17th International Workshop on Algebraic and Combinatorial Coding Theory,
October 11-17, 2020, online, Bulgaria, pp. 145-149

2020 Algebraic and Combinatorial Coding Theory (ACCT) | 978-1-6654-0287-3/20/$31.00 ©2020 IEEE | DOI: 10.1109/ACCT51235.2020.9383401

Modern Software Lifecycle Management leveraging the power of Blockchain

Biser Tsvetkov
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
Sofia, Bulgaria
biser@math.bas.bg

Jivko Jeliazkov
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
Sofia, Bulgaria
jivko@math.bas.bg

Hristo Kostadinov
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
Sofia, Bulgaria
hristo@math.bas.bg

***Abstract*—Many distributed ledger technologies (DLT) offer smart contract platforms, that are already used in various industries such as healthcare, transportation, government, banking, and insurance. Software Lifecycle Management (SLM) is another area where DLT and smart contract platforms could play an important role in the future. SLM processes are covering the initial setup, technical and business configuration and ongoing maintenance of software systems that often consist of many cloud services, on-premise servers, and/or edge devices, spread on many geographical locations. In SLM there are many participants that should communicate each other to achieve common goal, for example the software provisioning. In case something wrong happens in their communication, it is very hard to prove what is the reason and which party failed to deliver, since the environment is complex and the protocols of responsibilities are not bound to the software dependencies. The purpose of this paper is to demonstrate how DLT-based smart contract platforms supports SLM processes for complex business systems where various participants are involved.**

***Index Terms*—blockchain, distributed ledger technologies, software lifecycle management**

I. Introduction

In the last decade many researchers work in the area of Bitcoin [1] and Blockchain technologies [2].

Software Lifecycle Management is the area of installation, configuration and maintenance of software systems. These systems are servers, cloud services and/or edge devices, distributed on many locations and often have complex dependencies between them. High availability, disaster recovery, and other requirements are often requested by the customer adding this way to the complexity of the overall solution and to the complexity of the SLM processes. Important aspect of SLM is the way how participants are exchanging information between them and how is guaranteed that this information is accurate in terms of security, reliability and trust.

One challenge in SLM is the data visibility – who modifies what, who should deliver something by when, who have touched the binaries and the log files. Most important is whether everyone has delivered his part on time and quality.

All this challenges are by design included in many blockchains and DLTs as they offer features such as smart contract (SC) support, Ricardian contracts, distributed architecture, transaction finalization, high availability and disaster recovery [3]–[5]. All these features could be combined to resolve many of the SLM challenges, by applying innovative product architecture and design.

Our research shows that providers of software products and services, and also all involved parties in a typical software project, could benefit from Blockchain based architecture:

First: Establishing secure communications between non-fully trusted participants for the SLM processes and reducing the risks from malicious code reaching customer sites and their productive systems. **Second**: It allows sharing of SLM resources and services between involved parties thus greatly improving the overall interaction costs required for complex multi-party procedures. **Third**: The proposed system allows better time and resource prediction for upcoming SLM procedures by keeping historical data from previous runs of the same or similar procedures. In some cases payments for new software or financial penalties for unplanned downtimes or other SLA infringements could be executed automatically.

In the next section we present the challenges in the area of complex SLM procedures and describe how using DLT solves some SLM issues in the areas of data sharing, security and improved prediction. The concrete DLT selection and prove of concept is depicted in Section 3. Section 4 contains conclusion remarks.

II. Challenges of complex SLM procedures

In a very simple case, the customer's IT department have full access to their on-premise system and executes given SLM procedure by themselves. When it comes to real business software, unfortunately, the SLM scenarios are quite complex. One

