Received 1 October 2022, accepted 17 October 2022, date of publication 25 October 2022, date of current version 1 November 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3216874

SURVEY

Systematic Mapping of Testing Smart Contracts for Blockchain Applications

NICHOLAS PAUL IMPERIUS AND AYMAN DIYAB ALAHMAR, (Member, IEEE)
Department of Software Engineering, Lakehead University, Thunder Bay, ON P7B 5E1, Canada

Corresponding author: Nicholas Paul Imperius (npimperi@lakeheadu.ca)

ABSTRACT In the last few years, the technological future becoming apparent by the introduction of smart contracts into mainstream technology, specifically in the development of Web3 and the metaverse. Smart contracts will play a vital role in the decentralization and autonomy of the day-to-day tasks that must be completed. Several literature reviews, considered secondary sources, highlight the current state of testing methods for smart contracts made for Blockchain applications. In this paper, we present the results from a systematic mapping study to give structure to the information found from primary sources. Systematic mapping is a well-known method to identify and categorize research papers in a field with an increasing amount of literature. For this systematic mapping, we searched for studies between 2017 and present-day (March 2022) and were able to find 303 results, from which 47 were selected, by specific inclusion and exclusion criteria, to be relevant to this study. A concept map was created from the information gathered from primary sources to the attributes such as research type, contribution type, blockchain network, smart contract language, development process, testing methods, and testing environment. We also categorized the trends and demographics found in the selected papers based on publication year, author's country, and more. The results of this systematic mapping showed that this field is very new and quickly increasing with new research. The researchers that are interested in this field could use the results found to create opportunities for their future work.

INDEX TERMS Bitcoin, blockchain, cryptocurrency, digital asset, distributed ledger, ethereum, smart contract, solidity, systematic mapping.

I. INTRODUCTION

In the future, technologies that utilize blockchain technology could have the ability to increase the level of trust and transparency between entities [1]. The security and integrity of the smart contracts used to build these technologies should be of the highest priority. The first blockchain ever created was the Bitcoin blockchain in 2009 [2]. Blockchain technology provides us with the ability to create an agreement between two entities in the form of a smart contract. Compared to Bitcoin, smart contracts have been an idea for quite a long time. In the 1990s, a cryptographer and computer researcher, Nick Szabo, presented the idea of smart contracts as being a transaction protocol completed by a computer [2]. Nowadays, there are many cryptocurrency networks that are trying to be the strongest platform for smart contract development, such as Ethereum, Solana, and Cardano [3]. With an increase in popularity comes an increase of challenges that could arise, such as money loss, scams, and hacking.

The aim of this paper is to study and evaluate recent research available in the literature on blockchain smart contracts and the methods used to test them, their results, and the experimentation used to achieve those results. By comparing the results of the experimentation conducted in the papers, the testing method that achieves the highest level of performance can be determined and evaluated. By finding the best testing method for smart contracts, the overall security and reliability of the smart contract will be increased while minimizing user risk and cost as a result.

In literature about testing smart contracts, the number of primary studies increases year over year due to the increasing demand for this domain; therefore, it is beneficial to study the

The associate editor coordinating the review of this manuscript and approving it for publication was Barbara Guidi.

secondary studies too (i.e., literature review studies). In our literature review, it was found that there were different methods to test smart contracts that vary depending on the intended use case. In addition, there were no papers that indicated how to test smart contracts that would be used on multiple, different cryptocurrency networks, as most testing methods just focused on a single type of blockchain application. Based on these gaps, this paper aims to define a secondary study that we performed with the help of the systematic mapping process as our research methodology. We started our search by looking at articles that are from 2017 to March of 2022 in the following databases (in alphabetical order): ACM, Google Scholar, IEEE Xplore, and ScienceDirect. Of the 303 papers that were initially found, 47 were chosen for a more in-depth review. We created a concept map from the knowledge obtained by analyzing the primary studies on testing smart contracts for blockchain applications and were able to classify the papers based on a series of attributes; specifically, type of research or contribution type, blockchain network used, smart contract programming language, testing method, and testing environment. To the best of our knowledge, this is the only study that performs a more in-depth look into the performance associated with different testing methods that are used to test smart contracts. Furthermore, the key contributions of this systematic mapping are:

- A classification scheme that organizes the current papers related to testing smart contracts for blockchain applications,
- A systematic mapping of the most recent and relevant primary studies (until March 2022) that are surveyed based on the criteria in the classification scheme,
- A study into the trends and demographics of the relevant primary studies,
- Informing future researchers on the gaps present in current literature,
- An examination into the results and challenges of testing smart contracts for future research.

The rest of this paper is organized as follows: Section II gathers the information retrieved from related work and other background information. Section III provides reasoning to our search process, such as how we designed our search query. Section IV classifies how we analyzed and categorized the primary studies. Section V showcases our results and findings. Section VI displays the findings along with the challenges that they present. Finally, Section VII gives the conclusion of the paper.

II. BLOCKCHAIN SMART CONTRACTS AND RELATED WORK

A. OVERVIEW OF SMART CONTRACTS AND BLOCKCHAIN APPLICATIONS

Blockchain technology can be applied to numerous different applications, such as payment or money transfer, supply chain monitoring, insurance claims, copyright infringements, healthcare, and personal identification [4]. In these applications, smart contracts will be a core technology that provides the functionality required to perform the required task. In this paper, the main focus will be on smart contracts and their utility. A smart contract automatically enforces a contract between two entities, with the assistance of a credible ledger and without third party intervention, that will execute based on certain conditions [5]. For instance, a smart contract could be used for a Decentralized Finance (DeFi) application that could support borrowing money. Here, when a user intends to borrow money, a smart contract can be deployed to complete this task. In addition, Non-Fungible Tokens (NFTs) are also built off smart contracts. NFTs are digital assets that could have fiat, real value. These are just two of the increasingly many different use cases for smart contracts. Smart contracts are immutable and once a smart contract has been deployed, it cannot be changed or updated; therefore, testing these smart contracts is crucial to ensure that there are no errors that could arise. The consequences of having an incorrect smart contract could be catastrophic, e.g., a crypto platform that accepts payments in exchange for some product will create a smart contract to handle the payment transaction. This smart contract could have a bug that allows unauthorized users to access the payment funds, traps the funds in the smart contract since an incorrect wallet address was used, or transfer the purchased product to a bad actor's wallet. These are a few examples of the many issues that could arise from having a smart contract that has not been properly tested.

B. RELATED WORK

Although smart contracts have the potential to provide many advantages to future technology, only a select few studies reviewed the evidence on testing smart contracts in blockchain environments. We found seven secondary studies that reviewed the primary studies available in this domain. Table 1 presents the list of the secondary studies that are summarized below.

Macrinici et al. [6] provides a perspective of the problems and solutions for smart contracts within blockchain applications. A trend of increasing publications about this subject was presented within this study. In addition, they found that the most discussed problems and solutions regarded security, privacy and scalability of blockchains along with the ability to program smart contracts themselves.

Sánchez-Gómez et al. [7] discussed the Software Development Life Cycle and testing smart contracts built for blockchain applications. In addition, they found that there was not a clear methodology for validating and evaluating the methods used to test smart contracts as well as the development process. They showed that software developers would continue to create smart contracts that will have bugs or errors, proving to be a costly security vulnerability for the smart contract's customers.

Vacca et al. [8] discussed how the development process for smart contracts is not a standard Software Development Life Cycle which was found to be more error-prone and more

TABLE 1. A list of secondary studies on testing smart contracts and blockchain applications.

Type	Name of Article	Year	Ref
Systematic Mapping	Smart contract applications within blockchain technology: A systematic mapping study	2018	[6]
Literature Review	Model-Based Software Design and Testing in Blockchain Smart Contracts: A Systematic Literature Review	2020	[7]
Literature Review	A systematic literature review of blockchain and smart contract development: Techniques, tools, and open challenges	2020	[8]
Literature Review	Blockchain software patterns for the design of decentralized applications: A systematic literature review	2020	[9]
Survey	A comprehensive survey on smart contract construction and execution: paradigms, tools, and systems	2021	[2]
Survey	A Literature Survey on Smart Contract Testing and Analysis for Smart Contract Based Blockchain Application Development	2021	[10]
Survey	A survey of consensus algorithms in public blockchain systems for crypto-currencies	2021	[11]

resistant to updates. They aimed to show the current issues related to smart contract development processes.

