Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 184 (2021) 785–790

Procedia Computer Science

www.elsevier.com/locate/procedia

The 2nd International Workshop on the Advancements in Model Driven Engineering (AMDE)
March 23 - 26, 2021, Warsaw, Poland

Ethereum's Smart Contracts Construction and Development using Model Driven Engineering Technologies : a Review

Yassine Ait Hsain[a,*], Naziha Laaz[a], Samir Mbarki[a]

[a]*Information Modeling and Communication Systems Team, EDPAGS Laboratory, Faculty of Science, Ibn Tofail University, Kenitra, Morocco*

Abstract

In the Blockchain context, Smart Contracts are computer programs that run on the Ethereum platform. Benefiting from the properties of Blockchain, SCs development represents a major challenge to developers, as the code is deployed to an immutable system, besides the Ethereum platform is still evolving. This paper highlights how we can exploit model-driven engineering for generating long terms and high productivity smart contracts. It reviews researches on Smart Contracts generation in the Ethereum blockchain from a model-driven perspective. Based on the studied approaches, we defined a comparative framework to outline the advantages and disadvantages of each approach. The result can be used as a basis of tool selection for specific development aspects of SCs.

Keywords: Smart Contracts (SCs); Ethereum; Blockchain; Model-Driven Engineering (MDE); Modeling; Business Process Model and Notation (BPMN); Unified Modeling Language (UML).

1. Introduction

Blockchain has gained an extensive attention in many fields, making its first appearance in Bitcoin [21], a cryptocurrency that have reached over 650 billion dollar up today[6]. While the power of Bitcoin was recognizable, many were trying to move beyond cryptocurrency applications and there was Ethereum [30]. Over the few past years, Ethereum technology has attracted considerable attention and popularity in academic and industrial areas since its creation in 2015 [29]. The platform has now become the second largest cryptocurrency in circulation [5].

Based on blockchain, the Ethereum platform makes it possible to store and transmit information transparently, securely and without central control intermediaries. The Ethereum currency "Ether" is largely inspired by Bitcoin, but what makes this platform exceptional is the use and execution of Smart Contracts (SCs). Smart contracts represent automated programs that can send and receive transactions, offering developers the power of high availability, auditability, transparency, and neutrality [2].

* Corresponding author. Tel.: +212-610-208-630 ;
E-mail address: yassine.aithsain@uit.ac.ma

1877-0509
10.1016/j.procs.2021.03.097

One of the big challenges facing developers in Ethereum Blockchain is the nature of deployment system which is immutable. The development in this platform is structured in several stages, with major adjustments occurring at each stage. Each development step can include releases known as "hard forks", which radically change the platform, making the code of Smart Contracts no longer working [1]. Hence, developers have to deploy code taking into consideration the continuous evolution of the Ethereum platform. In other words, if a "hard fork" emerges, developers need to retain some control over their smart contracts in order to recycle and restart them.

Model-driven Engineering (MDE) can help meet the technical challenges and issues of Ethereum Blockchain development. Towards this end, MDE proposes a high level of abstraction representation to address heterogeneity and system complexity. This engineering goes beyond the pure development activities and encompasses other model-based tasks of a complete software engineering process. It is based on capitalization to enable re-usability, automatic adaptation of systems, and sustainable development in terms of time, cost and effort. This engineering is based on three concepts: Model, Metamodel and Model Transformation, which are essential for the automation of development processes and the generation of applications from models [24].

Having said that, many researchers have opted to use MDE to alleviate the complexity of Smart Contracts development. Indeed, researchers have proposed approaches that automatically generate SCs by identifying the elements of modeling a smart contract independently of the technical aspect of the Ethereum platform, thus treating both the static and behavior aspects of SC effectively. The challenges faced in this work, are the identification and the study of all these MDE-based approaches for modeling and generating SCs, in order to understand: what types of models are used to design smart contracts and how they are built? does it follow any standard? How these models are transformed? Are these transformations developed according to the MDE approach? Also, how the code generation is done?

To answer these questions, this paper presents a review study based on a comparative framework of most relevant researches, that have been conducted to solve this matter using MDE. It is organized as follows: After the introduction, Section 2 presents the studied approaches identified in the literature review of Blockchain modeling and SCs development using MDE. The analysis and results are discussed in Section 3. Section 4 concludes the paper.

