How Much Blockchain Do You Need? Towards a Concept for Building Hybrid DApp Architectures

Florian Wessling, Christopher Ehmke, Marc Hesenius, Volker Gruhn
paluno, University of Duisburg-Essen
Essen, Germany
firstname.lastname@paluno.uni-due.de

ABSTRACT

Adding blockchain technology to existing systems instead of building them from the ground up poses several challenges. It is difficult to find out which attributes of blockchains are important for a given use case (e.g. immutable, trustless, anonymous) and to decide which elements of an architecture should employ blockchain technologies. Current approaches generally only give a hint on whether blockchain technology makes sense for a given use case or not. This paper proposes a more fine-grained approach to decide which elements of an application architecture could benefit from the use of blockchain technology. We illustrate the first outline of our approach which identifies participants, their trust relations and interactions to derive a hybrid architecture (i.e., an architecture embedding blockchain technology in existing software systems or creating new systems using blockchain only in certain parts).

CCS CONCEPTS

• **Software and its engineering → Design patterns**; *Software design engineering*; *Reusability*; • **Computer systems organization →** *Distributed architectures*;

KEYWORDS

blockchain, architecture, Executable Distributed Code Contracts, EDCC, smart contract, decentralized application, DApp, design pattern, architecture pattern, blockchain-oriented software engineering, hybrid architectures

ACM Reference Format:
Florian Wessling, Christopher Ehmke, Marc Hesenius, Volker Gruhn. 2018. How Much Blockchain Do You Need? Towards a Concept for Building Hybrid DApp Architectures. In *WETSEB'18: WETSEB'18:IEEE/ACM 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB 2018), May 27, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3194113.3194121

1 INTRODUCTION

Blockchain technologies bring certain properties to create a decentralized, trustless, transparent and tamper-proof environment for building applications [10]. Those advantages come at the cost of additional technical complexity. When building blockchain-based applications (so-called DApps: decentralized applications) the goal is to benefit from the advantages and being able to handle the imposed technical challenges of blockchains (cf. [8]).

Current approaches focus on the question whether a blockchain is necessary for a given use case or not. Peck differentiates between three types of blockchain, that might be needed for an use case: "no blockchain", "permissioned blockchain" or "public blockchain" [6]. Wuest and Gervais add a fourth variant and distinguish between public and private permissioned blockchains [11]. This option is based on the question whether public verifiability is required. Xu et al. [12] argue that blockchain technology has many configurations and variants. To support the creation of blockchain-based systems the authors propose a design process for selecting and configuring the most suitable blockchain implementation. The main contribution of their work is a taxonomy of blockchain properties and a flowchart. Those elements serve as a guidance and initial questionnaire when designing a blockchain-based system by reflecting on aspects like authority, storage and decentralization. The result is very specific as it refers to the configuration of a single blockchain system with respect to technical details such as block creation time, block size, consensus algorithm, etc.

In this paper we focus on the architectural design for blockchain-oriented applications and propose an approach to decide which elements of an application architecture could benefit from the use of blockchain technology. We illustrate the first outline of our idea to derive a hybrid architectural draft by identifying participants, their trust relations and interactions. In contrast to the aforementioned approaches we propose a more fine-grained process. Instead of giving only a few general choices when deciding whether a blockchain is useful for a use case or not, we want to go into more detail and support developers in deciding which specific elements or areas of an architecture can benefit from the use of blockchain technology. This is especially relevant when applications are not built or rewritten from the ground up based on blockchain technology ("big-bang integration" [9]) but rather are extended with blockchain aspects for certain subsystems ("gradual integration" [9]). Those systems represent a hybrid architecture featuring elements both with and without blockchain technology (e.g. from existing software systems). This allows to benefit from blockchain properties in certain parts of an application and to decide which blockchain type and configuration fits best.

The paper is structured as follows: Section 2 explains our approach and its four steps in detail. In section 3 we conclude the paper and summarize our findings. Our ongoing research is outlined in section 4 as well as an outlook on our future work regarding architectural patterns and design patterns for code contracts.