Six et al. [9] discussed how software patterns are useful when designing software to ensure the quality meets expectations. They showed that there are not many software patterns that exist for this domain due to the lack of extensive testing of blockchain applications. They aimed to collect all available and relevant patterns across the literature to help in the pattern selection process for future blockchain application development processes.

Hu et al. [2] explored studies from 2008-2020 that related to the design patterns, design tools, testing methods, and privacy concerns associated with smart contract blockchain applications. They found that there were multiple challenges such as vulnerabilities, inefficient analysis tools, limited complexity, lack of testing, and lack of privacy. These challenges were argued to be the reason for the slow adoption of smart contracts into more applications.

Sujeetha et al. [10] discussed the issues that smart contracts face, specifically reliability, scalability, and security. They argued that the software development life cycle is not followed in the development process of smart contracts. They showcased current testing methods for smart contracts along with existing mutation testing methods that can be used for testing smart contracts.

Ferdous et al. [11] discussed the shortcomings that blockchain systems suffer from in terms of performance and security. They argued that these issues must be resolved for widespread adoption of blockchain applications built by incorporating smart contracts. Their goal was to provide an analysis of all blockchain systems and their functions to gain a greater understanding of the domain.

This study aims to contribute to the secondary studies that are presented in Table 1 because it presents the most recent state of current research in the domain of testing smart contracts based on a classification comprising of attributes attained from primary studies. Forty-seven (47) studies were reviewed and classified based off a classification scheme that was developed during the research process. Furthermore, this study is generalized with the goal of aiding in the creation of a robust classification scheme to use as a reference for structuring the information that would be obtained from primary studies currently available.

III. RESEARCH DESIGN

We applied systematic mapping as our research methodology in this work. This methodology is commonly used in software engineering research with the goal of identifying and categorizing research literature from primary studies, which is quite useful as the amount of literature on the topic is constantly increasing. Performing systematic mapping involves creating a search criterion that can provide good, concrete results when used within a library. This also includes a criterion for the screening and reviewing process so that irrelevant papers can be put aside by using an inclusion and exclusion criteria. A classification scheme can be created to classify the studies. The results from using the classification scheme on the studies with respect to the research topics allow us to analyze the trends, research challenges, and areas of opportunity in the research area.

Fig. 1 shows the systematic mapping process that is used in this paper as derived from the guidelines presented in [12]. The process used consists of the following steps:

- Define research questions (RQs),
- Define a search query, use the search query to search for papers,
- Review all papers that resulted from using the search query, creating a set of papers that are relevant to the topic,
- Use the classification scheme on the keywords found in the searched papers' abstracts and titles,
- Extract data and perform the systematic mapping.

The systematic mapping process began with defining the research questions and the search query that will be used. From using the search query in four online libraries and reviewing the results, we gathered 303 possible papers. To screen the retrieved papers, we developed a criterion to be used to determine what papers are relevant and which are not. This narrowed down the results to 47 highly relevant papers. Afterwards, the classification scheme was used to

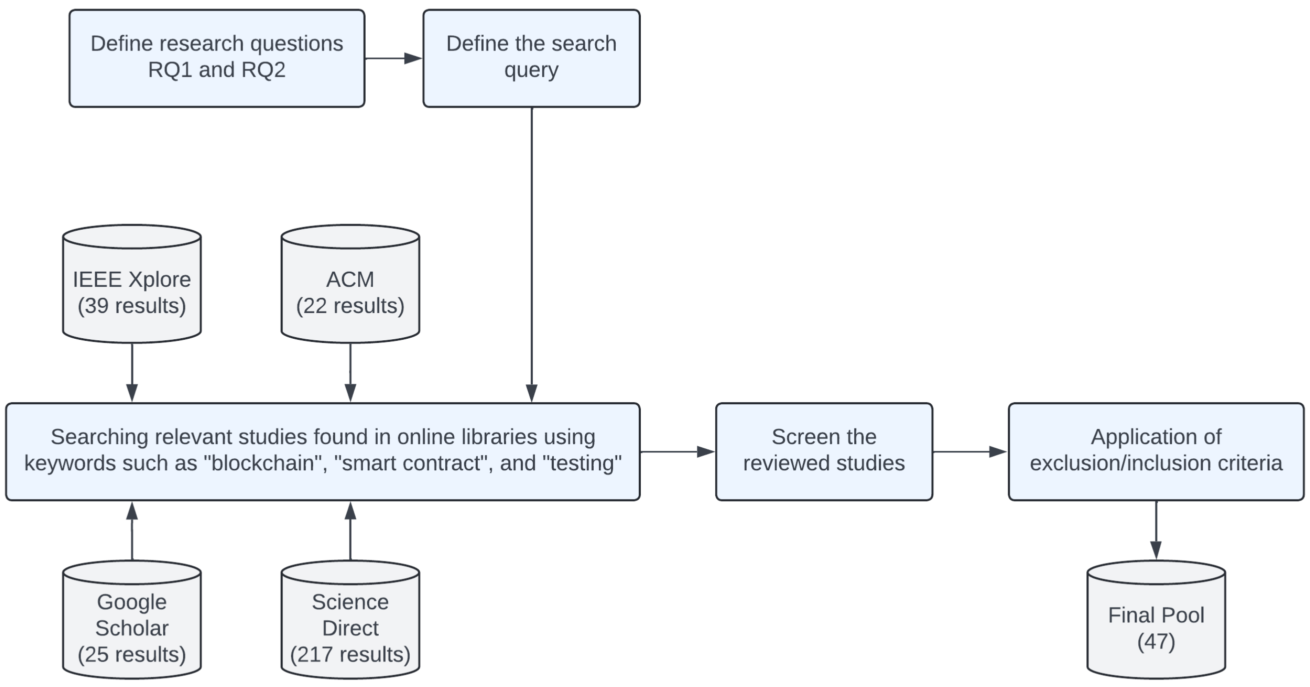

Define research questions RQ1 and RQ2
Define the search query
IEEE Xplore (39 results)
ACM (22 results)
Searching relevant studies found in online libraries using keywords such as "blockchain", "smart contract", and "testing"
Screen the reviewed studies
Application of exclusion/inclusion criteria
Google Scholar (25 results)
Science Direct (217 results)
Final Pool (47)

FIGURE 1. **The systematic mapping process used in this study.**

classify the papers by reading them and extracting terms from their contents (e.g., title, abstract, and body). Finally, once we have the classified data, we can analyze the results of this systematic mapping.

The rest of this section involves the research questions, the paper selection method, and the threats to validity.

A. RESEARCH QUESTIONS

The main goal of this systematic mapping process was to analyze and discuss any previous study on testing smart contracts for blockchain applications. Consequently, two main goals were set in relation to this paper: 1) To systematically review the related papers for mapping in this field and 2) To showcase an analysis of the useful publication trends in this field. With these goals in place, two main research questions were introduced along with several sub-questions to provide a further breakdown and obtain more specific information. Upon analyzing the literature, we produced a set of research questions that aim to help describe the current methodologies and practices being used for testing smart contracts. Furthermore, by analyzing the relevant literature and extracting useful information, we were able to categorize the papers according to the research questions presented below.

RQ1 Systematic Mapping: What is the literature's research space regarding testing smart contracts for blockchain applications? The sub-questions for RQ1 were:

RQ1.1 Research Type: What was the type of research method that was used in the paper? These levels of studies were introduced in [13] to be solution proposal, validation research, evaluation research, and experience papers.

RQ1.2 Contribution Type: What was the main contribution that the paper provides to the research field? Specifically, how many papers provided a method, model, tool, or process?

RQ1.3 Blockchain Network: In which blockchain network was the smart contract built on; e.g. the Ethereum network?

RQ1.4 Smart Contract Language: In which programming language was used to create the smart contract; e.g. Solidity, JavaScript, Rust?

RQ1.5 Test Case Creation Process: Were the test cases created by using automated methods or with Artificial Intelligence (AI)?

RQ1.6 Testing Methods: What testing methods were used to perform the testing on the smart contract? Which methods were used the most?

RQ1.7 Testing Environment: In which environment was the testing conducted in; e.g. emulators, simulators, public test networks, or as security analysis tools?

