2019 IEEE 12th International Conference on Cloud Computing (CLOUD)

Blockchain-based e-Vote-as-a-Service

Emanuele Bellini[1] IEEE Member, Paolo Ceravolo[2] IEEE Member, Ernesto Damiani IEEE Member [1]

***Abstract*— This article aims at introducing a new configurable and multipurpose electronic voting service based on the blockchain infrastructure. The objective is to design an architecture 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 article presents the preliminary results of the system and a SOA-based services definition implemented with smart contracts.**

I. INTRODUCTION

In recent years, businesses have entered into the cloud computing era, where virtually turning anything into an online service (XaaS) has become a distinct possibility [19]. Strangely enough, this is not yet true for the e-Voting services, mainly because the recent examples of adoption of e-Voting in political elections has revealed significant critical issues such as low transparency and vulnerability [5]. We, however, claim that there are a number of cases in daily life in which electronic vote may be successfully applied. Examples include elections in private or restricted organizations. For instance, the election of a company's executive officers where each shareholder is entitled to one vote per share multiplied by the number of officers 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 functioning of the organization, as the achievement of all the conditions required for a valid vote may delay the appointment of officers. In this respect, e-Voting systems can significantly reduce the costs of implementation and verification as well as ease and incentive the participation of voters thanks to its ubiquity. Moreover, the introduction of the Blockchain technology allows to robustly cover all the security and reliability requirements imposed by voting procedures. By contrast, in the current state of the art, Blockchain represents a static and monolithic technological solution strongly dependent on the contingent business case addressed. Since its inception, the Blockchain industry has witnessed the launch of 80,000 projects, according to the China Academy of Information and Communications Technology (CAICT). Only 8% of them are actively maintained while the remaining 92% failed with an average lifespan of 1.22 years. In fact, in Blockchain-based projects, parameters such as the numbers of peers, HW/SW requirements, the computational performance, and so forth, are typically addressed at the design time by a system administrator and are not appropriately tailored on the business process to be addressed. In order to overcome such issues, the proposed solution aims at considering those parameters as a matter of service. Our Blockchain-based e-Vote-as-a-Service relies on an architecture that dynamically selects, deploys and execute an e-Voting service. This way requirements are directly selected by the end users and 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 time-frame 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.

II. RELATED WORK

The The e-Vote base on blockchain is becoming a field of experiment. Several initiatives as commercial level exists as Bitcongress [1], Followmyvote [2] and TIVI [3]. Other implementations are [3], [20], [21]. However, none of them have considered blockchain in terms of service that has its own costs of design and execution and its life cycle. Moreover, they tend to conceive blockchain as a static tool neglecting its obsolescence and rigidity once it is deployed.

III. CONFIGURABLE E-VOTE

The e-Voting scenario defines three actors: a) the Voting Service Provider VSP, b) the organization O interested in the voting results, and c) the electoral body, i.e the voters V. These three actors interact with each other through the Voting Process VP. In fact, an organization O is composed of M members whose some of them are entitled to express a vote on a topic, with V$\leq$M. The voting mechanism $VMEC$ is defined by a set of rules (e.g. the method to define a winner, the definition of a quorum, the timing for express the vote, etc.) that are assumed well-know by the voters. The realization of a fully configurable BC-based e-Vote implies the definition of a platform-as-a-service environment addressing the properties of service at three abstraction levels.

- (A1) Business Process, defining the orchestration of autonomous services according to the voting protocol.
- (A2) Business Logic, defining the behavior of every single service (smart contract).
- (A3) Transaction data, defining the properties transactions have to guarantee (security, speed, privacy, etc.).

[1]E. Bellini is with Khalifa University, Center of Cyber-Physical Systems, UAE emanuele.bellini at ieee.org

[2]P. Ceravolo is with the Department of Computer Science EUniversity of Milan, Italy paolo.ceravolo at unimi.it

[2]E. Damiani is with the Khalifa University Center of Cyber-Physical Systems, UAE ernesto.damiani at ku.ac.ae

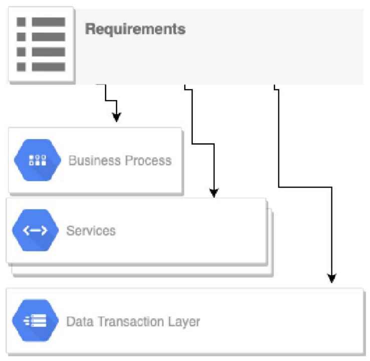

Fig. 1. An overview of the service

Since there are a number of functional and security constraints for a robust e-Voting service specified in [3],[4] that a VSP should respect, the most promising improvement in the current e-Voting service is represented by a method to minimize the internal operational costs. To this end, we identified the following challenges as the most relevant in the e-Voting context:

- (C1) Automating the implementation of a business process specifying the operational stages of a vote (A1).
- (C2) Semi-automating the generation of smart contracts (chaincode) addressing the logic of individual services (A2).
- (C3) Automating blockchain configuration (A3).
- (C4) Automating system deployment and optimizing the use of computational resources (A3).

