A Study on ĐApps Characteristics

Ilham Ahmed Qasse
RISE / OpenUAE Research Group
University of Sharjah
Sharjah, United Arab Emirates
iqasse@sharjah.ac.ae

Josef Spillner
Service Prototyping Lab
Zurich University of Applied Sciences
Winterthur, Switzerland
josef.spillner@zhaw.ch

Manar Abu Talib
RISE / OpenUAE Research Group
University of Sharjah
Sharjah, United Arab Emirates
mtalib@sharjah.ac.ae

Qassim Nasir
RISE / OpenUAE Research Group
University of Sharjah
Sharjah, United Arab Emirates
nasir@sharjah.ac.ae

***Abstract*—Repositories are important indicators for liveness and maturity in software development communities. They host user-facing applications or re-usable artefacts to build such applications. While rarely decentralised themselves, they are important for hosting code for decentralised applications. In this study, we investigate public repositories dedicated to decentralised applications, or *ĐApps*, executing on heterogeneous blockchain platforms. The study is the first to report aggregated metrics on the repository-level and application-level characteristics including ĐApps metadata, associated smart contracts composition and inconsistencies between repositories in both schema and content. The main contributions are data acquisition tools and an evolving public dataset along with an initial analysis to derive key metrics in a reproducible way. Insights provided encompass the dominance of Ethereum, the absence of smart contracts for a significant portion of applications, and unused application advertisement potential by absence from popular repositories. The insights can be exploited by developers to build high-quality and highly popular applications and set up corresponding quality checks.**

***Index Terms*—decentralised computing, cloud computing, blockchain, repository, artefacts**

I. Introduction

Software application development has shifted to interfacing convenient sets of application programming interfaces. In recent years, most of these APIs have evolved around managed platforms, primarily in the domain of cloud computing where Platform-as-a-Service (PaaS) has become a major paradigm spanning development, deployment and operation of applications [GDB19]. Following the trend towards smaller computing units, flavours such as Function-as-a-Service (FaaS) have emerged in which each function is executed in a rarely configurable environment in isolation but connected to stateful backends. The resulting simplicity and re-use factor resonates well with developers.

More recently, blockchains have been proposed as attractive application platforms in a similar vein. On top of blockchains, small decentralised applications are engineered in conventional or chain-specific programming languages, executed in isolation apart from a stateful backend, and made available for potential re-use [GD19]. Most global cloud service providers introduced Blockchain-as-a-Service (BaaS or occasionally BCaaS), fully managed on behalf of the users, primarily the application providers. Similar to PaaS, the BaaS model allows to access the blockchain services without the need to maintain the blockchain networks and nodes [OM19]. This lowers the cost required to access the technology and provides better scalability. However, BaaS models are subject to centralisation where any transactions will first go through the blockchain services host platform, thus reducing the advantages typically associated with decentralised platforms. The pricing and business models are also different, including per-node hosting cost and network traffic fees. Still, cloud and blockchain technologies are increasingly fused, such as using cloud elasticity for proofs [PNBT19] and emulation of large-scale public blockchains in few-node clouds [WAYZ19]. This raises questions on how to develop blockchain applications with the same re-use and simplicity advantages known from modern cloud delivery models.

A DApp uses the same technology and programming languages in traditional applications to build the application front end. One difference is that the DApp uses smart contracts to access the blockchain as stateful backend while the traditional application uses APIs to access the database or other backend services.

We study five public repositories for DApps in this paper to produce knowledge on this application model. We also investigate the quality and other characteristics of the available DApps in the market nowadays. Moreover, we will discuss the key challenges in developing DApps. The remainder of the paper is organised as follows. Section 2 discusses the background of DApps. The research method is provided in Section 3. Section 4 gives an inside about the extracted data and metrics on a repository, application and smart contract level. Finally, Section 5 concludes the paper.

Research partially supported by a Young Scientists Mobility Grant 'Quality Assessment of Digital Artefacts for Decentralised Software Applications' of the Swiss Leading House for the Middle East and North Africa, with funds of the Swiss State Secretariat for Education, Research and Innovation (SERI).

