Modelling the Development and Deployment of Decentralized Applications in Ethereum Blockchain: A BPMN-based approach

Nikolaos Nousias[1][0000-0002-6598-2098], George Tsakalidis[1][0000-0002-0889-7946], Sophia Petridou[1][0000-0002-2593-6150] and KostasVergidis[1][0000-0002-2755-499X]

[1]Department of Applied Informatics, University of Macedonia, Thessaloniki, Greece.
{nnousias, giorgos.tsakalidis, spetrido, kvergidis}@uom.edu.gr

Abstract. Decentralized Applications (DApps) have emerged as a new model for building massively scalable and profitable applications. A DApp is a software application that runs on a peer-to-peer blockchain network offering censorship resistance, resilience, and transparency that overcome the challenges of typical centralized architectures. Developing and deploying a DApp in a blockchain network is highly challenging. Developers need to initially decide if developing a DApp is justified, before considering different aspects of blockchain technology (e.g., cryptography, transactions, addresses, etc.). This adversity along with the plethora of previously published works highlight the need for new tools and methods for the development of blockchain-based applications. Throughout literature there is a lack of procedures that can guide practitioners on how to develop and deploy their own applications. This paper aims to address this research gap, by standardizing and modelling such processes, through the employment of the BPMN modelling technique. Initially, a DMN decision model is presented that can facilitate developers to determine whether developing a DApp is justified. Consequently, two BPMN models are introduced, namely the DApp development and the DApp deployment process models. The models can orchestrate new DApp initiatives and facilitate the developers' communication and implementation transparency. We expect that they can serve as a roadmap for enhancing the decision-making in the act of developing a DApp and reducing the implementation time and cost. Finally, we further discuss how the models implement a DApp for the registration and verification of academic qualifications and how BPMN can constitute a powerful tool for the development of DApps.

Keywords: Blockchain, Ethereum, DApp, DApp development, DApp deployment, BPMN, DMN, Modelling, Decision-making.

1 Introduction

Nowadays blockchain has attracted a lot of attention in both academia and industry. A blockchain is a digital immutable ledger of transaction data, shared across a network of untrusted users [1]. The ledger is replicated and synchronized across the nodes without the control of any third party [2]. As nodes broadcast transactions to the network, they

are validated and sealed into blocks using cryptographic primitives to maintain network integrity and avoid data tampering [3]. Once new blocks are appended, a chain of cryptographically linked blocks is established, hence creating the blockchain.

The most prominent application of blockchain is Bitcoin, introduced originally in 2008 by Satoshi Nakamoto [4]. The Bitcoin application constitutes the first generation of blockchains [5] and was initially delimited to the exchange of digital currencies. With the advent of Ethereum [6], the concept of smart contracts was introduced. As a result, the second generation of blockchains emerged, providing a deterministic and secure programming environment for building general-purpose DApps [7]. A DApp is a novel form of the blockchain-empowered software system [8] running on a decentralized peer-to-peer network, such as a blockchain network, and its backend code is employed in smart contracts. The app logic is executed deterministically in the blockchain, while it is guaranteed to be transparent, verifiable, and immutable [9].

As the technology evolves further, more and more applications will be decentralized [7]. However, the development of blockchain-oriented applications poses a new set of challenges that require more sophisticated knowledge compared to the traditional application development approaches [10]. Initially, a decision model is required to determine whether developing a DApp is justified over the development of conventional applications. Moreover, DApps require new ways of thinking about how to build, maintain, and deploy software [9]. Developer support in terms of blockchain applications is limited [11] and thus the development process is considered ambiguous [12] and challenging with a steep learning curve [13]. A plethora of previous works [14–17] highlights the need for specialized tools, techniques, modelling notations, and design patterns for the development of blockchain-based applications.

Traditional software development is enriched with software process models and design decision-making processes [18] that guide developers from the conception of an idea to the realization of the final product. However, to the best of our knowledge, there are no proposed processes and correlation approaches for DApp development and deployment. As of yet, developing a DApp is a composition of decisions which reside in the designers' reasoning and intuition. Hence, we consider the modelling of such processes a novelty and a timely contribution that can serve as a roadmap in new DApp initiatives. By standardizing and modelling such concepts, transparency and communication can be facilitated, resulting in the reduction of the implementation cost and time. In addition, as decision paths are explicitly modeled and documented with model constructs, decision-making can be enhanced. Specifically, developers can decide and enact their following action on the basis of the model logic, circumventing a time-consuming decision of what should be performed next. For this purpose, we employ BPMN as the state-of-the-art process modelling notation [19] and adopt Ethereum as the most popular, and mature blockchain for DApp development [20].

