Using DLT in Software Lifecycle Management

Biser Tsvetkov and Hristo Kostadinov(✉)

Institute of Mathematics and Informatics, Bulgarian Academy of Sciences (IMI-BAS), Acad. G. Bonchev Street, Bl. 8, 1113 Sofia, Bulgaria
hristo@math.bas.bg

Abstract. Motivated by the challenging business requirements in the area of software lifecycle management (SLM) we describe a system that is based on the use of distributed ledger technology (DLT) to address some of existing issues. DLTs and blockchain in particular allows industries to utilize distributed peer-to-peer networks where non-fully-trusting members may interact with each other without relying on the assistance of a trusted 3rd party. These interactions are kept verifiable and auditable both from the participants in the process and external players. One unexploited use of blockchain is in the area of SLM. Here providers of software products and services could benefit of some natural DLT properties. First: Establishing secure communications between non-fully trusted participants of the SLM processes and reducing the risks from malicious code reaching customer sites and their productive systems. Second: It would allow sharing of SLM resources and services between involved parties thus greatly improving the overall interaction costs required for complex multi-party procedures. Third: The system would allow better time and resource prediction for upcoming SLM procedures by keeping history execution data from previous runs of the same or similar procedures. Using DLT in SLM would allow simplification and automation of existing complex and time-consuming procedures in a cryptographically verifiable manner. Also, in some cases payments for new software or financial penalties for unplanned downtimes or other SLA infringements could be executed automatically.

1 Introduction

The distributed ledger technology is targeting now various scenarios, especially after the successful introduction of Bitcoin and blockchain technology in early 2009 where they proved that a trustless fully decentralized system could be build. Bitcoin is cryptographically protected and all participants in the peer-to-peer communication are incentivized to play fair for best results [1]. One of the areas where blockchain and Distributed Ledger Technologies are a good fit is the area of Software Lifecycle Management [2–4]. In this paper we will research and describe a system that uses DLT for secure sharing and distribution of SLM artifacts. Such system will also facilitate tracking of the changes and the status of the system regarding important process milestones.

I. Georgiev et al. (Eds.): BGSIAM 2018, SCI 961, pp. 393–404, 2021.
https://doi.org/10.1007/978-3-030-71616-5_35

"Classical blockchain", for which Bitcoin is the most popular example as a decentralizes peer-to-peer network of equal nodes, stores data in cryptographically linked data blocks. Its communication protocols and storage use public key cryptography and hashing algorithms to guarantee that the data on all nodes is accurate and is not been tempered with [5–7]. The written data uses proper consensus method in blockchain or DLT and it is considered to be immutable for practical purposes. Some enterprise systems requirements for high availability and disaster recovery could easily be guaranteed on a DLT protocol level since each blockchain node keeps a full copy of the same history of events as cryptographically linked blocks. In case the data of a particular node is lost it can be recovered by the information existing on other nodes. As result the Bitcoin data is transparently shared for all participants. Adding new blocks is done on predefined intervals by a consensus mechanism know as proof-of-work (PoW). PoW is pretty power-hungry mechanism, but it is a safe and now proven approach for defending blockchains integrity. After the success of the original Bitcoin network there are many new public blockchains implemented and in use such as Ethereum and EOS networks. Apart from the initial PoW approach, introduced with Bitcoin, the consensus for new blockchains offers new and less power-intensive approaches toward block consensus. Some of the new consensus methods are prove-of-stake (PoS), proof-of-authority (PoA) and delegated-proof-of-stake (DPoS).

At a later stage, driven by business requirements, some new flavors of the technology were invented. The so-called permissioned blockchains got popular in the enterprise world. Permissioned blockchains, unlike the free-to-join approach of the public blockchains, require explicit or implicit permissions to be granted for each node, user or organization to be able to use the network. Actions such as access to join network, read data, proposal to write data, participation in the consensus protocol (mining), voting, administration and others may be explicitly granted to some participating entities.

