Designing Process-Centric Blockchain-Based Architectures: A Case Study in e-voting as a Service

Emanuele Bellini[1,2(✉)], Paolo Ceravolo[3], Alessandro Bellini[1], and Ernesto Damiani[2,3]

[1] Mathema s.r.l., 50142 Florence, Italy
{Emanuele.bellini,abel}@mathema.com
[2] Center of Cyber Physical Systems, Khalifa University of Science, Technology and Research, Abu Dhabi 127788, UAE
ernesto.damiani@ku.ac.ae
[3] SesarLAB, University of Milan, Milan, Italy
paolo.ceravolo@unimi.it

Abstract. This article aims at introducing a new process-centric, trusted, configurable and multipurpose electronic voting service based on the blockchain infrastructure. The objective is to design an e-voting service using blockchain able to automatically translate service configuration defined by the end-user into a cloud-based deployable bundle, automating business logic definition, blockchain configuration, and cloud service provider selection. The architecture includes process mining by design in order to optimize process performance and configuration. The article depicts all the components of the architecture and discusses the impact of the proposed solution.

Keywords: Blockchain · Trust e-Vote · Configurability · Automation · Mirror model

1 Introduction

Processes span organizational boundaries, linking together heterogenous information systems. Business Process Management (BPM) has been largely earmarked to address the integration of inter-organizational processes. Coordination and interface of independent systems have been one of the key issues for years [59, 60] but it has been argued that technical integration is simply a precondition of organizational integration that requires trust and comprehension among parties [64]. Recently, the emerging blockchain technology has been acknowledged as the right solution for complementing BPM with support for trustworthy execution of business processes [61, 62]. Putting into operation BPM and Blockchain requires to engage with specific challenges. Following [62] we underline the need for (i) *developing a diverse set of execution and monitoring systems on blockchain*; (ii) *devising new methods for analysis and engineering business processes based on blockchain technology*; (iii) *defining appropriate methods for evolution and adaptation*. All these challenges call for an evolution of the

Published by Springer Nature Switzerland AG 2020
P. Ceravolo et al. (Eds.): SIMPDA 2018/2019, LNBIP 379, pp. 1–23, 2020.
https://doi.org/10.1007/978-3-030-46633-6_1

technological framework where Blockchain is specified and deployed. As part of this comprehensive view, this paper proposes a vertical study focused on an e-voting as-a-service application.

Different organizations, at different levels, have to organize elections in compliance with specific legal frameworks or standards. Even though we live today in a digital world, voting, in many and diverse contexts, continues to be implemented using the traditional process based on ballot boxes and manual verification. The reason behind the longevity of the traditional process is not without good reasons. It relates to the low transparency and the high vulnerability of distributed computer networks, as testified by the failure of e-voting in Estonia and Australia [7]. Even if its applicability for political pools remains an open challenge, there are a number of cases in daily life in which electronic vote may be successfully applied. Examples include the company's executive elections, where each shareholder is entitled to one vote per share multiplied by the number of executives to be elected. The rule for such voting system may be so rigid ending up to reduce their inclusiveness in some cases (the voting task is performed with a reduced number of voters available) or to affect the operativity of the organization in case of prolonged delay waiting that all the conditions for participation are satisfied. In this respect, electronic voting systems start to be attractive because they can significantly reduce the costs of implementation and verification as well as ease and incentivize the participation of voters thanks to its ubiquity. In recent years, the process of transformation has accelerated, and businesses have entered into the cloud computing era, where virtually turning anything into an online service (XaaS) has become a distinct possibility [25]. In this paper, we introduce HyperVote, a blockchain-based e-voting as a service in the framework of a scalable business model grounding the solution on three pillars: **configurability, automation, transiency.**

HyperVote is built on a Process Centric Architecture (PCA) "*where the entire IT system is conceptualized and centrally organized by the concept of the business process that is supported by the system and the business process component is the central component in the system*" [56]. It is able to support dynamic selection, deployment, and execution of an e-voting process whose requirements are tailored to the user needs. Thus, organizations of any dimension will be able to set-up a secure and transparent voting service using a pay-per-use configurable approach bringing on costs during the limited timeframe of an election. As regards of the service provider, this approach allows optimizing the allocation of resources as the same infrastructure can serve different customers in different time frames.

Electronic voting systems have been fraught with security concerns and controversy. In fact, a voting system has to guarantee three fundamentals requirements: **robustness**, **uniqueness** of the vote, and **transparency**. In this context, the anonymity that is commonly considered fundamental is here treated as an option in order to include all those cased in which the vote should be evident. Despite several technology and protocols was explicitly designed to meet the three of them simultaneously, this goal is difficult to be achieved in current e-voting systems. In fact, electronic vote is exposed to attacks such as double voting, Sybil attacks, and similar that affect its trustworthiness slowing down its adoption.

The three fundamental requirements of voting are however supported by an emerging technology that more and more observers expect will become a standard in the next future: the Blockchain. In general terms, a Blockchain is an immutable transaction ledger, maintained within a distributed network of peer nodes. Each node is requested to maintain a copy of the ledger by applying transactions that have been validated by a consensus protocol. This model offers great advantages in terms of transparency authenticity, nonrepudiation but imposes severe constraints in terms of performances especially when the identity of a participant must be assessed in conformance to regulations. Thanks to these features, Blockchain technologies are successfully adopted in very diverse domains [57–60]. In this respect, it represents a suitable tool for supporting e-voting, as evidenced by the multiplication of projects focusing on this integration. None of these projects has, however, demonstrated to scale-up, receiving the appropriate acceptance in the market. In fact, there is a large number of examples in which Blockchain is implemented as a static and monolithic technological solution. Startups and ICOs tend to shape their business around one specific solution considered suitable at that time.

This strong dependency between the business case addressed and the type of implementation of the Blockchain exposes those initiatives to a high risk of being run out from business very quickly because of the rapid technology obsolescence that usually characterizes emerging sector as the Distributed Ledger Technology domain. This is also particularly evident when you design and dimension your private Blockchain infrastructure. Parameters as numbers of peers, HW/SW requirements and maintenance, computational performance, and so forth, that have an impact on the sustainability a project in a long run, tend to be neglected in favor of quick delivery of the solution to market. The result is that 92% of all projects fail. Since inception roughly ten years ago, the Blockchain industry has witnessed the launch of 80,000 projects, according to the China Academy of Information and Communications Technology (CAICT). Of them, only 8% are survived while the remaining 92% failed with an average lifespan of 1.22 years [53].

This risk is here mitigated by the adoption of the *Process Thinking* perspective [56], which put the process at the core of the enterprise business process. This is a shift from the convectional *Functions Thinking* to the new process thinking, where functions are enablers to process and process performance and its optimization is more important than optimizing individual functions [56]. Thus in HyperVote, we do not consider Blockchain as a static installation but a function within the Hyper Vote Enterprise Business process that can be dynamically deployed and instantiated according to the service level requested by the end-users for a voting stage. Then, the Blockchain infrastructure is used to execute the HyperVote process in a trustworthy condition while parameters such as number of nodes, computational capacity and service availability offered by the cloud where the node runs, data preservation and so forth, become a matter of service (*process configurability*). This process needs to be deployed in an automatic way, negotiating, each time, the best price offered by the cloud service where the blockchain and the related smart contracts will be deployed (*automation*). Moreover, there is no reason to maintain historical data over its natural lifecycle that is

usually defined by the user or by the law. In HyperVote, even if the service is provided using blockchain, the data managed with the blockchain should be decommissioned at some point. In fact, the costs of data preservation should not be sustained by the service provider in a long run, thus the duration of data retention will be defined by the user and its costs computed in the final bill.