Overall, the purpose of this paper is twofold. Firstly, to propose two standardized processes of developing and deploying a DApp that can facilitate the process-thinking and decision-making of business analysts and software developers alike at their DApp initiatives. Secondly, to shift the discourse towards the applicability of BPMN for designing blockchain-based applications. The rest of the paper is structured as follows. In Section 2 we provide the theoretical background of our study. In Section 3 and 4, we standardize the processes of developing and deploying a DApp utilizing the BPMN. In

Section 5, we communicate the advantages of the proposed models and discuss their applicability on real-world applications. Section 6 concludes the paper, while providing directions for future work.

2 Background

In this section, we define the key concepts that will be discussed through the rest of the paper. Initially, we introduce Ethereum as a blockchain platform that enables the autonomous execution of smart contracts. Subsequently, we discuss what a smart contract is and how DApps treat them as first-class elements. Afterwards, we introduce BPMN as the standard process modelling technique that will be utilized to depict the processes of developing and deploying a DApp atop Ethereum. Finally, we present a DMN-based decision model for determining when the DApp development is justified, as a prerequisite decision before the DApp development and deployment processes unfold.

2.1 Ethereum

Ethereum was conceived by Vitalik Buterin in 2013 [6] as a general-purpose blockchain with a built-in Turing-complete programming language and a native cryptocurrency called Ether. Ethereum's vision was to allow anyone to write their own arbitrary rules in the so-called smart contracts, encoding rules for ownership, transaction formats, and state transition functions [6]. Loading and running the contracts in its distributed state machine (i.e., Ethereum Virtual Machine - EVM) results to state changes that are stored in its blockchain. Each node on the network runs a local copy of the EVM to validate the contract execution. To thwart a smart contact's infinite execution when a node attempts to validate it, Ethereum introduces the gas mechanism to limit the resources that any program can consume [7].

With its deterministic and secure programming environment, the Ethereum platform enables developers to build powerful DApps that constitute the culmination of the Ethereum vision [6]. Ethereum constitutes a protocol that is implemented by independent networks; multiple for testing and one for production. The transactions in a test network do not exchange real-value Ether, making it suitable for initial testing and experimentation. As the most mature Turing-complete programming blockchain, the vast majority of DApps are built atop Ethereum [20].

2.2 Smart Contracts

The concept of smart contracts was introduced by Nick Szabo in 1994 [21], by defining a smart contract as "a computerized transaction protocol that executes the terms of a contract". With the Ethereum foundation, this term was reinforced as "systems that can be autonomously executed and move digital assets according to arbitrary pre-specified rules" [6]. Smart contracts are deployed as data in a transaction and therefore are immutable. Their code can be inspected by every network participant while their execution is deterministic. To trigger their enactment, a message is sent to their address. Their

code activation allows to read and write to their internal storage, send other messages or create contracts in turn.

To deploy a smart contract in a blockchain network, developers encode initially their logic in a high-level programming language (e.g., Solidity). Subsequently, the code is translated into bytecode, before being deployed to the network. In case of Ethereum, the bytecode is executed on the EVM as part of the Ethereum network protocol.

2.3 Decentralized Applications (DApps)

The majority of web-based applications are centralized by design. They are based on client-server architecture, which entails that data processing mostly occurs on a single server hosting environment [1]. With the advent of the second generation of blockchains, DApps have emerged. In their most fundamental format, DApps consist of a web user interface (frontend UI) and a distributed backend (smart contract), which are usually bundled via the web3.js Javascript library [7]. Their backend code is distributed to overcome the challenges that arise from having a centralized server [13], such as low transparency or single point of failure [10]. The on-chain nature of DApps allows everyone to audit their code and inspect their functionality.

Blockchain has shown a great potential in enabling a wealth of DApps, related to games, finance, NFTs, health, energy, and supply chain, among others [20]. However, a major consideration remains the immutability of their smart contract code and their challenging development [13], thus necessitating more intricate and secure techniques in designing blockchain applications.