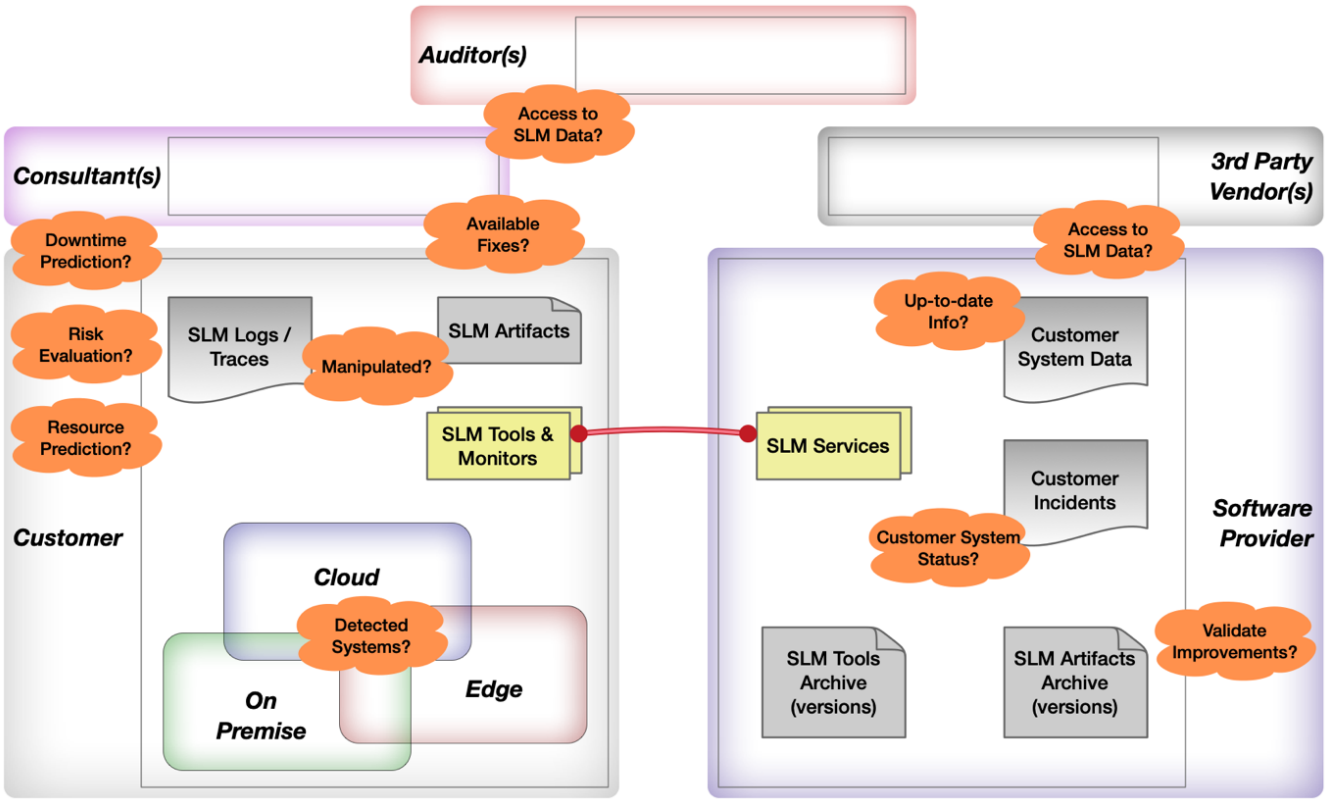

Fig. 1. Parties involved in a complex SLM procedure

thing that contributes to the complexity is the heterogenous environment, as it can be a mix on-premise systems, cloud services and IoT edge devices.

Another factor adding to the complexity is the number of involved parties. SLM participants ahave specific tasks and responsibilities as shown on Fig. 1. The **Software Provider** is the IT company that has developed the software and is offering ongoing support for its customers. The provider delivers software artifacts for the SLM procedures such as installation files, system upgrades, updates, patches, tools for configuration and software transports, and other. Responsibilities of the Software Provider include offering enterprise-level support for their products based on strict KPIs for handling each customer issue.

Typically **Auditing authority** are assigned to the project in government sector. Although **auditor(s)** do not contribute directly to the technical implementation, they are crucial participant in the project and need reliable information about status and resources as well as upcoming deadlines.

Technical consultants are developing the project on behalf of customer and are very important participant since apart of the very implementation they do the infrastructure setup and functional correctness tests.

Third party software and hardware vendors have a key role in the SLM processes as each complex system has strong dependencies on its software and hardware environment. Compatibility testing of the 3rd party hardware and software is an essential part of any SLM procedure since operating system, databases, server and IoT hardware add to the complexity of the SLM procedures.

