IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON BLOCKCHAIN TECHNOLOGY: PRINCIPLES AND APPLICATIONS

Received July 28, 2020, accepted August 17, 2020, date of publication September 3, 2020, date of current version September 21, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3021502

Model-Based Software Design and Testing in Blockchain Smart Contracts: A Systematic Literature Review

NICOLÁS SÁNCHEZ-GÓMEZ, JESUS TORRES-VALDERRAMA, J. A. GARCÍA-GARCÍA, JAVIER J. GUTIÉRREZ, AND M. J. ESCALONA

Web Engineering and Early Testing Research Group, Escuela Técnica Superior de Ingeniería Informática, University of Seville, 41012 Seville, Spain

Corresponding author: Nicolás Sánchez-Gómez (nicolas.sanchez@iwt2.org)

This work was supported in part by the Spanish Government's Ministry of Economy and Competitiveness through the Guided Solutions for the Systematization of Early Quality Assurance in Software Engineering (POLOLAS) Project under Grant TIN2016-76956-C3-2-R, and in part by the Andalusian Regional Ministry of Economy, Knowledge, Business and University through the Trop@ Project under Grant CEI-12-TIC021.

ABSTRACT Blockchain technology promises to spark a real revolution. One of most important concepts associated with this technology is smart contracts, which enable the automatic execution of agreements and augur a world without intermediaries. The conditions and rules of ''contracts'' are established in a computer codes and trust is enforced by consensus among the participants. One relevant feature associated with smart contract is the immutability property, which establishes the non-alteration of blockchain network data after the clauses of the contract are been approved by all parties or entities involved. For this reason, smart contract development requires more effort and care than the development of other common programs. They require systematic mechanisms to collect requirements and functional specifications. In addition, it is necessary to verify and validate the agreed functionality and the implemented code before they are deployed in the blockchain platform. This article presents a systematic literature review of primary studies in the field of Software Development Life Cycle, focusing on model-based software design and testing in the blockchain domain of smart contracts. This research aims to identify gaps and/or opportunities for further research. After carried out this review, it was observed that no clear methodology exists for evaluating and validating the quality either of this software or the overall development process. This means that software developers may implement smart contract code in which bugs and serious security vulnerabilities appear when the software is delivered to their customers.

INDEX TERMS Software engineering, software development life cycle, blockchain, smart contract, model-based software engineering, software testing, systematic literature review.

I. INTRODUCTION

Blockchain technology has well-known benefits in many areas (e.g., economic, political, humanitarian, or social, among others) and could reconfigure many aspects of today's society [2]. Blockchain technology, based on distributed ledger technologies (DLT), is a chain of cryptographically linked blocks. This technology provides mechanisms to define agreements using the concept of smart contract [61], which is computer programs that run on blockchain networks without intermediaries. A smart contract, in the case of legal agreement, tends to replace the printed document with legal language. The software requirements that this computer program must satisfy, from the engineering perspective, are analogous to the terms, rules, and conditions in a conventional legal contract. However, it is important to highlight that smart contracts can encode automated agreement execution, but not all smart contracts may necessarily be agreements and do not always necessarily codify actions between more than one party.

The first conceptualization of the smart contract term was published by Nick Szabo in 1994 [66]. In this article, the author defines a smart contract as a set of clauses, data and protocols. These protocols implement algorithms to automatically verify the fulfilment of each condition by each party/entity involved in the contract. In this sense, these

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

contracts are smarter than paper-based contracts because they automatically enforce the obligations of the parties involved. But a smart contract should not be seen as intelligent tools that can parse a contract's more subjective requirements.

On the other hand, it is important to note that the fact of eliminating the need for a trusted third party significantly reduces the transaction costs, that is, it facilitates the most economic exchanges.

In this context, a set of important concepts will be introduced:

- Software Development Life Cycle (SDLC) is a process that defines good practices to document, develop, maintain, and replace the software adequately [57]. As mentioned above, smart contracts are also software. Therefore, it is necessary to apply methodological processes for eliciting their requirements and functional specifications to improve their quality [36]. However, to the best of our knowledge, there are currently no guidelines for implementing smart contracts.
- One of the most important aspects, to consider in SDLC, is the Software Testing because it allows improving the quality of the software [28]. However, this testing phase often becomes less important due to delays in development. Then, it is usually performed at the end once the coding phase is finished and before the software is delivered to the customer. In this context, it is relevant to start testing software as soon as possible when SDLC is carried out. Early testing allows detecting many defects soon what helps to increase quality and customer's satisfaction [14].
- Over last years, Model-Driven software engineering [70] has become a practical, efficient, and successful methodology for software design and testing. Many proposals [7], [20], [21], [26] use this paradigm to define software requirements in structured, comprehensible, and formal models. This formalization allows establishing mechanisms to measure, check and verify these requirements from early stages. In addition, the definition of these formal models facilitates the definition of transformation rules to generate a test case systematically and automatically, and even software code associated with these test cases [19]. These methodological practices facilitate code reusability and the reduction of human errors.
- Other important benefits achieved by Model-Driven software engineering are: (1) systematized design, and continuous software testing to increase software development effectiveness, and (2) reusable models and software code which can reduce costs and time in the software development process.

In this context, this article presents the results of a study that analyzes the state-of-the-art of research works in the field of SDLC. Concretely, this study is focused on proposals that address the Model-Based software design and testing for smart contracts within blockchain domain. For this purpose, the Systematic Literature Review (SLR) method is used to identify gaps and to offer future research guidelines related to our research topic.

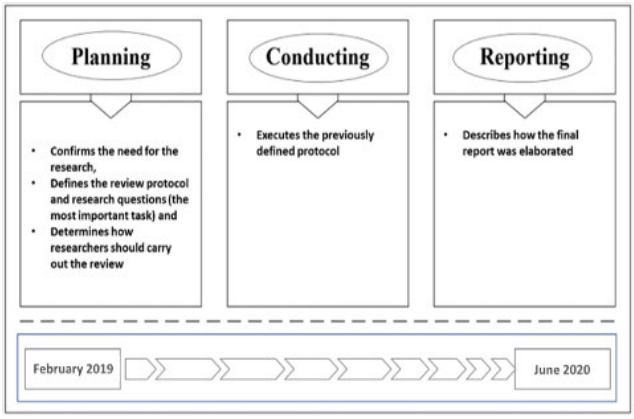


FIGURE 1. Schedule of the systematic review followed in this article.

The rest of the paper is structured as follows: Section II and III describe the method used for the systematic review and its planning, respectively; After defining the review protocol to be applied, it is conducted presenting in Section IV the results; Section V analyses and discusses the most relevant findings; finally, Section VI provides a set of conclusions and suggestions for future works.

II. REVIEW METHOD

The present study was carried out using one of the SLR methods most successfully and widely applied in software engineering field. Specially, the Kitchenham's method [30]. This method present rigorous stages to analyze research knowledge using a trustworthy and auditable methodology. Some authors, however, have criticized Kitchenham's method and/or proposed improvements on this one [15], [31].

In the wake of these criticisms and suggestions for improvement, Kitchenham published an updated version of her method in [30]. But, at present, some authors [15] admit that an important gap still exists regarding the evaluation of quality in studies based on empirical methods.

This SLR follows the latest version of Kitchenham's method, referenced above. It describes three phases for executing a systematic review: (1) planning, which defines aspects such as the need for the research, review protocol and research questions; (2) conducting, which the previously established protocol is carried out; and (3) reporting, which presents the final analysis to answer each research question.

Figure 1 shows these phases and their tasks on a timeline to achieve research objective of this article.

III. PLANNING THE SYSTEMATIC REVIEW

This section describes the planning process conducted in this SLR. During this process, the need to perform the review was identified, the research questions (RQs) were formed, and the review protocol was established and verified.

A. THE NEED FOR THE REVIEW

Over the last few years, much researches around the world to evaluate and identify challenges and obstacles to the application of blockchain smart contracts in different fields. This research has reported and evaluated the use of blockchain technology in multiple processes and services in different business areas (e.g., Internet of Things [39], Supply Chain [55], [64], [68], education [74], agriculture [72] or health [73], among others). In addition, one of the objectives of these investigations has been identifying challenges and barriers of this technology in different scenarios (e.g., quality [32], Big Data [29] or e-government [4]).

In addition, some works partially related to our proposal have been published. Alharby *et al.* [3], present a Systematic Mapping Study (SMS) of technology-oriented research in smart contracts. In this study, the authors followed the method presented by Petersen *et al.* [52]. However, the process followed is not well detailed which makes reproducibility difficult. Following the same method, Macrinici *et al.* [41] propose an SMS to know the state-of-the-art in smart contract applications within blockchain technology. Authors deeply reviewed 64 papers identifying some gaps but, there is no mention about testing. Leka *et al.* [35] develop an SLR but the authors follow the SMS method mentioned earlier, they do not follow the SLR process. Moreover, no explanation about the execution is described and results cannot be reproducible. Finally, Dhaiouir and Assar [17] present an SLR of Blockchain-Enabled Smart Contracts, focusing on platforms, languages, or applications among others. Again, there is no mention about testing.