The user can decide to keep the blockchain up and running for further analysis and verification after the end of the voting process. This period named *data retention period* (DRP) is part of the service configuration capability of HyperVote whose costs are opportunely computed in the final bill.

In order to manage such a case, HyperVote adopted the strategy to set up a different Hyperledger instantiation for every voting service provided instead of having a single Hyperledger network where multiple services are executed. In this way, as soon as the data retention period is expired, it is possible to automatically turn-off the entire services and concluding the contract with the cloud provider without affecting the other running services. The main aim is to keep under control all the operational costs of the service.

The article is organized as follow: in Sect. 2, a review of the current initiatives on e-voting based on blockchain is presented, in Sect. 3 the end to end verifiability and trust of the e-voting service has been introduced according to the mirror model [41]; Sect. 4 is devoted to defining the service requirements while Sect. 4 provides the reference architecture of HyperVote, Sect. 5 is dedicated to the evaluation and conclusions are provided in Sect. 6.

2 Related Work

There is a vast literature focused on properties of electronic vote protocols [34] and there are a number of functional and security requirements for a robust e-voting service that include transparency, accuracy, auditability, system and data integrity, secrecy/privacy, availability, and distribution of authority presented in [4–6, 13]. These properties are often expressed in terms of formal languages that could be mapped on a technological solution, but this mapping is seldom or only partially provided.

Another mapping that is often neglected is with BPs, meaning real-world operational or normed activities and procedures. The literature focusing on exploiting Blockchain for the electronic vote is a typical example of that. Relevant properties of voting protocols are matched but the integration of these properties with external services, that may be implied by specific organizational or legal constraints, typically is not addressed. We claim that the realization of a fully configurable electronic vote implies the definition of a platform-as-a-service environment addressing the properties of a protocol a three abstraction levels (see Fig. 1):

- Business Process, defining the orchestration of autonomous services;
- Business Logic, defining the behavior of each single service;
- Transaction data, defining the properties transactions have to guarantee.

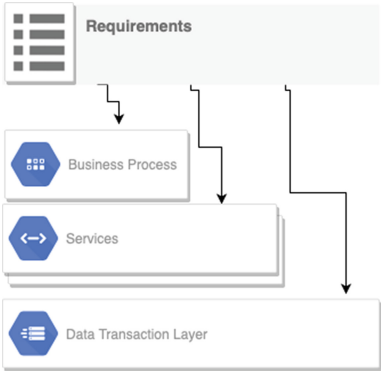

Fig. 1. Logic dependability of the architecture

Several initiatives at commercial level exist as Bitcongress [2], Followmyvote [3], Agora-Voting [43], VoteWarcher, E-VOX [42], and TIVI [4].

Agorà [43] is an end-to-end blockchain-based solution designed for public organizations according to a B2G view. The system is token-based, so that organizations should purchase these tokens for each individual eligible voter. The community of node operators is organized following a hybrid permission/permission-less model and are incentivized to verify election results by the VOTE token. Here the unique factor of scale for the user is the number of the voters and it is not possible to set other parameters such as the level of service required that is assumed standard and independent to any factors. Similarly, BitCongress [2] is a platform that binds together Bitcoin, Counterparty (to create tokens from Bitcoin), Blockchain with smart contracts. For every BitCongress participant, a token is released (called VOTE) which can be sent to a single address, thus not allowing the double vote. Once the elections are over, the tokens are returned to owners. In [27] a new protocol to support that utilizes the Blockchain as a transparent ballot box is provided. In addition to the fundamental properties of a voting system, the protocol allows for a voter to change or cancel one vote, replacing it with another. In [30] the Crypto-voting service is proposed. Crypto voting is built upon SideChain and consists of using two linked blockchains, one-way pegged sidechain, where the main blockchain is used to record eligible voters and record their vote operations, while the pegged one count the votes.

In [32] is designed a voting system based on Multichain. In order to perform a transaction in Multichain, it is needed to identify the address of the node and the related balance from where the asset (vote) will be sent. While sending the asset to the address, the transaction hash was generated carrying the transfer of vote. The balance of the receiving node was incremented by one vote (asset). The transaction becomes a part of the public ledger which shows that it has been mined.

Recently the "as a service" paradigm is emerging also in the blockchain domain in general and in e-voting in particular. For instance, in [31] and [44] blockchain is proposed as a service while in [33] the entire voting system based on blockchain is offered "as a service".

In the first two cases, the solution is based on Ethereum Private Network (EPN) version. In [31] the Proof of Authority that uses the Identity as a stake is introduced. Using EPN is possible to send hundreds of transactions per second onto the blockchain for free, exploiting every aspect of the smart contract to ease the load on the blockchain while keeping the access permissions more tightly controlled, with rights to modify or even read the blockchain state restricted to a few users.

However, the trustworthiness of the solution is strongly dependent on the number of nodes actually deployed. In fact, because of the militated number of nodes in the EPN, it is necessary to reduce the complexity of the Proof of Work (POW) consensus significantly to allow the miner to mine the block in due time. It means that the security and integrity of the private blockchain will not rely anymore on the blockchain itself but on external protections to prevent fraudulent nodes to join your network. Moreover, in POW based private blockchain network it is theoretically possible for a group of nodes to come together to collude and make unauthorized changes to past transactions on the Blockchain. A possible method to prevent this risk is to save a snapshot of the private Blockchain on the public Ethereum network so that a block hash on the private network can be counter-checked against what's written to the public Blockchain.

The second one is based on Hyperledge that is natively designed to support data privacy in a consortium-based network. The trust is granted by design and there is not any PoW or cryptocurrency embedded in the architecture. This means that with a limited number of nodes it is possible to achieve the appropriate level of reliability and security. The voting service including the underling blockchain infrastructure can be tailored by the users that can select a number of parameters as (a) the number of eligible voters, (b) the data retention period, (c) the level of performance (24/7), (d) the tallying method, (e) anonymity, etc. All these parameters concur with the final billing.

The solution proposed in [28] is also based on Hyperledge and stresses the aspects of portability of on different platform of the blockchain-based e-voting service, confirming the capability of Hyperledge project to be used in an on-demand business scenario.

According to this analysis, the paradigm "as-a-service" needs to be exploited in-depth automating most of the steps in setting up an e-voting service such as the blockchain architecture and the development of the related smart contracts in order to implement a fully automatized Decentralized Autonomous Organization (DAO) capable to be set up a voting service on demand.

3 End to End Verifiability and Trust in HyperVote

In our system, a voting event is composed by four pillars: a Ballot format, the set of Eligible voters, a Ballot box and the Tallying. In particular, a Ballot can be defined as:

$$B = \langle b_1, b_2, \ldots b_n \rangle$$

Where $B \neq \{\varnothing\}$ b_i is the i-th option (preference) of B and $n = |B|$ represents the cardinality of all the options by which a preference can be expressed. The element b is a boolean variable and represents the preference expressed (e.g. 1 to mark the preference, 0 otherwise). The condition to express the preference maybe several.

The second pillar of a voting event is the vector of eligible voters *EV* defined as follow:

$$EV = \langle ev_1, ev_2, \ldots ev_m \rangle$$

Where $EV \neq \{\varnothing\}$, ev_i is the i-th eligible voter in the vector *EV* and *m* is the cardinality of $|EV|$.

The third pillar is the Ballot Box *BB,* where all the votes are collected and can be formalized as follow:

$$BB = \langle B_1, B_2, \ldots, B_h \rangle$$

Where h is the cardinality of *BB* and $h \leq m$.