On Fig. 1 is shown one classification of the existing DLT technologies. As we can see there are blockchain and non-blockchain DLTs for both public DLT and enterprise DLT worlds. Some of the restrictions of all blockchain technologies lead to the evolution of the DLTs further away from Bitcoin, cryptocurrency and formal blockchain definition. There are certain limitations of original Bitcoin implementations that are later considered unfit for systems that have to fulfill different requirements. This led to some further developments of DLTs.

Here are some of the directions of DLTs has evolved since 2009:

- **Technology architecture.** While the majority of popular DLTs have architecture similar to the original Bitcoin blockchain there are several that differentiate cardinally from it and may not have even blocks. Some examples are Corda [8], Tangle [9], Nano [10].
- **Access.** AAs previously stated most corporate DLTs require some form of control for joining and operation [8,11].
- **Consensus mechanism.** The PoW consensus is well-known to be extremely power-hungry due to the concurrency between miners competing to produce

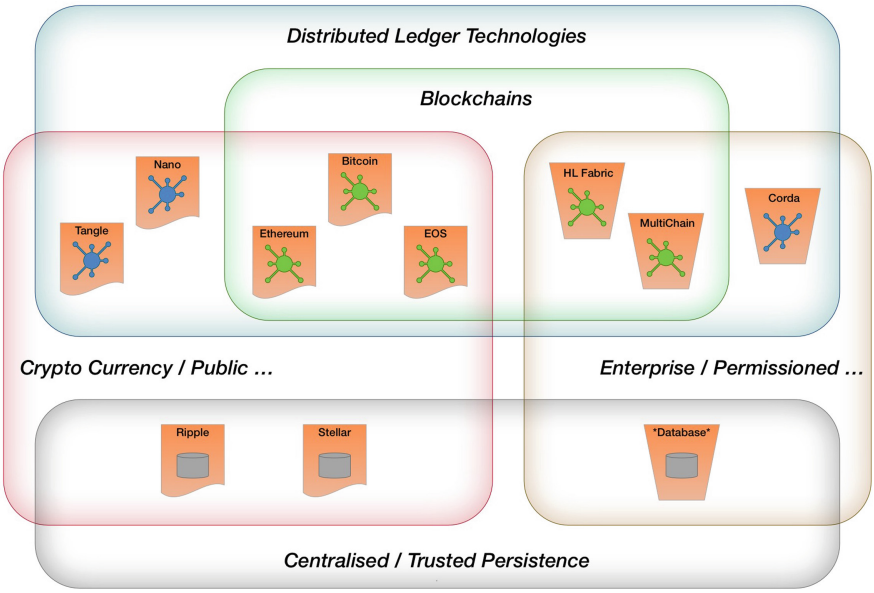

Fig. 1. DLT categories

valid blocks. Recent DLTs embrace more environmental-friendly mechanisms like distributed proof of stake, proof of authority, practical byzantine fault tolerance (PBFT) and others [8,11,12] .

- **ASIC resistance.** Bitcoin is mined mostly using ASICs at the moment. New algorithms are used in recent DLTs to guarantee fairer miner distribution (as long as PoW or similar techniques are used) [11,13].
- **Own crypto currency.** Bitcoin network is all about cryptocurrency generation and distribution as it is designed as such. Many modern DLTs do not introduce own currency and may operate without using any [8,11]. This is especially true for the enterprise/permissioned DLTs which normally do not have cryptocurrency attached.
- **Data visibility.** Unlike the full transparency of Bitcoin most Enterprise DLTs and some cryptocurrencies offer level of security that protect the data from being viewed by unauthorized parties [8].
- **Stored data.** While Bitcoin nodes store the full history on each node there are modern DLTs that allow only small part of data to be kept in each operating node [13].
- **Node equality.** All Bitcoin nodes store its full information and are considered equal as functions allowed. Many recent blockchains are built to have two or more layers of nodes. Their nodes may differ by having less features (Ethereum's light nodes) or offer unique functionality (Corda's Notaries, EOS's Block producers, Tangle's Coordinators).
- **Data locality.** WWhile Bitcoin operates on a global scale many DLTs are built with the idea to support sharding and data separation [12].