Conversely to the present literature, this article describes an SLR carried out in the field of SDLC, in particular on smart contract development life cycle. The focus is on the model-based design and testing of blockchain smart contracts. More specifically, we review the state of formal smart contract modelling and automatic code generation, together with the verification/validation of this code, in order to characterize and present the state-of-the-art (approaches) in this field and identify gaps and opportunities for further research.

B. RESEARCH QUESTIONS

Following Kitchenham's method, RQs needed to be stablished for clearly focus the research on the topic and improve our understanding of the proposals being studied. The general RQ guiding our whole SLR was: ≪What is the state-of-the-art as regards blockchain smart contracts in software engineering?≫. If software engineering is applied, it is fundamental to have specific analysis and design methods, quality control through testing and metrics, security assessment and overall development process. Therefore, this question could be considered very general, so it was reformulated in several more specific questions to provide a clear view of the most relevant aspects of proposals addressing the collection of requirements and functional specifications, analysis, design, coding and testing in the context of blockchain smart contracts. Table 1 shows together the defined RQs with their motivations and corresponding sub-questions. In addition, the possible answers to be responded are described.

C. REVIEW PROTOCOL

After formulating each research question, the review protocol must be specified to define search strategies, inclusion and exclusion criteria for primary studies (PS), and quality criteria. This section aims to define these aspects.

1) SEARCH STRATEGY

The search strategy is the procedure followed to locate the most relevant PS that have been indexed in different digital libraries. In this sense, this strategy is focused on locating papers published in peer-review journal and international conferences. The search strategy was divided into two phases:

- **Phase 1**. The first phase consists of defining keywords that are going to be used in the search protocol. This definition has been performed after running different pre-searches, which allow to refine the most appropriate set of keywords. In this sense, it is important to obtain appropriate keywords because these allow improving the quality of the results. Table 2 presents the keywords used in this SLR.

$$Equation_1 = [(V_{i=1}^{4} A_i) \wedge (V_{j=1}^{8} B_j)] \wedge (V_{k=1}^{2} C_k)$$
Boolean expression of keywords (1)

- **Phase 2**. After defining keywords, these have been systematically used to carry out searches in different digital libraries. For this purpose, keyword combinations have been used as Equation 1 states. This formula formalizes the boolean expression of the keywords used in the searches.

Moreover, regarding digital libraries to be used, some authors establish criteria to select the most appropriate libraries. For example, Ngai *et al.* [48] propose more than ten digital libraries (i.e., Academic Search Premier, ACM Digital Library, Emerald Full Text or IEEE Xplore Digital, among others) considering their relevance. However, during the process of carrying out preliminary searches, we noticed that many papers were repeatedly returned by different digital libraries. After considering this fact, we decided to use only the following databases: ACM Digital Library, IEEE Xplore Digital Library, ScienceDirect, Elsevier's Scopus, and Springer Link. References were managed using the Jabref tool [23] and a spreadsheet.

$$Equation_2 = keyword(E_1) \wedge abstract(E_1) \wedge Title(E_1)$$
Boolean expression for the metadata of a paper (2)

Once keywords and digital libraries were established, search expressions were formalized using Equation 1 and executed on each digital library. For this purpose, title-abstract-keyword metadata of each paper were selected as information sources. The use of these metadata is formalized in Equation 2. Each digital library has its own syntax for

TABLE 1. Research Questions (RQ).

RQ1	Are there approaches in the literature that promote the application of a SDLC? What do phases of the life cycle promote the different studies?
	This RQ aims to find proposals that have been published and to identify their general contexts and the objectives they achieved using SDLC, all in the context of blockchain smart contracts
	Sub-questions
RQ1.1	Is any type of smart contract development life cycle promoted? (single or multiple selection)
	A. The proposal does not describe or apply a life cycle
	B. The development process consists of several phases and there is a common vocabulary for each step
	C. These phases are arranged in order of precedence and are clearly defined I/O from one step to the next
	D. There is a deterministic "definition of done" (*) that can be used to confirm whether a step is truly complete
RQ1.2	Does the proposal promote any of the following phases? (single or multiple selection)
	A. Requirements gathering, analysis and/or design phases
	B. Software coding phase
	C. Software testing phase
	D. The proposal promotes other phases (or does not mention anything about phases)
RQ2	Do they promote model-based software engineering, early starting of the testing phase or automatic source code generation?
	The purpose of this RQ is to identify the techniques and guidelines applied in the different proposals, all in the context of blockchain smart contract
RQ2.1	Does the study apply model-based software engineering? (Boolean:Y/N)
RQ2.2	Does the proposal promote early starting of the testing phase? (known as early testing) (Boolean-Y/N)
RQ2.3	Does the study propose automatic smart contract code generation? (single or multiple selection)
	A. The proposal does not contemplate automatic code generation
	B. The proposal contemplates generation using domain-specific ontologies, glossaries and/or semantic rules
	C. The proposal contemplates generation through the application of model-based software engineering
	D. The proposal contemplates generation through templates and other utilities
RQ3	What scientific or empirical validation methods were used in the different proposals?
	The aim of this RQ is to determine the type of validation methods used in the different studies
RQ3.1	Does the study use a scientific validation method? (Boolean:Y/N)
RQ3.2	Does the study use an empirical validation method? (Boolean:Y/N)

(*) *The "definition of done" is usually clear when a product delivered to the client is complete. The definition is that the software engineer, following a methodological approach or previous guidelines, has 100% confidence that no more work is needed before the product can be handed to the client.*

TABLE 2. Keywords.

A	B	C
Engineering	Requirement	Blockchain
Model-based	Analysis	Smart Contract
Semantic	Design	
Ontology	Test	
	Debug	
	Check	
	Validation	
	Verification	

indicating custom search expressions. Moreover, they have certain limits on the maximum number of logical clauses in the same search. Therefore, Equation 2 was applied into different queries. Table 3 shows each of the queries executed in each of the databases selected in this SLR.

In addition, it was also necessary to extend each search query using filters due to the considerable number of search queries. These filters are provided by each digital library. For example, scientific area, specific topic, and year of publication greater than or equal to 2016 were some of the filters applied. Only from 2016, we started to identify paper that enhanced the predefined search criteria. Although Ethereum, for instance, was formally announced by Vitalik at The North American Bitcoin Conference in 2014 [8], the Ethereum network was not launched till 2015. In that year, developers began writing smart contracts and decentralized apps to deploy on the live Ethereum network.

Finally, the previously defined systematic search protocol has been extended using the "snowball" technique [71]. This technique involves extending the search process to cover both the reference lists and the citations in each paper under study. Section IV-A describes in detail how the search strategy was executed.

2) SELECTION PROCESS, EXCLUSION AND INCLUSION CRITERIA, AND QUALITY ASSURANCE

This section defines the selection process of relevant papers, which are going to be later analyzed considering the objectives of this SLR. Three researchers are responsible for carrying out this selection process. Table 4 summarizes each phase conducted in the process of selecting papers for study.

Our selection process includes some face-to-face meetings to offer a forum for discussion and agreement between researchers when there are doubts to evaluate a paper. The objective of these meetings is to reduce the bias of each researcher. On the one hand, the first meeting each face is made in the third phase (Ph3) of our selection process (see Table 4). As mentioned above, this phase aims to resolve any doubts when inclusion/exclusion criteria are applied. In these cases, a full reading of dubious papers is necessary. After this reading, all researchers decide — always jointly — to finally include or exclude the PS. The decision must be joint to avoid subjectivity. On the other hand, the second face-to-face meeting (Ph5) is carried out after applying the "snowball" technique and its objective is also to resolve any doubts when the papers returned by this technique are evaluated or when exclusion/inclusion criteria are applied.

Regarding exclusion/inclusion criteria, we have established some objective criteria which have been grouped by each phase of the selection process (Table 5). In short, we consider papers written in English and published in well-reputed

TABLE 3. Search queries.