The last pillar is the Tally process *T* that starts after the formal conclusion of the ballot and can be defined as follow:

$$T = f(BB)$$

where $f(BB)$ is a generic function in *BB* (e.g. one of the simplest is the cumulative frequency of $b \in B$) dedicated to counting the votes according to the algorithm defined and decrees the result (R).

A voting event should satisfy some very basic constraints to be considered reliable, such as:

(a) **Eligibility:** it is necessary to verify the eligible voter ev_{i} before allowing her to express the preference. Typically, this can be obtained by implementing an authentication service.
(b) **1:1 correspondence between *B* and *ev***. An ev_{i} can express one and only one vote B_{i} even if multiple preferences are included. Thus, it is necessary to manage the process in such a way to avoid double-voting.
(c) **Privacy (where necessary)**, in fact, not only the vote needs to be expressed in a safe and privacy-preserving condition, but it is necessary to pass through a mixing procedure shuffling the *BB* in order to avoid the possibility to guess the voter on the base of the arrival order of the vote collected. In fact, the temporal dependency between the event of voting by the voter and the event of collecting the vote is a well-known vulnerability that an e-voting system needs to address. In an offline system, the procedures foresee a physical shaking of the *BB* after the conclusion of the voting process and just before the counting phase.
(d) **Integrity:** Ensure that each vote is recorded as intended and cannot be tampered with in any manner, once recorded (i.e., votes should not be modified, forged or deleted without detection).
(e) **Transparency:** the tallying phase *T* needs to be managed in such a way the counting is performed with the guarantee that the calculation method is not affecting the final result.
(f) **Verifiable:** Allowing voters and third-party organizations to verify election results.

(g) **Auditable:** Supporting risk-limiting audits that help to assure the accuracy of an election.

All these constraints are a matter of trust and require a reliable solution able to guarantee it. In the present paper, we introduce the concept of End to End (E2E) trust in the e-voting system, i.e. a set features able to guarantee the appropriate level of protection of the votes collected and of the calculation algorithms. Here is where the Blockchain technology came to play.

The introduction of blockchain into a real-life scenario requires an in-depth analysis of the cases to verify (i) if the use of blockchain makes sense and (ii) how to introduce the technology in an appropriate way. In this respect, in [41] the Mirror Model (MM) has been introduced. In MM, three domains are identified: Reality, Representation, and Trust. In particular, "*Reality is where things objectively exist, independently of any efforts made by humans to design or control the Reality, Representation is a model conceived by architects or engineers or scientists to depict the aspects of interest or concern in the Reality and, finally, Trust is a structure of validation of the Representation of Reality whose aim is to make the Representation and its data dependable*." [41]. Through the Trust and Representation domains, we can act indirectly on the Reality to make it more reliable and secure for the benefit of humans and the environment in which they live.

Thus, according to the MM for blockchain adoption on real-life scenarios [41], the e-voting system can be entrusted as depicted in Fig. 2.

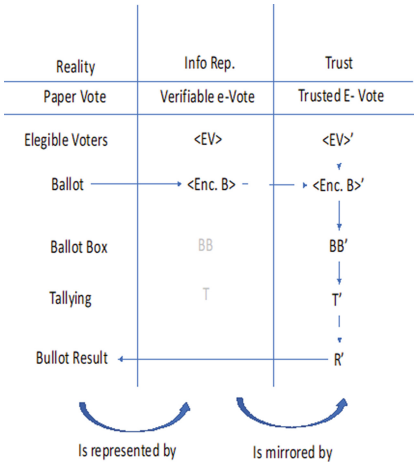

Fig. 2. Mirror model of e-voting scenario

In particular, the *Reality* column includes the main elements composing a paper-based voting event. In the *Information Representation* column, such elements are modeled to implement an e-vote application. Despite it is quite common that in a blockchain-based project only a few parts of the information system created to manage

an aspect of the reality are mirrored on the blockchain, in the case of a voting system, the use of the blockchain technology allows different considerations. In fact, because of the risk of introducing weaknesses at the information representation level (e.g. temporal dependencies of the database records link), several elements of the e-vote can be directly implemented within the blockchain infrastructure. However, some precautions must be taken as the generation of blocks has a direct temporal dependency with the vote expression that might be exploited to break the privacy constraint.

To address such issues, e-voting services are adopting cryptographic techniques to secure and end-to-end verifiability while preserving the privacy of the voters and avoiding the discovery of any possible link between a voter and her expressed preferences. One of the most used is *homomorphic encryption* [35, 45] that allows one to operate on ciphertexts without decrypting them. For a voting system, this property allows the encrypted ballots to be counted without leaking any information in the ballot [46, 47]. In fact, individual votes are encrypted votes and then combined to form an encrypted tabulation of all votes which can then be decrypted to produce an election tally that protects voter privacy. By running an open election verifier, anyone can securely confirm that the encrypted votes have been correctly aggregated and that this encrypted tabulation has been correctly decrypted to produce the final tally. This process allows anyone to verify the correct counting of votes by inspecting the public election record while keeping voting records secure. Other approaches include the *zero-knowledge proof* [49, 50] that requires that the voter should convince the authority that his vote is valid by proving that the ballot includes only one legitimate candidate without revealing the candidate information, or the *mix-net* approach [36, 37, 48] that aims to perform a re-encryption over a set of ciphertexts and shuffle the order of those ciphertexts. A mix node only knows the node that it immediately received the message from and the immediate destination to send the shuffled messages to.

However, the simple use of encryption technique, it is not a guarantee of reliability of the entire voting process. For instance, a security vulnerability has compromised 66,000 electronic votes in the New South Wales state in Australia in 2015 [51] even if encryption has been used. To this end, it is necessary to add a mechanism of securing an E2E trustworthy. In this respect, the *BB* implementation at Information Representation level (e-vote) that is usually managed with a database, needs to be avoided in favor of its implementation at the trust level (blockchain) so that the encrypted votes are collected with an untamperable tool. In this sense the mirroring is applied directly on the reality element. Moreover, to avoid attacks on the calculation method, the tallying function needs to be encoded at trust level T' (with a Smart Contract/Chaincode) as well. It will work on the mirrored Ballot Box (*BB'*), and thanks to the Homomorphic encryption the votes will be possible to obtain the results while preserving the privacy. The results of the counting are also managed in the trust level (*R'*) as a transaction in the blockchain that concludes the voting process. This way the entire process is communicated back to reality.

According to this process, the blockchain acts as an E2E trust layer where voters, vote, tallying and the result are stored in an untampered manner allowing further verification and audits.

In this way, it is possible to avoid any cyber-attack focused on votes removing or altering that could occur at the representation level as described in Sect. 2.

HyperVote Enterprise Business Process

In order to define the general business process of HyperVote it is necessary to decompose the entire voting process into self-consistent atomic steps that compose the Business Process. Then, according to MM, it is necessary to identify which of the steps can be implemented in a trustworthy environment using the blockchain. The result of such a decomposition and mapping is provided in Fig. 3. The picture shows that most of the tasks can be implemented in the Trust layer (dotted area), thus using blockchain. In this sense, business logic is defined by the smart contracts selected to implement the Business Process. This smart contract/chaincode can be seen also as a microservices architecture whose atomic services need to be composed to execute the final process.