- **Transaction finalization.** In Bitcoin network each transaction, once put into a block, is final and could not be changed [9]. Some DLTs, like EOS, allow some transactions to be reversed in specific situations.
- **Smart contracts/Oracles.** One of the major steps in DLT evolution after Bitcoin is the introduction of Smart Contracts by Ethereum network. It is a feature that most DLTs embrace and extend [12].

The rapid development of DLTs in recent time solved many shortcomings of the original bitcoin network and extended the class of problems DLTs can address. One of the new issues that architects now have is to properly choose the best suited DLT for specific tasks. In this work different DLT features will be investigated regarding their suitability to solve current problems in the area of the software lifecycle management.

The rest of this paper is organized as follows. In Sect. 2 we introduce some of the typical issues for complex projects running in software lifecycle management area. Demonstration of a DLT-based system for managing and coordination of SLM processes between separate organizations will be presented in Sect. 3. Conclusion remarks will be given in Sect. 4.

2 Software Lifecycle Management

In this section we describe some typical issues of Software Lifecycle Management (SLM) in multi-party environment. Software lifecycle management is the area of installation, configuration and maintenance of complex software systems that could consist of many servers, clouds or edge devices, spread on many locations that often have complex dependencies on each other. Quality requirements such as service's high availability and disaster recovery are often requested by customer thus increasing the complexity of the overall solution and the procedures used to install, configure and maintain it. In addition, in the enterprise world SLM is an area where often many different parties are involved in a single project. The SLM-relevant information about the system properties is often limited only to some of the participants, sometimes it is missing, intentionally or not, and in many cases there is wrong information as result of different parties' manipulations. Even more important – the history of the system as a series of SLM procedures further pinpoint specific issues that need to be targeted by any new party involved in its maintenance. Such info may not be shared by previous participants for various reasons.

Due to the ever-changing nature of the software systems the related issues are rarely well known, and responsibilities of each party are not always defined in detail for each project. The interest of each party is the project to be successful and each one contributes in different area for achieving the success. Sharing plans, data and commitments is important part of project's success. Also, specific deliveries in time and quality are often a milestone for passing responsibilities from one participant/partner to another.

On Fig. 2 is shown a typical setup for customer using enterprise applications. There are several parties involved in such procedure. The main participant is

the customer itself having the application running on servers in private landscape, somewhere in the cloud or possibly on some edge devices. The provider of the software is another involved party in the process. Typically, there are also technical consultants involved in the process which handle the technical aspects of the SLM on behalf of the customer. In addition, vendors of hardware and software (such as databases and operational systems) could be involved in some types of SLM processes.