WETSEB'18, May 27, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5726-5/18/05…$15.00
https://doi.org/10.1145/3194113.3194121

2 APPROACH

The approach of this paper is based on the hypothesis that the decision to use blockchain technology is not always useful to be made for a software system as a whole. Instead the decision should be considered for individual elements of a system. Our approach consists of four steps, which are explained in more detail in the next sections:

(1) Identify participants
(2) Identify trust relations between participants
(3) Identify interactions between participants
(4) Derive an architectural draft

(1) Identify Participants

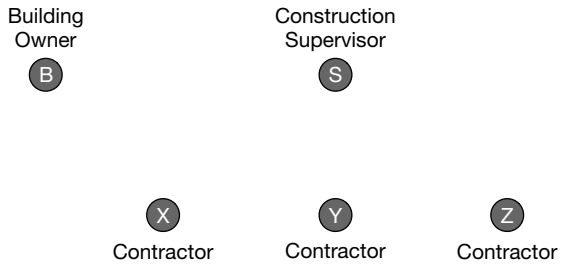

Figure 1: Identify participants

The first step is to identify the participants, i.e., peripheral systems or users, that interact with an application. To decide which participants or other systems are relevant to an use case, determine and consider the context boundary of the system (cf. [7]). In case the distinction does not make a difference it is possible to aggregate multiple participants (e.g. "Contractor X" and "Contractor Y") by simply using types of participants (e.g. "Contractors").

For our example we examine the creation of a new system that supports the coordination and payment of craftsmen constructing a building. As we do not add blockchain technology to an existing system, the selected participants focus on the people interacting with the planned application instead of peripheral systems. Figure 1 shows five participants: Building owner "B" who is legally responsible for the construction site and pays contractors. The construction supervisor "S" is responsible for coordinating the contractors "X", "Y" and "Z". The contractors carry out the construction works.

(2) Identify Trust Relations

Trust is a central aspect of building blockchain-based applications. Therefore identifying the trust relations among the participants of an application is a key task. The notion of trust exists in order to reduce the complexity of our environment, according to Luhmann [4]. He states that we are surrounded by many complex systems which must be considered when acting. The only way to reduce the complexity of these systems to a degree that enables us to act is through the concept of trust. Bierhoff and Vornefeld [2] point out that from a historical perspective there is a distinction between two forms of trust: relational trust, meaning the trust between different persons, and generalized trust. The latter refers to trust in social structures like the government or the trust that social conventions are generally followed. According to Bierhoff and Vornefeld, during the last decades a new form of trust arose: the so-called trust in abstract systems. This type of trust describes the problem that during the last years the number of complex systems around us increased significantly. For example many computer systems (like mobile phones, cars or even electronic door locks) began to interact with our daily life. These devices and systems are too complex to be checked for correctness by the common user. According to Bierhoff and Vornefeld people solved this problem by simply trusting that these systems work correctly.

These trust considerations can be transferred to software engineering as well. Classical software development means relying on the correctness or trustworthiness of the software host, the software itself, the hardware used to serve it, the developer or the software testing and quality assurance team. Blockchain technology proposed a shift in this trust relationship: Instead of trusting an abstract system (i.e., a server running the software) the users trust a protocol that encourages honest user behavior. The Bitcoin whitepaper explicitly states that blockchains propose a system that does not rely on trust and thus being trustless [5]. The fact that the resilience of a blockchain network is based on transparency and that every node is required to validate each transaction reduces the required trust both into other users and systems (cf. [1]).

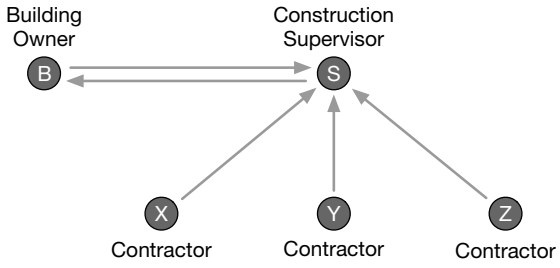

Figure 2: Identify trust relations of participants

When considering trust relations in this second step it can be useful to think about general trust classes the participants can be assigned to. For example the classes can range from the lowest trust in "unknown persons or companies", medium trust for "known partners with business relations in the past", higher trust for own "subsidiaries and holdings" and highest trust for the "own company". Generally these distinctions are not precise but usually sufficient for a first evaluation.

In our simplified example, several trust relations between participants identified in the previous step are illustrated in Figure 2. We assume a mutual and high trust relation between the building owner "B" and construction supervisor "S" as they work closely together while "B" delegates the coordination of contractors to "S". The contractors trust the construction supervisor as their contact person to assign tasks.