ACM Digital Library	
Q1	"query":keywords.author.keyword:(Engin* Model* Semantic Ontology) AND record Abstract:(Requirement Analysis Design Test* Debug* Check* Validation Verification) AND acmdlTitle:(Blockchain "Smart Contract")
Q2	"query":recordAbstract:(Engin* Model* Semantic Ontology) AND keywords.author.keyword:(Requirement Analysis Design Test* Debug* Check* Validation Verification) AND acmdlTitle:(Blockchain "Smart Contract")
IEEE Xplore	
Q3	((((("Author Keywords":engin*) OR "Author Keywords":model*) OR "Author Keywords":semantic) OR "Author Keywords":ontology) AND (((((((("Abstract":requirement) OR "Abstract":analysis) OR "Abstract":design) OR "Abstract":test*) OR "Abstract":debug*) OR "Abstract":check*) OR "Abstract":validation) OR "Abstract":verification) AND (("Document Title":blockchain) OR "DocumentTitle":smart contract)
Q4	(((((((("Author Keywords":requirement) OR "Author Keywords":analysis) OR "Author Keywords":design) OR "Author Keywords":test*) OR "Author Keywords":debug*) OR "Author Keywords":check*) OR "Author Keywords":validation) OR "Author Keywords":verification) AND ((((("Abstract":engin*) OR "Abstract":model*) OR "Abstract":semantic) OR "Abstract":ontology) AND (("Document Title":blockchain) OR "Document Title":smart contract)
Science Direct	
Q5	Find articles with these terms: ("Smart Contract") Title, abstract, keywords: ("Engin*" OR "Model*" OR "Semantic" OR "Ontology" OR "Requirement" OR "Analysis" OR "Design" OR "Test*" OR "Debug*" OR "Check*" OR "Validation" OR "Verification" OR "Blockchain" OR "Smart Contract")
Elsevier's Scopus	
Q6	(ABS("Requirement" OR "Analysis" OR "Design" OR "Test*" OR "Debug*" OR "Check*" OR "Validation" OR "Verification") AND KEY("Engin*" OR "Model*" OR "Semantic" OR "Ontology") AND TITLE("Blockchain" OR "Smart Contract"))
Q7	(KEY("Requirement" OR "Analysis" OR "Design" OR "Test*" OR "Debug*" OR "Check*" OR "Validation" OR "Verification") AND ABS("Engin*" OR "Model*" OR "Semantic" OR "Ontology") AND TITLE("Blockchain" OR "Smart Contract"))
Springer Link	
Q8	("Engin*" OR "Model*" OR "Semantic" OR "Ontology") AND ("Requirement" OR "Analysis" OR "Design" OR "Test*" OR "Debug*" OR "Check*" OR "Validation" OR "Verification") AND "Blockchain" AND "Smart Contract"

TABLE 4. Phases of the study selection.

Phase	Description	Participating researchers
Ph1	It automatically aims to execute our search strategies considering title-abstract-keyword metadata	Leading researcher
Ph2	Screening: exclusion of PS dealing with other subjects	Leading researcher
Ph3	First consensus meeting	All researchers
Ph4	Analysis for exclusion of PS based on full text and inclusion of PS based on the "snowball" technique	All researchers
Ph5	Second consensus meeting	All researchers

TABLE 5. Exclusion and inclusion criteria per phase.

Phase	Exclusion and inclusion criteria
Ph1	Automatic search was conducted in each digital library. In this sense, papers returned by search queries (see Table 3) are included in this phase.
Ph2	English only; year of publication greater than or equal to 2016, because after analyzing numerous papers from other years, only from 2016 onwards did we start to identify articles that enhanced the predefined search criteria; full text obtained. Papers not related to the subject were excluded. This exclusion phase included the elimination of duplicate papers and the reading of the title and abstract of the work. In case of any doubt about any document, that document would be preliminarily included. The final decision would be considered and evaluated in the next phase.
Ph3	No new exclusion criteria at this phase (1st meeting), but it is possible to include relevant papers. In this phase the researchers also analyzed all "doubtful" papers in detail, considering all their content.
Ph4	In this phase the "snowball" technique was applied, and it was, therefore, necessary tore-apply the P2 criteria.
Ph5	No new exclusion/inclusion criteria, in this second meeting, are considered. But all researchers analyzed all the "doubtful" papers in detail, considering all their content.

journals (i.e., journals indexed in Journal Citation Reports; JCR) or prestigious conferences (i.e., A*, A, B and C conference level categorized in CORE Conference Ranking). Regarding international conferences, we have considered conferences. Furthermore, we have decided to exclude surveys, discussion, reviews or opinion papers related to blockchain smart contracts.

Finally, Table 5 summarizes criteria defined to include and exclude PS during the selection process.

3) QUALITY QUESTIONS

Quality Criteria (QC) were defined to obtain the best results for future research. Table 6 shows the quality questionnaire applied in this SLR. The cumulative score for each criterion would make up the final quality score for each PS. It is important to note that these quality criteria are not used to exclude papers, but they are used to determine the most relevant and representative PS in future research.

4) DATA SCHEMA

The analysis of each PS could become a difficult task due to the heterogeneous information and different structures of each study. In this sense, we propose a characterization scheme (see Table 7) to homogenize this analysis and reduce the complexity of this task.

5) REVIEW PROTOCOL VALIDATION

Following the recommendations given in Kitchenham's method, the SLR protocol was reviewed to obtain a comprehensive review process. As mentioned above, some random searches were applied to refine definitive keywords and exclusion/inclusion criteria, among other aspects. However, we decided to seek advice from experts in the conduct of SLR to define a systematic, full and comprehensive process. In this sense, a Full Professor in Software Engineering at

TABLE 6. Quality criteria questionnaire.

Code	Question and Scores	Weight
QC1	About SDLC, does the paper clearly present the intended objectives, the scope of the solution provided and its conclusions? The possible answers are: Yes (+1); No (+0)	10%
QC2	Does the paper detail or contextualize any phase of SDLC? The possible answers are: All phases (+2); Some phases (+1); No (+0)	10%
QC3	Does the paper apply model-based software engineering? Does the proposal promote an early start to the testing phase? The possible answers are: Both questions (+2); One of the questions (+1); No (+0)	40%
QC4	Does the paper describe automatic source code generation using domain-specific ontologies, templates and/or model-based software engineering? The possible answers are: All techniques (+2); One or more techniques (+1); No (+0)	30%
QC5	Is the paper scientifically validated? The possible answers are: Case study or experiment (+1); Unvalidated (+0)	10%

TABLE 7. Data Scheme.

Feature	Description
Kind of publication	The journal or conference where the PS is published.
Business area	The business area to which the approach was applied.
Description and motivation	This feature represents a short description of each PS and its motivation.
Life cycle phase	This refers to the SDLC phase on which the proposal focuses.

TABLE 8. Summary of PS considered in the systematic review.

Data base	Ph1	Ph2	Ph3	Ph4	Ph5
ACM Digital Library	27	6	2	-	-
IEEE Xplore	39	7	3	-	-
ScienceDirect	372	31	7	-	-
Elsevier's Scopus	352	42	6	-	-
SpringerLink	243	24	4	-	-
Snowball technique	-	-	-	10	3
Subtotals	1,033	110	**22**	10	**3**
TOTAL	**25**				

the University of Seville (Spain) participated as an external expert to validate our review protocol.

IV. CONDUCTING AND QUALITY RESULTS

This section aims to present the results obtained after executing the review protocol defined in the previous section. This section also presents the set of PS obtained, as well as the quality score of each PS according to our quality criteria questionnaire (cf. Table 6). Finally, Section IV-B describes some threats to this work's validation process.

A. DETECTION, SELECTION OF PRIMARY STUDIES (PS) AND DATA EXTRACTION

Table 8 presents the distribution of PS obtained after applying inclusion/exclusion criteria in each phase of the selection protocol. For each digital library, it is showed the number of papers thrown up in each stage of the review protocol. Table 8 also includes a record of papers obtained after the "snowball" technique had been applied (to streamline the handling of the results, these papers were not classified by digital library).

Figure 2 graphically displays the evolution of the considered PS in the search protocol.

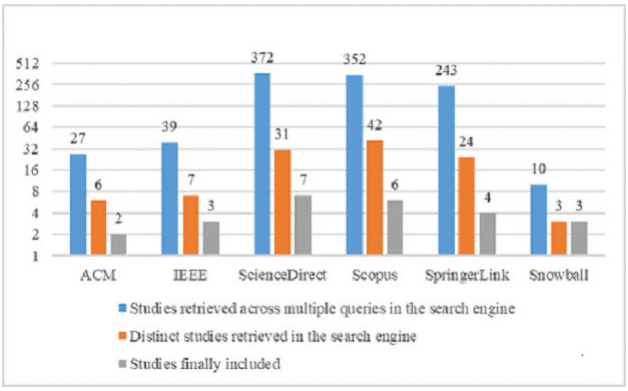


FIGURE 2. Papers retrieved using the digital libraries.

Figure 3 displays the number of PS retrieved from each digital library, and finally included in the analysis divided by the number of papers selected from all the digital libraries.

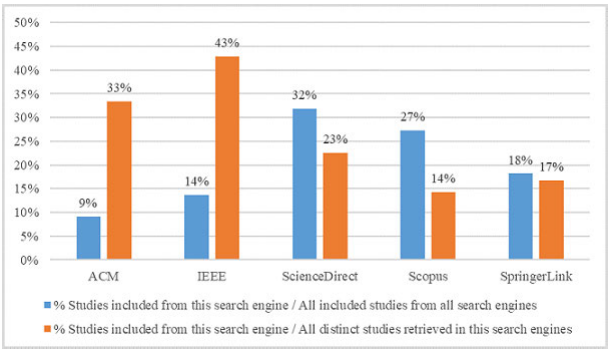


FIGURE 3. Analysis of results retrieved from each digital library as a percentage of the total papers included.