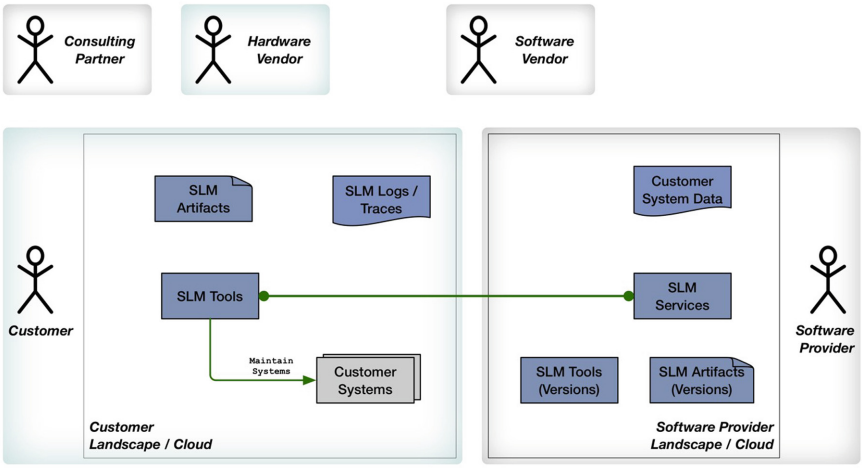

Fig. 2. Typical setup for multi-party SLM process

Using DLTs as a medium for sharing data and SLM artifacts such as archived executables and customized configurations will solve many of the issues above. Permissioned DLTs provide such environment sharing process data and SLM artifacts that is verifiable, transparent and protected from the unauthorized access. Smart contracts could be used for safeguarding SLM procedures at customer site and to verify that only approved executables and valid configurations are used by customer and its consulting partners. Setting proper policies for the DLT allow explicit approvals to be requested and granted by other involved parties.

Example: The Provider of the Software detects a security issue in a specific version of their product. The provider should notify all its users of that specific version to evaluate the effects of the issues found. There is no need this information to go public immediately as it may generate unwanted attention to these customers' systems. Once a fix is ready it is sent using the same channel to both the end customers and their consultants. They evaluate the degree of the risk, test it on their test systems and decide if and when to apply the fix to their productive systems. Customer could choose downtime interval that consultants

will use to apply the fix. During the selected downtime window, the consultant is allowed to apply the requested fix on the productive system. At the end of the procedure, using the DLT notifications, Software provider will be notified that the security fix is applied at the customer system by the specific consultant. The whole procedure is well documented and transparent for all involved parties.

Let's have a look at some of the key issues of multi-party SLM procedures that could be addressed by the DLT-based system presented in the next section:

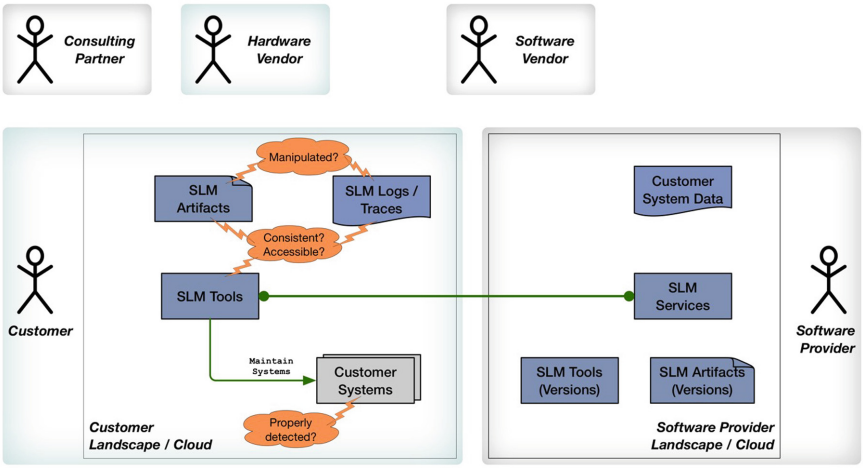

Fig. 3. SLM process: consistency and security

- **System Consistency and Security.** It is often the case when some SLM artifacts – executables, configurations scripts, execution logs and traces are manipulated at customer site. Sometimes deletion of such files is even part of the established procedures at customer site. On Fig. 3 are shown the SLM artifacts, logs and traces that could benefit from additional security coming from using DLT as a medium to track their consistency. Older versions of the files sometimes overwrite the later ones causing issues during the use of the systems. In some cases, person that have access to the tools may choose not to apply (by accident or deliberately) certain fix or patch to the system. This way the system is left in malfunctioning state or have its security compromised. If the person has access to the logs and traces, he could cover all his traces and hide such manipulations from customer and other parties. Later, in case of dispute, the Software Provider could be kept responsible for any damages caused by using the compromised system. Using DLT it guarantees that the downloaded SLM artifacts are authentic, they are used properly, and the customer system is properly updated, patched, configured to their latest known secure version. All manipulation of critical parameters such as firewall settings can also be tracked in detail and verified.