IV. System Design

A. System Configurability

Finding a near-optimal configuration for highly configurable systems requires to deal with exponential space complexity [6]. This relates to the size of the combinations to be considered (the power set of the assessed features) but also to the fact the feature interactions introduce performance dependencies with other features so in determining performances it is not possible to simply aggregate the contributes provided by every single feature in a combination. To make performance prediction practicable two general strategies have been followed.

In sampling-centric solutions the goal is generating a model from a restrict set of samples, as restricted as possible, keeping at the same time the prediction accuracy significant. Random sampling is regarded as the standard operating procedure [7] however, true sampling approach should consider only valid configurations. Filtering out invalid configurations is a possible answer [8] but this solution does not avoid to generate the entire set of configurations [9]. Encoding configurations in a feature model that using propositional formulas can reveal inconsistent configurations, i.e. conjunctions of features in the model, avoiding to explore the entire configuration space [6]. Statistical procedures have been adopted in order to sampling using the most significant variance [10] if necessary in conjunction with other variables, such as acquisition cost, for instance, [11], the prediction power of a method can be improved by exploiting the information that is gained from the system itself.

Model-centric solutions start from the assumption that 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 [13]. Feature Models can be exploited to represent the interdependencies between features in configurations restricting the sampling space to valid configurations or identifying the minimum set of features required to determine a performance [14]. Our approach implements a model-centric view as it guarantees a fast application at the different specification levels handled by our architecture. The models we use for connecting features and performance are however updated based on the results obtained during by running the system and monitoring its operating results.

B. Service components

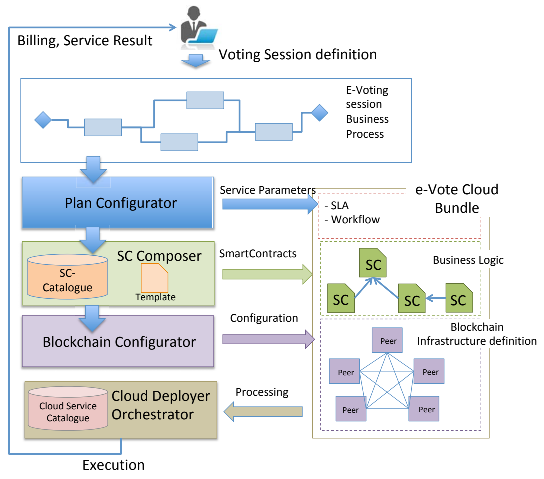

Fig. 2. Architecture overview

1) Plan Configurator (C1): The Plan Configurator is the interface a customer can use to specify the properties required by the desired voting session. This is shaped as a business process, then it is translated into parameters used to compose the e-Vote Cloud Bundle. For each block included in the business process, a number of performance and feature options can be selected. Each selection is reflected in a change in the performances constraining the Service Level Agreement (SLA) and impacts on the final pricing.

2) Smart Contract Composer (C2): The specifications defined in the Plan Configurator input the Smart Contract Composer. The component is devoted to automating the $VMEC$ and includes a catalog of pre-defined Smart Contracts (SC) to be used as a template for a common voting mechanism. Customization of a voting mechanism can be supported by a specific wizard. The Smart Contract selection and implementation follow the Service Oriented Architecture model specification [15]. There are different approaches for service modeling, as surveyed in [16]. In the present work, we adopted SOMF nomenclature since it provides a formal method of service identification at different levels of abstraction, from holistic to detailed [17], [18]. Then the SC can be divided into Atomic, Composite and Clustered Services according to their business logic. Examples of Atomic Services are:

- $VoterAuthentication$: the service verifies the credentials of the participant entitled to vote.
- $PoolCreation$: the service acquires the accounts of the voters from a list and set a create a new pool.
- $TokenGeneration$: the service generates a token for a given voter and Pool ID.

Examples of a Composite Service could be the $VoteCollection$ that takes as input the Pool ID and provided as output the matrix of the votes collected, and to do so needs to execute atomic services such as $PoolCreation$. The Cluster Services are represented by the entire BP defined by the end-user. An example could be a $VotingSessionSimple$ where, in input are provided (Voters list, counting methods, Date start, date end) and as output, the user obtains the final Report.

3) Blockchain configurator (C3): The Blockchain Orchestrator translates some user parameters like performance, data preservation, security and costs into specific infrastructure characteristics (e.g. number of peer nodes, computation capability of the hosts, etc.). In this way, it is possible to calibrate the complexity and the performance of the infrastructure according to the user request. Once the right configuration is decided, it is included in the bundle.

4) Cloud Deployer Orchestrator (C4): This component performs a matchmaking analysis between the criteria defined in the bundle and the could services hosted in the internal catalog. The matching algorithm should respect all the criteria expressed by the end-user as well as the gain the maximization of the e-vote service provider Thus the Cloud Deployer Orchestrator automates the deployment of virtual machines, their configuration, provisioning, and intra-service orchestration according to the e-Voting Cloud Bundle.