All this is done in favour of the end **customers** who also arrange financial background of the SLM procedures and benefits from the provided software. Customers are the participants who use productively delivered software and as such they define the important key performance indicators (KPI) for the SLM procedure and the system maintained. The exact downtime window, availability requirements (high availability and/or disaster recovery), the access to company resources and the final decision based on the risks and benefits for running each SLM procedure are ultimately decided by the customer.

In the described hybrid landscape and multi-party project environment there are several types of challenges that are part of these complex SLM projects.

System Consistency and Security: Malicious manipulation of key system artifacts such as important executables, system configurations, logs and traces are always at risk. This is why the validated version and patch level detection of system components is important aspect of overall system security. Hacking the indication that a certain critical patch or fix has already been applied to the customer productive systems is one possible attack on SLM data. If successful it will leave the system unpatched for a long period and exposed to known malicious attacks. Simple and reliable method of verifying

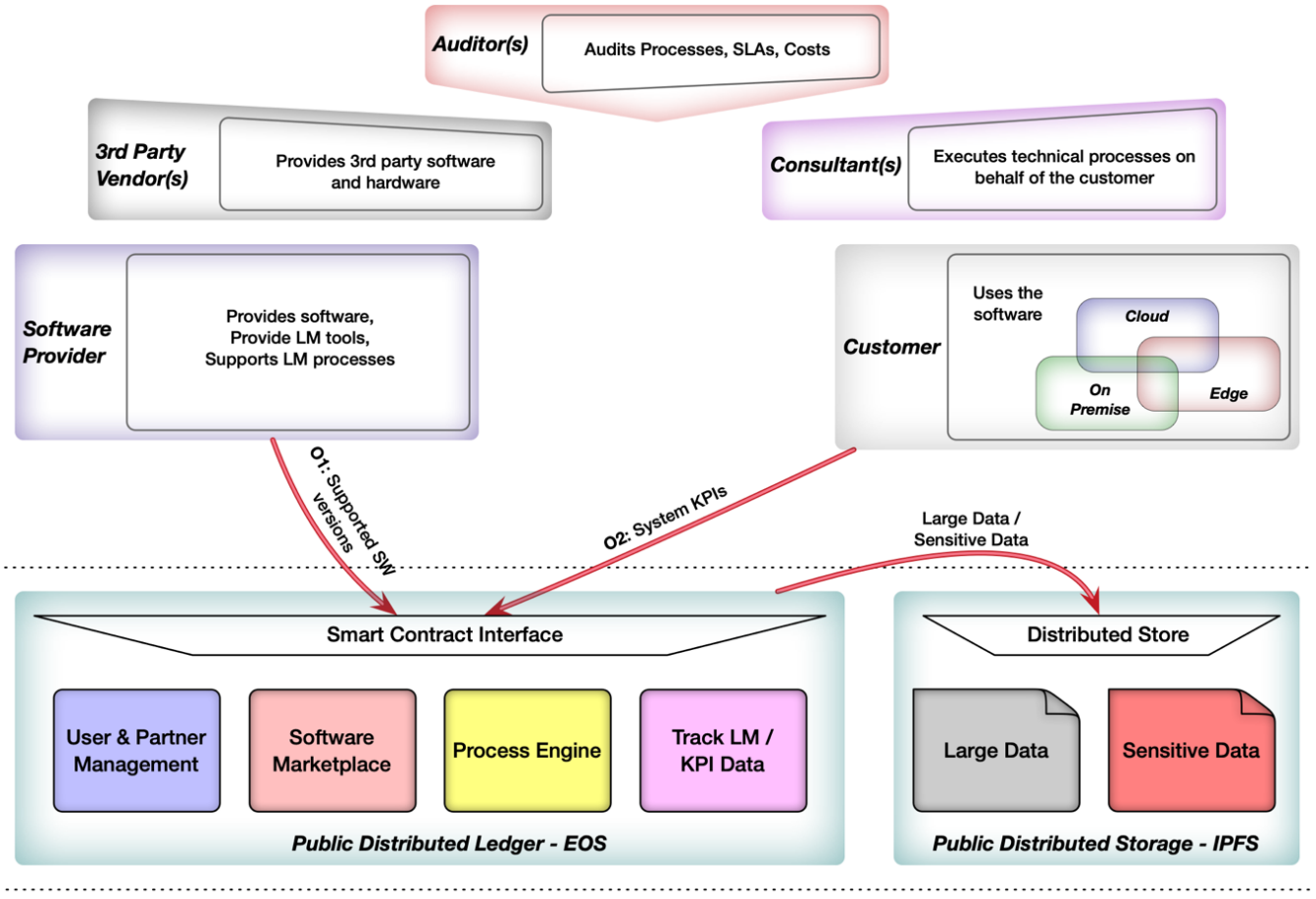

Fig. 2. DLT-based SLM System based on EOS and IPFS

system integrity to avoid such attacks is essential.

Data Visibility: In a typical complex SLM process involved parties have access only to the pieces of the overall information/data that is related directly to their contribution. For that reason, the "big picture" is not transparent and available to all key decision makers. For example, the exact component and product versions are used by specific customer are not always known by their software provider.