2. Modeling Approaches for Smart Contracts Development using MDE

In this section, we will discuss the studied approaches regarding the generation of Ethereum's Smart Contracts using Model-Driven Engineering (MDE), as well as other works related to modeling and Blockchain. These papers were identified after a deep review of the literature, searching by several criteria ranging from general to specific, in order to find all the relevant approaches applying MDE in Blockchain platforms, in particular those proposing solutions for the generation of smart contracts.

The first approach we encountered during our research was Lorikeet [28]. Starting from a BPMN models and fungible/non-fungible registry data schemas, they generated standardised ERC-20/ERC-721 compliant asset registry smart contracts. Lorikeet's BPMN Modeler is based on bpmn-js modeling library. This library is licenced to bpmn.io which by itself a branch of Camunda. This leads us to the next tool which is Caterpillar [16], another work of the same author that uses Camunda which is an open-source workflow and decision automation platform. The advantage of Camunda beside being upgradable is the fact that it relies on OMG standards CMMN and DMN. These two studied tools [28, 16] focused on using BPMN, which is constrained to support only the concepts of modeling that are applicable to Business Processes. This means that other types of modeling done by organizations for business purposes is out of scope for BPMN [22].

BlockME [9] another approach that supports the modeling and execution of blockchain-aware business processes to allow external applications, to communicate with blockchains systems while taking care of blockchain specificities. The approach focuses on generating a BPMN 2.0-based business process that can communicate with the Blockchain Access Layer (BAL), a middleware allow external application to exchange transactions with public blockcahin sytems.

BMPN2 process engine, Activiti (http://activiti.org/) is redesigned by [11], so that business processes are implemented as a SC maintaining the properties of the Blockchain. They have defined a pipeline model allowing to separate the management of states on several BCs and the execution of SCs by creating blocks in concurrent manner.

Adding to the mentioned approaches above we have FSolidM, an Open Source Framework [18] which like [28, 16] is built on top of a Model-Driven tool (WebGME) [17]. FSolidM uses Finite State Machine Model that is represented graphically and Data attributes as inputs to generate Smart Contracts code. In addition the framework extends a set

of security plugins that can alleviate some common security issue namely the "DAO" attack. the main features of FSolidM are: Formal Model, Graphical Editor, Code Generator and Security Plugins [19]. Lacking from formal operational semantics, FSolidM was introduced in 2019 as VeriSolid or FSolidM / VeriSolid Framework [20], to address formal verification capabilities thereby providing an approach for correct-by-design smart contracts development.

Another studied approach is [11], that differentiate from others in terms of input models with the use of UML Statechart model. The used elements are simple and composite states (including history states) alongside with state transitions in order to express the state of a cyber-physical. These transitions will be mapped to blockchain transaction to which the code is generated. The authors of this work highlight the behavioral aspect of SC by choosing one of the dynamic models proposed by UML which is the Statchart model. Furthermore, the particularity of this approach is that it defines mapping rules between Statechart and Solidity elements, which is not the case for the other approaches.

In this work [25], the authors propose an open source environment designing a smart contract model and generating equivalent code automatically. They elaborate a class diagram to define a visual domain specific language, and for the process design part, this approach uses a set of BPMN and DEMO (DMN) models.the authors have created four models to define elements of Smart Contract, taking into account its different modeling views; Transaction kinds, actor roles, processes, data model and action rules. By analysing this approach, we notice that the repository of the mentioned environment is not based on MDE Technologies. Hence, it uses DEMO technologies. Also, the presented metamodel is nothing but a class diagram with different concepts of a smart contract.

[15] represents an MDA-based approach defines a unifying model as a UML class diagram. Based on literature from legal informatics and blockchain research, this model describes legal States, actions, roles and data sources which represent the most important component of legal smart contracts. The main goal of this approach is to compare and assess existing modeling languages for legal smart contracts development with regards to the proposed unifying model.

The approach [26] founded on MERODE (a method that relies on MDE and artifact-centric BPs to design and implement intra-organizational Enterprise Information Systems) is a recent published approach consisting of model-driven engineering and artifact-centric business processes to generate Smart contracts [26] and blockchain-based information systems [8] supporting cross-organizational collaborations.This approach is based on five layers: Domain Layer, Permission Layer, Core Information System Services Layer, Information System Services Layer and Business Process Layer.

In [10], the authors have introduced a process that automates the translation of institutional constructs into codified machine- readable contractual rules to address smart contracts to non-developers as it express human-readable institutional rules, regulations, and laws in terms of ADICO statements that capture high-level institutional semantics then those sentences are mapped and transcribed into Solidity code.