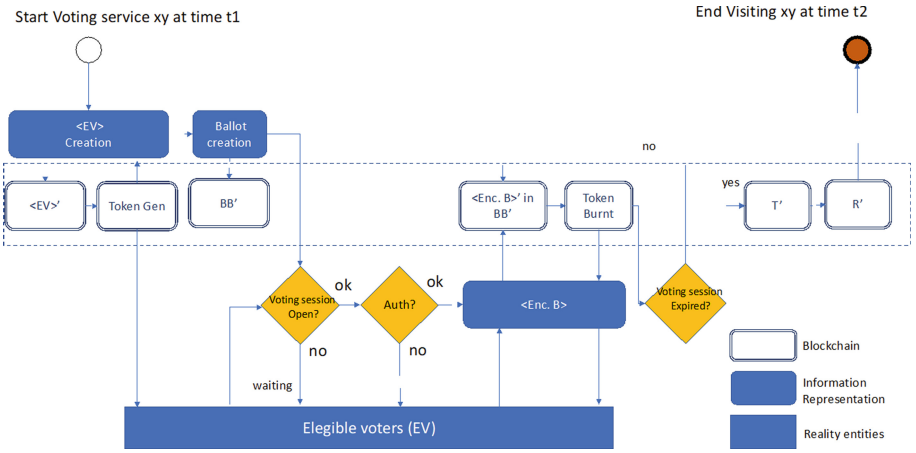

Fig. 3. e-voting business process

Business Process to SOA Translations

The second step includes the translation of the BP into a Service Oriented Architecture model (SOA). SOA [21] is an approach for defining and implementing services over a network linking business and computational resources (principally organizations, applications, and data) to achieve the results requested by service consumers (which can be end-users or other services). The definition, given in [21], now common for Web services and the related architectures, can be also exploited in the blockchain framework for chaincode design. There are different approaches for service modeling, as surveyed in [22] e.g. SOA-RM, SOA-RFA, SOMF, PIM4SOA, SoaML, SOA ontology, and SOMA. In the present work, we adopted SOMF since it provides a formal method of service identification at different levels of abstraction, from holistic to detailed, including meta-model concept and specific notation [22–24]. More specifically, we extended the SOMF notation introducing the Trusted Atomic Service (TAS) (see Fig. 4). A TAS is an irreducible service resulted from the decomposition of the tasks executed on the blockchain. It is implemented by a chaincode and, according to the MM, may represent a mirrored Atomic Service. According to the extended SOMF, the BP is translated into the service architecture depicted in Fig. 5.

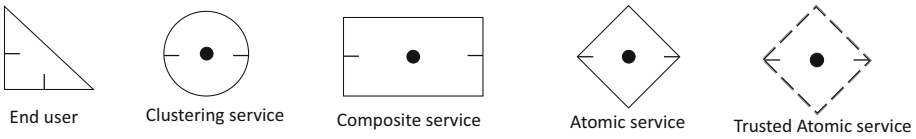

Fig. 4. SOMF notation plus trusted atomic service

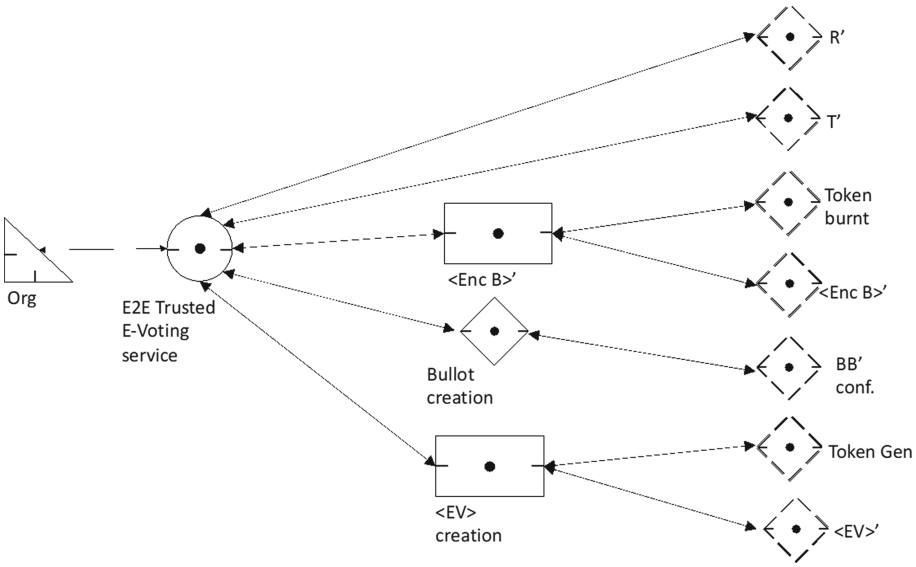

Fig. 5. SOMF-based HyperVote business process translation

Process Analytics Flow

The subject of measurement in process analytics are *events* that occur during the process execution. An event represents the change of state of objects such as data, activities, parameters, users, and so forth, within a given context [55]. For instance, log-in/log-out, process start/end, CRUD operations are *events* that need to be detected and then analysed to monitor system performance and compliance against business KPIs. Because of the complexity of the e-voting, it is useful to define a model organizing the steps in the knowledge flow. In HyperVote there are three classes of *events* to be detected: *process configuration events*, *process execution events*, and *compliance check events*. Process configurations events are recorded each time one of the parameters that the user can manipulate and customise is altered. Tracking these changes generates data that can be used to understand user intention and preferences. In addition, Cloud and Hyperledger Fabric infrastructures should be monitored and assessed against their performances. For instance, the cloud service on which the Fabric is deployed can be evaluated against parameters such as *availability*, *reliability*, *response time*, *security*, *throughout*, *capacity*, s*calability*, and *latency*. Similarly, the chaincodes executed in the blockchain can be analysed against their memory and CPU consumption. However,

such data are interrelated with complex and sometimes not linear dependencies. In fact, the performance of the blockchain is only partially affected by the computational infrastructure on which it is deployed, the selected configuration (e.g. number of nodes) and the chaincode implementation style/logic also play a crucial role. In other words, to optimize the HyperVote service for both user and service provider (e.g. MAX(*Profit*), MIN (*SLA deviation*)), it is necessary to continuously collect through an ETL mechanism. The absence of logging libraries in blockchain frameworks makes event logs generation a project critical task requesting dedicated development effort. The technological scenario is however in evolution with initial logging solutions devoted to blockchain technology [63] (Fig. 6).

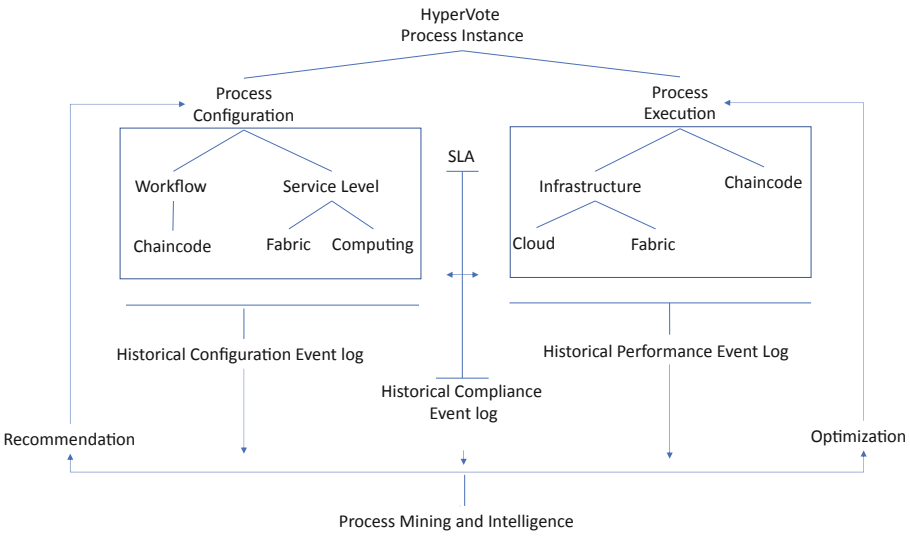

Fig. 6. HyperVote process analysis flow

4 HyperVote Process Centric Architecture

The concept of Process Centric Architecture (PCA) to create software systems that enable business process has been introduced in [56]. The objective of PCA is to support the creation of architectures able to execute business processes of an enterprise. Figure 7 depicts the architecture with the main components and their relationship with the Hypervote Cloud Bundle (CB), that will be used in the Cloud Deployer Orchestrator for the process execution. The HyperVote CB is an XML based data structure containing all the information required by the execution environment to deploy, execute, account, and monitor the service. The HyperVote CB binds a Workflow Model with the concrete Atomic Services (chaincode) and Hyperledger Network structure and Cloud Infrastructures that will be invoked and selected by the workflow, respectively, when executed. The HyperVote CB is structured as follow:

- Section 1 - Voting Process parameters. This section includes all the parameters defining the Voting Event such as Date time for start/end, and the parameters defined in Table 1.
- Section 2 - Voting Process model. Such a model is an executable BPMN 2.0 file already configured with all the technical details required to be executed by a BPM Engine. It might contain invocation to atomic service selected from the Smart Contracts included in the process.
- Section 3 – Trusted Business Logic. This section includes the reference to the selected chaincodes that implement each step in the BPMN model.
- Section 4 – Fabric Architecture parameters. This section includes the definition of the number of peers to be deployed, which will be configured as endorser, i.e. the Certification Authority and the Order. It is important to remark that a core architecture is defined and includes three peer nodes, one Order and one CA that are executed within the service provider.
- Section 5 – Cloud Services. This section includes information about the cloud provider, the services to be used, and the negotiated best SLA on the Cloud Providers Catalog. The parameters can conform to a combination of WS-Agreement (WS-Agreement 2015) and OWL-Q [51]. The allocation-related information is represented according to CAMEL [29], a multipurpose DSL that comprises existing DSLs such as CloudML.

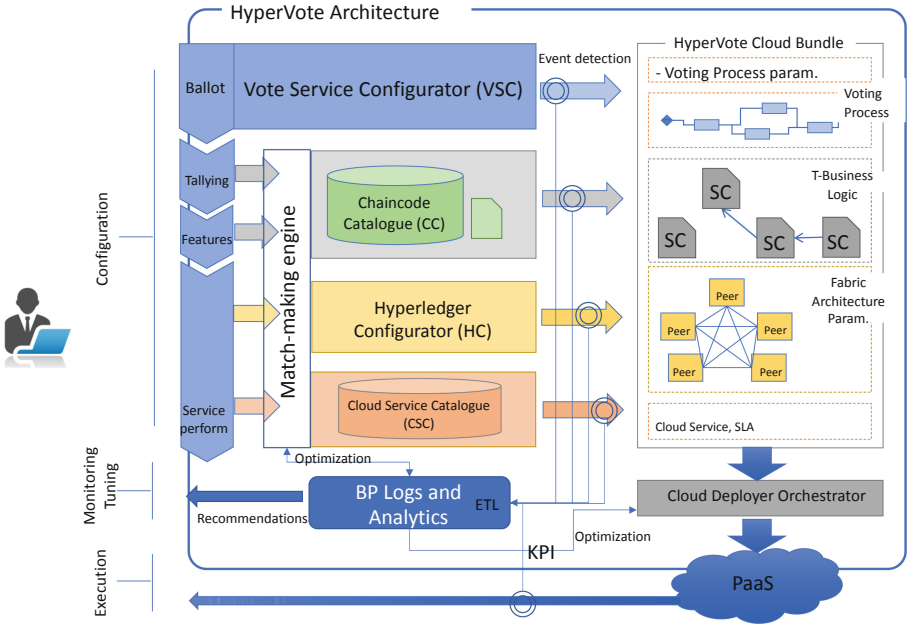

Fig. 7. HyperVote architecture

As defined in PCA, the Business Process components are here represented by the Vote Service Configurator (VSC) and the related Voting Process. The Business Functions components are represented by the Chaincode Catalog and the related Trusted Business Logic included in the CB as well as the Match-making engine. The User Interface is represented by VSC frontend and the HyperVote end-user application for voting.

4.1 Vote Service Configurator (VSC)

The Vote Service Configurator (VSC) is the interface a customer can use to specify the properties required by the system. Each selection is reflected in a change in the Service Level Agreement and the Pricing Scheme to be signed by the parties. In particular, the configuration is supported by a wizard organized in five steps reflecting the HyperVote Cloud Bundle organization. Thus, each step is devoted to trigger a platform component that manages the related section in the bundle.

Through the VSC it is possible to select the appropriate tallying mechanism searching into the Chaincode Catalogue. In fact, there are a relevant number of different tallying process. For instance, in a Ranked Choice Voting (RCV) mechanism, voters can rank as many candidates as they want in order of choice. Candidates do best when they attract a strong core of first-choice support while also reaching out for second and even third choices. When used as an "instant runoff" to elect a single candidate like a mayor or a governor, RCV helps to elect a candidate that better reflects the support of a majority of voters. Other mechanisms are related to exclusive vote so that the voter can express a mutually exclusive preference for only one candidate. In this regard, during the configuration phase, the user will be looking for the chaincode that better reflects the intent of the user.

VSC allows the user to specify additional parameters, as defined in Table 1, and supports the selection of the Quality of Service that will be translated into an SLA once the negotiation phase is concluded. When the configuration is completed, the prize of the e-voting solution can be computed and presented to the user.

4.2 Chaincode Catalogue (CC)

The programmability of the Blockchain was introduced in 2014 by Vitalik Buterin [5] with the permissionless (or open) Ethereum Blockchain, and later by IBM with the Hyperledger permissioned (or private) Blockchain [6]. According to Hyperledge definition, *Chaincode* is a software (written in one of the supported languages such as Go, Node.js or Java) that defines assets and related transactions; in other words, it contains the business logic of the system. It is installed and instantiated onto a network of Hyperledger Fabric peer nodes, enabling transactions with that network's shared ledger. A chaincode typically handles business logic agreed to by members of the network, so it may be considered as a "smart contract". In order to maximize the reuse of the chaincode when similar voting services are required, HyperVote implements a

catalog where it is possible to retrieve the chaincode that match the configuration selected by the user. In particular, the chaincodes implementing the tallying process are the most critical. In fact, the users need to select the right tallying mechanism and in case parametrize it.

4.3 Hyperledger Configurator (HC)

The Hyperledger Fabric is a permissioned consortium driven private blockchain network [26], built from the peers owned and contributed by the different organizations that are members of the network. The network exists because organizations contribute their individual resources to the collective network. Peers are a fundamental element of the network because they host ledgers and smart contracts. A peer executes chaincode, accesses ledger data, endorses transactions, and interfaces with applications. Peers have an identity (digital certificate) assigned by a Certificate Authority to authenticate member identity and roles. The validators are known, so any risk of a 50 + 1 attack arising from some miner collusion applies.

Because of that, the Blockchain Configurator is devoted to translating the configuration choices into parameters automating the infrastructure deployment.