Historical information for better prediction: Having access to the data of SLM procedures executed by hundred of customers gives possibility for prediction of important aspects of given procedure like downtime, risk analysis, participant involvement and system load.

Responsibilities and SLAs: When the system is running every milestone and fulfilment of a given SLA could be easily tracked and penalties could be automatically triggered as a compensation to affected participant.

The four aspects mentioned above will be addressed by using a SLM systems based on a DLT.

III. Architecture of a DLT-based SLM system and concrete use case

Now, knowing the main issues in a real business environment, we have designed a DLT-based system that addresses all of the mentioned issues. More concrete we choose to base our solution on EOSIO technology and use the Inter-Planetary File System (IPFS) as distributed storage for the large or sensitive data. The choice of EOSIO as a platform for running smart contract is based on the features it supports, its performance, popularity and flexible price model. Some functional requirements considered are flexible support for smart contracts and Ricardian contracts, support for Oracles and proper security features. Some gaps of pure EOSIO-based solution are filled by using IPFS for storage of large or sensitive data. Each relevant file stored in IPFS is referenced by the data of smart contracts of one or more dapps running on EOS Mainnet. As usual the location of the file is its IPFS hash value. The SLM system consists of an EOSIO dapp with several related smart contracts. The overall setup and involved parties are shown on Fig. 2.

To make any changes to the information DLT requires that the participant has to be properly identified and authorized. Thus, on every step, it is known who is doing what and all artefacts could be tracked to the very participant. Furthermore, it is easy to decide on what phase to move the responsibility from one participant to another. Another security-related case is when there is a critical fix for specific version of an installed