Last but not least, the Smart Contract Engineering (SCE) is introduced in [12]. This method presents a roadmap of SC verification from its formal modeling with Event-B to model transformation, to verification, to solidity code generation and conformance testing. The main goal of SCE method is to meet the requirements of the design and development of large-scale smart contracts.

3. Analysis and Discussion

This Section discusses all studied approaches cited in the section above, by proposing a comparative framework while considering various criteria. First, we have identified criteria representing both the basis for the development of SCs following model-driven engineering as well as a baseline for an objective comparison and evaluation of the studied approaches.

The main criteria of the considered approaches are : year of publication, MDE principals, code generation from input to output to the implementation of security concepts, and used technologies. Some criteria are divided into sub-features as follows :

- Year of publication
- Implementing MDE : Model definition, Metamodel definition and Model Transformation.
- Code generation : input, output and the implementation of security concepts.
- Used technologies : exploited or extended.

Table 1. Comparative table of the studied approaches using MDE for the generation of Smart contracts.

Studied Approaches/Tools	Year	Implementing MDE			Code generation			Used technologies
		Model Definition	Metamodel definition	Model Transformation	input	output	Implementing Security	
SC Engineering [12]	2020	no	no	yes	Event-B model	Solidity code	no	ATL language
B-MERODE [26]	2020	yes	no	no	Existing Dependency Graph, Object-Event Table and FSM	-	-	-
The Development of SCs for Heterogeneous BCs [27]	2019	yes	yes	yes	UML Class diagram, OCL Constraints	Solidity code, Hyperledger code	no	Obeo's Acceleo, Papyrus
BlockME [9]	2019	yes	no	yes	BlockME Process Model	Standard-Compliant Process Model	no	Camunda's BPMN-modeler [3]
VeriSolid [20]	2019	yes	no	yes	Smart Contracts as Transition System, Properties	Solidity code	yes	WebGME [17]
Das Contract [25]	2019	yes	-	-	Contract Structure Diagram (DEMO OCD Model), Contract Process Diagram (BPMN and DMN Models.), Contract Data Model (DEMO OFD Model), Contract Action Diagram(Brockly)	Solidity code	no	Angular and Nodejs
FSolidM [18]	2018	yes	no	yes	FSM Model	Solidity code	yes	WebGME [17]
Lorikeet [28]	2018	yes	no	yes	Business process model, Asset registry data model	Solidity code	no	BPMN-JS [4]
Towards Model-Driven Engineering of Smart Contracts for Cyber-Physical Systems [11]	2018	yes	no	yes	Uml Statchart Model	Solidity code	no	YAKINDU Statechart Tools [31]
Caterpillar [16]	2017	yes	no	yes	Business process model, process variable definitions, solidity code for script tasks, code specifying the exchange of information from/to user/service tasks.	Solidity code	no	Camunda's BPMN-modeler [3]
From Institutions to Code: Towards Automated Generation of Smart Contracts [10]	2016	no	no	no	ADICO statements	Solidity code	no	ADICO[7], Scala[23]

(-) : refers to an ambiguity while analysing the paper for the specified criteria.

The result of the evaluation criteria analysis is summarized in Table 1, which is deducted from all the selected approaches. By constructing this table, which represents a comparative table of all the approaches studied according to each criterion, we observed that the majority of these articles validated our proposed criteria, which justifies our relevant choice of articles for this review study.

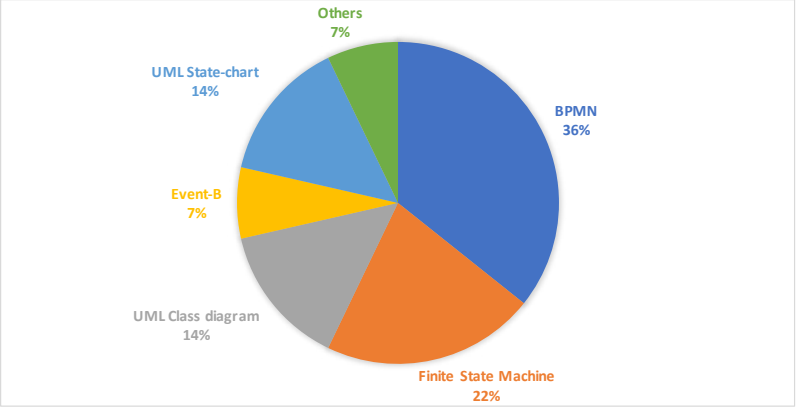

Fig. 1. Distribution of Models type used for the SCs modeling.