RQ2 Useful Publication Trends: The following set of sub-questions were formulated by analyzing existing papers:

RQ2.1 Publication Count Sorted by Year: What is the yearly number of publications in the field?

RQ2.2 Ranking of Cited Publications: Which publications have been the most cited by other papers?

RQ2.3 Most Contributing Countries: Which countries have contributed the most papers in this field?

B. PUBLICATION SELECTION PROCESS

We performed our search for papers published in journals and conference proceedings from 2017 to March 2022 in

TABLE 2. Number of studies initially retrieved and selected.

Digital Library	# Initially Retrieved	# Initially Selected	# Uniquely Selected
ACM	22	11	11
IEEE Xplore	39	26	26
Google Scholar	25	14	5
ScienceDirect	217	5	5
	303	**56**	**47**

the online libraries from ACM, IEEE Xplore, ScienceDirect, and Google Scholar. Other databases were considered, such as the Web of Science and SpringerLink; however, it was found that there were a large number of studies that were already retrieved in the aforementioned databases used in this paper. Web of Science retrieves papers in a similar fashion to Google Scholar; therefore, using Google Scholar along with the 3 other databases gives a broad enough spectrum of papers. Furthermore, it is worth mentioning that Google Scholar accesses and retrieves articles from multiple digital libraries, including SpringerLink, which makes this study generic enough to have helpful conclusions.

The following search query was used to collect all papers that would be given a further analysis: (‘‘blockchain’’ OR ‘‘block chain’’) AND (‘‘smart contract’’) AND (‘‘testing’’). These keywords were used because they focused primarily on papers that discussed smart contracts with blockchain technology and consisted of a testing discussion. In addition, using additional keywords like Solidity or Ethereum, had no effect on the number of search results. Table 2 shows the number of studies that were initially found, initially selected, and uniquely selected from using this search query.

By analyzing the title and abstract of the paper we were able to filter the 303 initially retrieved papers as either being relevant or irrelevant to this systematic mapping. If the paper was deemed relevant, then we would do a further in-depth analysis into the body of the paper and perform out classification process on it. For a paper to be considered relevant, it must satisfy the following inclusion/exclusion criteria: (i) a method for testing smart contracts is described; (ii) an analysis of the method is shown; (iii) language must be English; (iv) no duplicate papers between libraries. For instance, if a paper mentioned that a smart contract was tested but gave no further analysis into how the testing was conducted, this would be considered as an irrelevant paper to this study. Criteria (i)-(iii) resulted in 56 potential papers; 26 were from IEEE Xplore; 11 from ACM; 5 from ScienceDirect; and 14 from Google Scholar. When applying criteria (iv), 47 papers were uniquely selected for a more thorough analysis, as will be shown in the remainder of this paper.

C. THREATS TO VALIDITY

We systematically selected and assessed any threats to the four main types of validity in our research based on the guidelines studied in [14] and [15]. In addition, we describe the steps that were taken to lower the risks of these threats, as shown in [16]. The four types of validity are as follows: Internal, Construct, Conclusion, and External Validity.

1) INTERNAL VALIDITY

If we limit the search terms and the number of different search engines that we use to search, we can end up with incomplete sources. To avoid this situation, a specific set of keywords were used along with a manual review process for each study in the search results. To minimize the possible threat that may occur from using a search engine, an online scholarly database that has access to many libraries from which we can search for papers. An example of this was the use of Google Scholar which is a comprehensive academic database.

2) CONSTRUCT VALIDITY

In this study, construct validity refers to how suitable the research questions are and the categorization scheme that was used for extracting data from the papers. The research questions have been engineered for the specific goal and different aspects of testing smart contracts.

3) CONCLUSION VALIDITY

We must ensure that the methods used to review and analyze the data is reliable. This can be done by following the systematic mapping approach procedures so that this mapping can be replicated without significantly different results.

4) EXTERNAL VALIDITY

When using the search terms described earlier in this paper, all the results were written only in English; however, there is an issue with the fact that English is not the only language that contains papers in this field. We do consider that all information found represents a well-enough representation of the overall field.

IV. CLASSIFICATION SCHEME

To perform the systematic mapping process, a robust classification scheme had to be created. Table 3 shows the different research questions along with their respective attributes of concern, the possible classification types, and descriptions of each classification type. Each row of Table 3 is explaining the RQs possible types of answers which provides a further breakdown of the data shown in Section V. This classification scheme is important in ensuring the replicability of this study and that the results are unambiguous.

As discussed in [13], a research facet represents the different types of research approaches that are used in this paper. RQ1.1 relates back to this. The following facets were analyzed: solution proposal, validation research, evaluation research, and experience paper. A solution proposal is a paper that provides a new solution. If the paper contains weak testing of the hypothesis by using empirical evidence, this would be validation research. If the papers are studied extensively with empirical methods and have the advantages and

TABLE 3. **Classification scheme.**

RQ No.	Attribute of Concern	Possible Answers	Description of Answers
RQ 1.1	Research Type	Solution Proposal	Description of a new solution and showing its applicability by an example.
		Validation Research	Virtually not implemented, new methods that are used for sample experiments or weak empirical studies.
		Evaluation Research	Evaluation of the methods for implementing a testing solution and the evaluation of this solution.
		Experience Paper	The author's personal experiences in implementing testing methods for smart contracts.
RQ 1.2	Contribution Type	Method	New method is proposed.
		Model	New model is proposed.
		Tool	New tool is proposed for this domain and is implemented.
		Process	Definition of the new steps taken for testing smart contracts.
RQ 1.3	Type of Blockchain Network	Ethereum Not Specified	The blockchain network that the smart contract will be tested on.
RQ 1.4	Type of Smart Contract Language	Solidity JavaScript Rust Not Specified	The programming language used to create the smart contract.
RQ 1.5	Type of Test Case Creation	Automated Manual Not Specified	Were the test cases designed for testing the smart contracts created with AI or automated techniques.
RQ 1.6	Type of Testing Method	Mutation Testing Validation Testing Fuzz Testing Acceptance Testing SOCRATES Automation Testing Unit Testing	The testing method performed on the smart contract.
RQ 1.7	Type of Testing Environment	Emulators Simulators Public Test Networks Security Analysis Tools	The testing environment that the smart contract was tested in.
RQ 2	Trends and Demographics	Author	Name of all authors.
		Authors' Country	Country of the author.
		Year	The year of publication, grouped by count.
		# Of Citations	The most cited papers.

disadvantages outlined, it is classified to be evaluation research. Lastly, if the paper solely studies applications or experiences, these are called experience papers.

RQ1.2 deals with the type of contribution that the paper provided; this relates to the following: method, model, tool, process, or other [13]. For instance, some papers showed new testing methods, while some showed the use of an existing testing method. Furthermore, the following contribution facets were used: method, model, tool, or process.

RQ1.3 deals with the different blockchain networks that the smart contract would deploy on. This is important since each blockchain network has a different architecture. By understanding how different testing methods operate across a series of networks, we can compute the average performance for each one.

RQ1.4 concerns the programming language with which the smart contract will be constructed. There are smart contract-specific languages, e.g., Solidity, and existing languages, e.g., JavaScript, that also could be used.

RQ1.5 concerns the method by which the test cases were created for testing the smart contracts. Specifically, were the test cases created manually or by automation software or AI? The answer is either automated, manual, or not specified. Automating the test case creation process allows for a broader range and an increased number of test cases compared to manually creating each one. In addition, is it more time-efficient to have an automated method create them instead of doing it manually? This could lead to a decrease in the number of errors that are found once the smart contract has been implemented.

RQ1.6 follows the studies' different types of testing methods. Many papers focused on mutation testing or fuzz testing while others studied the more traditional validation and acceptance testing. Mutation testing is a fault-based testing method [17]. Fuzz testing is providing invalid or incorrect input data to the smart contract to observe its quality [20]. Validation testing ensures that the smart contract provides the use case required. Automation testing is using software to execute the test cases on the smart contract. Lastly, acceptance testing is confirming the smart contract meets the requirements set out by the owner of the smart contract.