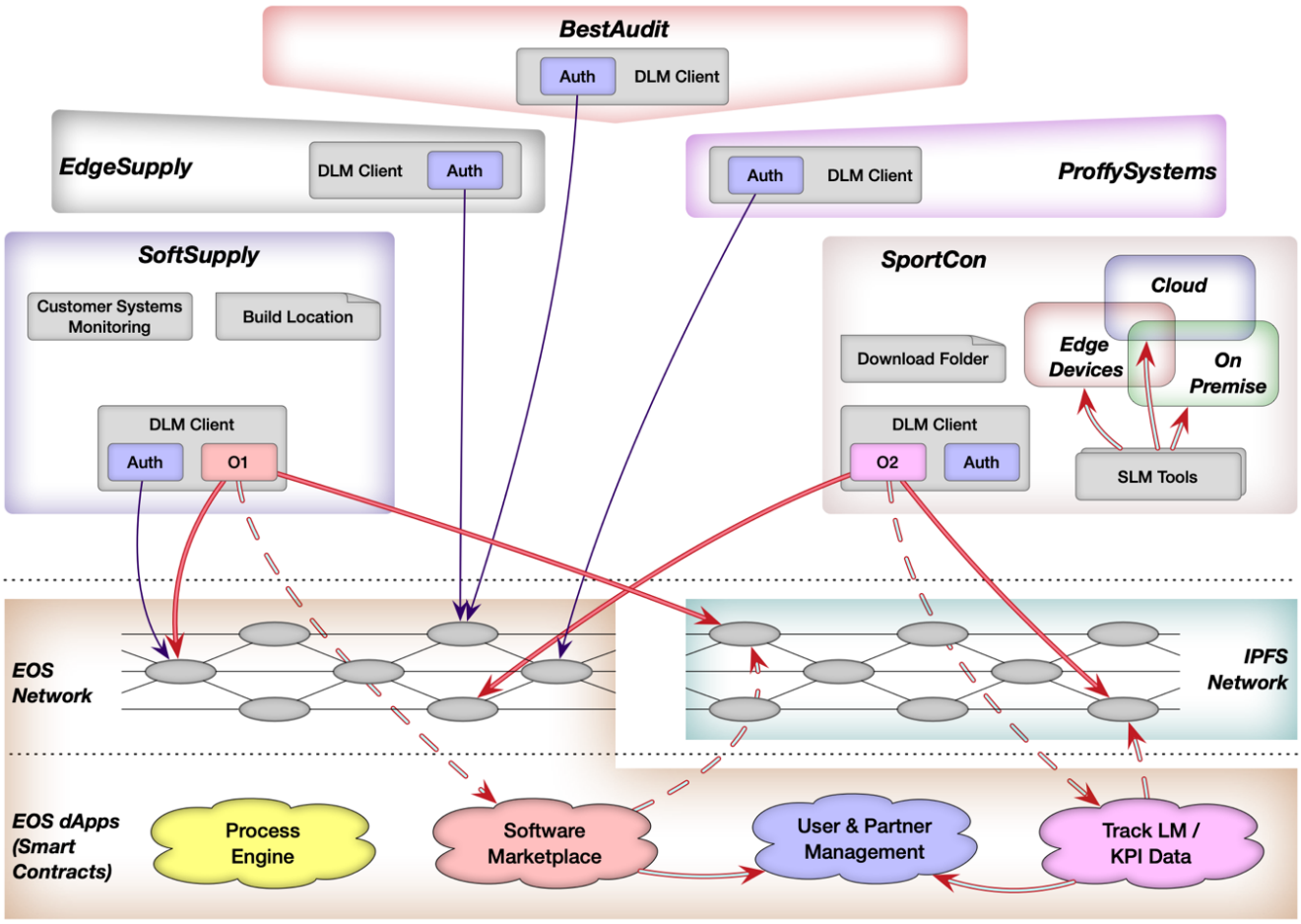

Fig. 3. Sample case study for using EOS and IPFS-based SLM system

component. The software provider may contact initially only the directly affected customers to offer a fix before going public and getting potentially unwanted attention. Sharing unencrypted data is straight forward with this architecture. There are popular security methods (such as zero knowledge proofs) for the cases where encrypted data has to be shared or a statement to be proved to a third party providers. Last but not least processing of anonymized information is ensured by the provided architecture and could be used for reliable downtime, resource and risk prediction based on processing of real data. This is especially important when personal data is involved. Using zero knowledge proof elements, integrated in the smart contracts that process collected data, is one natural approach available to modern Blockchain/DLT platforms.

Some of the key types of smart contracts of the proposed dapp are User Management module, a simple Process Engine, a smart contract providing Secure Communication, Software Marketplace and a module containing data from customer for Tracking LM/KPI Data. The User Management smart contract keeps track of relationship between different parties, roles and processes. It is used to validate authorization for EOSIO accounts to trigger certain changes of the state. The Process Engine keeps track of processing and data related to specific processes. It may create new processes and change process statuses and user roles during process execution. It is also responsible for automatic payments as result of manual interaction or triggered by a scheduler for reaching a deadline. When communication or notifications between involved parties has to be kept encrypted, they use the Secure Communication functionality. The Software Marketplace smart contract stores information for available SLM artifacts and versions. It is updated by oracles running at Software Provider and are used for automatic validation of artifacts installed at customer and checks for fixes and upgrade recommendations. Finally, any data from customers' systems is collected by the Track Data smart contract either by manual operations or by established oracles running at customer's site. The potentially confidential data itself is eventually encrypted and uploaded to IPFS. Keys to decrypt data in each file are generated and saved to the info in Track Data for each party.