- **Data Visibility.** Visibility of system's data is important aspect of any systems maintenance. On Fig. 4 are shown parties and data that is shared during the SLM procedures and after it. The Software Provider needs to know what applications are used by its customers, at what version they are and if they are properly patched and configured. On the other hand, the customer and its consulting parties are interested to know about any new versions of the provided software. Any security-relevant fixes have to be visible and applied as soon as possible preferably before they are made public and potentially exploitable.
- **Downtime and resource prediction, risk analysis.** FFor many SLM procedures it is important to predict properly the duration of the expected procedure steps and their resource usage. On Fig. 5 are presented parties and sample landscape that benefit from having access to the historical data for the SLM processes. For upcoming SLM procedure one of the most important feature is the prediction of the period of system downtime. In addition, some of the procedures may have critical steps that could potentially cause issues. It would be in best interest that the executions of the procedure at other customers are tracked by the software provider so that similar risk-evaluation could be done based on statistics. Also, in this way software provider validates the effect of certain performance and other improvements by getting real customer data for SLM tool usage.

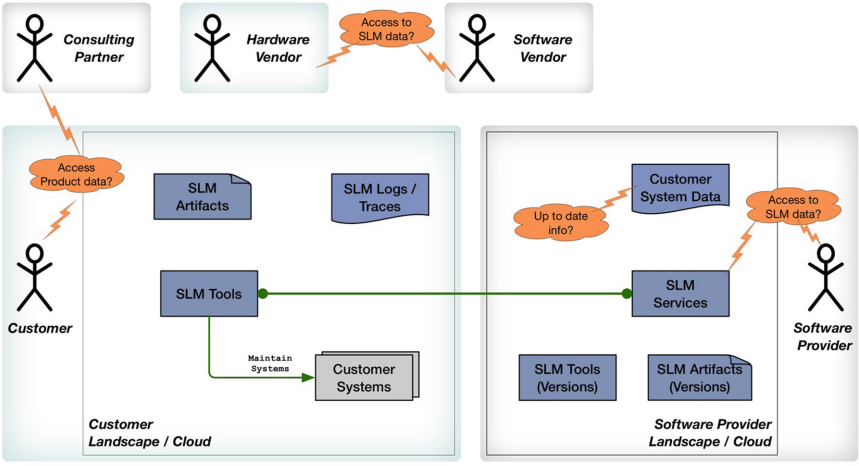

Fig. 4. SLM process: data Visibility

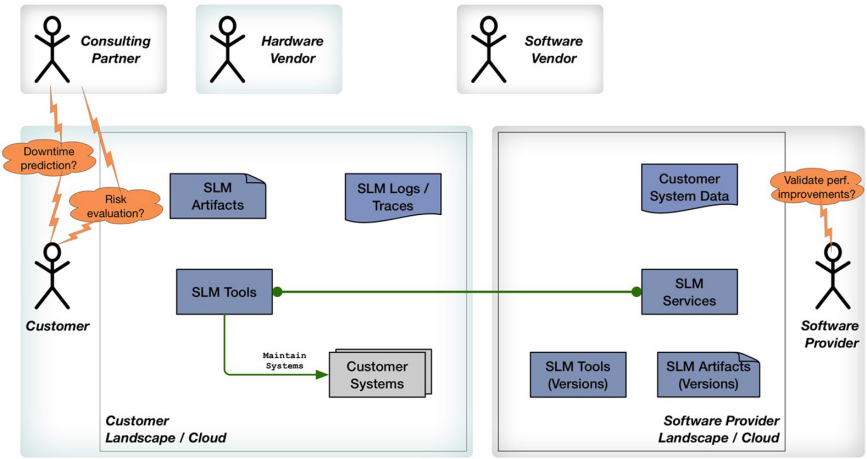

Fig. 5. SLM process: resource prediction, risk estimation

3 DLT-based System for Managing Complex SLM Processes

In this section we propose a DLT-based system for solving most of the typical SLM issues. Our approach is presented on Fig. 6. On that figure are shown the main players and network nodes of the proposed DLT-based system share. Real time data is shared between involved parties during the SLM procedures. That can be used for keeping records for further analysis and as a proof of origin for every artefact involved in the process. As it is depicted, each party have a DLT node keeping a copy of all the information. In the proposed setup the software provider is creating the DLT landscape and corresponding nodes are also created according to the number of involved partners. In such a way each partner is authorized to have its node to join the DLT. Thus, a trusted environment is established between parties, so that communication flow is configured in secure manner and every party can either see all the info or only the info it is authorized to see. Such setup covers the requirement for data