2.4 Business Process Model & Notation (BPMN)

The BPMN standard [22] is a contemporary notation for capturing business processes in a graphical and executable format. Introduced by OMG in 2006, BPMN has been widely adopted as the de facto process modelling notation in both academia and industry. The primary goal is to provide an understandable notation for various business users (i.e., analysts, developers, managers, etc.) [23]. Previous works have employed BPMN for modelling IoT processes [24], RPA initiatives [25], and blockchain-based applications [26], among others.

Beyond its primary goal of modelling business processes, BPMN models serve as inputs to software development projects [27]. Since modelling is an intrinsic part of designing a software [15], BPMN models are handed over to software developers. System requirements, process flow and decision paths, are specified in a graphical representation before developers translate their logic to code execution.

The application of standard BPMN diagrams for designing and developing DApps has evolved as an intriguing challenge in the aspect of existing blockchain limitations [28]. In particular, the usage of BPMN constructs may prove an efficient method in addressing blockchain usability and complexity issues, due to its well-defined steps and comprehensible notation for the communications between project stakeholders. Bearing in mind the emerging interest of the blockchain community in BPMN, we employ BPMN for the modelling of the DApp development and deployment processes, introduced in sections 3 and 4.

2.5. A DMN-based decision model for DApp development

Developing a DApp is a decision-making process towards committing to optimal decisions in specific time frames. Importantly, an initial decision should be taken whether blockchain application is justified, before design decisions emerge during the DApp development and deployment phases. In this regard, a Decision Model and Notation (DMN) - based decision model (Fig.1) is introduced to guide developers determine whether developing a DApp is a correct decision. DMN [29], emerged as an OMG standard in 2015 for decision modelling, where its primary goal is to provide a common notation for the graphical representation of decisions. The most fundamental constructs are decision nodes, represented by rectangles, and input data, represented by ovals.

In our context, an initial decision path [30] should be followed to justify the blockchain applicability after reviewing the requirements (i.e., use case) and the blockchain peculiarities. Specifically, on the basis of a need for a shared database, the involvement of multiple untrustworthy participants, and the need for disintermediation, the blockchain applicability should be decided. Subsequently, developers need to take design decisions in the act of developing and deploying their own applications. For this purpose, the DApp development and deployment processes unfold in the following sections to standardize the process flow and facilitate the design decision-making.

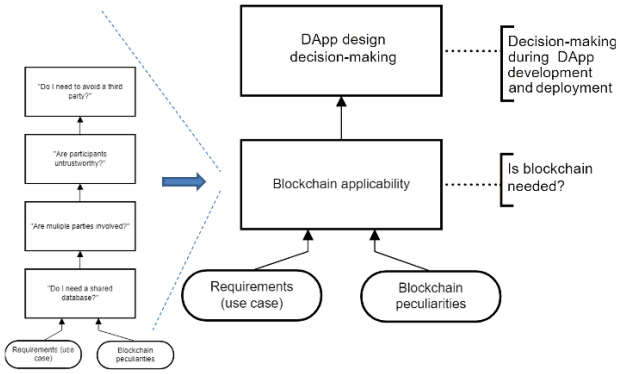

Fig. 1. A DMN-based decision model for DApp development

3 Modelling the DApp development process

This section discusses the DApp development process on the Ethereum blockchain and introduces the respective BPMN model. The aim is to model the flow and the decision paths that a developer should follow while developing a DApp, combining blockchain concepts (transactions, cryptocurrencies, public – private keys, addresses, etc.) in a visual and intuitive manner. Thus, the model can shed light on the process of developing a DApp, become a roadmap for DApp development initiatives, and facilitate the decision-making by explicitly determining a sequence of activities and flow paths.

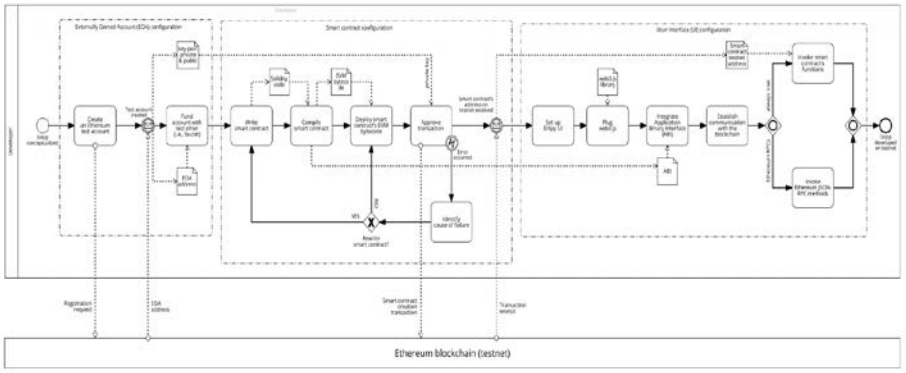

Fig. 2. The DApp development process[1]