Usually, the Hyperledger configuration is a time-consuming task that needs to be tailored on the base of the actual business requirements. For instance, if the number of running chaincodes on the peers exceeds the real need of duplication (e.g. not all the peers needs to host a chaincode in the network), this leads to an increment of the costs of operating the service without, however, obtaining a benefit for either the end-user or the service manager. However, such kind of recommendation is related to consulting activities that could be costly and require effort that might be difficult to estimate. To this end, the blockchain configuration will be a matter of service. Thus, HyperVote may provide some recommendations (or a default configuration), but the final choice remains on the user side. Expert users will be more capable to reduce the costs of the service optimizing the system configuration. Examples of parameters that can be configured by the users are:

(a) *# of running peers*: there is a minimum of three peers that are managed by the service provider.
(b) *chaincode level of distribution*: how many peers will run the chaincode in parallel. This parameter may affect the performance of the service. In fact, having multiple peers executing the chaincode reinforces the robustness of the process but, in turn, the speed to complete the process is reduced.
(c) *# of endorser node*: this parameter defines how many nodes will be set as endorsers. The number of endorsers affects the computational speed but, on the other hand, the process may result in a more reliable data processing.

4.4 Cloud Service Catalogue (CSC)

This component is organised according to the results of the CloudSocket project [54]. The Cloud Service Catalogue is defined by two sub-components: the Cloud Provider

Registry (CPR) and the Atomic Service Registry (ASR). The CPR is responsible to store and describe the different cloud provider configurations (local or remotes) as the login, front-end, the definition of the APIs. This registry is related to the complete management information (instead of being a simple configuration description); only the cloud providers included in the catalog will be used in the HyperVote Cloud Bundle. The ASR describes the cloud services that a Service Provider offers. There are three essential components to the CSC:

- The data model (in a self-describing JSON format) holding the information for the Service Catalogue to describe standard and extended attributes of Service Providers and Cloud Service offerings.
- The standard set of structures to describe the Service Provider and Service Provider's products and services.
- The API to interact with the CSC.

The different workflows use them to be part of the service tasks in the business process definition. Hence, these descriptions have to include both functional descriptions, business details and technical specifications (as the interface description). Besides, this is required certainly in case of adaptation reasons, as we might need to substitute one service with another one. Additionally, it can also be used to draw additional information about a particular service, which could then facilitate its (adaptive) execution [54].

4.5 Cloud Deployer Orchestrator (CDO)

This component is devoted to executing the HyperVote Cloud Bundle. In particular, it interacts with the cloud service where the Hyperledge infrastructure and the chaincodes will be deployed. It will set up all the connections and configuration and will execute a rule-based workflow engine to monitor the steps of the service. Moreover, it will continue to monitor the SLA and provides alerts in case of anomalies reducing the effort to manage running services.

4.6 Match-Making Engine (MME)

The MME component implements the PCA Business Rule layer, where the service provider business rules manage the entire life cycle of the service.

4.7 Business Process Logs and Analytics

Finding a near-optimal configuration for highly configurable systems, such as for instance Software Product Lines, requires dealing with exponential space complexity [8]. This relates to the size of the combinations to be considered (the power set of the assessed features) but also to the fact the interactions between features introduce performance dependencies. Thus, for determining performances, it is not possible to

simply aggregate the contributes provided by every single feature. Software systems require to be adapted to a variety of environments, constraints, or objectives. For this reason, modern software is designed to support different configurations and a key goal is reasoning about the performances lead by a specific configuration. A naive approach may consider the system simply as a black box and produce a set of observations for measuring the performances obtained by the system in specific configurations. This approach is however infeasible in any context where the number of configurations is so high that their combinations become intractable.

To make performance prediction practicable two general strategies have been followed, even if, in practice, several works apply a combination of the two.

- **Sampling-centric.** The goal is generating a model from a restrict set of samples keeping, at the same time, the prediction accuracy significantly. Random sampling is regarded as the standard operating procedure [9, 10], however, the true sampling approach should consider only valid configurations. Filtering out invalid configurations is a possible answer [11] but this solution does not avoid generating the entire set of configurations [12]. Encoding configurations in a feature model that using propositional formulas can reveal inconsistent configurations, i.e. conjunctions of features in the model, avoiding exploring the entire configuration space [8]. Statistical procedures have been adopted in order to sampling using the most significant variance [14], if necessary, in conjunction with other variables, such as acquisition cost, for instance [15]. The prediction power of a method can be improved by exploiting the information that is gained from the system itself. Machine Learning has been often used in this respect as it allows an iterative approach with a training set used to create the model and testing set to assess the sample and increase it if the achieve accuracy is low [16].
- **Model-centric.** Measuring the performances of a real system is typically time-consuming, costly and implies risks of damaging the system itself. For this reason, model-driven approaches may be preferred. For example, performance annotated software architecture models (e.g. UML2) can be transformed into stochastic models (e.g. Petri Nets) to then apply analytical or simulation-based optimization methods. A comparative assessment of different approaches with the trade-offs they entail is proposed in [17]. Feature Models can be exploited to represent the inter-dependencies between features in configurations restricting the sampling space to valid configurations [18] or identifying the minimum set of features required to determine a performance [18]. Clearly, the drawback is that not all inconsistencies are known a priori.

Our approach implements a model-centric view as it guarantees a fast application at the different specification levels handled by our architecture. An example of the parameters that can be defined provided in Table 1.

Table 1. Service configuration parameter

Param	Value	Description
Secrecy/Privacy	Yes/no	This option consists of allowing end-users in disclosing their vote when it is required by the voting system (e.g. qualified vote). In any case, the vote is collected and managed as encrypted
Tallying methods	<list>	This parameter allows the end-user to select the tallying method suitable for the event. A pre-filled list is presented and includes simple and complex calculation methods In case, none of the presented methods works for the case, a wizard for self-defined tallying, method is provided
Under-voting alert	Yes/no	The vote organizer may receive a warning of not voting. However, the system must not prevent under-voting
Receipt	Yes/no	The system may issue a receipt to the voter if and only if it can be ensured that vote-coercion and vote-selling are prevented, so that he may verify his vote at any time and also contend, if necessary
Distribution of Authority	#external running peers	The minimum number of peers to execute Hyperledger is 3. These nodes are executed within the Hyperledger Service; however, a number of external nodes can be instantiated for transparency allowing process inspection to the voter organization(s)
Data retention period	#Days	This feature represents the number of days the instantiated blockchain needs to be kept up and running for any audit and evidence analysis
Reporting	- Simple - Complete - Analytics	This parameter defines the amount of information to be sent back to the user after the end of the tallying process. The simple option reports the winner and ranks the other options; the complete option includes also statistics such as the number of voters, Analytics option add some flow analysis (timeslot based), etc.

To collect events, this component uses an ETL (Extract Transform and Load) process (e.g., Penthao Kettle) invoking a web service via HTTP Post. In most cases, the real-time data are pushed directly into the mapping process to feed a temporary SQL store. They are typically streamed both into a traditional SQL store and then converted into triples in the RDF (using the Karma tool) final store and in a NoSQL database (HBase).

5 HyperVote Evaluation

HyperVote needs to be considered as an agile and cost-effective solution to run a reliable and ubiquitous e-voting service. In order to evaluate the benefit of the proposed approach, we identify the following criteria derived by [38]:

(a) *Non-regression property (NRP)*: the validity of the model developed should not be affected by the addition of new elements (e.g. a new tallying function).
(b) *Rebound effect (RE)* [39]: the increased efficiency in performing the voting service creates an additional demand for new services. It is a proxy indicator of the achieved efficiency.
(c) *Value of the Service* (VoS) [40] defined as $VoS = \mathcal{F}\{t, c\}$ where t is time needed to obtain the result R after the vote collection; c is the costs to run the service.
(d) *Accuracy (Acc)*: it is related to the capacity of the system to produce a result that is the exact expression with a low number of votes considered not valid or lost.
(e) *Effort*: this criterion refers to the total effort in days needed to run an election service.