As stated, the proposed solution addresses the main challenges of SLM and in order to demonstrate that lets consider a simple case study where we could trace in concrete steps the interaction between specific participants. The company

SportCon is selling tickets for sport events and also branded merchandise items via several channels in the thematic shops, online orders or using special hardware devices located in the city. Their productive landscape consists of some cloud-based services, some critical systems running on premise and their ticket and merchandise selling machines as IoT edge devices. The edge device hardware is provided by the *EdgeSupply* company. The generic software for the ticket distribution is provided by *SoftSupply*. The implementation project of *SoftSupply* for *SportCon* is responsibility of the consulting company *ProffySystems*. *SoftSupply* tries to offer the most competitive software on the market and they are able to track the key parameters of the real use of their customers systems. They also need a fast, reliable and secure channel to make the new versions of their products available to their customers. In case of future conflicts, they also need to be able to prove what binaries they are providing to their customers and at what time. The customer *SportCon* aims to provide great user experience for the people purchasing various types of tickets via their system. They want the systems to work reliable and secure. The technical consultant *ProffySystems* uses the tools and archives provided by *SoftSupply* and need an auditable proof that they use only genuine artefacts for installations and updates at *SportCon*. The case study diagram is shown on Fig. 3.

As part of the installation at *SportCon*, several agents report the system's KPIs to the smart contract platform via oracle **O2** interface. *SoftSupply* finds security-related issues and identifies in which of the delivered components they are located. Of course, *SoftSupply* inds security-related issues and identifies in which of the delivered components they are located based on already gathered information. Of course, *SoftSupply* immediately produces a new version, which contains fixes for the detected security issues. The new version of the patch passes all security and functional tests and is ready to be delivered to customers. In order the delivery to be done in secure and reliable manner, the very upload is done in the IPFS, hash code is stored in the blockchain and passed into notification to interested customers. All needed interaction with the blockchain in regards to software catalog are done via **O1** oracle. There is no need of duplication of big files in the blockchain since the hash code is secure reference to the IPFS and gives all needed security to the end customer.

Clients receive all relevant notifications via locally installed Distributed LM.

SportCon in coordination with *ProffySystems* review all important financial and IT aspects and decide on applying patch. The very review could involve the download from IPFS and extensive tests in *SportCon* test infrastructure. Part of the tests will give better prediction for the impact of applying patch for example downtime of the system, price, security aspects and others.

When patch is applied the notification about the new version, installation time, and others goes via **O2** oracle. Selected data could be encrypted, but can be anonymously accessed via ZKP.

This concept is covering the typical interactions in SLM, but the architecture is open and easily additional functionality could be added like machine learning or automated financial operations in favour of affected party if important KPIs are not kept.

IV. Conclusion

Our paper shows that smart contract platforms we can used for design and implementation of a distributed system, leveraging modern DLTs to resolve typical SLM issues in a multi-party no-trust project environment. The challenges that such environment overcomes are - security and consistency of tools and systems, data sharing and historical data collection. By combining ZKP techniques and Artificial Intelligence elements in the Smart contracts we can predict resource usage and downtime duration. In addition, smart contracts platforms are used to model and track all interactions between parties and automate certain steps of the process such as payment for certain services or penalties for SLA violations as the system keeps track of all responsibilities that are modelled. As result of our research we are convinced that in the future SLM will rely more and more on DLT-based smart contract platforms.

Acknowledgment

The first and second authors were partially supported by the National Science Fund of Bulgaria under Grant KP-06-N32/2-2019. The third author was supported by the National Scientific Program "Information and Communication Technologies for a Single Digital Market in Science, Education and Security (ICTinSES)", financed by the Bulgarian Ministry of Education and Science.

References

[1] Y. Borissov, M. Markov, "An Approach to Computing the Number of Points on Elliptic Curve $y^2 = x^3 + a(modp)$, via Explicit Formula for That Number Modulo p", Proceedings of the Ninth IWSDA, Dongguan China, 19-24 Oct. 2019.
[2] B. Tsvetkov, H. Kostadinov, "DLT smart platforms for software lifecycle management", AIP Conference Proceedings, 2116, 2019
[3] I. Bashir, "Mastering Blockchain," Packt Publishing, 2018.
[4] R. Etwaru, "Blockchain: Trust Companies," Dog Ear Publishing, 2017.
[5] M. Corrales, M. Fenwick, and H. Haapio "Legal Tech, Smart Contracts and Blockchain," Springer, 2019.