B. THREATS TO VALIDITY

The review and validation protocol presented in previous sections may involve weaknesses or threats on this protocol because the tasks have been conducted by people. Due to this human factor, the selection of papers could be affected by errors during the process of classification and selection of PS. These risks have been mitigated with the execution of several iterations in the review process and several meetings between researchers when there were doubts about the categorization of any PS.

Although the review process has been exhaustively defined and executed, it is not possible to guarantee full coverage of the scientific literature about a topic (e.g., non-indexed papers or grey literature are not considered in this SLR).

Moreover, Schmucker *et al.* [60] states that those types of publications are very rarely relevant in SLR results. That is

TABLE 9. Primary studies (PS) included in the SLR and their quality assessment scores.

PS	Authors	Title	Year	SC
PS01	Marchesi, M. et al.	An Agile Software Engineering Method to Design Blockchain Applications [43]	2018	5
PS02	Liu, Y. et al.	Applying Design Patterns in Smart Contracts [38]	2018	1.5
PS03	Choudhury, O. et al.	Auto-Generation of Smart Contracts from Domain-Specific Ontologies and Semantic Rules [9]	2018	6
PS04	Tateishi, T. et al.	Automatic smart contract generation using controlled natural language and template [67]	2019	6
PS05	Tsai, W. et al.	Beagle: A New Framework for Smart Contracts Taking Account of Law [69]	2019	4
PS06	Koul, R.	Blockchain Oriented Software Testing - Challenges and Approaches [33]	2018	2.5
PS07	Dolgui, A. et al.	Blockchain-oriented dynamic modelling of smart contract design and execution in the supply chain [18]	2019	1.5
PS08	Porru, S. et al.	Blockchain-Oriented Software Engineering: Challenges and New Directions [54]	2017	3.5
PS09	Shishkin, E.	Debugging Smart Contract's Business Logic Using Symbolic Model-Checking [62]	2018	1.5
PS10	Mavridou, A. et al.	Designing Secure Ethereum Smart Contracts: A Finite State Machine Based Approach [44]	2018	5
PS11	Parizi, R. M. et al.	Empirical vulnerability analysis of automated smart contracts security testing on blockchains [50]	2018	1.5
PS12	Lee, S. et al.	Formal Specification Technique in Smart Contract Verification [34]	2019	0.5
PS13	Mavridou, A. et al.	FSolidM for Designing Secure Ethereum Smart Contracts: Tool Demonstration [45]	2018	6
PS14	Sillaber, C. et al.	Life Cycle of Smart Contracts in Blockchain Ecosystems [63]	2017	0
PS15	Grigg, I.	On the intersection of Ricardian and Smart Contracts [25]	2015	0.5
PS16	Kruijff, J. et al.	Ontologies for Commitment-Based Smart Contracts [16]	2017	2
PS17	Clack, C. D.	Smart Contract Templates: Legal semantics and code validation [10]	2018	1.5
PS18	Clack, C. D. et al.	Smart Contract Templates: Foundations, design landscape and research directions [12]	2016	2
PS19	Syahputra, H. et al.	The Development of Smart Contracts for Heterogeneous Blockchains [65]	2019	6
PS20	Liao C. et al.	Toward A Service Platform for Developing Smart Contracts on Blockchain in BDD and TDD Styles [37]	2017	2.5
PS21	Al Khalil, F. et al.	Trust in Smart Contracts is a Process, As Well [1]	2017	1.5
PS22	Mavridou, A. et al.	VeriSolid: Correct-by-Design Smart Contracts for Ethereum [46]	2019	6
PS23	Permenev, A. et al.	VerX: Safety Verification of Smart Contracts [51]	2019	2.5
PS24	Mao, D. et al.	Visual and User-Defined Smart Contract Designing System Based on Automatic Coding [42]	2019	1.5
PS25	Clack, C. D. et al.	Smart Contract Templates: essential requirements and design options [11]	2016	2

why in our study the search terms were used in five online digital libraries covering a wide range of topics enough to be reasonably considered exhaustive for the research field on which this SLR focused. Furthermore, it is important to note that all authors have been involved in the definition of the search protocol, RQs and search terms. This decision has allowed to increase the objectivity of the review process.

V. DISCUSSION AND ANALYSIS

This section aims to answer and discuss each research question to identify state-of-the-art weaknesses according to the objective of this SLR.

A. RQ1. ARE THERE APPROACHES IN THE LITERATURE THAT PROMOTE THE APPLICATION OF A SMART CONTRACT DEVELOPMENT LIFE CYCLE? WHAT PHASES OF THE LIFE CYCLE DO THE DIFFERENT PROPOSALS PROMOTE?

Table 9 shows the PS that were found and finally included in this SLR following all the quality criteria described in Section III-C.

It is important to mention that the number of PS has not been reduced after applying quality measurements. These measures allow us to score the most relevant and representative PS for consideration in future research. In this sense, Table 9 shows the quality score (chosen in the last column of Table 9; SC column) for each PS. The maximum score for quality measurement is ten points according to the criteria established in Section III-C.

For instance, Grigg (PS15) proposes to use the Ricardian contracts. A Ricardian contract places the defining elements of a legal agreement in a format that can be expressed and executed in software. This work is very relevant to be considered in future research. In this studies list, highlights several papers as:

- Marchesi (PS01) proposes a software development process to gather the requirement, analyze, design, develop, test and deploy blockchain applications.
- Choudhury (PS03) provides a novel framework for auto-generating smart contracts by enabling the seamless translation of constraints encoded in knowledge representation to blockchain requirements. This framework uses ontologies and semantic rules to encode domain-specific knowledge and then leverages the structure of abstract syntax trees to incorporate the required constraints.
- Tateishi (PS04) proposes a technique to automatically generate a smart contract from a human-understandable contract document that is created using a document template and a controlled natural language (CNL). The automation is based on a mapping from the document template and the CNL to a formal model that can define the terms and condition in a contract including temporal constraints and procedures.
- Mavridou (PS13) argues, in practice, the smart contracts are riddled with vulnerabilities comprising a critical issue. To facilitate the development of secure smart contracts, we have created a framework, which allows developers to define contracts as finite state machines (FSMs) with rigorous and clear semantics.
- Syahputra (PS19) discusses the development process of a smart contract platform that aims to generate smart contracts for heterogeneous blockchain technologies.

- Mavrodou (PS22) introduces a framework for the formal verification of smart contracts that are specified using a transition-system based model with operational semantics. This framework allows the generation of Solidity code from the verified models, which enables the correct-by-design development of smart contracts.

After the detailed analysis of numerous studies, it can be said that software development practice has progressed steadily, and many methods and models have been recommended to enhance its productivity and effectiveness. Royce [58] is acknowledged as the first person to introduce a methodology specifically conceived for software development. Known as the waterfall model, this method was subsequently redefined by many different organizations and people, resulting in several variations known collectively as SDLC methodology.

Despite the appearance of some new agile models it is also important to mention that SDLC is still the most widely used development methodology in most organizations [5]. The main phases of SDLC are requirements gathering, system analysis and design, coding, testing, deployment, and maintenance. Table 10 shows the distribution of PS about the phases they support, in the context of smart contract Development Life Cycle.

As illustrated in Table 10, the most addressed phases by authors were, respectively: (A1) Requirements gathering, analysis or design phases (68%), (A2) Coding phase (40%), (A3) Testing phase (28%), and (A4) Other phases (12%).

In these aspects, highlights the work of Marchesi (PS01) and Tsai (PS05). Marchesi proposes a software development process to collect the requirements, analyze, design, develop, test and deploy blockchain applications. On the other hand, Tsai (PS05) proposes a framework. This framework has 5 stages: smart contract template development from domain analysis, formal smart contract model and code development from templates and verification/validation.

It can be concluded that the scientific community's efforts are currently aimed at implementing some kind of SDLC. But in the smart contract context, this process consists only of a certain number of unlinked phases, lacking in a common vocabulary. These phases are not arranged in a clear order of precedence and the inputs/outputs are not clearly defined from one step to the next. In addition, there is no deterministic ''definition of done'' that can be used to confirm whether a step is truly completed.

On the other hand, it is important to highlight the fact that, despite its importance to efficient blockchain development, the phase with the least impact in the literature is that of software testing. Blockchain differs from other, traditional applications, requiring testers to address specific requirements and acceptance criteria. Once a smart contract is implemented, its execution cannot be reversed. This calls for robust testing with emphasis on code debugging, but blockchain software testing is a highly specialized domain which requires proven expertise and a rigorous approach. Moreover, smart contract

TABLE 10. Distribution of primary studies (PS) in relation to the Smart Contract Development Life Cycle phases.