RQ1.7 concerns the environment in which the testing took place. Some studies showed that an emulator was used, while others attempted to simulate the smart contract's behaviour. It was also found that they could have been testing on public test networks or even with security analysis tools. The Ethereum blockchain is designed around smart contracts, and to help developers create and test their smart contracts, the Ethereum Virtual Machine (EVM) was created for this sole purpose. Using an emulator to create a test environment is another method that developers could use to test their smart contracts. Simulators provide a similar effect to emulators; however, in a simulator, the developer has control over different variables in the environment, which allows for the testing of more specialized scenarios.

In RQ2.1 to RQ2.3, the useful publication trends and demographics with respect to the types described in each research question were answered. Specifically, the year, the number of citations, and the researchers and countries that have contributed to research in this field.

V. SYSTEMATIC MAPPING RESULTS

In this section, the results of the systematic mapping are showcased in relation to RQ1 and RQ2. For reference, all papers that were used in this study can be found in the excel spreadsheet located at: https://bit.ly/35zQ38q

A. RQ1 SYSTEMATIC MAPPING

1) RQ1.1 RESEARCH TYPE

Fig. 2 shows the different types of research that were found in the 47 articles that were uniquely selected. The majority (51.1%; n = 24) were categorized as solution proposal, then evaluation research (25.5%; n = 12), followed by validation research (21.3%; n = 10), and experience paper (2.1%; n = 1). Considering the research content, just over half of the selected papers were classified as solution proposals. This shows that, in this field, many authors are attempting to bring new solutions to help test smart contracts appropriately

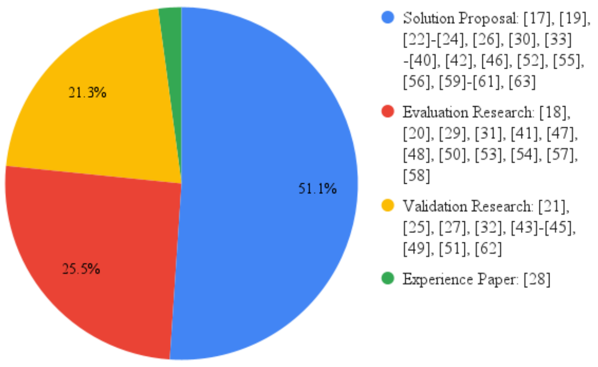


FIGURE 2. Distribution of studies by research type.

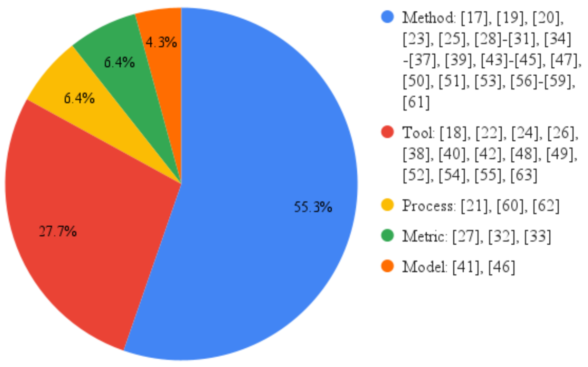


FIGURE 3. Distribution of studies by contribution type.

and effectively. It is worth noting that there are not many experience papers since this domain is quite new, and there has not been enough time for individuals to gain experience in this domain.

2) RQ1.2 CONTRIBUTION TYPE

Fig. 3 illustrates the distribution of contribution types for the uniquely selected papers. As shown in Fig. 3, the majority (55.3%; n = 26) of the papers have been classified to contribute the method type, then tool (27.7%; n = 13), process (6.4%; n = 3), metric (6.4%; n = 3), and model (4.3%; n = 2). From these results, we can conclude that the key contribution that the papers are providing relate to new methods in the field of testing smart contracts. We can see that some new tools have been introduced by authors as well. Since the testing of smart contracts is quite new, for there to be a larger amount of method contribution types is not surprising.

In Table 4, a cross-analysis of the research and contribution types is shown. In this table, we can see that the combination with the highest occurrence (n = 12) are solution proposal and method with the second highest count of combinations (n = 9) was solution proposal and tool. The cross analysis illustrates that the majority of solution proposals contributed either a new method or tool.

TABLE 4. Cross analysis of research and contribution type.

Research Type	Contribution Type	Count
Solution Proposal	Method	12
Solution Proposal	Tool	9
Evaluation Research	Method	8
Validation Research	Method	5
Evaluation Research	Tool	3
Validation Research	Metric	2
Validation Research	Process	2
Evaluation Research	Model	1
Experience Paper	Method	1
Solution Proposal	Metric	1
Solution Proposal	Model	1
Solution Proposal	Process	1
Validation Research	Tool	1

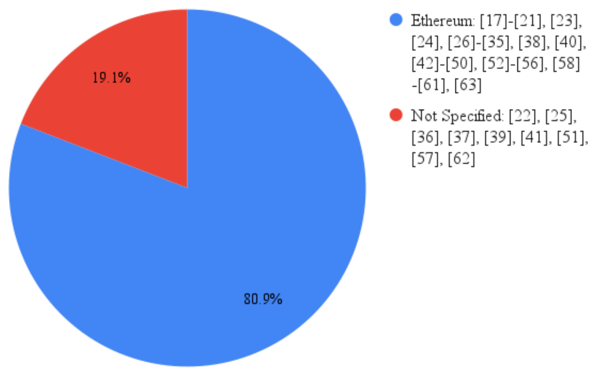


FIGURE 4. Distribution of the blockchain networks.

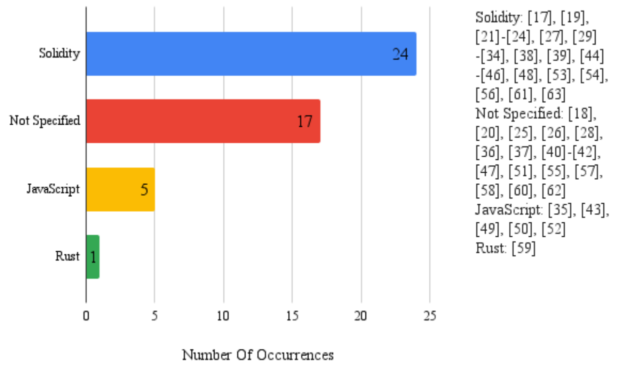


FIGURE 5. Distribution of the smart contract programming languages.

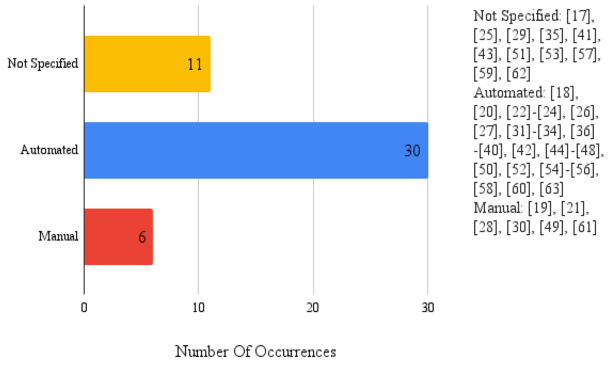


FIGURE 6. Distribution of type of test case creation process.

3) RQ1.3 BLOCKCHAIN NETWORK

In Fig. 4, the distribution of the different blockchain networks is shown. From the papers researched, the testing was either done on the Ethereum blockchain or was not specified. We can conclude that the Ethereum network is one of the easiest networks to build smart contracts on. It is worth noting that newer networks, e.g., Solana, IOTA, and Assembly, are all becoming increasingly popular for smart contract applications.

4) RQ1.4 SMART CONTRACT LANGUAGE

The different smart contract languages must be considered too. In Fig. 5, the distribution of the different smart contract languages is shown. Solidity is the most popular ($n = 24$) smart contract programming language, largely due to that it has a suite of tools to aid in smart contract development. JavaScript was another language ($n = 5$) that was found in the literature, followed by one paper that used Rust. It was also found that many did not specify a language for the smart contract to build on; this could be that their testing method was not specific to individual programming languages or blockchain networks but rather a generalized process. Since the most common blockchain network was Ethereum, we can assume that the majority of development on the Ethereum network uses the Solidity programming language.

5) RQ1.5 TEST CASE CREATION PROCESS

From the literature, we can review each paper to determine if the test cases that were used were created manually or with assistance from automation software or AI. Fig. 6 shows that most ($n = 30$) of the papers used automated methods to create test cases for their testing method; however, the other portion of the papers reviewed were either not specified ($n = 11$) or created manually ($n = 6$). With the majority using automated methods, we can conclude that this is the best possible solution for this process, likely due to the increased efficiency in creating a large number of test cases for boundary cases.