V. CONCLUSIONS

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 cloud with a fee based on the number of transactions, the challenge of a dynamic and on-demand system configuration and optimization based on end users business requirements remains. The present approach allows the end users to specify functional and non-functional requirements of services and the Cloud infrastructure where services are deployed on demand to optimize costs and service performance.

REFERENCES

[1] BitCongress. Control the world from your phone. [Online]. Available: http://www.bitcongress.org/BitCongress Whitepaper.pdf

[2] FollowMyVote.com, Tech. Rep., 2017. [Online]. Available: https://followmyvote.com

[3] Tivi - verifiable voting: Accessible, anytime, anywhere, TIVI, Tech. Rep., 2017. [Online]. Available: https://tivi.io

[4] K.-H. Wang, S. K. Mondal, K. Chan, and X. Xie, A review of contemporary e-voting: Requirements, technology, systems, and usability, in Data Science and Pattern Recognition, vol. 1, no. 1, pp. 3147, 2017.

[5] J. A., Halderman (2016). "Practical attacks on real-world e-voting", in Real-World Electronic Voting (pp. 159-186). Auerbach Publications.

[6] J. Oh, D. Batory, M. Myers, and N. Siegmund, "Finding near-optimal configurations in product lines by random sampling", in Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. ACM, 2017, pp. 6171.

[7] Y.Zhang, J.Guo, E.Blais, and.Czarnecki,"Performance prediction of configurable software systems by Fourier learning (t)", in 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 365373.

[8] A. S. Sayyad, T. Menzies, and H. Ammar, "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. IEEE Press, 2013, pp. 492501.

[9] C.Henard, M.Papadakis, M.Harman, andY.LeTraon,"Combining multi-objective search and constraint solving for configuring large software product lines", in 37th International Conference on Software Engineering. IEEE Press, 2015, pp. 517528.

[10] J. Guo, K. Czarnecki, S. Apel, N. Siegmund, and A. Wasowski, Variability-aware performance prediction: A statistical learning approach, in Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on. IEEE, 2013, pp. 301 311.

[11] A. Sarkar, J. Guo, N. Siegmund, S. Apel, and K. Czarnecki, Cost-efficient sampling for performance prediction of configurable systems (t), in Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on. IEEE, 2015, pp. 342352.

[12] P. Jamshidi and G. Casale, An uncertainty-aware approach to optimal configuration of stream processing systems, in Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2016 IEEE 24th International Symposium on. IEEE, 2016

[13] F. Brosig, P. Meier, S. Becker, A. Koziolek, H. Koziolek, and S. Kounev, Quantitative evaluation of model-driven performance analysis and simulation of component-based architectures. IEEE Trans. Software Eng., vol. 41, no. 2, pp. 157175, 2015.

[14] R. Schroter, S. Krieter, T. Thum, F. Benduhn, and G. Saake, Feature-model interfaces: the highway to compositional analyses of highly-configurable systems, in 38th International Conference on Software Engineering. ACM, 2016, pp. 667678.

[15] M. Bell (2008). "Introduction to Service-Oriented Modeling". Service-Oriented Modeling: Service Analysis, Design, and Architecture. Wiley and Sons. ISBN 978-0-470-14111-3.

[16] Mohsen Mohammadi Muriati Mukhtar, "A Review of SOA Modeling Approaches for Enterprise Information Systems" - Elsevier Procedia Technology - 2013, Volume 11, Pages 794-800

[17] E. Bellini, C. Luddi, C. Cirinn, M. Lunghi A. Felicetti, B. Bazzanella, P. Bouquet (2012) "Interoperability Knowledge Base for Persistent Identifiers Interoperability Framework" in 8th IEEE SITIS Conference, DOI: 10.1109/SITIS.2012.130

[18] Truyen. F, (2011) Enacting the Service-Oriented Modeling Framework (SOMF) using Enterprise Architect

[19] Y. Duan, Y. Cao, and X. Sun. "Various aas of everything as a service". In Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2015 16th IEEE/ACIS International Conference, pages 16,

[20] B. Yu, J. Liu, A. Sakzad, S. Nepal, R. Steinfeld, P. Rimba, and M. Ho Au, (2018) "Platform-Independent Secure Blockchain-Based Voting System". In: Chen L., Manulis M., Schneider S. (eds) Information Security. ISC 2018. LNCS, vol 11060. Springer

[21] R. Hanifatunnisa, B. Rahardjo (2017), "Blockchain based e-voting recording system design", 11th International Conference on Telecommunication Systems Services and Applications (TSSA)

[22] F. Fusco, M. L. Lunesu, F. Eros Pani and A. Pinna (2018) Crypto-voting, a Blockchain based e-Voting System, 10th International Conference on Knowledge Management and Information Sharing

[23] Fririk . Hjlmarsson, Gunnlaugur K. Hreiarsson, Mohammad Hamdaqa, Gsli Hjlmtsson (2018), Blockchain-Based E-Voting System, IEEE 11th International Conference on Cloud Computing