PS	A1	A2	A3	A4
[PS01]	X	X	X	X
[PS02]	X			
[PS03]	X	X		
[PS04]	X	X		
[PS05]	X	X	X	X
[PS06]			X	
[PS07]				X
[PS08]	X	X	X	
[PS09]			X	
[PS10]	X	X		
[PS11]			X	
[PS12]	X			
[PS13]	X	X		
[PS14]				
[PS15]	X			
[PS16]				
[PS17]				
[PS18]	X			
[PS19]	X	X		
[PS20]	X		X	
[PS21]				
[PS22]	X	X		
[PS23]	X	X		
[PS24]	X			
[PS25]	X			
TOTAL	**17**	**10**	**7**	**3**

A1: Requirements gathering, analysis, and/or design phases
A2: Coding phase
A3: Testing phase
A4: Other phases

testing involves simulating all possible expected and unexpected variables for every smart contract and for the triggers that execute transactions.

Smart contract testing requires expert knowledge of scenarios, business, and transaction variables specific to blockchain networks. In this kind of network, with numerous nodes and combinations, automatic testing should predominate. Since smart contracts enforce a set of rules through strong cryptography, it is also necessary to validate encryption codes using robust testing methodologies. Smart contract testing is complicated and requires specialized validation capabilities. Testers need QA (quality assurance software) skills and API skills in addition to regulatory, business process management, security, and compliance skills.

B. RQ2. DO THEY PROMOTE MODEL-BASED ENGINEERING, EARLY STARTING OF THE TESTING PHASE OR AUTOMATIC SMART CONTRACT GENERATION?

Since a few years ago, modelling tools [59] have been helping document enterprise process functionality and using model transformations to automate software code generation with Unified Model Languages (UML) and other modelling standards. For example, Marchesi (PS01) and Syahputra (PS19) make use of UML-Diagrams to describe the requirements of the applications.

UML for testing, known as the UML 2.0 Testing Profile (U2TP) [13] has also closed the gap between designers and testers by providing a good reason for using UML for

both system modelling and test specification. This allows UML-Design documents to be reused for testing and, most importantly, enables test development in an early phase of system development.

Since models are usually easier to understand than software source code [24], it also makes it possible to improve development productivity and quality. It is easier to check the correctness of a model, and modelling tools can ensure that the deployed code, from the model, has not been modified after its generation [40].

Model-based software engineering [53], [56] is important in blockchain-oriented applications for the following reasons:

- Best practices can be implemented, and well-tested codes generated, thereby reducing the occurrence of vulnerable code. Moreover, model-based tools can improve the verification/validation of smart contract code by applying testing techniques right from the early stages of the SDLC, a practice known as early testing.
- The software code is more difficult to understand than the models. It is, therefore, easier to check the correctness of a model. Software code can also be generated automatically from model-based tools, thus ensure that the generated code has not been modified after it is obtained.
- As they are platform-agnostic, models can avoid lock-in to specific blockchain technology and model-based engineering can be applied across multiple blockchain platforms.

Table 11 shows the distribution of PS concerning this type of proposals.

As the table shows, the types of proposals most addressed by authors are, respectively: (B1) Application of model-based software engineering (48%), (B2) Promotion of early testing (5%), and (B3) Proposal of automatic code generation (48%).

It can be concluded that although early testing helps to reduce the number of defects and, ultimately, cuts the cost of final revisions, the scientific community's efforts are currently not aimed at this approach. Every software engineer knows that if a bug is detected in the final stage of testing [59], changes may then need to be made in the design and analysis phase. It is therefore important to carry out testing in every phase to ensure that software will run according to expectations and will not fail once it has been delivered to the client. When testing is performed at the end of the SDLC, i.e., after the coding phase, there may not be enough time to do it properly, resulting in compromises which could affect the quality of the software [27]. Early testing will provide enough time to identify the absence or inadequacy of any functional requirements. Moreover, test cases obtained from the requirements and shared with the developer's team before the coding phase may reveal new possibilities and help them to estimate the chances of failure in their code [27].

In this context, Koul (PS06) highlights the need to deliver first-time quality while minimizing the impact of testing on the delivery teams. This PS stands out the challenges currently faced in testing such applications. It also acknowledges the need to devise specialized tools and techniques for blockchain-oriented software testing to ensure high standards of quality. In short, the role of software testing is not only to verify the "rightness" of the software but rather to discover defects in time for then be rectified. The goal should be to develop smart contracts with higher quality code and as few errors as possible. Exhaustive testing, covering 100% of a software's functionality, is not possible, but it is important to eliminate the highest number of errors as soon as possible. All members of the development team should be involved early in the SDLC. This will, in turn, have a positive impact on the development of the smart contract. If there are testers at the beginning of the development cycle, then errors can be reported at every step and team members may contribute to remedying those errors. By including early testing throughout this process, smart contracts can be implemented with higher reliability, and lower development costs [27].

TABLE 11. Distribution of primary studies (PS) according to the scope of their proposal.

PS	B1	B2	B3
[PS01]	X		
[PS02]			
[PS03]	X		X
[PS04]	X		X
[PS05]	X		
[PS06]		X	
[PS07]			
[PS08]	X		
[PS09]			
[PS10]	X		X
[PS11]			
[PS12]			
[PS13]	X		X
[PS14]			
[PS15]			
[PS16]	X		
[PS17]			X
[PS18]			X
[PS19]	X		X
[PS20]			
[PS21]			X
[PS22]	X		X
[PS23]			
[PS24]			
[PS25]			X
TOTAL	**10**	**1**	**10**

B1. Application of model-based software engineering
B2: Promotion of early testing
B3: Proposal of automatic smart contract generation

Another important aspect to consider in connection with automatic smart contract generation is the technique used. Table 12 shows the distribution of PS regarding the different techniques used in the proposals.

In our opinion, another important aspect is an automatic smart contract code generation using a model-based software engineering process. This eliminates the manual effort required in coding from design and therefore accelerates the process while decreasing the possibility of errors in comparison with manual coding from requirements or models. As the table shows, the techniques most proposed by authors are, respectively: (C1) Automatic code generation

TABLE 12. Distribution of primary studies (PS) according to the techniques proposed.

PS	C1	C2	C3
[PS01]			
[PS02]			
[PS03]	X		X
[PS04]			X
[PS05]			
[PS06]			
[PS07]			
[PS08]			
[PS09]			
[PS10]			X
[PS11]			
[PS12]			
[PS13]			X
[PS14]			
[PS15]			
[PS16]			
[PS17]			
[PS18]			X
[PS19]		X	
[PS20]			
[PS21]			
[PS22]			X
[PS23]			
[PS24]			
[PS25]			X
TOTAL	**1**	**1**	**7**

Automatic code generation:
C1. Using domain-specific ontologies and/or semantic rules
C2. Using model-based software engineering
C3. Through templates and other utilities

using domain-specific ontologies and/or semantic rules (4%), (C2) Automatic code generation using model-based software engineering (4%), and (C3) Automatic code generation through templates and other utilities (28%).

Model-based software engineering has been gaining traction in the development of embedded software in industries, especially in safety-critical domains [6]. Automatic code generation with model-based development is an important technology that offers software engineers advanced options for requirements gathering and software deployment and verification. In this context, Syahputra (PS19) discusses the development process of a smart contract platform that aims to generate smart contracts for heterogeneous blockchain technologies. With the modeling approach they are using in their paper, UML, and OCL (Object Constraint Language), they implement the workflow and algorithm in a supply chain demo sample. It is important to understand the potential applications of code generation, but technology alone is not going to improve software quality processes. Developers must also establish an SDLC that leverages code generation technologies and yet adheres to well-established software engineering principles like the reduction of complexity, requirements traceability, efficient configuration management and version control.

C. RQ3. WHAT SCIENTIFIC OR EMPIRICAL VALIDATION METHODS WERE USED IN THE DIFFERENT PROPOSALS?

After analyzing the PS shown in Table 13, it can be seen that only 12% of the papers used scientific validation. More

TABLE 13. Distribution of primary studies (PS) according to validation method.

Scientific Validation Method	Primary Studies (PS)	Total	%
Experiment	[PS07][PS09][PS10]	3	12%
Case studies / Proof-of-concept	[PS01][PS02][PS03][PS04] [PS11][PS13][PS19][PS20] [PS22][PS23][PS24]	11	44%
No evaluation	[PS05][PS06][PS08] [PS12][PS14][PS15][PS16] [PS17][PS18][PS21][PS25]	11	44%
	TOTAL	**25**	

specifically, the table details the distribution of each of the PS with respect to the evaluation methods used to validate proposals. As can be seen, 44% of the PS carry out their evaluation by means of experimental case studies or proofs of concept, and yet the same percentage of studies does not even have a full validation plan, making it difficult to verify their assertions.

VI. CONCLUSION AND OPEN ISSUES

This paper presents the results of an SLR which identifies and analyses the state-of-the-art of scientific publications in the field of software design, coding, testing and SDLC phases, all in the context of blockchain smart contracts. To achieve its objectives, the paper follows Kitchenham's most recent method for carrying out an SLR.