Developing a DApp on top of the Ethereum can be regarded as a process with concrete boundaries, a distinct trigger event (e.g., conceptualization of a DApp) and a distinguishable output (e.g., successful DApp development). The DApp development process is modelled in BPMN (Fig. 2) to better conceptualize and standardize the process. Embarking on the DApp development initiative, a developer needs to create an Externally Owned Account (EOA), develop a smart contract, and to configure the interface for the DApp. To communicate better the above three discrete phases, we compartmentalized the process model with BPMN group artifacts. In more detail:

(i) The Externally Owned Account (EOA) configuration aims at the creation of an account to propagate a valid transaction to the blockchain;

(ii) The Smart Contract configuration aims at the encapsulation and the deployment of the backend application to the Ethereum network;

(iii) The User Interface (UI) configuration aims at the establishment of an interface, facilitating the interaction with a smart contract that is operating on the blockchain.

3.1 Externally Owned Account (EOA) configuration

Initializing the development process, a developer should primarily create their own test Ethereum account to be able to interact with the Ethereum blockchain and propagate valid transactions to every node. Specifically, a message (registration request) is propagated to Ethereum, where an EOA is generated (rendered with an intermediate catching message event) and a relative key-pair: private and public (modelled using a data object), is received. Considering that even in a test network, transactions require fees calculated on Ether, a developer is subsequently requested to fund their account. However, due to the testing nature of such networks, developers can reach services (i.e., faucets) that dispense funds in the form of free Ether, instead of buying real-value Ether.

[1] For readability purposes, the DApp development process model has been uploaded to verde.uom.gr/dapp/development.html

3.2 Smart contract configuration

Once the EOA is generated and funded with test Ether, the Smart Contract configuration phase unfolds. At this stage, the developer initially writes a smart contract, typically in Solidity, as the most frequently utilized language for Ethereum smart contracts. Solidity code needs to be further compiled into EVM bytecode to be executed by Ethereum's execution environment, namely the EVM. In this regard, the developer should pass the Solidity code to a Solidity compiler, which in turn produces - as outputs - the EVM bytecode and a contract interface, namely the Application Binary Interface (ABI). Rendered with BPMN data objects, they are further manipulated as the development process unfolds. Thereon, the developer is requested to deploy the previously generated bytecode to the network and approve the smart contract creation transaction. Utilizing their private key, they output a digital signature which verifies that they have the authorization to generate such a transaction. Once successfully created, a transaction receipt is acquired (shown in the model with an intermediate catching message event), indicating the smart contract's address (modelled using a data object) on the Ethereum test network.

However, the propagation of the transaction to the Ethereum network might be interrupted by a plethora of errors. Such errors are mapped with a BPMN error boundary interrupting event, attached to the border of the transaction approval task. The developer should identify the cause of failure and proceed fundamentally either with the code modification or with the gas limit increase.

3.3 User Interface (UI) configuration

Once the smart contract has been successfully deployed on the Ethereum test network, the configuration of the application's interface is the final step. The developer should establish (task: set-up DApp UI) its interface to the external users and configure its core functionality. Thereafter, the web3 JavaScript library and the previously generated contract's ABI (modelled as BPMN input data objects) should be integrated into the application's logic. Specifically, the former enables the programmatic interaction with the Ethereum blockchain, while the latter is a JSON-based description of the available smart contract's functions. This description defines the methods that the application can invoke so as to interact with the deployed contract [7]. Once successfully integrated, the developer is further requested to formalize the interaction with the blockchain, communicating directly either with the deployed contract or with the blockchain itself. Hence, utilizing the formerly generated smart contract's address, its functions can be invoked, while optionally JSON-RPCs (Remote Procedure Calls) can be conducted to query the blockchain-related information (e.g., current block, current gas price, etc.). For this purpose, an inclusive (OR) gateway is utilized. Importantly, the one sequence flow is always triggered (i.e., condition is always true, considering that the interaction with the smart contract is the developer's ultimate aim), while the other one is an optional flow, indicating the conditional invocation of Ethereum RPCs (i.e., condition is 'Ethereum RPCs'). Once the interaction is successfully established, the DApp development process is completed, triggering at the same time the DApp deployment process. The DApp deployment process unfolds subsequently in section 4.