II. Background on DApps

A. Application Model

After defining the goal of the application and identifying the actors, the system is divided into two parts, the smart contract system and the front end of the decentralised application. Building the smart contract system includes defining the required data structure, messages, actors and assessing the security of the system. The front end of the DApp consists of designing the UI, architecture, models and the database structure. The main characteristics of the DApp are [LXCL17]:

- DApps are multiparty, where more than one party are participating in the application.
- DApps do not require trusted authority.
- The DApps data are immutable and non-repudiable.
- The DApps data are transparent, but they are not confidential.

In the following, the typical lifecycle of a DApp (design, development, deployment and operation) will be briefly presented.

a) Design and architecture: Lu et al. [Lu19] and Wang et al. [WYW+18] discussed the architecture of blockchain ecosystems. They have divided them into layers. From the proposed blockchain architecture, Zeng et al. [ZZ19] defined the architecture of the DApps into three layers, as shown in Figure 1: Service layer (blockchain platform), contract layer (transaction code referring to blocks), application layer (frontend).

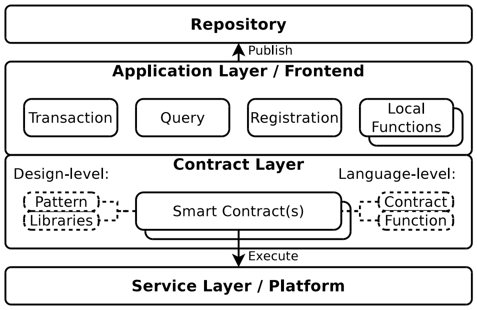

Fig. 1: DApp architecture – refinement of [ZZ19]

DApps can run single or multiple smart contracts to achieve specific functionality. Bartoletti et al. [BP17] and Daniel et al. [DG19] introduced the different design patterns of smart contracts. Bartoletti et al. [BP17] concluded that 80% of smart contracts in Ethereum [W+14] at least used one of the designated design patterns such as oracle pattern or termination pattern. The data from our study will allow for data-driven identification of further patterns and similarities.

b) Development: DApps can be engineered in different programming languages and be deployed and executed on multiple blockchains. The most common combination is represented by DApps written in Solidity, a JavaScript-like language, executing on the Ethereum blockchain. Not all the blockchain platforms support the concept of libraries, inheritance and other complex user-defined types. Ethereum supports importing libraries which are calling other predefined smart contracts and reuse them in the current smart contract. Libraries in Ethereum are stateless where internal variables are not stored. However, deploying libraries will add to the cost of implementing the smart contract or the DApp. The metadata on the DApp level does not differ much from the one in the web applications, as it contains the page's content description, as well as keywords that are linked to the content. DApps showed some vulnerabilities and issues in their code, causing damage [PD+18]. The vulnerabilities are due to that the developers are not mature enough to write a bug-free smart contract. Parizi et al. [PD+18] and Bartoletti et al. [BP17] showed that Solidity programming language has many bugs and security issues compared to other programming languages such as Pact [pac] and Liquidity [liq]. However, Solidity has much support and used by many developers. Angelo and Salzer compared 27 tools for analysing smart contracts in Ethereum [AS19], and improved tooling can be expected.

c) Deployment: In contrast to PaaS, where providers determine the deployment and runtime cost, Ethereum charges a certain amount of *gas* in proportion to the DApp's public portion size, encompassing public functions but not private or constructor functions. The cost of deploying the DApp varies from one blockchain platform to another. Unlike the cloud application, it does not cost anything to keep the DApp in the blockchain. In general, there is a required cost to deploy, update and use the DApp in most of the blockchain platforms. In Ethereum, the developer needs to pay for the gas cost to deploy the DApp. Furthermore, the users need to pay the gas cost to use the application, except for read-only operations. In Ethereum, gas is a transaction fee that measured based on the work required for action to perform. Unlike Ethereum, in EOS [io218], the DApp users do not need to pay for using the application. However, the DApp owner needs to pay to receive storage, bandwidth and CPU for their DApp. Some blockchain platforms have fixed costs for deploying DApps such as Neo blockchain [NEO]. BaaS helps the developers to build their applications without setting up the blockchain network and save them the maintenance that the nodes require regularly. The cost needed to set up the network is reduced using the BaaS. Moreover, BaaS models provide pre-configured infrastructures and networks, which will reduce the development time. However, it does not save the cost of deploying and using the application.