This study, however, presents some limitations. Following Kitchenham's method, we did not consider grey literature, which might contain other significant results. Nor papers from other languages rather the English. After conducting a search of potential studies and having selected a few PS, we identified several types of proposals addressing SDLCs in the target context. Specifically, 25 PS were identified once the search protocol described in this paper had been executed. These studies were also classified according to the phases of the SDLC for which they offer support. After conducting the review, open issues were identified.

Requirements gathering and, above all, software testing is among the most important aspects of smart contract development, but they are almost always overlooked. For example, Marchesi (PS01) proposes a software development process to gather the requirements, analyze, design, coding, testing and deploying blockchain applications. The process is based on several Agile practices. But it makes use of UML-Diagrams to describe the design of the applications. This work moves toward this direction providing full modeling of interactions among traditional software and blockchain environment, including Class diagrams, Sequence diagrams, Smart Contracts diagrams, etc. Other authors as Syahputra (PS05) discusses on the development process of smart contract platform that aims to create a smart contract for heterogeneous blockchain technologies. The author starts the process by creating blueprint design and modelling with UML and OCL.

Whenever we create a smart contract, we must make sure that it works properly. Emphasis should be placed on the functionality, security, and performance of smart contracts,

and testing is the only way to reduce the risk. Functional tests and non-functional tests are both necessary for validating and correcting a contract's behaviour before it is released. Smart contract testing is crucial in the blockchain development process, because blockchain, with its immutability, forgives no errors. The only way to fix a bug on an already deployed smart contract is to deploy a new release of that agreement; the old version with the bug will remain on the blockchain (and will stay there forever).

Efficiently testing a smart contract before deploying it will ensure that it works as expected, following the established requirements. In functional testing, all business rules or requirements - including valid/invalid arguments, boundary values, and argument combinations - should be verified in various test cases. In this context, Koul (PS06) highlights the need to deliver first-time quality while minimizing the impact of testing on the development teams. This PS stands out the challenges currently faced in testing such applications. It also acknowledges the need to devise specialized tools and techniques for blockchain-oriented software testing to ensure high standards of quality.

In traditional development software processes, analysis and design information is usually transferred and handled in the form of text-based documents, which are difficult to understand and subject to interpretation bias. Engineers create embedded code from those text-based documents, leading to error-prone processes. There is also little scope to ascertain whether or not functionality is being implemented correctly. In this regard, the main benefit of using model-based development software is the auto-generation of code, which can eliminate human error and facilitate code reusability. Other important benefits of model-based software engineering are (i) disciplined analysis, design and continuous testing to improve development effectiveness, (ii) the possibility of using verification as a parallel activity taking place throughout the development process, because test cases can be automatically generated from models, and (iii) reusable models which can reduce development times and costs. Given all this, the following conclusions were from this SLR:

- The efforts of the scientific community should be aimed at implementing a smart contract Development Life Cycle with clearly defined phases and a common vocabulary for each step.
- In this context, the software testing phase is critical to efficient blockchain development. Smart contract testing requires expert knowledge of scenarios, business, and transaction variables specific to blockchain networks.
- Model-based design and model-based testing should be promoted by the scientific community because model-based software engineering makes it possible to find errors in design and code. Model-based tools can also improve the verification/validation of smart contract code through the application of testing techniques right from the early stages of the SDLC. Early testing provides enough time to identify the absences and the inadequacies of functional requirements. In addition, test cases composed amid prerequisites and shared with the development team before the coding phase can reveal new possibilities and help developers estimate the chances of failure in their code.
- Automatic smart contract (code) generation in a model-based software engineering process is vital for effective quality development. It eliminates the manual effort required when coding from design and therefore accelerates the process while decreasing the chance of error in comparison with manual coding from requirements or models.

Having completed this SLR, we plan to explore a new line of potentially very interesting research: the possibilities offered by the model-based software engineering paradigm, which may facilitate mechanisms for validating smart contracts by applying early testing techniques before a contract code is deployed in the blockchain network. The use of this approach has given very satisfactory results in other technologies and its application would appear to be of great interest in blockchain technology.

Such application constitutes our future objective, and we also intend to apply other methods such as the one published by ISD2014 [22] to enrich this study, where not only new articles should appear, but also the existing grey and commercial literature. For example, during the last few months, new relevant papers have appeared as Pankov [49] lists several existing tools for testing and verifying blockchain systems and smart contracts, and also identifies the problem of the lack of an appropriate normative base and standards in this area. Miraz and Ali [47] explores the 6 traditional SDLC models and advocates that there is an urgent need to develop a new standard model(s). This paper concludes that the traditional SDLC models are unsuitable for blockchain-enabled smart contract-based applications. These studies confirm the current need and indicate that we are on the right track in our investigations.

REFERENCES

[1] F. Al Khalil, T. Butler, L. O'Brien, and M. Ceci, "Trust in smart contracts is a process, as well," in *Proc. Financial Cryptogr. Workshops*, 2017, pp. 510–519.
[2] W. Al-Saqaf and N. Seidler, "Blockchain technology for social impact: Opportunities and challenges ahead," *J. Cyber Policy*, vol. 2, no. 3, pp. 338–354, Sep. 2017.
[3] M. Alharby, A. Aldweesh, and A. V. Moorsel, "Blockchain-based smart contracts: A systematic mapping study of academic research (2018)," in *Proc. Int. Conf. Cloud Comput., Big Data Blockchain (ICCBB)*, Nov. 2018, pp. 1–6.
[4] F. Batubara, J. Ubacht, and M. Janssen, "Challenges of blockchain technology adoption for e-government: A systematic literature review," in *Proc. 19th Annu. Int. Conf. Digit. Government Res., Governance Data Age*, 2018, pp. 1–9.
[5] O. Benediktsson, D. Dalcher, and H. Thorbergsson, "Comparison of software development life cycles: A multiproject experiment," *IEE Proc.-Softw.*, vol. 153, no. 3, pp. 87–101, 2006.
[6] M. Bialy, V. Pantelic, J. Jaskolka, A. Schaap, L. Patcas, M. Lawford, and A. Wassyng, "Software engineering for model-based development by domain experts," in *Handbook of System Safety and Security*. Amsterdam, The Netherlands: Elsevier, 2017, pp. 39–64.

[7] R. Blanco, J. G. Enríquez, F. J. Domínguez-Mayo, M. J. Escalona, and J. Tuya, "Early integration testing for entity reconciliation in the context of heterogeneous data sources," *IEEE Trans. Rel.*, vol. 67, no. 2, pp. 538–556, Jun. 2018.

[8] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *Ethereum White Paper*, vol. 3, no. 37, 2014. [Online]. Available: https://cryptorating.eu/whitepapers/Ethereum/Ethereum_white_paper.pdf

[9] O. Choudhury, N. Rudolph, I. Sylla, N. Fairoza, and A. Das, "Auto-generation of smart contracts from domain-specific ontologies and semantic rules," in *Proc. IEEE Int. Conf. Internet Things (iThings), IEEE Green Comput. Commun. (GreenCom), IEEE Cyber, Phys. Social Comput. (CPSCom), IEEE Smart Data (SmartData)*, Jul. 2018, pp. 963–970.

[10] C. D. Clack, "Smart contract templates: Legal semantics and code validation," *J. Digit. Banking*, vol. 2, no. 4, pp. 338–352, 2018.

[11] C. D. Clack, V. A. Bakshi, and L. Braine, "Smart contract templates: Essential requirements and design options," 2016, *arXiv:1612.04496*. [Online]. Available: https://arxiv.org/abs/1612.04496

[12] C. D. Clack, V. A. Bakshi, and L. Braine, "Smart contract templates: Foundations, design landscape and research directions," 2016, *arXiv:1608.00771*. [Online]. Available: https://arxiv.org/abs/1608.00771

[13] A. Cockburn, *Writing Effective Use Cases*, vol. 303. Boston, MA, USA: Addison-Wesley, 2001, pp. 158–161.

[14] C. Cutilla, J. A. G. García, J. J. G. Rodríguez, P. D. Mayo, M. J. E. Cuaresma, L. Rodríguez, and F. J. D. Mayo, "Model-driven test engineering: A practical analysis in the AQUA-WS project," in *Proc. 7th Int. Conf. Softw. Paradigm Trends (ICSOFT)*, 2012, pp. 111–119.

[15] F. Q. B. da Silva, A. L. M. Santos, S. Soares, A. C. C. França, C. V. F. Monteiro, and F. F. Maciel, "Six years of systematic literature reviews in software engineering: An updated tertiary study," *Inf. Softw. Technol.*, vol. 53, no. 9, pp. 899–913, Sep. 2011.