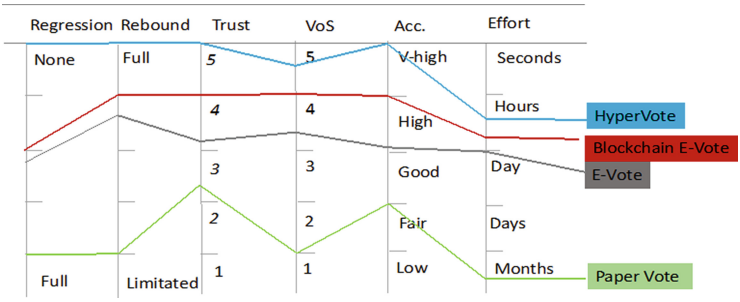

Fig. 8. HyperVote evaluation

According to the evaluation framework, HyperVote is absolutely not regressive, since changes in the Service configuration (e.g. a different tallying algorithm), does not invalidate the rest of the Service process. In fact, configurability is one of the main features of the system.

On the contrary, for paper-based/offline voting system, every change (e.g. preference expression mechanism) requires the entire re-execution of the simulation and the results. Blockchain-based voting service, as discussed in Sect. 2, cannot be entirely not regressive because of the use of the Distributed Ledger Technology as a monolith that reduces the possibility of extension and technology adaptation in case of changing conditions (e.g. technology obsolescence).

The possibility to have results in a cost-effective, trusted and verifiable/auditable way triggers the so-called rebound effect [39] in HyperVote that can be expressed in a request of even more configuration levels. Rebound effect is also present in blockchain-based voting system and in simple e-Vote service because of the use of technology. However, the effect is also dampened since the services do not easily scale in presence of a high number of voters (e.g. ELIGO e-voting service requires a project-based approach for more than 2000 voters involving a consulting phase that increases the costs [19]). Because of automation, the VoS is maximum in HyperVote. It is also significant for e-voting services. The combination of E2E verifiability with the E2E trust assigns the maximum value for the accuracy to Hypervote. The simple use of

blockchain increases the accuracy in e-voting services, while for paper vote, the voting process mechanisms are able to guarantee just a fair level. Finally, the effort required to configure lunch and obtain the results of the consultation in HyperVote is quantifiable in terms of hours. In the case of a relevant number of voters, in e-voting service, the effort required might be increased. The difference between the paper-based voting process is incomparable. The result of this evaluation is depicted in Fig. 8.

6 Conclusion

Our proposal aims at developing an e-Vote-as-a-Service based on Blockchain overcoming the limitations of the current projects using a cloud-based approach. Even if a number of cloud providers such as IBM and Oracle are offering ready-to-use blockchain installation on the Cloud with a fee based on the number of transactions, the challenge of a dynamic, cross-platform and on-demand system configuration and optimization based on end user's business requirements remains. HyperVote aimed at:

- keeping the operational costs (transaction, maintenance, etc.) is entirely under control of the organizations that provide the service. In particular, since Fabric is based on certified nodes, it is possible to have high performance and scalability respect to the permission less Blockchains,
- combining Privacy by Homomorphic encryption and Trust blockchain allowing tallying process on encrypted votes without the disclosure of the voter identity,
- modeling complex Business Logic with chaincode treated as an atomic service,
- enabling high configurability of the system and cross-platform agile portability.

The proposed architecture aims also to propose a new perspective for the blockchain-based application moving from a static to a dynamic and configurable view to optimize costs and service performance in view of long-term sustainability of the service provided.

References

1. Hardwick, F.S., Gioulis, A., Akram, R.N., Markantonakis, K.: E-voting with blockchain: an e-voting protocol with decentralisation and voter privacy. In: IEEE Conference on Internet of Things, Green Computing and Communications, Cyber, Physical and Social Computing, Smart Data, Blockchain, Computer and Information Technology, Congress on Cybermatics (2018)
2. BitCongress: Control the world from your phone. http://www.bitcongress.org/BitCongress Whitepaper.pdf
3. FollowMyVote.com: Technical report (2017). https://followmyvote.com
4. Tivi - verificable voting: Accessible, anytime, anywhere, TIVI, Technical report (2017). https://tivi.io
5. Wang, K.-H., Mondal, S.K., Chan, K., Xie, X.: A review of contemporary e-voting: requirements, technology, systems and usability. Data Sci. Pattern Recognit. **1**(1), 31–47 (2017)

6. Gritzalis, D.A.: Principles and requirements for a secure e-voting system. Comput. Secur. **21** (6), 539–556 (2002)
7. Halderman, J.A.: Practical attacks on real-world e-voting. In: Real-World Electronic Voting, pp. 159–186. Auerbach Publications (2016)
8. Oh, J., Batory, D., Myers, M., Siegmund, N.: Finding near-optimal configurations in product lines by random sampling. In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, pp. 61–71. ACM (2017)
9. Zhang, Y., Guo, J., Blais, E., Czarnecki, K.: Performance prediction of configurable software systems by fourier learning (t). In: 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 365–373. IEEE (2015)
10. Siegmund, N., Grebhahn, A., Apel, S., Kästner, C.: Performance influence models for highly configurable systems. In: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, pp. 284–294. ACM (2015)
11. Sayyad, A.S., Menzies, T., Ammar, H.: On the value of user preferences in search-based software engineering: a case study in software product lines. In: Proceedings of the 2013 International Conference on Software Engineering, pp. 492–501. IEEE Press (2013)
12. Henard, C., Papadakis, M., Harman, M., LeTraon, Y.: Combining multiobjective search and constraint solving for configuring large software product lines. In: Proceedings of the 37th International Conference on Software Engineering, vol. 1, pp. 517–528. IEEE Press (2015)
13. Anane, R., Freeland, R., Theodoropoulos, G.: E-voting requirements and implementation. In: The 9th IEEE CEC/EEE 2007, pp. 382–392. IEEE (2007)
14. Guo, J., Czarnecki, K., Apel, S., Siegmund, N., Wasowski, A.: Variability-aware performance prediction: a statistical learning approach. In: 2013 IEEE/ACM 28th International Conference on Automated Software Engineering (ASE), pp. 301–311. IEEE (2013)
15. Sarkar, A., Guo, J., Siegmund, N., Apel, S., Czarnecki, K.: Cost efficient sampling for performance prediction of configurable systems (t). In: 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 342–352. IEEE (2015)
16. Jamshidi, P., Casale, G.: An uncertainty-aware approach to optimal configuration of stream processing systems. In: 2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 39–48. IEEE (2016)
17. Brosig, F., Meier, P., Becker, S., Koziolek, A., Koziolek, H., Kounev, S.: Quantitative evaluation of model-driven performance analysis and simulation of component-based architectures. IEEE Trans. Softw. Eng. **41**(2), 157175 (2015)
18. Schröter, R., Krieter, S., Thüm, T., Benduhn, F., Saake, G.: Feature-model interfaces: the highway to compositional analyses of highly-configurable systems. In: Proceedings of the 38th International Conference on Software Engineering, pp. 667–678. ACM (2016)
19. https://www.eligo.social/
20. Nuseibeh, B., Easterbrook, S.: Requirements engineering: a roadmap ICSE'00. In: Proceedings of the Conference on the Future of Software Engineering, pp. 35–46 (2000)
21. Bell, M.: Introduction to service-oriented modeling. In: Service-Oriented Modeling: Service Analysis, Design, and Architecture. Wiley (2008). ISBN 978-0-470-14111-3
22. Mohammadi, M., Mukhtar, M.: A review of SOA modeling approaches for enterprise information systems. Elsevier Procedia Technol. **11**, 794–800 (2013)
23. Bellini, E., et al.: Interoperability knowledge base for persistent identifiers interoperability framework. In: IEEE Eighth International Conference on Signal Image Technology and Internet Based Systems, SITIS 2012r, vol. 6395182, pp. 868–875 (2012)
24. Truyen, F.: Enacting the Service Oriented Modeling Framework (SOMF) using Enterprise Architect (2011)