6) RQ1.6 TESTING METHOD

Testing smart contracts is the primary focus of this paper; therefore, it is important to gather information from the literature about the different testing methods that could be used. In Table 5, the distribution of the different testing methods is shown. Fuzz testing ($n = 12$) and validation testing ($n = 11$) were the most popular testing methods found in the literature. Followed by mutation testing ($n = 8$), unit testing ($n = 7$),

TABLE 5. Testing methods used by the reviewed studies.

Testing Method	# Of Papers	Papers
Fuzz Testing	12	[20], [21], [23], [24], [26], [41], [42], [48], [53]-[56]
Validation Testing	11	[22], [27], [32], [33], [38], [39], [44], [46], [47], [57], [63]
Mutation Testing	8	[17], [19], [29]-[31], [34], [45], [61]
Unit Testing	7	[36], [43], [49], [51], [59], [60], [62]
Automation Testing	6	[35], [37], [40], [50], [52], [58]
Acceptance Testing	2	[25], [28]
SOCRATES	1	[18]

and automation testing (n = 6). The least occurring testing methods were acceptance testing (n = 2) and SOCRATES (n = 1) which is a modular testing framework that relies on interacting bots that play a distinct role. The bots are ways for smart contracts to exhibit different behaviours since they can play different roles [18]. Furthermore, mutation and fuzz testing are new testing methods that performed well for smart contract testing. Unit testing is also a proven method that was commonly used in the researched papers too.

In [39], mutation testing was defined as purposefully inserting small, artificial defects into the code, creating variants called mutants. These variants would be based on a set of operators called mutant operators. This testing method is very effective in ensuring the code does not contain any errors; however, in [24], the authors explain that mutation testing is unable to simulate all possible faults that could arise in real world scenarios but instead focus on simpler faults that can represent the entirety of all faults. To conclude, mutation still proved to be a powerful method that showcased strong performance when testing smart contracts.

New methods were proposed that introduced fuzz testing or fuzzing. Ji et al. [15] suggested a new method called Targy, which is an efficient, targeted mutation strategy to better solve the issues that arise with nested conditional statements that have a big depth to them. By using their new method, Targy, they were able to increase the coverage of the fuzzer. In the end, new methods such as mutation and fuzz testing have gained more interest compared to traditional methods, such as unit testing, and continue to increase the performance of the applications made with smart contracts.

7) RQ1.7 TESTING ENVIRONMENT

The final research question from RQ1's sub-questions is about the testing environment. The testing environment is important since any errors could potentially create a cascading effect that could affect other applications on the network too. In Fig. 7, most of the environments (38.3%; n = 18) that the smart contracts were tested in were public test networks. Followed by security analysis tools (31.9%; n = 15), emulators (23.4%; n = 11), and simulators (6.4%; n = 3). It shows that public test networks along with security analysis tools provide the best experience and feedback when conducting tests on smart contracts. Simulators do not seem to be a good environment to conduct the smart contract testing procedures.

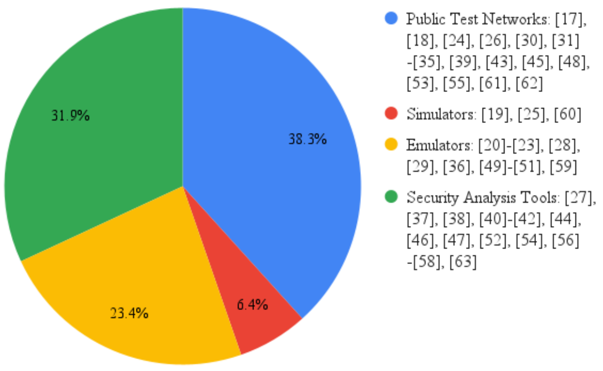


FIGURE 7. Distribution of the testing environments.

B. RQ2 SYSTEMATIC MAPPING

1) RQ2.1 PUBLICATION COUNT BY YEAR

Fig. 8 displays the annual trend of cumulative publication counts over time for the relevant studies found from 2017 to March 2022. Note that Fig. 8 starts at 2018 instead of 2017 because there were no relevant papers to this topic in 2017 (the same trend is observed in Fig. 9 to come later). Interestingly, papers that were published in earlier years were more commonly found to be solution proposals and in later years, there were more validation and evaluation research papers being published. Considering these papers are gathered over the past five years, the amount of literature is consistently increasing over time, especially over the last couple years. We can assume this upward trend in the number of papers reported will increase for many years to come due to the increasing interest in this domain.

2) RQ2.2 RANKING OF CITED PUBLICATIONS

In this research question, the top-cited publications were identified on a scatter plot organized by the publication year in Fig. 9. The citation data was gathered from Google Scholar for every paper that was reviewed in this systematic mapping. As shown in Fig. 9, the number of citations per year trended downwards as the publication year increased, most likely due to their recent publication time. Having a high number of citations also shows that the paper is highly influential in the academic community.

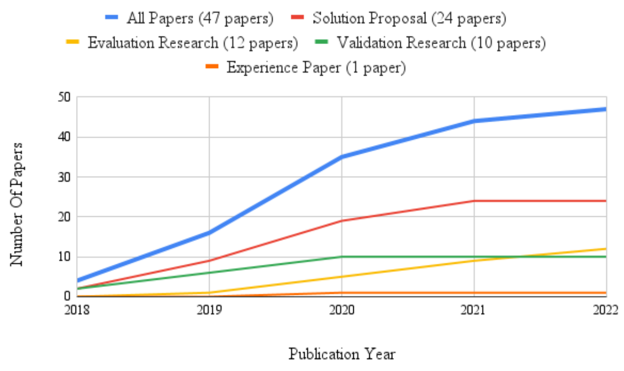


FIGURE 8. Annual trend of cumulative publication counts by research type.

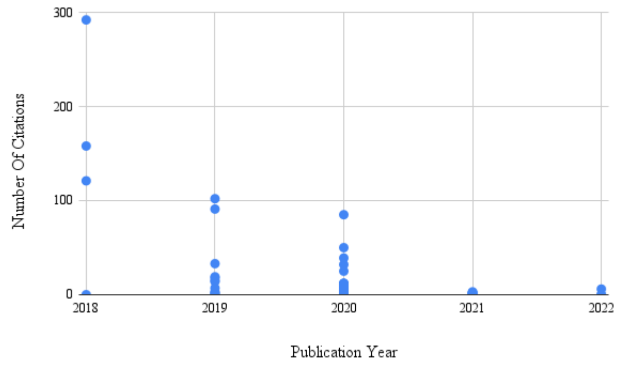


FIGURE 9. Citations by publication year.

3) RQ2.3 MOST CONTRIBUTING COUNTRIES

To select the country of a paper, we looked at the university/organization location of all the papers' authors and assigned a score based on the country (i.e., if all authors of a paper are from the same country, that country is given a score of one (1), if the paper has two authors from two different countries, each country is assigned a score of 0.5, and so on). As shown in Fig. 10, China was the most contributing country with a total score of 16.17. USA was the second with a total score of 5.32, then Canada and United Kingdom with total scores of 3.27 and 3.17, respectively. Of the 47 papers that were reviewed, 33 had all authors from one country while the remaining 14 papers have authors from more than one country.

VI. SUMMARY OF FINDINGS AND CHALLENGES

A. SUMMARY OF FINDINGS

This systematic mapping process aimed to answer various research questions in the domain of testing smart contracts. The findings for each RQ are stated as follows:

RQ1.1 (Research Type): In our findings, 51.1% of the reviewed papers were solution proposals, followed by 46.8% consisting of either evaluation research (25.5%) or validation research (21.3%). Only 2.1% of papers were based on experience. This result confirms the lack of maturity of research methods in this field since most papers were proposing new solutions; therefore, research needs to head in the direction of evaluating these new proposals.