[16] J. de Kruijff and H. Weigand, "Ontologies for commitment-based smart contracts," in *Proc. OTM Confederated Int. Conf. Move Meaningful Internet Syst.*, Oct. 2017, pp. 383–398.

[17] S. Dhaiouir and S. Assar, "A systematic literature review of blockchain-enabled smart contracts: Platforms, languages, consensus, applications and choice criteria," in *Proc. Int. Conf. Res. Challenges Inf. Sci.* Cham, Switzerland: Springer, 2020, pp. 249–266.

[18] A. Dolgui, D. Ivanov, S. Potryasaev, B. Sokolov, M. Ivanova, and F. Werner, "Blockchain-oriented dynamic modelling of smart contract design and execution in the supply chain," *Int. J. Prod. Res.*, vol. 58, no. 7, pp. 2184–2199, Apr. 2020.

[19] I. Drave, S. Hillemacher, T. Greifenberg, S. Kriebel, E. Kusmenko, M. Markthaler, P. Orth, K. S. Salman, J. Richenhagen, B. Rumpe, C. Schulze, M. von Wenckstern, and A. Wortmann, "SMArDT modeling for automotive software testing," *Softw., Pract. Exp.*, vol. 49, no. 2, pp. 301–328, Feb. 2019.

[20] J. G. Enríquez, R. Blanco, F. J. Domínguez-Mayo, J. Tuya, and M. J. Escalona, "Towards an MDE-based approach to test entity reconciliation applications," in *Proc. 7th Int. Workshop Automating Test Case Design, Selection, Eval. (A-TEST)*, 2016, pp. 74–77.

[21] M. J. Escalona and G. Aragón, "NDT. A model-driven approach for Web requirements," *IEEE Trans. Softw. Eng.*, vol. 34, no. 3, pp. 377–390, May 2008.

[22] M. J. Escalona, J. A. Garcia-Garcia, F. J. D. Mayo, and I. Ramos, "Technical tool surveys and comparative studies: A systematical approach," in *23rd Int. Conf. Inf. Syst. Develop. (ISD CROATIA)*. ISD, Sep. 2014.

[23] S. Feyer, S. Siebert, B. Gipp, A. Aizawa, and J. Beel, "Integration of the scientific recommender system Mr. DLib into the reference manager JabRef," in *Proc. Eur. Conf. Inf. Retr.* Cham, Switzerland: Springer, 2017, pp. 770–774.

[24] A. Forward and T. C. Lethbridge, "Problems and opportunities for model-centric versus code-centric software development: A survey of software professionals," in *Proc. Int. Workshop Models Softw. Eng.*, 2008, pp. 27–32.

[25] I. Grigg. (Feb. 2015). *On the Intersection of Ricardian and Smart Contracts*. [Online]. Available: https://iang.org/papers/

[26] J. Gutierrez, C. Nebut, M. Escalona, M. Mejías, and I. Ramos, "Visualization of use cases through automatically generated activity diagrams," in *Proc. Int. Conf. Model Driven Eng. Lang. Syst.* vol. 5301, Sep. 2008, pp. 83–96.

[27] S. Jain and H. Joshi, "Impact of early testing on cost, reliability and release time," in *Proc. 5th Int. Conf. Rel., Infocom Technol. Optim., Trends Future Directions (ICRITO)*, Sep. 2016, pp. 318–322.

[28] T. Jindal, "Importance of testing in SDLC," *Int. J. Eng. Appl. Comput. Sci.*, vol. 1, no. 2, pp. 54–56, Dec. 2016.

[29] E. Karafiloski and A. Mishev, "Blockchain solutions for big data challenges: A literature review," in *Proc. 17th Int. Conf. Smart Technol.*, 2017, pp. 763–768.

[30] B. Kitchenham and P. Brereton, "A systematic review of systematic review process research in software engineering," *Inf. Softw. Technol.*, vol. 55, no. 12, pp. 2049–2075, Dec. 2013.

[31] B. Kitchenham, R. Pretorius, D. Budgen, O. P. Brereton, M. Turner, M. Niazi, and S. Linkman, "Systematic literature reviews in software engineering—A tertiary study," *Inf. Softw. Technol.*, vol. 52, no. 8, pp. 792–805, 2010.

[32] B. Koteska, E. Karafiloski, and A. Mishev, "Blockchain implementation quality challenges: A literature review," in *Proc. 6th Workshop Softw. Qual., Anal., Monit., Improvement, Appl. (SQAMIA)*, 2017, pp. 11–13.

[33] R. Koul, "Blockchain oriented software testing–challenges and approaches," in *Proc. 3rd Int. Conf. Converg. Technol. (ICT)*, Apr. 2018, pp. 1–6.

[34] S.-M. Lee, S. Park, and Y. B. Park, "Formal specification technique in smart contract verification," in *Proc. Int. Conf. Platform Technol. Service (PlatCon)*, Jan. 2019, pp. 1–4.

[35] E. Leka, B. Selimi, and L. Lamani, "Systematic literature review of blockchain applications: Smart contracts," in *Proc. Int. Conf. Inf. Technol. (InfoTech)*, Sep. 2019, pp. 1–3.

[36] W. E. Lewis, *Software Testing and Continuous Quality Improvement*. Boca Raton, FL, USA: CRC Press, 2017.

[37] C.-F. Liao, C.-J. Cheng, K. Chen, C.-H. Lai, T. Chiu, and C. Wu-Lee, "Toward a service platform for developing smart contracts on blockchain in BDD and TDD styles," in *Proc. IEEE 10th Conf. Service-Oriented Comput. Appl. (SOCA)*, Nov. 2017, pp. 133–140.

[38] Y. Liu, Q. Lu, X. Xu, L. Zhu, and H. Yao, "Applying design patterns in smart contracts," in *Proc. Int. Conf. Blockchain*. Cham, Switzerland: Springer, 2018, pp. 92–106.

[39] S. K. Lo, Y. Liu, S. Y. Chia, X. Xu, Q. Lu, L. Zhu, and H. Ning, "Analysis of blockchain solutions for IoT: A systematic literature review," *IEEE Access*, vol. 7, pp. 58822–58835, 2019.

[40] Q. Lu, I. Weber, and M. Staples, "Why model-driven engineering fits the needs for blockchain application development," *IEEE Blockchain Tech. Briefs*, p. 3, Jul. 2018.

[41] D. Macrinici, C. Cartofeanu, and S. Gao, "Smart contract applications within blockchain technology: A systematic mapping study," *Telematics Informat.*, vol. 35, no. 8, pp. 2337–2354, Dec. 2018.

[42] D. Mao, F. Wang, Y. Wang, and Z. Hao, "Visual and user-defined smart contract designing system based on automatic coding," *IEEE Access*, vol. 7, pp. 73131–73143, 2019.

[43] M. Marchesi, L. Marchesi, and R. Tonelli, "An agile software engineering method to design blockchain applications," in *Proc. 14th Central Eastern Eur. Softw. Eng. Conf. Russia ZZZ (CEE-SECR)*, 2018, pp. 1–8.

[44] A. Mavridou and A. Laszka, "Designing secure ethereum smart contracts: A finite state machine based approach," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Berlin, Germany: Springer, 2018, pp. 523–540.

[45] A. Mavridou and A. Laszka, "Tool demonstration: FSolidM for designing secure Ethereum smart contracts," in *Proc. POST*, 2018, pp. 270–277.

[46] A. Mavridou, A. Laszka, E. Stachtiari, and A. Dubey, "VeriSolid: Correct-by-design smart contracts for Ethereum," in *Financial Cryptography and Data Security*. Cham, Switzerland: Springer, 2019, pp. 446–465, doi: 10.1007/978-3-030-32101-7_27.

[47] M. H. Miraz and M. Ali, "Blockchain enabled smart contract based applications: Deficiencies with the software development life cycle models," 2020, *arXiv:2001.10589*. [Online]. Available: http://arxiv.org/abs/2001.10589

[48] E. W. T. Ngai, Y. Hu, Y. H. Wong, Y. Chen, and X. Sun, "The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature," *Decis. Support Syst.*, vol. 50, no. 3, pp. 559–569, Feb. 2011.

[49] K. N. Pankov, "Testing, verification and validation of distributed ledger systems," in *Proc. Syst. Signals Generating Process. Field Board Commun.*, Mar. 2020, pp. 1–9.

[50] R. M. Parizi, A. Dehghantanha, K.-K. Raymond Choo, and A. Singh, "Empirical vulnerability analysis of automated smart contracts security testing on blockchains," 2018, *arXiv:1809.02702*. [Online]. Available: http://arxiv.org/abs/1809.02702

[51] A. Permenev, D. Dimitrov, P. Tsankov, D. Drachsler-Cohen, and M. Vechev, "VerX: Safety verification of smart contracts," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2020, pp. 18–20.

[52] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *Proc. 12th Int. Conf. Eval. Assessment Softw. Eng. (EASE)*, 2008, pp. 1–10.