(3) Identify Interactions

As the third step any interactions between the participants have to be identified. An interaction can be seen as any exchange of

data using the planned application. This can be a function call that adds, removes or modifies data in any of the peripheral systems. These interactions can occur between users, systems or a mix of both. It is part of our ongoing research to decide if types of interaction, e.g. reading or writing data, need to be differentiated for further decisions in our approach. For example it could be useful to allow reading operations but restrict writing operations only to participants considered trustworthy in the previous step.

Figure 3 shows an example of the main interaction relations between participants identified in the first step. Contractors report to the construction supervisor that the assigned tasks have been carried out. The construction supervisor frequently sends updates about the current state of the construction site and overall project progress to the building owner. The interactions between the building owner and the contractors result from the legal responsibility for the construction site and obligation to pay the contractors.

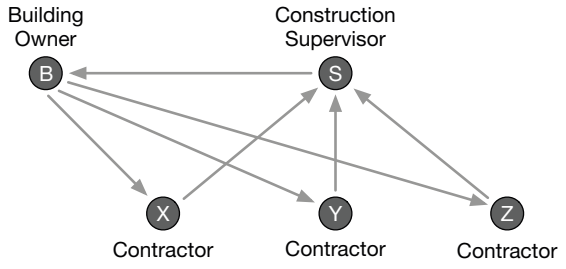

Figure 3: Identify interactions between participants

(4) Derive an Architectural Draft

In this last step the results gathered in steps 1–3 are combined to derive a first architectural draft. This task supports the identification of specific areas where blockchain technology can benefit the system architecture. For a given use case it needs to be considered which value the blockchain should secure and be able to transport (e.g. representation and transfer of ownership, documentation of process execution or execution of payments). This also influences the scope of the blockchain, i.e., the participants that will read data from or write data to the blockchain.

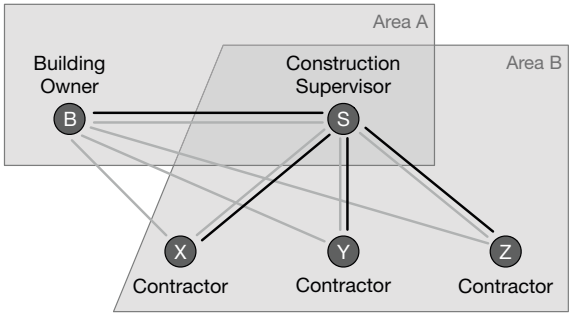

Figure 4: Emerging areas of the trust and interaction overlay as a first hint towards an architectural draft

Figure 4 shows the overlay of trust relations (in black) and interactions (in grey) from our example. Two areas emerge: Area A covers the mutual trust and interaction relation between the building owner and the construction supervisor. As all participants trust each other and can freely interact, it does not make sense to employ blockchain technology. Here the identified trust and interaction relations suggest the use of a centralized software system as the easiest and most cost-effective solution. Area B covers the trust and interaction between the construction supervisor and the contractors. This area is different from area A as only a small subset of the participants trust each other with the construction supervisor as a central actor. What is not covered, is the interaction between the building owner and the contractors. Their trust relation exists only transitively through the construction supervisor but the owner is responsible for paying the contractors (as "B" legally entered a contract with "X", "Y" and "Z" that was arranged by "S").

At the intersection between area A and B the construction supervisor "S" serves as a connector between those two areas of trust and interaction. There are two solutions that can be considered to enable the interaction between the building owner and the contractors: (a) All participants trust the construction supervisor that therefore can serve as a "trusted third-party" and coordinate all interactions. (b) Use blockchain technology to connect the building owner directly to the contractors for payment purposes and getting automatically informed about the project's progress. In the latter case using blockchain technology can be justified as it enables trustless interaction between participants without the need of a centralized system or trusted third-party. Instead of replacing the construction supervisor with blockchain technology this solution reduces the power and trust that is necessary towards this participant. The supervisor can still be responsible for setting up the legal contracts, hire contractors on behalf of the building owner and configure the blockchain infrastructure. This covers the configuration and deployment of the Executable Distributed Code Contracts on the blockchain (=EDCCs: a more precise term for Smart Contracts[1]). These code contracts support a more decentralized project management by allowing the supervisor to assign tasks, contractors to confirm finished tasks, enable automatic and tamper-proof progress documentation and execute payments automatically.

All insights from this step give a first hint towards an architectural draft for the system in our example. Based on the trust and interaction relations the use of blockchain technology seems useful in a certain area of our system, specifically for documentation and payment purposes.