25. Duan, Y., Cao, Y., Sun, X.: Various AAS of everything as a service. In: 2015 16th IEEE/ACIS International Conference Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), p. 16, June 2015
26. Buterin, V.: On public and private Blockchains (2015)
27. Hardwick, F.S., Gioulis, A., Akram, R.N., Markantonakis, K.: E-voting with blockchain: an e-voting protocol with decentralisation and voter privacy. In: IEEE Confs on Internet of Things, Green Computing and Communications, Cyber, Physical and Social Computing, Smart Data, Blockchain, Computer and Information Technology, Congress on Cybermatics (2018)
28. Yu, B., Liu, J.K., Sakzad, A., Nepal, S., Steinfeld, R., Rimba, P., Au, M.H.: Platform-independent secure blockchain-based voting system. In: Chen, L., Manulis, M., Schneider, S. (eds.) ISC 2018. LNCS, vol. 11060, pp. 369–386. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99136-8_20
29. Rossini, A., et al.: D2.1.2– CloudML Implementation Documentation. Passage project deliverable, April 2014
30. Fusco, F., Lunesu, M.I., Pani, F.E., Pinna, A.: Crypto-voting, a blockchain based e-voting system. In: 10th International Conference on Knowledge Management and Information Sharing (2018)
31. Hjlmarsson, F., Hreiarsson, G.K., Hamdaqa, M., Hjlmtysson, G.: Blockchain-based e-voting system. In: IEEE 11th International Conference on Cloud Computing (2018)
32. Khan, K.M., Arshad, J., Khan, M.M.: Secure digital voting system based on blockchain technology. Int. J. Electron. Gov. Res. **14**(1) (2018)
33. Bellini E., Ceravolo, P., Damiani, E.: Blockchain-based e-Vote-as-a-Service. In: IEEE CLOUD Conference (2019)
34. Delaune, S., Kremer, S., Ryan, M.: Verifying privacy-type properties of electronic voting protocols. J. Comput. Secur. **17**(4), 435–487 (2009)
35. Aziz, A., Qunoo, H., Samra, A.A.: Using homomorphic cryptographic solutions on e-voting systems. Int. J. Comput. Netw. Inf. Secur. **10**(1), 44–59 (2015)
36. Sako, K., Kilian, J.: Receipt-free mix-type voting scheme. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-49264-X_32
37. Catalano, D., Jakobsson, M., Juels, A.: Coercion - resistant electronic elections. In: Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, pp. 61–70. ACM (2005)
38. Bellini, E., Coconea, L., Nesi, P.: A functional resonance analysis method driven resilience quantification for socio-technical systems. IEEE Syst. J. (2019). https://doi.org/10.1109/JSYST.2019.2905713
39. Gossart, C.: Rebound effects and ICT: a review of the literature. In: Hilty, L.M., Aebischer, B. (eds.) ICT Innovations for Sustainability. AISC, vol. 310, pp. 435–448. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-09228-7_26
40. Sajko, M., Rabuzin, K., Bača, M.: How to calculate information value for effective security risk assessment. J. Inf. Organ. Sci. **30**(2)
41. Bellini, A., Bellini, E., Gherardelli, M., Pirri, F.: Enhancing IoT data dependability through a blockchain mirror model. Future Internet **11**, 117 (2019)
42. http://e-vox.org/
43. https://www.agora.vote/
44. Hjálmarsson, F.Þ., Hreiðarsson, G.K., Hamdaqa, M., Hjálmtýsson, G.: Blockchain-based e-voting system. In: 11th IEEE Conference on Cloud Computing (2018). https://doi.org/10.1109/cloud.2018.00151
45. Gentry, C.: A fully homomorphic encryption scheme. Stanford University (2009)

46. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. Trans. Emerg. Telecommun. Technol. **8**(5), 481–490 (1997)
47. Katz, J., Myers, S., Ostrovsky, R.: Cryptographic counters and applications to electronic voting. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 78–92. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_6
48. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM **24**(2), 84–90 (1981)
49. Chow, S.S., Liu, J.K., Wong, D.S.: Robust receipt-free election system with ballot secrecy and verifiability. In: NDSS, vol. 8, pp. 81–94 (2008)
50. Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: Proceedings of the 8th ACM conference on Computer and Communications Security, pp. 116–125. ACM (2001)
51. NSW election result could be challenged over ivote security flaw (2015). https://www.theguardian.com/australia-news/2015/mar/23/nsw-election-result-could-bechallenged-over-ivote-security-flaw
52. Kritikos, K., Plexousakis, D.: Semantic QoS metric matching in ECOWS. IEEE Comput. Soc. **2006**, 265–274 (2006)
53. https://hacked.com/92-of-blockchain-projects-fail-according-to-new-chinese-study/retrieved. Accessed 19 June 2019
54. Woitsch, R.: D4.1 First CloudSocket Architecture (2015). https://site.cloudsocket.eu/documents/251273/350509/CloudSocket_D4.1_First-CloudSocket-Architecture-v1.0_BOC-20150831-FINAL.pdf/0bdd6c7b-a349-47ab-bed9-631e567365d8?download=true. Accessed 19 June 2019
55. zur Muehlen, M., Shapiro, R.: Business Process Analytics. In: Rosemann, M., vom Brocke, J. (eds.) Handbook on Business Process Management, vol. 2, pp. 137–157. Springer, Berlin (2010). https://doi.org/10.1007/978-3-642-01982-1_7
56. Seshan, P.: Process-Centric Architecture for Enterprise Software Systems. Auerbach Publications, Boca Raton (2010)
57. Bellini, E.: A blockchain based trusted persistent identifier system for big data in science. Found. Comput. Decis. Sci. **44**(4), 351–377 (2019)
58. Brotsis, S., et al.: Blockchain solutions for forensic evidence preservation in IoT environments. In: Proceedings of the 2019 IEEE Conference on Network Softwarization: Unleashing the Power of Network Softwarization, NetSoft, June 2019, Article number 8806675, pp. 110–114 (2019)
59. Kettinger, W.J., Grover, V.: Toward a theory of business process change management. J. Manag. Inf. Syst. **12**(1), 9–30 (1995)
60. Vukšić, V.B., Bach, M.P., Popovič, A.: Supporting performance management with business process management and business intelligence: a case analysis of integration and orchestration. Int. J. Inf. Manag. **33**(4), 613–619 (2013)
61. Weber, I., Xu, X., Riveret, R., Governatori, G., Ponomarev, A., Mendling, J.: Untrusted business process monitoring and execution using blockchain. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNCS, vol. 9850, pp. 329–347. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45348-4_19
62. Mendling, J., et al.: Blockchains for business process management-challenges and opportunities. ACM Trans. Manag. Inf. Syst. (TMIS) **9**(1), 4 (2018)
63. Klinkmüller, C., Weber, I., Ponomarev, A., Tran, A.B., van der Aalst, W.: Efficient logging for blockchain applications (2020). https://arxiv.org/abs/2001.10281. Accessed 28 Jan 2020
64. Ghosh, A., Fedorowicz, J.: The role of trust in supply chain governance. Bus. Process Manag. J. **14**, 453–470 (2008)