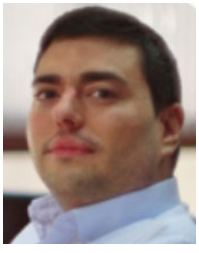
[53] K. Pohl, H. Hönninger, R. Achatz, and M. Broy, *Model-Based Engineering of Embedded Systems. The SPES 2020 Methodology*. Berlin, Germany: Springer, 2012, doi: 10.1007/978-3-642-34614-9.
[54] S. Porru, A. Pinna, M. Marchesi, and R. Tonelli, "Blockchain-oriented software engineering: Challenges and new directions," in *Proc. IEEE/ACM 39th Int. Conf. Softw. Eng. Companion (ICSE-C)*, May 2017, pp. 169–171.
[55] S. Pranto, L. Jardim, T. Oliveira, and P. Ruivo, "Literature review on blockchain with focus on supply chain," in *Proc. 19th Conf. Portuguese Assoc. Inf. Syst. (CAPSI)*. Lisboa, Portugal: CAPSI, 2019.
[56] A. L. Ramos, J. V. Ferreira, and J. Barceló, "Model-based systems engineering: An emerging approach for modern systems," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 1, pp. 101–111, Jan. 2012.
[57] S. Barjtya, A. Sharma, and U. Rani, "A detailed study of software development life cycle (SDLC) models," *Int. J. Eng. Comput. Sci.*, vol. 6, no. 7, pp. 1–4, 2017.
[58] W. W. Royce, "Managing the development of large software systems: Concepts and techniques," in *Proc. 9th Int. Conf. Softw. Eng.*, 1987, pp. 328–338.
[59] N. Sánchez-Gómez, L. Morales-Trujillo, J. J. Gutiérrez, and J. Torres-Valderrama, "The importance of testing in the early stages of smart contract development life cycle," *J. Web Eng.*, vol. 6, no. 2, pp. 215–242, Jun. 2020.
[60] C. M. Schmucker, A. Blümle, L. K. Schell, G. Schwarzer, P. Oeller, L. Cabrera, E. von Elm, M. Briel, and J. J. Meerpohl, "Systematic review finds that study data not published in full text articles have unclear impact on meta-analyses results in medical research," *PLoS ONE*, vol. 12, no. 4, Apr. 2017, Art. no. e0176210.
[61] H. Sheikh, R. M. Azmathullah, and F. Rizwan, "Smart contract development, adoption and challenges: The powered blockchain," *Int. Res. J. Adv. Eng. Sci.*, vol. 4, no. 2, pp. 321–324, May 2019.
[62] E. Shishkin, "Debugging smart contract's business logic using symbolic model checking," *Program. Comput. Softw.*, vol. 45, no. 8, pp. 590–599, Dec. 2019.
[63] C. Sillaber and B. Waltl, "Life cycle of smart contracts in blockchain ecosystems," *Datenschutz und Datensicherheit-DuD*, vol. 41, no. 8, pp. 497–500, 2017.
[64] M. Surjandy, A. Hidayanto, and H. Prabowo, "The latest adoption blockchain technology in supply chain management: A systematic literature review," *ICIC Exp. Lett.*, vol. 13, no. 10, pp. 913–920, 2019.
[65] H. Syahputra and H. Weigand, "The development of smart contracts for heterogeneous blockchains," in *Enterprise Interoperability VIII*. Cham, Switzerland: Springer, 2019, pp. 229–238.
[66] N. Szabo, "Smart contracts," unpublished manuscript, 1994. [Online]. Available: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html
[67] T. Tateishi, S. Yoshihama, N. Sato, and S. Saito, "Automatic smart contract generation using controlled natural language and template," *IBM J. Res. Develop.*, vol. 63, nos. 2–3, pp. 6:1–6:12, Mar. 2019.
[68] Y. Tribis, A. El Bouchti, and H. Bouayad, "Supply chain management based on blockchain: A systematic mapping study," in *Proc. MATEC Web Conf.*, vol. 200. EDP Sciences, 2018, p. 20.
[69] W.-T. Tsai, N. Ge, J. Jiang, K. Feng, and J. He, "Invited paper: Beagle: A new framework for smart contracts taking account of law," in *Proc. IEEE Int. Conf. Service-Oriented Syst. Eng. (SOSE)*, Apr. 2019, pp. 134–145.
[70] R. Van Der Straeten, T. Mens, and S. Van Baelen, "Challenges in model-driven software engineering," in *Proc. Int. Conf. Model Driven Eng. Lang. Syst.* Berlin, Germany: Springer, 2008, pp. 35–47.
[71] C. Wohlin and R. Prikladniki, "Systematic literature reviews in software engineering," *Inf. Softw. Technol.*, vol. 55, pp. 919–920, Jun. 2013.
[72] V. Yadav and A. Singh, "A systematic literature review of blockchain technology in agriculture," in *Proc. Int. Conf. Ind. Eng. Oper. Manage.*, 2019, pp. 973–981.
[73] S. Yaqoob, M. Murad, R. Talib, A. Dawood, S. Saleem, F. Arif, and A. Nadeem, "Use of blockchain in healthcare: A systematic literature review," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 5, pp. 644–653, 2019.
[74] H. Yumna, M. Khan, M. Ikram, and S. Ilyas, "Use of blockchain in education: A systematic literature review," in *Proc. Asian Conf. Intell. Inf. Database Syst.*, in Lecture Notes in Computer Science: Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, 2019, pp. 191–202.

NICOLÁS SÁNCHEZ-GÓMEZ received the degree in computer engineering and the master's degree in engineering and software technology from the University of Seville.

He has developed a large part of its professional career in the technology and process consultancy sector, both in the private and public sectors. Throughout more than 30 years of professional experience, he has gone from implementing ICT solutions to supervising work teams, managing clients, and leading ICT projects. He is currently a member of the Web Engineering and Early Testing Research Group. In recent years, he has been coordinating different projects of the research group, including the Project Management Office of the Ministry of Culture (Andalusian Regional Government). He currently has a broad knowledge of the functions and processes that make up the activity environment of the sectors in which he has participated and has completed his studies in computer engineering. He has knowledge and skills of people management, ICT project management, customer management, and practical application of computer engineering methodologies and techniques, in addition to obtaining the Prince2® Foundation Certified, the ISTQB® Certified Tester, Foundation Level and the PMP® Certified. From 2001 to 2009, he developed his professional activity as the Manager of the company Everis, Spain, being responsible for different accounts in public and private sectors. From 1990 to 2001, he has worked for the company Coritel (Accenture Group), where he also carried out management and project management activities.

JESUS TORRES-VALDERRAMA received the M.Sc. and Ph.D. degrees in computer systems in 1997.

He has been working with the Department of Computer Languages and Systems, Seville's University, since 1991, where he is currently a Senior Lecturer. He was the Dean of the School of Computer Engineering, Seville's University, from 2006 to 2014, where he has been the Manager of the Foundation for Research and Development of Information Technology in Andalusia, since 2016. His main research interests include requirements engineering, web-based systems development, user interfaces, usability, and early software testing. In these areas, he has directed several Ph.D. theses and published numerous papers in journals and conferences. He has managed and participated in a high number of projects related to his areas of research.



J. A. GARCÍA-GARCÍA received the Ph.D. degree in computer science from the University of Seville, Spain, in 2015. He is currently a Lecturer and a Researcher with the Department of Computing Languages and Systems, University of Seville. Since 2008, he has participated in Research and Development projects as a Researcher in the Web Engineering and Early Testing Group (IWT2). His current research interests include business process management, business process modeling, model-driven engineering, and quality assurance. He is responsible for the BPM area and responsible for security in IWT2. He also participates as a member of committee in several international conferences and journals.

JAVIER J. GUTIÉRREZ received the Ph.D. degree in computer science from the University of Seville, Spain, in 2011.

Since 2004, he has been a Professor with the Department of Computer Languages and Systems, University of Seville, where he has been a collaborating Professor, since 2006. He is currently a member of the Web Engineering and Early Testing Research Group. Among his most notable research results, it is worth mentioning his transfer to the business world. With the development of the concept of early testing and its integration with the NDT methodology also developed within the research group, he has managed to develop a set of methodological solutions for the development and quality assurance that has been widely used in the Andalusian and national business network or even by international companies. This can be measured not only in the transfer projects, but also in the number of publications with companies and in the tools registered.

M. J. ESCALONA received the Ph.D. degree (Hons.) in computer engineering from the University of Seville, in 2004. She is currently a Full Professor with the Department of Computer Languages and Systems, University of Seville. She is also the Director of the Web Engineering and Early Testing Research Group. Her main research interests include software engineering, specifically to software requirements, software early quality assurance, and early software testing. In these areas, she has directed several Ph.D. theses and published numerous papers in journals and conferences. She is also a member of the editorial board of JWE and IEEE IT Professional and she collaborates as a regular reviewer with several conferences and journals. She has managed and participated in a high number of projects related to her areas of research.