visibility as each party have immediate access to the data shared by other parties as part of the established procedure. Sharing SLM data with DLT has a consensus mechanism in place to agree on the veracity of the data. All information once written in the DLT can no longer be deleted or modified by any party or parties. In addition, each participant in the process have its own copy of the data that can be used long after the collaboration during the SLM procedure is finished. There are no further dependencies in time so the collected data can be used in the future independently by each party for accessing the information it has access to. Also, the DLT architecture allows to keep track of historical information, that can later be used for estimations about resource usage and downtime predictions for similar procedures.

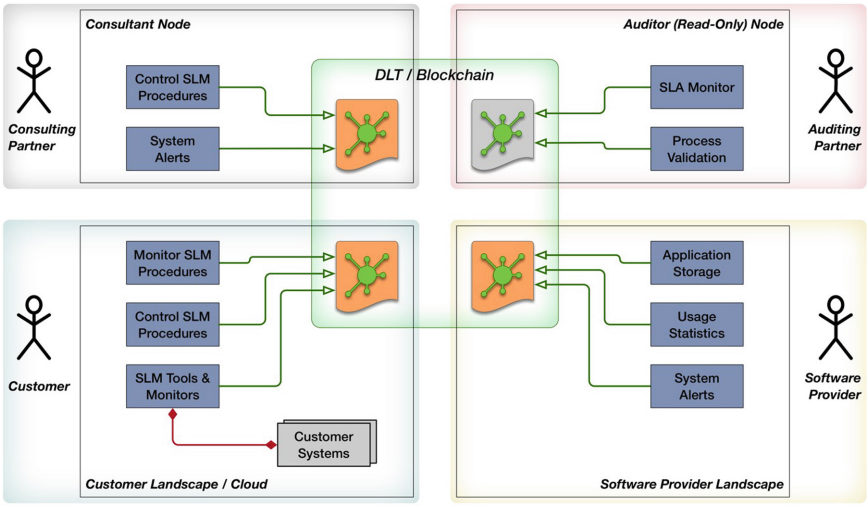

Fig. 6. Managing SLM processes using DLT

One key decision for having the right architecture and optimized setup is choosing the proper type of a DLT. As we have pointed out in the introduction there are many different properties of the DLTs that could bring advantages and disadvantages to the overall proposed solution. There are generic requirements for every DLT such as performance, stability and security. Participants are always known so public DLTs are not suitable as they focus on fully trustless environment. Enterprise blockchains/DLTs are better fit due to their simplified approach for reaching consensus. There is no need for PoW or similar complex consensus mechanism since the overall setup is controlled. Cryptocurrency support is not an essential feature for described SLM processes. Anyway, cryptocurrency may be a useful addition in some cases. For example, downloading SLM artifacts or consulting efforts could be paid automatically as a part of an established smart contract. This way using an agreed-upon SLM smart contracts could simplify and automate some payments between involved parties. The software provider could establish a DLT oracle that provides a service for delivering executables and other artifacts automatically on request. Consulting efforts could also be a paid service, provided by the selected consultants.

As we demonstrate on Fig. 1 complex ecosystem of various permissioned DLT offerings with their advantages and disadvantages. Specifics of the SLM processes are also different and vary depending on the environment they operate and the main purpose that the DLT-based system is built to solve. Some of the main requirements for a DLT used in SLM context are overall performance and scalability, security, simplicity to use, flexibility, maintainability and support for smart contracts and DLT oracles.