Based on the used model, these approaches can be divided into three categories. A dominant category interested by the behavioral aspect of smart contracts, modeling business processes by both BPMN and UML statechart models [9, 25, 28, 16]. A second category no less important than the first one, which focuses on the modeling of the static aspect using the class diagram of the UML standard [27]. While the third and last category designs the formal aspect of smart contracts using several representations such as finite state machine model [18, 26], Events B [12], OCL or Ontologies [27].

For the meta-model definition criteria, we see that most approaches does not cover it. However, only one work proposes a meta-model for Ethereum Smart Contracts [25], which is nothing but a class diagram with the different concepts of SCs. As for the third criteria related to transformation between models, we observe that the majority of approaches do not focus on the model-to-model transformation, and others defines a mapping rules using a natural language such as [27, 9]. Elsewise, the only approach that apply an automatic transformation (M2T) using ATL Transformation language is [12]. The other works transform its models directly implementing model to text transformation.

In terms of code generation, we notice that [26] addresses it in future works, without showing any tangible results, other than [12, 27, 9, 20, 25, 18, 28, 11, 16, 10] that showed the applicability of their method reaching significant results. Another point is that all the studied approaches have selected Solidity programming language as the targeted code output.

As shown in Figure 1, we illustrate by a pie chart the distribution of the studied approaches according to the used model type. As reported, you may notice that 36% use BPMN models to represent Smart contract processes and 22% employ Finite State Machine, while approaches based on UML State Chart and Class diagram represent 14% each. Finally, few works implemented other methods covering formal aspects by a percentage of 7%.

We conclude that most of the approaches commonly based on BPMN which is considered as an OMG standard for the representation of the business process. While BPMN shows the flow of data (Messages), and the association of data artifacts to Activities, it is not a data flow language. In addition, operational simulation, monitoring and deployment of Business Processes are out of scope of this specification [14, 13]. As for the other approaches they implement different types of model that distinguish each one of them.

4. Conclusion

This paper highlights the use of MDE aspects such as metamodelling and model transformation for the early validation of smart contract properties and the possibility of automatic generation of code in Blockchain platforms. It provides a review of the research status on this matter. While filtering the approaches that use MDE technologies for generating SCs from the collected papers, we have observed that few studies have been conducted concerning this topic.

After analysing all the studied approaches, we were convinced that Ethereum can benefit from MDE technologies particularly in specifying the concepts of SCs on different metamodelling levels, which in turn allows for the

application of automated managing and maintaining. Moreover, we have emphasized the most used solutions and technologies that automatically generate SCs. In the future, our investigations will address the question on how to extend the existing approaches to include other model-based aspects during the construction and generation of SCs.

References

[1] Antonopoulos, A.M., Wood, G., 2019. Mastering Ethereum. volume 53. URL: https://github.com/ethereumbook/ethereumbook, arXiv:arXiv:1011.1669v3.

[2] Buterin, V., et al., 2014. A next-generation smart contract and decentralized application platform. white paper 3.

[3] Camunda, . Modeler | Camunda. URL: https://camunda.com/products/camunda-bpm/modeler/.

[4] Camunda Services GmbH, 2016. BPMN 2.0 rendering toolkit and web modeler | bpmn.io. URL: https://bpmn.io/toolkit/bpmn-js/ https://bpmn.io/.

[5] Chen, W., Zhang, T., Chen, Z., Zheng, Z., Lu, Y., 2020. Traveling the token world: A graph analysis of ethereum erc20 token ecosystem, in: Proceedings of The Web Conference 2020, pp. 1411–1421.

[6] CoinMarketCap, 2020. Cryptocurrency Prices, Charts And Market Capitalizations. URL: https://coinmarketcap.com/.

[7] Crawford, S.E.S., Ostrom, E., 1995. A Grammar of Institutions. American Political Science Review 89, 582–600. doi:10.2307/2082975.

[8] De Sousa, V.A., Burnay, C., Snoeck, M., 2020. B-merode: A model-driven engineering and artifact-centric approach to generate blockchain-based information systems, in: International Conference on Advanced Information Systems Engineering, Springer. pp. 117–133.