d) Operation: Many DApps support different interaction features using message-based protocols [DG19]. The supported interaction features are transactions, events, message calls and delegate calls. Transactions are used by blockchain users (clients) to create or use existing smart contracts. If the transaction is approved, it will be added to the blockchain and can be accessed publicly. Events are used as response to a transaction to send information to the outside blockchain. Once the transaction is available publicly, the event will also become publicly accessible. Message calls are an interaction protocol where contracts can send messages and interact with each other. The calls are locally executed in the blockchain node without cryptocurrency. Delegate calls are used by the contracts to deploy and use libraries. These calls are also locally executed in the blockchain node without cryptocurrency.

B. Application Repositories

For any software application technology, there is a need to find the most suitable application based on the functional and non-functional requirement. Marketplaces, hubs, application stores and software repositories have already proliferated for mobile phone applications as well as for microservices in cloud environments. Consequently, repositories are also used to categorise and advertise DApps. In addition to the smart contract implementation, they are characterised by developer-provided and runtime-augmented metadata, including a human-readable name, author and development status information, social media contacts, associated transactions and potentially a logo or representative image. Among the most popular DApps repositories are the following ones:

1) State of the DApps with around 2100 DApps, primarily for Ethereum but also for minority target platforms including EOS and Steem.
2) DApp Radar with around 2800 DApps, primarily again for Ethereum but also EOS and TRON among others.
3) DApp.com with around 2700 DApps from seven major blockchain platforms including TomoChain, IOST and Blockstack.
4) DApp Store which is a store for only Ethereum DApps. Currently, the store has over 1500 Ethereum DApps.
5) Universal DApp Store (App.co) with around 310 DApps from multiple blockchain platforms.

The numbers correspond to the last observation on Nov 29th 2019.

III. Research Method

Given the activity on blockchain platforms, cloud blockchain offerings and application repositories, a systematic study will assist software developers in making appropriate design and implementation decisions. We follow an information-centric applied research path by combining public data scraping over a long period of time with automated statistical inference and manual extraction of findings. Our research aims at three levels of information: repositories, applications (DApps) and implementation units (smart contracts).

Figure 2 demonstrates the flow of acquiring the insights into DApps by scraping repositories, aggregating and correlating raw metrics, and performing additional manual analysis. The public repositories for the DApps are dynamic websites which load data after executing JavaScript code. Scraping these websites necessitates to automate the interaction with the browser to perform tasks like scrolling and hovering. We have used Selenium software to automate the interaction, an open-source automation testing software that can be used to automate web pages and applications for testing purposes, in repository-specific Python scripts. After extracting the data from the repositories, the scripts analyze the extracted data and generate plots and figures that describe them. Moreover, the script will compare newly extracted data with previous records to highlight the changes and define newly added and removed DApps.

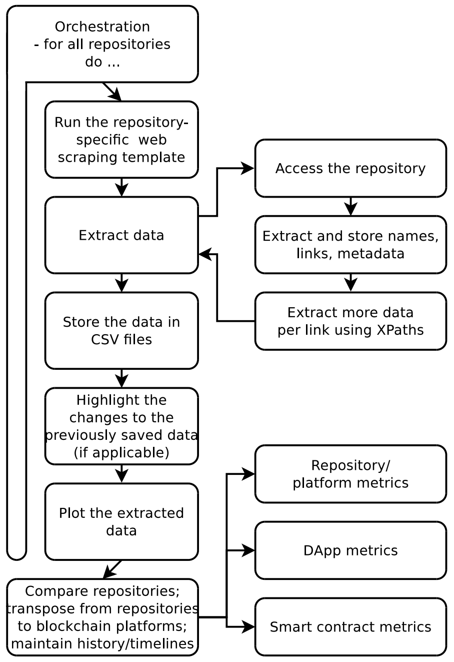
Orchestration
- for all repositories
do ...
Run the repository-specific web scraping template
Extract data
Access the repository
Extract and store names, links, metadata
Extract more data per link using XPaths
Store the data in CSV files
Highlight the changes to the previously saved data (if applicable)
Plot the extracted data
Repository/ platform metrics
DApp metrics
Smart contract metrics
Compare repositories; transpose from repositories to blockchain platforms; maintain history/timelines

Fig. 2: Repositories website scraping flow

IV. Data and Metrics

All study scripts[1] and raw results[2] from which the following metrics and findings have been derived are publicly available.