We have chosen for our research three of the most popular open source DLTs that are available - Hyperledger Fabric, MultiChain and R3 Corda. Below, we

shall evaluate them based on their suitability for serving SLM procedures in different contexts.

Hyperledger Fabric. Hyperledger Fabric is part of the Hyperledger open source community. It was contributed by IBM and was built specifically targeting enterprise use. As such it does not have its own cryptocurrency. The smart contracts are called chaincode in Fabric. Using chaincode is the only way to access and modify data in Hyperledger Fabric's blockchains. Supported languages include Golang, Node.js and Java. It has concepts for Ledger, State DB and Side DB. Accessing the Hyperledger Fabric's Ledger for example gives access to all transactions that happened in the past regardless of the current state of the objects involved (such as "deleted" ones). The State DB is rebuilt for each node based on the Ledger info and it contains only the current state of the objects and does not include objects that are set as "deleted". Side DB is a storage for off-chain data that still may be transmitted and validated by Hyperledger Fabric, but it is not part of the its blockchain structures. This is useful when big amounts of data or sensitive personal data has to be transferred securely between involved parties. Such data should not be set as part of the blockchain due to its size or to be able to fulfil legal requirements such as GDPR's "right-to-be-forgotten". The concept for Hyperledger Fabric Channels allows fine granular configuration of processes. Technically each channel is a separate blockchain with own chaincode and users. Parties that have access to the channel see all the information that is part of this channel. In order to restrict the visibility of certain info additional channels could be used. The consensus protocols for Hyperledger fabric are highly customizable by using the endorsement, ordering, validation phases for adding info in the Fabric. To each node/party may be assigned one or more of the consensus functions. Custom policies may be set to cover customized scenarios with complex consensus rules. For example: In a three organization process it may require nodes of at least 3 different organization to validate/endorse certain critical transactions.

The flexibility of Hyperledger Fabric is an advantage for complex SLM scenarios. Its ability to add whole organizations as participant and its LDAP support allows easy integration in different companies. As Hyperledger Fabric is a blockchain each node/user has a copy of and access to all the information that is relevant for it. This makes it a good fit for guaranteeing system consistency and security. It provides a good medium for sharing information between involved parties and also common store of information that could be used for collecting historical data for past SLM procedures. Even if Hyperledger Fabric is still considered a blockchain it has several performance improvements that make it suitable for handling bigger loads compared to Multichain. Often the performance of the system may be improved considerably by properly configuring its smart contracts and policies. Its main advantage is its flexibility and adaptability to different and changing processes. The main reason for such flexibility is its configurable model for smart contracts and policies. In addition, it is designed to easily integrate with already established enterprise systems.

MultiChain. Multichain is another open source permissioned blockchain. It is based on the original coding of the Bitcoin blockchain. Its consensus mechanism is a configurable hybrid between classical Proof-of-Work (without the reward for block producing) and simple round robin. This is achieved by setting a limitation for producing new block for each node that had already produced a block in the near past. MultiChain has built-in mechanisms for permissions, assets and streams. Instead it supports smart filters programmed on JavaScript. In short: MultiChain is easy to setup and automate but it is not very customizable and flexible compared to the other permissioned blockchains. One of the key differentiators of Multichain compared to the other permissioned DLTs are its built-in mechanisms for permissions, assets and streams. This allows for easier/automatic validation of Multichain installation compared to the systems that are driven by smart contracts and complex policies. The drawback of Multichain is its restricted flexibility that may be compensated by more complex accompanying applications. Being a classical blockchain it has the natural scalability issues that could make it unfit for a heavy-load use such as SLM for IoT devices. In short MultiChain is a good fit for automating simple multi-party SLM processes that do not require smart contracts.