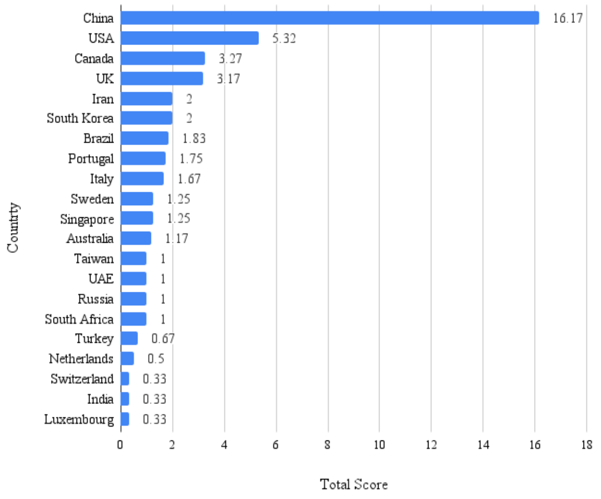


FIGURE 10. Distribution of countries in the relevant papers.

RQ1.2 (Contribution Type): Most papers contributed a newer method (55.3%), although it was poorly validated. The remaining 44.7% consisted of new tools, new validation processes, and new metrics. The ratio from this study is mostly positive. Again, since this is a new domain, most reports were focused on the introduction of new methods compared to validating previous ones by looking at empirical results. This confirms that more validation research is needed in this domain.

RQ1.3 (Blockchain Network): Ethereum is one of the largest cryptocurrency networks. One of the factors that entice developers and other users to use the Ethereum network is the large number of applications and smart contract capabilities; therefore, 80.9% of reports designed their testing methods for testing smart contracts on the Ethereum network. Interestingly, no other networks were mentioned, even though there are others that incorporate smart contracts. As more blockchain networks are created, the dominance of Ethereum may lower as developers may seek other options that could provide benefits in other areas compared to the Ethereum network.

RQ1.4 (Smart Contract Language): Research into the specific programming language that the smart contract was built with is important in understanding how to perform white-box testing since each programming language has a different syntax associated with them. It was found that the majority (51.1%) of the papers were using the Solidity programming language, which is specifically designed for smart contract development. In a recent paper [59], the Rust programming language was used. Rust is a relatively new programming language that is gaining popularity for its high performance and safety [64], [65] which makes it a good choice for smart contracts. It is noted that the smart contract programming language was not specified 36.2% of the time. When the smart contract language was Solidity in the papers, the blockchain network was Ethereum rather than not specified. Solidity is the primary programming language for the Ethereum network. This shows that the best-performing programming language for most reviewed applications is Solidity.

RQ1.5 (Test Case Creation Process): Creating test cases with the assistance of automated or AI-driven software can prove to be very valuable for testing. It was found that most papers used this type of test creation process in their studies. The benefits of automated test case creation are: can easily be updated with feature updates or code changes, can exhaustively test the entirety of the system, and ease of use; however, if the system is not very complex, manually creating test cases is not a bad idea, but in the field of smart contract development, every solution is quite complex.

RQ1.6 (Testing Methods): The testing method is important in discovering any errors in the smart contract before deployment. 48.9% of the papers were either fuzz testing or validation testing, while mutation testing, unit testing, and automation testing contributed to 44.6% of reported papers. From this, we can see that the fuzz testing and validation testing have performed the best for most cases.

RQ1.7 (Testing Environment): The most common environment (38.3%) was a public test network. The Ethereum network has a unique derivative called the Ethereum Virtual Machine, or EVM, which is a testing environment for developers. Security analysis tools were the next popular choice found at 31.9%. To get the best performance, the smart contract must be tested in an environment that resembles the production environment as much as possible to ensure that the smart contract performs well; therefore, the best environment to test a smart contract on the Ethereum network will be a public test network such as the EVM.

RQ2 (Trends and Demographics): The cumulative annual publication volume of testing smart contracts papers showed that there is an increasing interest in this field. There were almost 300 citations of [56] with five papers being cited around 90-160 times. The most contributing country was China, with approximately 34.4% of the papers. In China, blockchain and cryptocurrency have been popular due to the low costs of electricity for cryptocurrency mining purposes, while in other countries, the field of smart contract development and testing is quite new.

B. CHALLENGES OBSERVED

From the observations found in the reviewed papers, we identified two challenges related to testing smart contracts for blockchain applications. Firstly, insufficient data gathered from testing smart contracts. Secondly, many testing methods that have few supporting studies. A summary of each is below.

1) TESTING DATA CHALLENGES

From the papers in this study, it was found that there are many ways to design testing methods for smart contracts as well as where to test them. There is not a large library of research behind which methods prove to be the best here; thus, the main challenge in research currently is the lack of experience-based knowledge in this domain. Over time, with more research being conducted, this will change, and the research will start to focus on key methods.

Dealing with the errors or delays associated with testing methods that do not properly suit the smart contract's purpose could lead to bugs being overlooked, which would result in a poorly built smart contract. That is why it is crucial for developers to understand what works the best and for what purpose they are designed. In the papers [17], [19], [29], [30], [31], [34], [45], and [61], a newer method called mutation testing was introduced for the specific purpose of testing smart contracts. In addition, in the papers [20], [21], [23], [24], [26], [41], [42], [48], [53], [54], [55], and [56], another method called fuzz testing was introduced as well.

There is already preliminary research into new testing methods that are better suited for testing smart contracts. With time, these methods will be rigorously tested to ensure they are the best, or new ones will rise and be the best.

2) RESEARCH RELATED CHALLENGES

There are many ways to test smart contracts, as shown by the papers in this study. There were many papers (n = 14) proposing new solutions specifically in this domain. Having an influx of new solutions is good, but it can also lead to issues for potential users of the methods. There must be a place for the supporting evidence and comparisons between methods to ensure the highest quality. Creating strong benchmarks for more solutions to be proposed remains a challenge.

VII. CONCLUSION

With smart contract-enabled blockchain applications most likely impacting our future lives, their reliability must be proven. This study aimed to provide an analysis of the related literature around the testing methods used for smart contracts in blockchain applications and help identify any gaps in the current literature by applying the systematic mapping procedure to the studies in this domain. In this study, 47 most relevant studies were selected from an initial 303 results. These articles were analyzed with respect to their research and contribution types, the type of network and language they were built with, the methods in place to test the smart contracts as well as new ones, and the trends and demographics of the selected papers.

The mapping of the selected papers showed that the domain is very new, and the amount of literature is increasing year over year. Many papers reported on new solutions for testing methods. Furthermore, additional studies into the evaluation of these new solutions will prove vital for future research. There appeared to be certain areas of the research/industry that were monopolized by an attribute or one technology. For example, the blockchain network that was mentioned was only Ethereum.

Many of the papers proposed new methods, and few introduced new tools, processes, or metrics. There was a variety of testing methods already in place, and development processes ranged from more traditional to more modern such as Agile development. These facts showcase the ever-changing, ever-growing nature of the literature in this domain.

There are opportunities to introduce new testing methods in our studies as the research indicated four new methods such as mutation testing, fuzz testing, automation testing, and SOCRATES. There is a need for more research to validate the current research since it was found that the majority of the literature was solution proposals.

With the evidence of the utility of effective testing methods, this introduced two key challenges. The challenges observed are related to the data behind the current and new testing methods and the challenges related to the research type. This mapping will be useful for future work to facilitate an increased performance in smart contracts for blockchain applications after the testing phase. Furthermore, this study will help in closing the gaps found in this domain by highlighting key areas of focus for researchers.

REFERENCES

[1] C. Dilmegani. (Jun. 14, 2022). *Smart Contracts: What are They & Why They Matter in 2022*. AIMultiple. Accessed: 2022. [Online]. Available: https://research.aimultiple.com/smart-contracts/

[2] B. Hu, Z. Zhang, J. Liu, Y. Liu, J. Yin, R. Lu, and X. Lin, "A comprehensive survey on smart contract construction and execution: Paradigms, tools, and systems," *Patterns*, vol. 2, no. 2, Feb. 2021, Art. no. 100179.

[3] E. Newbery. (Jan. 15, 2022). *7 Smart Contract Cryptos to Watch in 2022*. The Motley Fool. Accessed: 2022. [Online]. Available: https://www.fool.com/the-ascent/cryptocurrency/articles/7-smart-contract-cryptos-to-watch-in-2022/

[4] Kalyanicynixit. (Aug. 17, 2020). *Applications of Blockchain Technology*. Medium. Accessed: 2022. [Online]. Available: https://medium.com/@kalyanicynixit/applications-of-blockchain-technology-5e123aef146d

[5] D. Austin. Nov. 19, 2020. *What are Smart Contracts?* Medium. Accessed: 2022. [Online]. Available: https://medium.com/swlh/what-are-smart-contracts-6c13f6c725d7

[6] D. Macrinici, C. Cartofeanu, and S. Gao, "Smart contract applications within blockchain technology: A systematic mapping study," *Telematics Inform.*, vol. 35, no. 8, pp. 2337–2354, 2018, doi: 10.1016/j.tele.2018.10.004.