A. Repositories Analysis

We investigate five public DApps repositories (State of the DApps, DApp Radar, DApp.com, DApp Store and Universal DApp Store) to gain insight into the production and consumption trends as well as technical characteristics of a broad set of DApps and associated smart contracts. Table I shows the different public repositories and the number of DApps in each covered major blockchain platform. Minor platforms (WAVE, ThunderCore, VeChain, NEO, Waves, xDai, ONT among others with only one repository presence) are omitted from the table. The dominance of Ethereum as blockchain platform of choice for most application developers, accounting for about 74.5% of all DApps, is evident. It should be noted that while State of the DApps does not contain multi-platform listings, DApp Radar includes 2.7% and DApp.com 1.9% multi-platform DApps.

Additionally, owing to the dominance of Ethereum, we investigate the authoritative Ethereum database on smart contracts, Etherscan, to gain insights into smart contract implementations. Our preliminary analysis focuses not only on the volume of entries per repositories, but also on the consistency. Comparing the two largest ones, 'DApp Radar' and 'DApp.com', we note that the intersection of unique entries

[1]DApps Scraping: https://github.com/serviceprototypinglab/dapps-scraping
[2]DApps Dataset: https://github.com/serviceprototypinglab/dapps-dataset

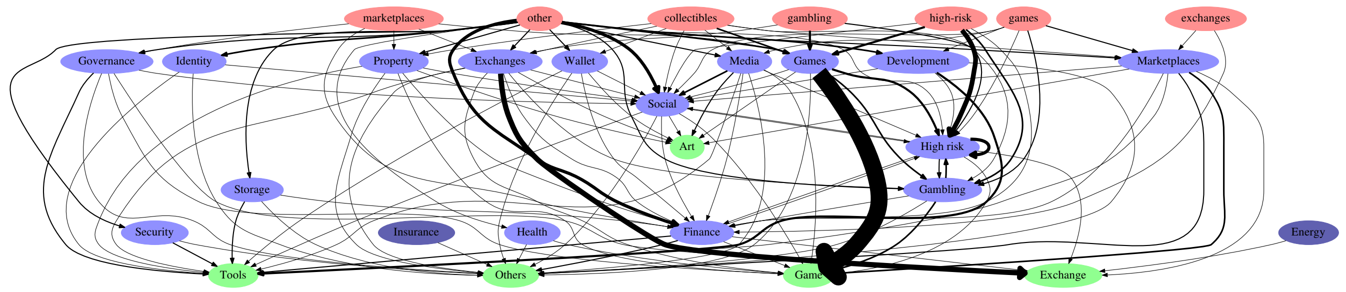

Fig. 3: Differences in categorisation between the 'DAppRadar', 'State of the DApps' and 'DApp.com' websites

TABLE I: Public DApps repositories and blockchain platforms [on Aug 16th 2019]

Blockchain platform	SotD	DR	Dcom	DS	UDS
Ethereum	1855	1557	1384	1516	309
EOS	255	462	336		37
Steem	81		86		29
TRON		407	385		6
LOOM	8	8			
IOST		30	26		

is 1699, or 62.3/64.6% relative to the respective totals. This means that in each of the repositories, around one third of the DApps listed on the respective other one is missing, and a multi-repository abstraction is necessary to gain a full view of available DApps. When including the third-largest repository, 'State of the DApps', the intersection set in DApps is 9.4%. This means that nine out of ten DApps are not properly registered in all of these repositories. Moreover, the use of categories is not consistent. For 1015 DApps of the intersection set of only the two largest repositories (59.1%), different categories are used. Fig 3 shows the mismatches where thicker associative lines mean more DApps are affected. The red categories represent 'DApp Radar', the blue ones 'State of the DApps', and the green ones 'DApp.com'. Dark blue are categories for DApps appearing in 'State of the DApps' but not 'DApp Radar' despite the latter's larger set of entries. Evidently, there is a high correlation between 'Game' and 'Games', but also a high split between 'collectibles', 'gambling' and 'high-risk' to the same 'Games' category. This means that often users will not find the right application even when using multiple portals due to habits on which keyword to use for searching. Moreover, assuming the microservice view on DApps to foster re-use of the underlying smart contract services, software developers will equally be affected.