R3 Corda. Corda is a non-blockchain DLT built by R3 Consortium for use primary in financial institutions. Its architecture naturally supports sharding for scalability and its performance is greatly improved compared to any classical blockchain. Unlike the Hyperledger Fabric and Multichain the consensus for Corda is reached on transaction level (there are no blocks) and always involve interaction with one or more nodes of the so-called Notary cluster. In Corda transactions involve at least one notary. Each Corda transaction is by default visible only to the involved parties, and there is possibility on request the full history of funds to be tracked. The transaction info may safely and verifiably be shared explicitly with other entities. This consensus approach is perceived as more secure as it does not "leak" any extra info and allows for very fine-grain tweaks for each process. The Corda smart contracts are typically written in Kotlin or Java. By design the Corda smart contracts (as well as those of Hyperledger Fabric) are not guaranteed to be deterministic. As such it is responsibility of the smart contract developer to write non-probabilistic algorithms to avoid complicated attacks on the system. To improve SLM processes Corda may offer the best performance under stress, and configurable consensus based on Notary Cluster. Its security model on data visibility is the most complicated and as such is much harder to be tested, analyzed and proven to be flowless. Considering the performance aspect, when properly configured Corda may be the best choice for DLT supporting complex multi-party system for SLM processes operating under heavy load.

4 Conclusion

Our research shows that by using permissioned DLTs a setup of multiple nodes can be constructed in such a way that it will solve some typical SLM issues in

multi-party project environment. The challenges that such environment overcomes are security and consistency of tools and systems, data sharing and also collecting historical data. It can also improve estimation for resource usage and downtime prediction during the planning of similar procedures. In addition, smart contracts could be used to model interactions between parties and automate certain steps of the process such as payment for certain services. It could also automatically cover financial costs related to not fulfilling certain service level agreements (SLAs).

In this paper we show how distributed systems based on DLT can be used to automate and optimize the execution of multi-party SLM processes. All the three DLTs that are analyzed in this article are fulfilling the requirements for a complex SLM procedures. The choice of a proper DLT depends on the environment and the challenges that have to be solved.

Acknowledgements. This research was partially supported by the National Scienti c Program "Information and Communication Technologies for a Single Digital Market in Science, Education and Security (ICTinSES)", financed by the Ministry of Education and Science, Bulgaria.

References

1. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System (2008). https://bitcoin.org/bitcoin.pdf
2. Morrell, B.: How to Strengthen Product Lifecycle Management using Blockchain (2017). https://www.ey.com/en-gl/life-sciences/how-to-strengthen-product-life-cycle-management-using-blockchai
3. Bee, R.: Asset Management: 3 ways Blockchain will Enhance your Asset Management Efforts (2018). https://www.ibm.com/blogs/internet-of-things/iot-blockchain-enhances-asset-management/
4. Boudguida, A.: Towards better availability and accountability for IoT updates by means of a blockchain. In: IEEE European Symposium on Security and Privacy (2017)
5. Etwaru, R.: Blockchain: Trust Companies. Dog Ear Publishing, Indianopolis (2017)
6. Mougayar, W.: The Business Blockchain. Wiley, New Jersey (2016)
7. Tatro, P.: Blockchain Unchained. Book Councelor, USA (2018)
8. Brown, R.: The Corda Platform (2018). https://www.corda.net/content/corda-platform-whitepaper.pdf
9. Popov, S.: The Tangle (2015). https://www.iota.org/research/academic-papers
10. LeMahieu, C.: Nano: A Feeless Distributed Cryptocurrency Network (2018). https://nano.org/en/whitepaper
11. Greenspan, G.: MultiChain Private Blockchain? White Paper (2015). http://www.multichain.com/white-paper/
12. Larimer, D.: EOS.IO Technical White Paper (2017). https://eoscollective.org/papers
13. Buterin, V.: A Next-Generation Smart Contract and Decentralized Application Platform (2013). https://github.com/ethereum/wiki/wiki/White-Paper