3 CONCLUSION

The need for building blockchain-based applications as well as adding blockchain technology to existing software systems is increasing. Nonetheless, for creating these systems a holistic engineering approach is currently missing. It needs to be examined whether existing methods for software engineering are applicable for blockchain-oriented software engineering [8], if adjustments are necessary or completely new methods and processes are required.

[1] cf. https://www.ethnews.com/edcc

As a first step towards a blockchain-oriented software engineering approach we motivated in this paper the need for a more fine-grained approach when deciding whether blockchain technology is useful for a given use case or not. We presented the first outline of a concept that can be used to derive a high-level hybrid architecture of a blockchain-based application by identifying participants, their trust relations and interactions.

4 FUTURE WORK

In this section we will explain the focus of our future research, peripheral ideas and how the architectural approach outlined in this paper can be extended on an implementation and code level.

The approach as presented in section 2 is currently under development. Therefore several open questions remain and are subject of future work. In contrast to the given example we will test our approach in a situation where blockchain technology should be added to an existing system, instead of being built from scratch. We also want to examine the potential target audience of our approach. Maybe developers perceive the trust relations of the participants differently from project managers.

Architectural Patterns

The simplified example from section 2 showed a transitive trust relation between the building owner, construction supervisor and contractors. Although there is no direct trust relation between the building owner and contractors, both need to interact with each other. After identifying the trust relations and interactions we explained why the use of blockchain technology makes sense in this case and can solve this issue.

We assume that challenges like this will come up regularly where patterns in specific situations can be identified (e.g. using trust relations and interactions as a hint). Therefore we assume that multiple patterns for architectures of blockchain-based applications will emerge. Some authors have similar ideas, e.g. Xu et al. mention "design patterns for applications based on blockchain" as part of their future work [12]. These architectural patterns will support developers when deciding which elements of their architectures will benefit from using blockchain technology and how existing systems can be combined to benefit from both approaches.

Design Patterns for EDCCs

The high-level hybrid architectural draft is a first step towards a system using blockchain technology. As the next step the lower-level, i.e., the implementation and code level, has to be considered. In blockchain-based systems the business logic is usually implemented as a set of Executable Distributed Code Contracts (EDCCs or Smart Contracts). Similar to the discussion whether methods from known software engineering processes need to be adapted for building blockchain-based applications, the question also arises on a code level. Design patterns are a common instrument to solve reoccurring problems (cf. the "Gang of Four" patterns [3]). For our ongoing research we examine these design patterns, consider whether they are applicable for EDCCs and measure their impact on transaction costs.

ACKNOWLEDGMENTS

The European Union supported this work through the project CPS.HUB NRW, EFRE No. 0-4000-17.

REFERENCES

[1] Andreas M. Antonopoulos. 2014. *Mastering Bitcoin: Unlocking Digital Crypto-Currencies* (1st ed.). O'Reilly Media, Inc.
[2] Hans-Werner Bierhoff and Bernd Vornefeld. 2004. The social psychology of trust with applications in the internet. *Analyse & Kritik* 26, 1 (2004), 48–62.
[3] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1995. *Design Patterns: Elements of Reusable Object-oriented Software.* Addison-Wesley Longman Publishing Co., Inc.
[4] N. Luhmann. 2017. *Trust and Power.* John Wiley & Sons.
[5] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).
[6] M. E. Peck. 2017. Blockchain world - Do you need a blockchain? This chart will tell you if the technology can solve your problem. 54, 10 (2017), 38–60. https://doi.org/10.1109/MSPEC.2017.8048838
[7] Klaus Pohl. 2010. *Requirements Engineering: Fundamentals, Principles, and Techniques* (1st ed.). Springer Publishing Company, Incorporated.
[8] S. Porru, A. Pinna, M. Marchesi, and R. Tonelli. 2017. Blockchain-Oriented Software Engineering: Challenges and New Directions. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C).* 169–171. https://doi.org/10.1109/ICSE-C.2017.142
[9] Victoria Stavridou. 1999. Integration in software intensive systems. *Journal of Systems and Software* 48, 2 (1999), 91–104. https://doi.org/10.1016/S0164-1212(99)00049-7
[10] Melanie Swan. 2015. *Blockchain: Blueprint for a New Economy.*
[11] Karl Wüst and Arthur Gervais. 2017. Do you need a Blockchain? (2017). https://eprint.iacr.org/2017/375
[12] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba. 2017. A Taxonomy of Blockchain-Based Systems for Architecture Design. In *2017 IEEE International Conference on Software Architecture (ICSA).* 243–252. https://doi.org/10.1109/ICSA.2017.33