B. DApp Analysis

In this section, we discuss the extracted data from the repositories. The informal schema for metadata provided in most of the repositories are shown in Table II. We note that the comparison between repositories is thus restricted to metadata keys present in all or at least a qualified majority of repositories. Based on the acquired metadata per repository across blockchain platforms, we have transposed the view to metrics per platform across repositories to identify the platform activity. Specifically, we have calculated the total number of users, transactions, and the total amount of volume, as shown in Table III. We have only covered the blockchain platforms that have more than 20 DApps to eliminate bias. The resulting observation is that there is no proportional relation between the metrics per platform. Although Ethereum has the most significant number of DApps, the number of daily active users and the number of transactions daily are very low compared to the other platforms. The number of active daily users and transactions in EOS are the largest.

We also studied the current DApps status which represents the maturity level specifically for the 'State of the DApps' repository, although it relies on self-proclamation by submitters. Figure 4 illustrates the status. Most DApps are marked as 'live' or usable in commercial operation. The remaining codes are 'prototype', 'work in progress' (WIP), 'beta', and 'concept', with no clear semantics on when which status should be given. Additional codes include 'broken' and 'abandoned' which clearly imply unsuitability for production use but without visible reasoning. Therefore, a deeper analysis in terms of DApps behaviour and reliability would be needed to automate setting a meaningful status. Further, we have aggregated the different categories of DApps within a single repository, again 'State of the DApps', thus notwithstanding the mentioned inconsistencies between repositories. Figure 5 demonstrates the number of DApps in each major category. The dominance of leisure applications (games, gambling, social) over business applications (finance, exchanges, governance) is apparent.

C. Smart Contract Analysis

We also investigated DApps at the level of smart contracts, independently from the repositories, by extracting all the blocks in the Ethereum blockchain and filtering the created contracts from these blocks. The size of the extracted contracts is 23.5GB. Table IV demonstrates the structure of the dataset. We have cleaned the dataset from any irrelevant smart contract such as duplicate, destroyed and token exchange contracts. The count after cleaning is 102,010 as of Aug 16th 2019.

TABLE II: Public DApp repository metadata

Attribute	Description
DApp Name	The name of the DApp
Category	The category of the DApp
Balance	Total amount of cryptocurrencies held in the DApp
Users	The number of daily unique active users
Volume	Total amount of cryptocurrencies transferred to the DApp within the last 24 hours or 7 days
Transactions	The number of transactions made to the DApps within the last 24 hours or 7 days
Development activity	The number of events generated by the project repository in GitHub in the last 30 days (such as code pushes, pull requests, issues, etc)
Status	The status of the DApp
Date	The launch date of the DApp
Social media links	The social media links provided by the DApp

TABLE III: Quantitative analyses for the extracted data

Blockchain platform	# of DApps	Users	Volume	Transactions
Ethereum	1,863	23,832	34,701,610	33,803
EOS	452	29,574	11,564,355	2,531,519
TRON	384	19,803	12,319,051	391,005
Steem	87	16,676	16,582	238,596
IOST	27	5,296	149,512	208,441

