Smart Contract Languages: A Multivocal Mapping Study

ÁNGEL JESÚS VARELA-VACA and ANTONIA M. REINA QUINTERO, Universidad de Sevilla

Blockchain is a disruptive technology that has attracted the attention of the scientific community and companies, as proven by the exponential growth of publications on this topic in recent years. This growing interest is mainly due to the promise that the use of blockchain enables it to be verified, without including any trusted intermediaries, that the information received from the network is authentic and up-to-date. In this respect, blockchain is a distributed database that can be seen as a ledger that records all transactions that have ever been executed. In this context, smart contracts are pieces of software used to facilitate, verify, and enforce the negotiation of a transaction on a blockchain platform. These pieces of software are implemented by using programming languages, which are sometimes provided by the blockchain platforms themselves. This study aims to (1) identify and categorise the state-of-the-art related to smart contract languages, in terms of the existing languages and their main features, and (2) identify new research opportunities. The review has been conducted as a multivocal mapping study that follows the guidelines proposed by Garousi et al. for conducting multivocal literature reviews, as well as the guidelines proposed by Kitchenham and Charters for conducting mapping studies. As a result of the implementation of the review protocol, 4,119 papers were gathered, and 109 of them were selected for extraction. The contributions of this article are twofold: (1) 101 different smart contract languages have been identified and classified according to a variety of criteria; (2) a discussion on the findings and their implications for future research have been outlined. As a conclusion, it could be stated that a rigorous and replicable overview of the state-of-the-art of smart contract languages has been provided that can benefit not only researchers but also practitioners in the field, thanks to its multivocal nature.

CCS Concepts: • **Software and its engineering → Context specific languages**;

Additional Key Words and Phrases: Smart contract language, multivocal literature mapping study, systematic literature review, blockchain

1 INTRODUCTION

In recent years, blockchain and its surrounding technologies (i.e., cryptocurrency) are becoming the cornerstones of digital transformation in the industry [22, 41]. Blockchain could be seen as

Both authors contributed equally to this research.

This research was partially supported by the Ministry of Science and Technology of Spain with projects ECLIPSE (No. RTI2018-094283-B-C33), by the Junta de Andalucía with COPERNICA and METAMORFOSIS projects, and by the European Regional Development Fund (ERDF/FEDER).

Authors' address: Á. J. Varela-Vaca and A. M. Reina Quintero, Universidad de Sevilla, Dpto. Lenguajes y Sistemas Infor-máticos, ETSII Informática. Avda. Reina Mercedes, s/n, Seville, Spain, 41012; emails: {ajvarela, reinaqu}@us.es.

a distributed database that records all the transactions that occur in the network in which smart contracts operate in an autonomous and decentralised way. It can therefore be stated that smart contracts [38] constitute one of the best applications for blockchain technologies and are also crucial to facilitate, verify, and enforce the negotiation of a transaction without third parties in a blockchain: transactions that are trackable and irreversible.

Blockchain [32] has emerged as a disruptive technology for the development of decentralised, secure, and trusted distributed applications. From the business perspective, blockchain smart contracts are conceived to provide trust and transparency on the operations of the organisation. Deloitte states that "blockchain-based smart contracts are a critical step forward, streamlining processes that are currently spread across multiple databases and systems" [34]. In fact, Gartner predicts that organisations that use smart contracts will increase data quality by 50% [16]. From the academia perspective, smart contract research is gaining the interest of the research community: in 2017, publications thereon have almost tripled in number since the year 2014 [28].

