual lifecycle for rolling back collaboration conflicts [20].

Pertaining to Figure 4, there are one disruptive and three conflict calming business-semantics rollbacks. The first type of disruptive business-semantics rollback commences after the decision to replace a current eCommunity partner with the objective to set up a new DGI. Thus, the disruptive business-semantics rollback dismantles the existing DGI and rolls back to the *negotiation*-service. Such a business-semantics rollback implies it is possible to have multiple eCommunity partners choose to discontinue their involvement in a newly emerging smart contract. Note that a policy-violating partner may again be part of the new smart contract. We infer, the reason for a disruptive partner change is caused by a policy violation of a severity that does not permit an eCommunity to continue collaboration.

Of the remaining three conflict calming business-semantics rollbacks, one also replaces an eCommunity-partner in a way where it does not dismantle and recreate the DGI. Likewise, the other conflict calming business-semantics rollbacks equally leave the existing DGI intact. If the eCommunity-partners vote to ignore a respective policy violation, the related smart-contract enactment resumes without any modification. The third type of conflict calming business-semantics rollback allows the complete replacement of a local smart contract copy with a new one as part of the existing DGI that remains otherwise unchanged. Thus, the new local smart contract and related policies, BNMA and monitor must adhere to the main DGI-coordinating smart-contract agent.

6 Discussion

So far, blockchain-technology innovation stem from free-market, open-source initiatives. Only very recently, also traditional computer-science academia realizes that blockchain-technology can no longer be ignored. Thus, Section 6.1 gives recent academic related work and Section 6.2 discusses recent smart-contract innovation that employs blockchain technology.

6.1 Academic related work

The authors in [27] map a running case of a collaborative process onto a smart contracting scripting language. That approach addresses the trust-issue in collaborative processes in that no single third-party entity must monitor events. Instead, the blockchain enables trustless process collaboration because of no single entity being in control. Additionally, the mapping from collaborative processes to blockchains enables the monitoring of process enactment and an auditing of related events.

The role of smart contracts supported by blockchains is also stressed in [30] for trust evaluation based on a diverse input of social-media facts. The authors present a proof-of-concept prototype that employs a blockchain ID for combining in a weighted way facts such as the amount of video feeds, number of likes and followers, online reputation, and so on, to calculate a reputation that diverse users may consider, e.g., future employers. The prototype helps to secure the identity of individual from hijacking, from online character assassination and provides the real reputation of the individual. For smart contracts, such a secure reputation system is important for evaluating the potential eCommunity partners during the setup phase and also during enactment-phase conflict resolutions.

Finally in [3], the authors stress that decentralized smart-contract validation requires formal methods. As a mentioned example, game theory is potentially useful to analyze the behavior of contracting parties under the assumption of perfect rationality and return maximization, which is not a realistic assumption given the nature of human action [13] that differs from trivial machines. Such game-theoretical verifications are only suitable when the contracting parties are

fully automated software agents. Yet, it is important to formally check the soundness of decentralized smart contracts during the setup phase [18] to assure the enactment of a smart contract is sound to assure a desired termination state.

6.2 Latest blockchain-based smart-contract innovation

We briefly discuss some notable blockchain-technology driven smart-contract innovations without claiming completeness. A very prominent example for a smart contract solution is Ethereum [28] that operates on top of a public blockchain-based distributed computing platform to execute peer-to-peer contracts using a crypto-currency called ether. Solidity[8] is the smart-contract language employed in Ethereum that is a statically typed object-oriented language resembling Java in its syntax.

Another smart-contract system is Lisk[9] that differs from Ethereum in that that every blockchain application is on its own sidechain. Informally, sidechaining allows tokens from one blockchain to be securely used within a completely separate blockchain and still moved back to the original chain. Furthermore, Lisk also differs from Ethereum by using JavaScript for smart-contract specifications.

More recently, Synereo [11] emerged from a failed implementation of a censorship-free social media such as facebook using Ethereum. The failure reasons are lack of Ethereum scalability because of full replication of the blockchain into every node, combined with costly proof-of-work block validations. Synereo uses RChain[10] that is a concurrent and compositional blockchain. Furthermore, Synereo adopts as a smart-contract language Rholang[11] that is a concurrency-oriented programming language with a focus on message-passing and asynchrony.

7 Conclusion

This keynote paper investigates the benefits and existing gaps in the use of blockchain-technology supported smart contracts. Traditional contracts that are not machine-readable typically merely have a symbolic character and pose severe problems such as inherent ambiguity that inevitably leads to conflicts, lack of enforceability, and so on. As a solution, machine-readable smart contracts force contracting parties into more detailed specifications yielding a reduction of conflict causes that occur during the enactment stage.

While blockchain-technology driven solutions rapidly penetrate industry-domains such as finance, e-governance, cyber-physical systems solutions such as for Industry 4.0, traditional academia [6, 7] only slowly considers blockchain-technology focused research. That is surprising given the interdisciplinary and

[8] https://github.com/ethereum/solidity
[9] https://lisk.io/
[10] https://blog.synereo.com/2016/09/05/meet-rchain-the-first-scalable-blazing-fast-turing-complete-blockchain/
[11] https://github.com/synereo/rholang

scientifically challenging application cases for disruptive concepts such as smart contracts. Specifically, this paper points out the existing gap in industry solutions pertaining to the application layer for smart contracts, in particular smart-contract lifecycle management. Note that the presented application-layer smart-contract lifecycle is based on pre-existing scholarly literature.