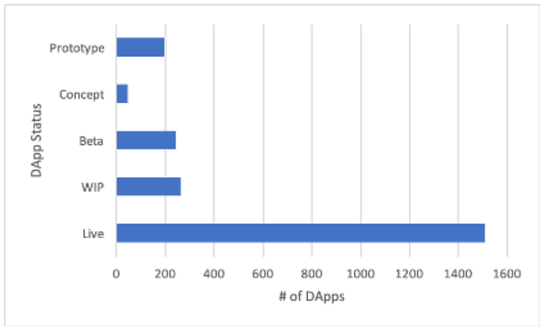

Fig. 4: DApps maturity status

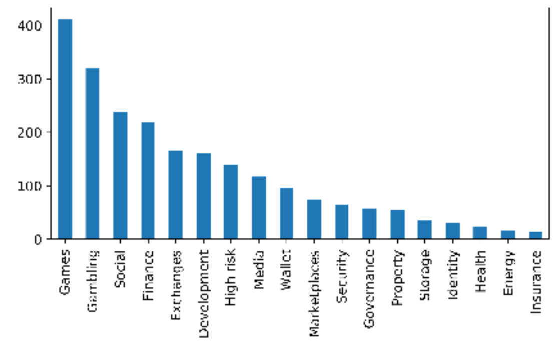

Fig. 5: Number of DApps in each category

TABLE IV: Ethereum smart contracts dataset

Name	Description
Address	Address of the contract
Bytecode	Bytecode of the contract
Function-sig#	4-byte function signature hashes
is_erc20	Whether this is an ERC20 contract
is_erc72	... and/or an ERC721 contract
Block timestamp	Timestamp of the block where this contract was created
Block number	... and corresponding block number
Block hash	... and corresponding hash sum

Moreover, we studied the smart contracts of the top 10 ranked DApps, according to DApp Radar on the basis of number of users, on Ethereum in order to combine the analysis of the blockchain network with the one of the repository. The total number of contracts in these DApps is 58, or 5.8 contracts per DApp, although with significant deviation. The top-ranked DApp, 'My Crypto Heroes', has 25 smart contracts. Drilling down further into the smart contracts requires either direct access to the code, which is provided by some DApps, or decompilation. Etherscan offers built-in decompilation based on the Panoramix decompiler, bytecode to UML transformation and other tools, although the decompilation is limited and lossy. Either way, further analysis is then made possible on the code level, referring to the Solidity programming language in the case of Ethereum-based smart contracts. The average number of actual language-level contracts (classes in object-oriented programming, resulting from using separate .sol files) in each contract block is around 10, again with deviations such as 24 for the first smart contract (0x946048A7) of 'My Crypto Heroes'. Some contracts contain libraries. However, the number of imported libraries is small, as they cost more gas. Each language-level contract is again subdivided into a number of functions as lowest-level units of execution apart from

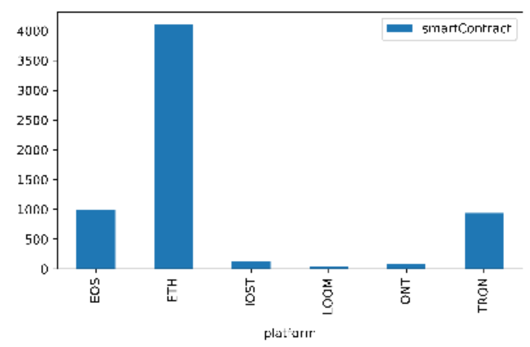

Fig. 6: Number of smart contracts for each blockchain platform

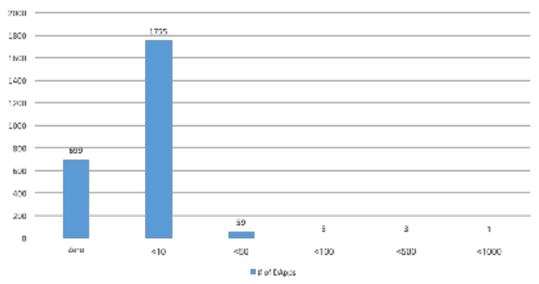

Fig. 7: Number of smart contracts in each DApp

individual statements. Beyond the Ethereum-specific analysis we have studied the total number of smart contracts across all DApps for each blockchain platform, as shown in Figure 6. We also investigated the related number of smart contracts in each DApp, again across all platforms. Figure 7 demonstrates that most of the DApps have far less than 10 contracts. The assumption here is that along with rising maturity of an application, the implementation gains complexity, yet many DApps are for test purposes or in initial development. It also shows that many of the existing DApps do not have any smart contracts, which is surprising given the general architecture of the DApps, but can equally be explained with the immaturity.

V. Conclusion

In this study, we have explored five public repositories of DApps and the characteristics of the available DApps, and discussed the challenges of developing and using DApps. DApps are not mature enough to be adopted like traditional applications. There is a massive difference in active users between DApps, traditional and cloud applications. This discrepancy is due to the lack of promotion tools and the lower user experience provided by blockchain platforms. This decreases the number of potential users and developers. Moreover, there are some issues in blockchain application performance, scalability and security. A negative trait common to blockchain, cloud and cloud-hosted blockchain applications is the vast heterogeneity of technologies, which is typical for a pre-consolidation phase. As future direction for DApps, blockchain platforms should address scalability and performance issues to reduce the entry barrier by not requiring full initial replicas. Moreover, the platforms should improve the available user experience and provide more promotion tools for developers, leading to a higher integration with repository websites.