4 Modelling the DApp deployment process

This section introduces the DApp deployment process (Fig. 3), modelled in BPMN. Once the development process is completed, the next step is the DApp migration to the main Ethereum network. The process follows the same process compartmentalization (EOA, Smart Contract and UI configuration). The deployment process should maintain the application's functionality with the minimum number of modifications. For this purpose, the introduced model aims to guide developers on identifying the required changes (e.g., fund account with real Ether, replace smart contract's testnet address with the mainnet one, etc.) and taking consistent decisions, to make their DApps run on a real-value transactions environment.

4.1 Externally Owned Account (EOA) configuration

With the trigger of the DApp deployment process, the EOA configuration phase is initiated. An Ethereum account is applicable to different networks, maintaining the same address with a different balance. Considering that transactions on the main Ethereum network are executed with real-value Ether, the developer is requested to fund their precedently generated account with real Ether.

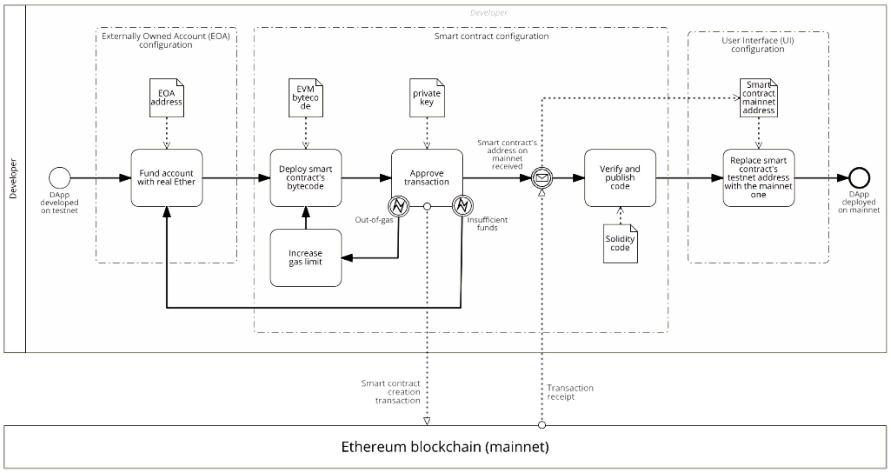

Fig. 3. The DApp deployment process[2]

4.2 Smart contract configuration

Once the EOA is funded with real-value Ether, the developer is requested to deploy the smart contract on the main Ethereum network. At this phase, no code modification is needed given that the smart contract has been successfully developed during the DApp

[2] For readability purposes, the DApp deployment process model has been uploaded to verde.uom.gr/dapp/deployment.html

development process. The developer should exclusively deploy the smart contract's bytecode, generated during the development process, on the mainnet, and approve the contract's creation transaction by signing it with the account private key. Intuitively, the transaction (through a BPMN message flow) is propagated to the main Ethereum network, where a contract account and a relative address is generated. At this stage, it is considered as best practice to verify and publish the contract's Solidity source code, inciting the entire network to entrust its encapsulated process logic [7].

In case of an error occurrence during the propagation of the transaction to the entire network, a recovery (i.e., error-handling) procedure is initialized. At this stage, any code-related errors are typically not expected as they are mitigated at the development process. Any transactional-based errors (e.g., out-of-gas, insufficient funds, etc.) are depicted with interrupting error events, attached to the boundary of the transaction approval task. This acts as a warning to modify the transaction details (e.g., increase the transaction's gas limit) or increase the balance of their account, before redeploying the contract's bytecode.

4.3 User Interface (UI) configuration

Once the smart contract configuration phase is successfully completed, the user interface needs to interact with the contract that is deployed on the mainnet. The process guides the developer to update the smart contract address without modifying neither the application's interface nor its core functionality. As a result, the DApp is able to operate on the main Ethereum network with minimum modifications.

5 Discussion

The process dimension of software engineering is well recognized by researchers and practitioners [18, 31], contrary to the decentralized application software that lacks this perspective. This paper presented the DApp development and deployment process utilizing the BPMN technique. Considering the lack of competing approaches, the paper presents a novelty, offering four major advantages.