Considering the state of industry solutions, it is evident that the lack of academic involvement is a reason for suboptimal solutions pertaining to achieving a scalable management of blockchains, finding less expensive but still secure means of validating blocks and transactions, developing suitable and expressive Turing-complete smart-contract languages, and so on. We hope this paper raises awareness and results in an enhanced engagement of academia in blockchain research and development.

References

1. S.H. Ammous. Blockchain technology: What is it good for? *Available at SSRN 2832751*, 2016.
2. M. Atzori. Blockchain technology and decentralized governance: Is the state still necessary? *Available at SSRN*, 2015.
3. G. Bigi, A. Bracciali, G. Meacci, and E. Tuosto. *Validation of Decentralised Smart Contracts Through Game Theory and Formal Methods*, pages 142–161. Springer International Publishing, Cham, 2015.
4. V. Butterin. A next-generation smart contract and decentralized application platform, 2014.
5. W. Dai, V. Vyatkin, C. Pang, and J. H. J. H. Christensen. Time-stamped event based execution semantics for industrial cyber-physical systems. In *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, pages 1263–1268, July 2015.
6. D. Dirk. *Smart Business Process Management*, pages 207–223. Workflow Management Coalition, 2012.
7. D. Draheim. *Business process technology: A unified view on business processes, workflows and enterprise applications*. Springer Science & Business Media, 2010.
8. R. Eshuis, A. Norta, O. Kopp, and E. Pitkanen. Service outsourcing with process views. *IEEE Transactions on Services Computing*, 99(PrePrints):1, 2013.
9. R. Eshuis, A. Norta, and R. Roulaux. Evolving process views. *Information and Software Technology*, 80:20 – 35, 2016.
10. P.A. Hamburger. The development of the nineteenth-century consensus theory of contract. *Law and History Review*, 7(2):241–329, 10 2011.
11. D. Konforty, Y. Adam, D. Estrada, and L. G. Meredith. Synereo: The decentralized and distributed social network. 2015.
12. L. Kutvonen, A. Norta, and S. Ruohomaa. Inter-enterprise business transaction management in open service ecosystems. In *Enterprise Distributed Object Computing Conference (EDOC), 2012 IEEE 16th International*, pages 31–40. IEEE, 2012.
13. Ludwig von Mises. *Human action*. Ludwig von Mises Institute, 1949.
14. O. Morten. How firms overcome weak international contract enforcement: repeated interaction, collective punishment and trade finance. *Collective Punishment and Trade Finance (January 22, 2015)*, 2015.

15. S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Consulted*, 1(2012):28, 2008.
16. A. Norta. *Creation of Smart-Contracting Collaborations for Decentralized Autonomous Organizations*, pages 3–17. Springer International Publishing, Cham, 2015.
17. A. Norta. *Establishing Distributed Governance Infrastructures for Enacting Cross-Organization Collaborations*, pages 24–35. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.
18. A. Norta and R. Eshuis. Specification and verification of harmonized business-process collaborations. *Information Systems Frontiers*, 12:457–479, 2010.
19. A. Norta, L. Ma, Y. Duan, A. Rull, M. Kõlvart, and K. Taveter. eContractual choreography-language properties towards cross-organizational business collaboration. *Journal of Internet Services and Applications*, 6(1):1–23, 2015.
20. A. Norta, A. B. Othman, and K. Taveter. Conflict-resolution lifecycles for governed decentralized autonomous organization collaboration. In *Proceedings of the 2015 2Nd International Conference on Electronic Governance and Open Society: Challenges in Eurasia*, EGOSE '15, pages 244–257, New York, NY, USA, 2015. ACM.
21. B.S. Panikkar, S. Nair, P. Brody, and V. Pureswaran. Adept: An iot practitioner perspective, 2014.
22. T. Roxenhall and P. Ghauri. Use of the written contract in long-lasting business relationships. *Industrial Marketing Management*, 33(3):261 – 268, 2004.
23. T. Ruokolainen, S. Ruohomaa, and L. Kutvonen. Solving service ecosystem governance. In *Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2011 15th IEEE International*, pages 18–25. IEEE, 2011.
24. A. Schaub, R. Bazin, O. Hasan, and L. Brunie. *A Trustless Privacy-Preserving Reputation System*, pages 398–411. Springer International Publishing, Cham, 2016.
25. M. Swan. Blockchain thinking: The brain as a dac (decentralized autonomous organization). In *Texas Bitcoin Conference*, pages 27–29, 2015.
26. N. Szabo. Formalizing and securing relationships on public networks. *First Monday*, 2(9), 1997.
27. I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, and J. Mendling. *Untrusted Business Process Monitoring and Execution Using Blockchain*, pages 329–347. Springer International Publishing, Cham, 2016.
28. G. Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 2014.
29. A. Wright and P. De Filippi. Decentralized blockchain technology and the rise of lex cryptographia. *Available at SSRN 2580664*, 2015.
30. A. Yasin and L. Liu. An online identity and smart contract management system. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 192–198, June 2016.
31. X. Yue, H. Wang, D. Jin, M. Li, and W. Jiang. Healthcare data gateways: Found healthcare intelligence on blockchain with novel privacy risk control. *Journal of Medical Systems*, 40(10):1–8, 2016.
32. Y. Zhang and J. Wen. The iot electric business model: Using blockchain technology for the internet of things. *Peer-to-Peer Networking and Applications*, pages 1–12, 2016.