[7] N. Sanchez-Gomez, J. Torres-Valderrama, J. A. Garcia-Garcia, J. J. Gutierrez, and M. J. Escalona, "Model-based software design and testing in blockchain smart contracts: A systematic literature review," *IEEE Access*, vol. 8, pp. 164556–164569, 2020.

[8] A. Vacca, A. Di Sorbo, C. A. Visaggio, and G. Canfora, "A systematic literature review of blockchain and smart contract development: Techniques, tools, and open challenges," *J. Syst. Softw.*, vol. 174, Apr. 2021, Art. no. 110891.

[9] N. Six, N. Herbaut, and C. Salinesi, "Blockchain software patterns for the design of decentralized applications: A systematic literature review," *Blockchain: Res. Appl.*, vol. 3, no. 2, Jun. 2022, Art. no. 100061.

[10] R. Sujeetha and C. A. S. Deiva Preetha, "A literature survey on smart contract testing and analysis for smart contract based blockchain application development," in *Proc. 2nd Int. Conf. Smart Electron. Commun. (ICOSEC)*, Oct. 2021, pp. 378–385.

[11] M. S. Ferdous, M. J. M. Chowdhury, and M. A. Hoque, "A survey of consensus algorithms in public blockchain systems for crypto-currencies," *J. Netw. Comput. Appl.*, vol. 182, May 2021, Art. no. 103035.

[12] T. G. Erdogan and A. Tarhan, "Systematic mapping of process mining studies in healthcare," *IEEE Access*, vol. 6, pp. 24543–24567, 2018.

[13] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *Proc. Electron. Workshops Comput.*, Jun. 2008, pp. 1–10.

[14] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Inf. Softw. Technol.*, vol. 64, pp. 1–18, Aug. 2015.

[15] B. A. Kitchenham, D. Budgen, and O. Pearl Brereton, "Using mapping studies as the basis for further research—A participant-observer case study," *Inf. Softw. Technol.*, vol. 53, no. 6, pp. 638–651, 2011.

[16] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. New York, NY, USA: Springer, 2012.

[17] J. Sun, S. Huang, C. Zheng, T. Wang, C. Zong, and Z. Hui, "Mutation testing for integer overflow in ethereum smart contracts," *Tsinghua Sci. Technol.*, vol. 27, no. 1, pp. 27–40, Feb. 2022.

[18] E. Viglianisi, M. Ceccato, and P. Tonella, "Summary of: A federated society of bots for smart contract testing," in *Proc. 14th IEEE Conf. Softw. Test., Verification Validation (ICST)*, Apr. 2021, pp. 282–283.

[19] E. Andesta, F. Faghih, and M. Fooladgar, "Testing smart contracts gets smarter," in *Proc. 10th Int. Conf. Comput. Knowl. Eng. (ICCKE)*, Oct. 2020, pp. 405–412.

[20] S. Ji, J. Dong, J. Qiu, B. Gu, Y. Wang, and T. Wang, "Increasing fuzz testing coverage for smart contracts with dynamic taint analysis," in *Proc. IEEE 21st Int. Conf. Softw. Qual., Rel. Secur. (QRS)*, Dec. 2021, pp. 243–247.

[21] X. Mei, I. Ashraf, B. Jiang, and W. K. Chan, "A fuzz testing service for assuring smart contracts," in *Proc. IEEE 19th Int. Conf. Softw. Qual., Rel. Secur. Companion (QRS-C)*, Jul. 2019, pp. 544–545.

[22] Y. Hassanzadeh-Nazarabadi, K. Kshatriya, and O. Ozkasap, "Smart contract-enabled LightChain test network," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, May 2021, pp. 1–2.

[23] W. K. Chan and B. Jiang, "Fuse: An architecture for smart contract fuzz testing service," in *Proc. 25th Asia–Pacific Softw. Eng. Conf. (APSEC)*, Dec. 2018, pp. 707–708.

[24] J.-W. Liao, T.-T. Tsai, C.-K. He, and C.-W. Tien, "Soliaudit: Smart contract vulnerability assessment based on machine learning and fuzz testing," in *Proc. 6th Int. Conf. Internet Things: Syst., Manag. Secur. (IOTSMS)*, Oct. 2019, pp. 458–465.

[25] P. Vilain, J. Mylopoulos, and H.-A. Jacobsen, "A preliminary study on using acceptance tests for representing business requirements of smart contracts," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, May 2020, pp. 1–6.

[26] T. Zhou, K. Liu, L. Li, Z. Liu, J. Klein, and T. F. Bissyande, "SmartGift: Learning to generate practical inputs for testing smart contracts," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Sep. 2021, pp. 23–34.

[27] K. B. Kim and J. Lee, "Automated generation of test cases for smart contract security analyzers," *IEEE Access*, vol. 8, pp. 209377–209392, 2020.

[28] X. Wang, Z. Xie, J. He, G. Zhao, and R. Nie, "Basis path coverage criteria for smart contract application testing," in *Proc. Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discovery (CyberC)*, Oct. 2019, pp. 34–41.

[29] M. Barboni, A. Morichetta, and A. Polini, "SuMo: A mutation testing strategy for solidity smart contracts," in *Proc. IEEE/ACM Int. Conf. Autom. Softw. Test (AST)*, May 2021, pp. 50–59.

[30] P. Chapman, D. Xu, L. Deng, and Y. Xiong, "Deviant: A mutation testing tool for solidity smart contracts," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Jul. 2019, pp. 319–324.

[31] M. Fooladgar, A. Arefzadeh, and F. Faghih, "TestSmart: A tool for automated generation of effective test cases for smart contracts," in *Proc. 11th Int. Conf. Comput. Eng. Knowl. (ICCKE)*, Oct. 2021, pp. 476–481.

[32] W. Zhang, S. Banescu, L. Pasos, S. Stewart, and V. Ganesh, "MPro: Combining static and symbolic analysis for scalable testing of smart contract," in *Proc. IEEE 30th Int. Symp. Softw. Rel. Eng. (ISSRE)*, Oct. 2019, pp. 456–462.

[33] X. Wang, H. Wu, W. Sun, and Y. Zhao, "Towards generating cost-effective test-suite for ethereum smart contract," in *Proc. IEEE 26th Int. Conf. Softw. Anal., Evol. Reeng. (SANER)*, Feb. 2019, pp. 549–553.

[34] Z. Li, H. Wu, J. Xu, X. Wang, L. Zhang, and Z. Chen, "MuSC: A tool for mutation testing of ethereum smart contract," in *Proc. 34th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Nov. 2019, pp. 1198–1201.

[35] J. Gao, "Guided, automated testing of blockchain-based decentralized applications," in *Proc. IEEE/ACM 41st Int. Conf. Softw. Eng., Companion (ICSE-Companion)*, May 2019, pp. 138–140.

[36] Z. Wu, J. Zhang, J. Gao, Y. Li, Q. Li, Z. Guan, and Z. Chen, "Kaya: A testing framework for blockchain-based decentralized applications," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Sep. 2020, pp. 826–829.

[37] S. Akca, A. Rajan, and C. Peng, "SolAnalyser: A framework for analysing and testing smart contracts," in *Proc. 26th Asia–Pacific Softw. Eng. Conf. (APSEC)*, Dec. 2019, pp. 482–489.

[38] J. F. Ferreira, P. Cruz, T. Durieux, and R. Abreu, "Smartbugs: A framework to analyze solidity smart contracts," in *Proc. IEEE/ACM Int. Conf. Automated Softw. Eng.*, Dec. 2020, pp. 1349–1352.

[39] X. Wang, X. Yang, and C. Li, "A formal verification method for smart contract," in *Proc. 7th Int. Conf. Dependable Syst. Their Appl. (DSA)*, Nov. 2020, pp. 31–36.