References

[AS19] Monika Di Angelo and Gernot Salzer. A survey of tools for analyzing ethereum smart contracts. In *IEEE International Conference on Decentralized Applications and Infrastructures, DAPPCON 2019, Newark, CA, USA, April 4-9, 2019*, pages 69–78, 2019.

[BP17] Massimo Bartoletti and Livio Pompianu. An empirical analysis of smart contracts: platforms, applications, and design patterns. In *International conference on financial cryptography and data security*, pages 494–509. Springer, 2017.

[DG19] Florian Daniel and Luca Guida. A service-oriented perspective on blockchain smart contracts. *IEEE Internet Computing*, 23(1):46–53, 2019.

[GD19] Luca Guida and Florian Daniel. Supporting reuse of smart contracts through service orientation and assisted development. In *IEEE International Conference on Decentralized Applications and Infrastructures, DAPPCON 2019, Newark, CA, USA, April 4-9, 2019*, pages 59–68, 2019.

[GDB19] David Gesvindr, Jaroslav Davidek, and Barbora Buhnova. Design of scalable and resilient applications using microservice architecture in paas cloud. In *Proceedings of the 14th Intl. Conf. on Software Technologies, ICSOFT 2019, Prague, Czech Republic, July 26-28, 2019*, pages 619–630, 2019.

[io218] Eos.io technical white paper v2. https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md, 2018.

[liq] liquidity Programming Language. https://www.liquidity-lang.org. Accessed: 2019-07-15.

[Lu19] Yang Lu. The blockchain: State-of-the-art and research challenges. *Journal of Industrial Information Integration*, 2019.

[LXCL17] Sin Kuang Lo, Xiwei Xu, Yin Kia Chiam, and Qinghua Lu. Evaluating suitability of applying blockchain. In *2017 22nd International Conference on Engineering of Complex Computer Systems (ICECCS)*, pages 158–161. IEEE, 2017.

[NEO] NEO White Paper. http://docs.neo.org/en-us/. Accessed: 2019-07-15.

[OM19] Md Mehedi Hassan Onik and Mahdi H Miraz. Performance analytical comparison of blockchain-as-a-service (baas) platforms. In *International Conference for Emerging Technologies in Computing*, pages 3–18. Springer, 2019.

[pac] Pact Programming Language. https://docs.pact.io. Accessed: 2019-07-15.

[PD$^+$18] Reza M Parizi, Ali Dehghantanha, et al. Smart contract programming languages on blockchains: An empirical evaluation of usability and security. In *International Conference on Blockchain*, pages 75–91. Springer, 2018.

[PNBT19] Matthias Pohl, Abdulrahman Nahhas, Sascha Bosse, and Klaus Turowski. Proof of provision: Improving blockchain technology by cloud computing. In *Proceedings of the 9th International Conference on Cloud Computing and Services Science, CLOSER 2019, Heraklion, Crete, Greece, May 2-4, 2019*, pages 523–527, 2019.

[W$^+$14] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project*, 151(2014):1–32, 2014.

[WAYZ19] Xinying Wang, Abdullah Al-Mamun, Feng Yan, and Dongfang Zhao. Toward accurate and efficient emulation of public blockchains in the cloud. In *Cloud Computing - CLOUD 2019 - 12th International Conference, Held as Part of the Services Conference Federation, SCF 2019, San Diego, CA, USA, June 25-30, 2019, Proceedings*, pages 67–82, 2019.

[WYW$^+$18] Shuai Wang, Yong Yuan, Xiao Wang, Juanjuan Li, Rui Qin, and Fei-Yue Wang. An overview of smart contract: architecture, applications, and future trends. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 108–113. IEEE, 2018.

[ZZ19] Yue Zeng and Yue Zhang. Review of research on blockchain application development method. *Journal of Physics: Conference Series*, 1187(5):052005, 2019.