First, modelling such processes in BPMN can mitigate any issues arising from their free-form textual description. This approach corroborates Nordsieck's [32] statement that visual models can reveal the notion of a subject matter in a more comprehensive way than any other form of representation. Exploiting the cognitive effectiveness of BPMN [33], developers can intuitively follow the process flow of the models to orchestrate their DApp initiatives. Second, standardizing their flow can make the models serve as an established point of reference, eliminating the necessity of designing, communicating, and agreeing on the software process, each time the development and deployment of a DApp take place. As a result, it is expected that the implementation time and cost can be reduced, while developers' communication and implementation transparency can be improved. Third, decision-making can be facilitated as time-consuming decisions are taken on the basis of the model logic. Being aware of the process control flow, developers identify the forthcoming steps on time, circumventing the need for pondering on their next step. Fourth, the lack of constraints or dependencies in a

particular Integrated Development Environment (IDE) (e.g., Remix, EthFiddle, etc.), or Ethereum client (e.g., Geth, Parity, MetaMask, etc.), enriches the applicability of the proposed processes to all DApp initiatives atop Ethereum. Developers can adhere to their process logic irrespective of the selected tools to implement their applications.

To investigate the applicability of the proposed processes, the authors employed them for the prototype implementation of the VerDe (Verified Degrees) application; a proposed decentralized application for the registration and verification of academic qualifications. As initially presented in [34], VerDe is envisioned as a decentralized application that securely registers and verifies degrees atop Ethereum. The goal is to mitigate fraud and mobility issues inflicted by the current way in which degrees are circulated. Blockchain can serve as a technology enabler for such issues, since it is resistant to the modification of data it holds. Specifically, the VerDe architecture is conceived as a smart contract running on the Ethereum network, offering two distinct user interfaces for writing (i.e., degree registration) and reading (i.e., degree verification) from it. From the conceptual design towards its actual implementation, we followed the previously proposed models. Initially, adhering to the introduced DMN decision model, we decided that developing a DApp is justified, considering that: (i) a shared single source of truth is needed, (ii) universities, students, and companies are involved, (iii) fake degrees are circulated, and (iv) disintermediation from bureaucratic nostrification agencies is required. Subsequently, the introduced BPMN process compartmentalization facilitated the controlled development of the VerDe application in three discrete phases. In detail, we configured our own EOA, developed and deployed the VerDe smart contract, and configured its interfaces to the external world. This approach revealed that planning was promoted ahead of time, allowing the definition and evaluation of each phase's goals. Additionally, issues were detected and fixed quickly, as error-handling procedures were explicitly specified in the models. Thus, implementation time and cost were significantly reduced. Currently, a functional demo of the VerDe platform has been released[3], while the DApp development[1] and deployment[2] processes are publicly available under the same project directory.

Overall, a distinct feature of our work is the employment of BPMN for blockchain modelling. The research conducted in this paper, proved that in contrary to the findings in [28], BPMN diagrams may constitute a useful and efficient method for both the design and development of DApps. What is highlighted is that the interpretation of different blockchain concepts can be achieved through the usage of sophisticated BPMN constructs. Among others, we employed BPMN message flows to model the propagation of transactions to a blockchain network. BPMN error events were introduced to model blockchain failures. Additionally, BPMN message events were utilized to model transaction receipts. The proposed approach can standardize the depiction of such blockchain concepts and inspire researchers to the modelling of their own blockchain-based applications. However, considering that blockchain modelling is recognized as a nascent research domain [35], further research is needed to investigate the applicability of BPMN for the modelling of more complex blockchain concepts (e.g., consensus in the distributed network, mining, etc.).

[3] A functional demo of the VerDe application can be found on verde.uom.gr

6 Conclusion

Blockchain opens an opportunity to create DApps that can benefit from its distributed and immutable nature. Compared to traditional software engineering, their development poses new challenges with respect to different blockchain aspects. As DApp software engineering is still a nascent area, new tools, methods, and design patterns are needed for their optimal development and deployment in a blockchain network.

The work presented in this paper allows developers to follow a systematic step-by-step process for developing and deploying a DApp atop the Ethereum network. For this purpose, a DMN decision model was presented to help developers decide whether developing a DApp is justified. Moreover, two BPMN process models were introduced, the DApp development process model and the DApp deployment. By standardizing and modelling their flow, we expect that these models can serve as a roadmap for DApp developers, while eliminating the need to devise and decide on the process flow each time a new DApp initiative unfolds. To investigate the applicability of our proposed approach, we employed the introduced models for a prototype implementation of a DApp for the registration and verification of academic degrees. Our approach proved to facilitate decision-making and decrease implementation time and cost, advancing further the idea of bringing together process modelling and DApp development.