[40] M. Zhang, P. Zhang, X. Luo, and F. Xiao, "Source code obfuscation for smart contracts," in *Proc. 27th Asia–Pacific Softw. Eng. Conf. (APSEC)*, Dec. 2020, pp. 513–514.

[41] J. Choi, D. Kim, S. Kim, G. Grieco, A. Groce, and S. K. Cha, "SMARTIAN: Enhancing smart contract fuzzing with static and dynamic dataflow analyses," in *Proc. 36th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Nov. 2021, pp. 227–239.

[42] C. Liu, H. Liu, Z. Cao, Z. Chen, B. Chen, and B. Roscoe, "ReGuard: Finding reentrancy bugs in smart contracts," in *Proc. IEEE/ACM 40th Int. Conf. Softw. Eng., Companion (ICSE-Companion)*, Jun. 2018.

[43] L. Fritz, "A decentralized application for verifying a matching algorithm—Programming and testing a smart contract on the ethereum blockchain," M.S. thesis, Dept. Comput. Sci. Eng., Univ. Gothenburg, Gothenburg, Sweden, 2018.

[44] R. M. Parizi, A. Dehghantanha, K.-K. R. Choo, and A. Singh, "Empirical vulnerability analysis of automated smart contracts security testing on blockchains," in *Proc. 28th Annu. Int. Conf. Comput. Sci. Softw. Eng. (CASCON)*, 2018, pp. 103–113.

[45] P. Hartel and R. Schumi, "Mutation testing of smart contracts at scale," in *Tests Proofs*. Cham, Switzerland: Springer, 2020, pp. 23–42.

[46] A. Suliman, Z. Husain, M. Abououf, M. Alblooshi, and K. Salah, "Monetization of IoT data using smart contracts," *IET Netw.*, vol. 8, no. 1, pp. 32–37, Jan. 2019.

[47] E. Solaiman, T. Wike, and I. Sfyrakis, "Implementation and evaluation of smart contracts using a hybrid on- and off-blockchain architecture," *Concurrency Comput., Pract. Exper.*, vol. 33, no. 1, p. e5811, 2020.

[48] M. Ren, Z. Yin, F. Ma, Z. Xu, Y. Jiang, C. Sun, H. Li, and Y. Cai, "Empirical evaluation of smart contract testing: What is the best choice?" in *Proc. 30th ACM SIGSOFT Int. Symp. Softw. Test. Anal.*, Jul. 2021, pp. 566–579.

[49] H. Medeiros, P. Vilain, J. Mylopoulos, and H.-A. Jacobsen, "SolUnit: A framework for reducing execution time of smart contract unit tests," in *Proc. 29th Annu. Int. Conf. Comput. Sci. Softw. Eng. (CASCON)*, Nov. 2019, pp. 264–273.

[50] J. Gao, H. Liu, Y. Li, C. Liu, Z. Yang, Q. Li, Z. Guan, and Z. Chen, "Towards automated testing of blockchain-based decentralized applications," in *Proc. IEEE/ACM 27th Int. Conf. Program Comprehension (ICPC)*, May 2019, pp. 294–299.

[51] H. Medeiros, P. Vilain, and V. C. Pereira, "Reducing the execution time of unit tests of smart contracts in blockchain platforms," in *Proc. XV Brazilian Symp. Inf. Syst.*, May 2019, pp. 1–9.

[52] Y. Liu, Y. Li, S.-W. Lin, and Q. Yan, "ModCon: A model-based testing platform for smart contracts," in *Proc. 28th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, Nov. 2020, pp. 1601–1605.

[53] S. Akca, C. Peng, and A. Rajan, "Testing smart contracts," *Proc. 15th ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas. (ESEM)*, Oct. 2021, pp. 1–11.

[54] A. Leid, B. van der Merwe, and W. Visser, "Testing ethereum smart contracts: A comparison of symbolic analysis and fuzz testing tools," in *Proc. Conf. South Afr. Inst. Comput. Scientists Inf. Technologists*, Sep. 2020, pp. 35–43.

[55] G. Grieco, W. Song, A. Cygan, J. Feist, and A. Groce, "EchiDNA: Effective, usable, and fast fuzzing for smart contracts," in *Proc. 29th ACM SIGSOFT Int. Symp. Softw. Test. Anal.*, Jul. 2020, pp. 557–560.

[56] B. Jiang, Y. Liu, and W. K. Chan, "ContractFuzzer: Fuzzing smart contracts for vulnerability detection," in *Proc. 33rd ACM/IEEE Int. Conf. Automated Softw. Eng.*, Sep. 2018, pp. 259–269.

[57] A. Ghaleb and K. Pattabiraman, "How effective are smart contract analysis tools? Evaluating smart contract static analysis tools using bug injection," in *Proc. 29th ACM SIGSOFT Int. Symp. Softw. Test. Anal.*, Jul. 2020, pp. 415–427.

[58] T. Durieux, J. F. Ferreira, R. Abreu, and P. Cruz, "Empirical review of automated analysis tools on 47,587 ethereum smart contracts," in *Proc. ACM/IEEE 42nd Int. Conf. Softw. Eng.*, Jun. 2020, pp. 530–541.

[59] T. Hu, X. Liu, T. Chen, X. Zhang, X. Huang, W. Niu, J. Lu, K. Zhou, and Y. Liu, "Transaction-based classification and detection approach for ethereum smart contract," *Inf. Process. Manage.*, vol. 58, no. 2, Mar. 2021, Art. no. 102462.

[60] A. Singh, R. M. Parizi, Q. Zhang, K.-K.-R. Choo, and A. Dehghantanha, "Blockchain smart contracts formalization: Approaches and challenges to address vulnerabilities," *Comput. Secur.*, vol. 88, Jan. 2020, Art. no. 101654.

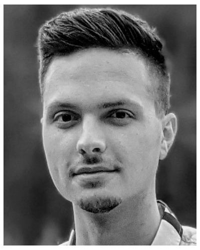
[61] Y. Ivanova and A. Khritankov, "RegularMutator: A mutation testing tool for solidity smart contracts," *Proc. Comput. Sci.*, vol. 178, pp. 75–83, Jan. 2020.

[62] M. A. Walker, D. C. Schmidt, and A. Dubey, "Testing at scale of IoT blockchain applications," in *Advances in Computers*. Amsterdam, The Netherlands: Elsevier, 2019, pp. 155–179.

[63] P. Praitheeshan, L. Pan, X. Zheng, A. Jolfaei, and R. Doss, "SolGuard: Preventing external call issues in smart contract-based multi-agent robotic systems," *Inf. Sci.*, vol. 579, pp. 150–166, Nov. 2021.

[64] W. Bugden and A. Alahmar, "Rust: The programming language for safety and performance," in *Proc. 2nd Int. Graduate Stud. Congr.*, Jun. 2022, pp. 1–9.

[65] W. Bugden and A. Alahmar, "The safety and performance of prominent programming languages," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 32, no. 5, pp. 713–744, May 2022, doi: 10.1142/S0218194022500231.



NICHOLAS PAUL IMPERIUS was born in Thunder Bay, ON, Canada. He received the Diploma degree in software engineering technology from Lakehead University, Thunder Bay, in 2021, where he is currently pursuing the B.S. degree in software engineering.

In Summer 2016, he worked as a Desktop Support Analyst Intern at Shaw Communications Inc., where he was a ServiceNow Report Specialist Intern, in Summer 2017. In 2022, he pursued a summer IT job at the Thunder Bay Regional Health Sciences Center in Thunder Bay.

AYMAN DIYAB ALAHMAR (Member, IEEE) received the M.B.A. degree in business administration and the Ph.D. degree in software engineering from Lakehead University, ON, Canada, and the Ph.D. degree in mechanical engineering from Middle East Technical University, Ankara, Turkey. He is currently a Faculty Member with the Department of Software Engineering, Lakehead University. He has a long academic and industrial experience as an Associate Professor and a Serial Entrepreneur. His current research interests include artificial intelligence in engineering, health informatics and software engineering. He received numerous awards, including the Excellence in Teaching Award, the Dean's Award for Excellence, and the IEEE Best Presentation Award for the paper published in the IEEE International Conference on Cloud and Big Data Computing (CBDCom 2020). He serves as a Reviewer for several journals, including *Computers in Biology and Medicine* and *Informatics in Medicine Unlocked.*