[9] Falazi, G., Hahn, M., Breitenbücher, U., Leymann, F., 2019. Modeling and execution of blockchain-aware business processes. SICS Software-Intensive Cyber-Physical Systems 34, 105–116.

[10] Frantz, C.K., Nowostawski, M., 2016. From institutions to code: Towards automated generation of smart contracts, in: 2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS* W), IEEE. pp. 210–215.

[11] Garamvölgyi, P., Kocsis, I., Gehl, B., Klenik, A., 2018. Towards model-driven engineering of smart contracts for cyber-physical systems, in: 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), IEEE. pp. 134–139.

[12] Hu, K., Zhu, J., Ding, Y., Bai, X., Huang, J., 2020. Smart contract engineering. Electronics 9, 2042.

[13] Kharmoum, N., El Bouchti, K., Laaz, N., Rhalem, W., Rhazali, Y., 2020. Transformations' study between requirements models and business process models in mda approach. Procedia Computer Science 170, 819–824.

[14] Laaz, N., Kharmoum, N., Mbarki, S., 2020. Combining domain ontologies and bpmn models at the cim level to generate ifml models. Procedia Computer Science 170, 851–856.

[15] Ladleif, J., Weske, M., 2019. A unifying model of legal smart contracts, in: International Conference on Conceptual Modeling, Springer. pp. 323–337.

[16] López-Pintado, O., García-Bañuelos, L., Dumas, M., Weber, I., 2017. Caterpillar: A blockchain-based business process management system., in: BPM (Demos).

[17] Maróti, M., Kecskés, T., Kereskényi, R., Broll, B., Völgyesi, P., Jurácz, L., Levendovszky, T., Lédeczi, Á., 2014. Next generation (meta) modeling: web-and cloud-based collaborative tool infrastructure. MPM@ MoDELS 1237, 41–60.

[18] Mavridou, A., Laszka, A., 2018a. Designing secure ethereum smart contracts: A finite state machine based approach, in: International Conference on Financial Cryptography and Data Security, Springer. pp. 523–540.

[19] Mavridou, A., Laszka, A., 2018b. Tool demonstration: Fsolidm for designing secure ethereum smart contracts, in: International Conference on Principles of Security and Trust, Springer. pp. 270–277.

[20] Mavridou, A., Laszka, A., Stachtiari, E., Dubey, A., 2019. Verisolid: Correct-by-design smart contracts for ethereum, in: International Conference on Financial Cryptography and Data Security, Springer. pp. 446–465.

[21] Nakamoto, S., Bitcoin, A., 2008. A peer-to-peer electronic cash system. Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf 4.

[22] OMG, 2011. Business Process Model and Notation (BPMN). Object Management Group, formal/2011-01-03. URL: http://www.omg.org/spec/BPMN/2.0.

[23] Scala, 2011. The Scala Programming Language. URL: https://www.scala-lang.org/http://www.scala-lang.org/.

[24] Schmidt, D.C., 2006. Model-driven engineering. doi:10.1109/MC.2006.58.

[25] Skotnica, M., Pergl, R., 2019. Das contract-a visual domain specific language for modeling blockchain smart contracts, in: Enterprise Engineering Working Conference, Springer. pp. 149–166.

[26] de Sousa, V.A., Burnay, C., Snoeck, M., 2020. B-merode: A model-driven engineering and artifact-centric approach to generate smart contracts, in: Conference on Advanced Information Systems Engineering, LNCS-Springer-Verlag.

[27] Syahputra, H., Weigand, H., 2019. The development of smart contracts for heterogeneous blockchains, in: Enterprise Interoperability VIII. Springer, pp. 229–238.

[28] Tran, A.B., Lu, Q., Weber, I., 2018. Lorikeet: A model-driven engineering tool for blockchain-based business process execution and asset management., in: BPM (Dissertation/Demos/Industry), pp. 56–60.

[29] Vujičić, D., Jagodić, D., Ranđić, S., 2018. Blockchain technology, bitcoin, and ethereum: A brief overview, in: 2018 17th international symposium infoteh-jahorina (infoteh), IEEE. pp. 1–6.

[30] Wood, G., 2014. Ethereum: a secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper , 1–32arXiv:arXiv:1011.1669v3.

[31] Yakindu, . YAKINDU Statechart Tools (SCT) – state machine tool. URL: https://www.itemis.com/en/yakindu/state-machine/.