As future work, we intend to further utilize the BPMN technique for DApp development by exploring how BPMN message flows can define the functionality of smart contracts, as the most critical elements of a DApp. Modelling a smart contract with a BPMN pool, incoming and outgoing message flows can indicate the parameters to be passed and returned from a smart contract's methods. As a result, we plan to investigate how the graphical message flows can be translated to Solidity code, thus establishing a sound communication between a DApp's distributed backend and its external environment.

References

1. Zheng, Z., Xie, S., Dai, H., Chen, X., Wang, H.: An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. In: 2017 IEEE International Congress on Big Data (BigData Congress). pp. 557–564 (2017). https://doi.org/10.1109/BigDataCongress.2017.85.
2. Böhme, R., Christin, N., Edelman, B., Moore, T.: Bitcoin: Economics, Technology, and Governance. Journal of Economic Perspectives. 29 (2), 213–238 (2015). https://doi.org/10.1257/jep.29.2.213.
3. Zhang, K., Jacobsen, H.-A.: Towards Dependable, Scalable, and Pervasive Distributed Ledgers with Blockchains. In: 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), pp. 1337–1346 (2018).
4. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. (2008). http://bitcoin.org/bitcoin.pdf, last accessed 2022/01/22.
5. Xu, M., Chen, X., Kou, G.: A systematic review of blockchain. Financ Innov. 5, 27 (2019). https://doi.org/10.1186/s40854-019-0147-z.

6. Buterin, V.: A next-generation smart contract and decentralized application platform. Ethereum White Paper (2014). https://ethereum.org/en/whitepaper/, last accessed 2022/01/22.
7. Antonopoulos, AM., Wood, G.: Mastering Ethereum: building smart contracts and dapps. first Ed. O'Reilly Media. United States (2018).
8. Cai, W., Wang, Z., Ernst, J.B., Hong, Z., Feng, C., Leung, V.C.M.: Decentralized Applications: The Blockchain-Empowered Software System. IEEE Access. 6, 53019–53033 (2018). https://doi.org/10.1109/ACCESS.2018.2870644.
9. Karger, E., Jagals, M., Ahlemann, F.: Blockchain for AI Data – State of the Art and Open Research. 18 (2021).
10. Cai, C., Duan, H., Wang, C.: Tutorial: Building Secure and Trustworthy Blockchain Applications. In: 2018 IEEE Cybersecurity Development (SecDev). pp. 120–121 (2018). https://doi.org/10.1109/SecDev.2018.00023.
11. Mendling, J., Weber, I., Aalst, W.V.D., Brocke, J.V., Cabanillas, C., Daniel, F., … Zhu, L.: Blockchains for Business Process Management - Challenges and Opportunities. ACM Trans. Manage. Inf. Syst. 9, 4:1-4:16 (2018). https://doi.org/10.1145/3183367.
12. Antal, C., Cioara, T., Anghel, I., Antal, M., Salomie, I.: Distributed Ledger Technology Review and Decentralized Applications Development Guidelines. Future Internet. 13, 62 (2021). https://doi.org/10.3390/fi13030062.
13. Why Model-Driven Engineering Fits the Needs for Blockchain Application Development - IEEE Blockchain Initiative, https://blockchain.ieee.org/technical-briefs/september-2018/why-model-driven-engineering-fits-the-needs-for-blockchain-application-development, last accessed 2022/02/18.
14. Porru, S., Pinna, A., Marchesi, M., Tonelli, R.: Blockchain-Oriented Software Engineering: Challenges and New Directions. In: 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C). pp. 169–171 (2017). https://doi.org/10.1109/ICSE-C.2017.142.
15. Rocha, H., Ducasse, S.: Preliminary Steps Towards Modeling Blockchain Oriented Software. In: 2018 IEEE/ACM 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB). pp. 52–57 (2018).
16. AL-Ashmori, A., Basri, S., Dominic, P.D.D., Muneer, A., Al-Tashi, Q., Al-Ashmori, Y.: Blockchain-Oriented Software Development Issues: A Literature Review. In: Silhavy, R., Silhavy, P., and Prokopova, Z. (eds.) Software Engineering Application in Informatics. pp. 48–57. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-90318-3_6.
17. Koul, R.: Blockchain Oriented Software Testing - Challenges and Approaches. In: 2018 3rd International Conference for Convergence in Technology (I2CT). pp. 1–6 (2018). https://doi.org/10.1109/I2CT.2018.8529728.
18. Sommerville, I.: Software process models. ACM Comput. Surv. 28, 269–271 (1996). https://doi.org/10.1145/234313.234420.
19. Kocbek, M., Jošt, G., Heričko, M., Polančič, G.: Business process model and notation: The current state of affairs. Computer Science and Information Systems. 12, 509–539 (2015).
20. State of the DApps, https://www.stateofthedapps.com/, last accessed 2022/02/01.

21. Szabo, N.: Smart Contracts. (1994). http://archive.is/X3lR2, last accessed 2022/02/10.
22. Business Process Model and Notation Specification Version 2.0.2, https://www.omg.org/spec/BPMN/About-BPMN/, last accessed 2022/02/01.
23. Chinosi, M., Trombetta, A.: BPMN: An introduction to the standard. Computer Standards & Interfaces. 34, 124–134 (2012). https://doi.org/10.1016/j.csi.2011.06.002.
24. Martins, F., Domingos, D.: Modelling IoT behaviour within BPMN Business Processes. Procedia Computer Science. 121, 1014–1022 (2017). https://doi.org/10.1016/j.procs.2017.11.131.
25. Friedrich-Alexander-University Erlangen-Nuremberg, Chair of Digital Industrial Service Systems, Nuremberg, Germany, Hindel, J., Cabrera, L.M., Stierle, M.: Robotic Process Automation: Hype or Hope? In: WI2020 Zentrale Tracks. pp. 1750–1762. GITO Verlag (2020). https://doi.org/10.30844/wi_2020_r6-hindel.
26. Turkanović, M., Hölbl, M., Košič, K., Heričko, M., Kamišalić, A.: EduCTX: A Blockchain-Based Higher Education Credit Platform. IEEE Access. 6, 5112–5127 (2018). https://doi.org/10.1109/ACCESS.2018.2789929.
27. Ouyang, C., Dumas, M., Aalst, W.M.P.V.D., Hofstede, A.H.M.T., Mendling, J.: From business process models to process-oriented software systems. ACM Trans. Softw. Eng. Methodol. 19, 2:1-2:37 (2009). https://doi.org/10.1145/1555392.1555395.
28. Udokwu, C., Anyanka, H., Norta, A.: Evaluation of Approaches for Designing and Developing Decentralized Applications. In: Proceedings of the 2020 4th International Conference on Algorithms, Computing and Systems. pp. 55–62 (2020). https://doi.org/10.1145/3423390.3426724.
29. Decision Model and Notation Specification Version 1.3, https://www.omg.org/spec/DMN, last accessed 2022/02/03.
30. Pedersen, AB., Risius, M., Beck, R.: A ten-step decision path to determine when to use blockchain technologies. MIS Quarterly Executive, 18 (2), 99–115 (2019).
31. Fuggetta, A.: Software process: a roadmap. In: Proceedings of the Conference on The Future of Software Engineering. pp. 25–34. Association for Computing Machinery, New York, NY, USA (2000). https://doi.org/10.1145/336512.336521.
32. Nordsieck, F.: Die Schaubildliche Erfassung und Untersuchung der Betriebsorganisation. Organisation - Eine Schriftenreihe. C. E. Poeschel Verlag, Stuttgart (1932).
33. Genon, N., Heymans, P., Amyot, D.: Analysing the Cognitive Effectiveness of the BPMN 2.0 Visual Notation. In: Malloy, B., Staab, S., and van den Brand, M. (eds.) Software Language Engineering. pp. 377–396. Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19440-5_25.
34. Michoulis, G., Petridou, S., Vergidis, K.: Verification of Academic Qualifications through Ethereum Blockchain: An Introduction to VerDe. XIV Balkan Conference on Operational Research (BALCOR 2020), 429–433. Thessaloniki, Greece (2020).
35. Post, R., Kas, S., Smit, K.: The Role of Modeling in Blockchain Process Design. In: Asatiani, A., García, J.M., Helander, N., Jiménez-Ramírez, A., Koschmider, A., Mendling, J., Meroni, G., and Reijers, H.A. (eds.) Business Process Management: Blockchain and Robotic Process Automation Forum. pp. 52–66. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-58779-6_4.