As blockchain grows in use and smart contracts spread across different domains (see, for example, the initiative of five global banks to build proof-of-concept platforms that use smart contracts [34]), several solutions and technologies that have emerged in the industry and academy sectors related to smart contracts and the blockchain itself. In parallel, multiple literature reviews have been published in the scientific literature to identify open problems, challenges, gaps, and issues related to blockchain in general, and to smart contracts in particular [3, 4, 28, 39]. Nevertheless, none of these reviews is focused on smart contract languages, nor do any of them integrate "grey literature" to provide insight into the state-of-the-practice. The main aim of this study is to identify and carry out an exhaustive and systematic analysis of the state-of-the-art of smart contract languages by complementing the academic literature with the industrial "grey literature" on the topic. Thus, this study focuses on smart contract languages, their main features, and their technical stuff and puts the emphasis on the high-level, compared with assembly/bytecode languages. As smart contracts are an emerging, practitioner-oriented and application-oriented field, we conduct this review as a multivocal literature mapping study following the guidelines proposed by Garousi et al. [14], Kuhrmann et al. [27], and Kitchenham and Charters [25]. The main difference with traditional mapping studies is that multivocal mapping studies include both scientific literature and "grey literature." Including "grey literature" for combining the academic state-of-the-art-and-practice is essential in practitioner-oriented fields to include the state-of-practice, because a large majority of practitioners do not publish in academic forums [13].

The rest of the article is structured as follows: Section 2 presents the background of this work. Section 3 discusses related work. In Section 4, the methodology we followed to conduct the review is illustrated, with special focus on research questions, the search process, the study selection, the quality assessment, and the data extraction process. Section 5 introduces the data extraction results, and research questions are discussed in Section 6. After that, Section 7 then analyses the threats to the validity of this study, and, finally, Section 8 gives the final remarks, conclusions, and future proposals for this work.

2 BACKGROUND

In this section, the terminology needed to understand the full article is introduced. While Section 2.1 explains the main concepts related to blockchain and smart contracts, Section 2.2 presents the terminology related to multivocal literature reviews.

2.1 Blockchain and Smart Contracts

The term *blockchain* was first introduced by Satoshi Nakamoto in Reference [32]. The blockchain is informally defined as a distributed database that records all the transactions that occur in the

network. Distribution in this context means that: (1) each node in the network has the same copy of the database at the same time; and (2) all the changes in the blockchain are reflected in all the copies of the network. The main characteristic of a blockchain is that the blocks introduced into the database are immutable and previously verified. Thus, there is a process in which the new blocks are verified (i.e., proof-of-work) by a set of members (i.e., miners) of the network without human intervention. Once the block is verified, it is introduced into the chain. The main issue is that the verified blocks cannot be modified after having been introduced into the chain. For that reason, a blockchain can be seen as a distributed ledger. There are different types of blockchain: (1) public, with no access restrictions; (2) private, which requires an invitation to access; and (3) hybrid, which combines private and public characteristics.

The term *smart contract* was introduced by Szabo in Reference [38] as follows: *"Smart contracts combine protocols with user interfaces to formalise and secure relationships over computer networks."* A smart contract can be defined as a piece of software (i.e., executable program) employed to facilitate, verify, and enforce the negotiation of a transaction in a blockchain.[1] The main concern on the smart contract is the verification process (i.e., proof-of-work), which is carried out without third parties and without human intervention. In general, each blockchain platform provides a programming language to specify smart contracts. Thus, a smart contract language such as Solidity was devised for Ethereum, while Script was devised for Bitcoin. Smart contract languages are critical in the functioning of a blockchain, and hence a bug or malfunction in the smart contract can produce security problems, as demonstrated with the DAO attack [5].

2.2 Multivocal Literature Review

Studies in software engineering can be classified as primary studies if they present a primary work, or as secondary studies if they summarise and classify published primary studies. Secondary studies are gaining attention as a tool to acquire knowledge in a systematic and reproducible way. Hence, secondary studies in software engineering can be surveys, Systematic Literature Reviews (SLRs) or Systematic Mapping Studies (SMSs). Surveys are not conducted systematically and with an explicit protocol [24]. In contrast, both SLRs and SMSs aims to provide in-depth reviews of primary studies in a way that is unbiased and repeatable. The main difference between SMSs and SLRs is that a systematic mapping is a method to build a classification schema for topics studied in a field of interest, that is, it usually provides the big picture of an area of research, while a systematic review is concerned with the analysis of the specific details of the primary studies.

Traditionally, SLRs and SMSs review primary studies that are formally published literature (i.e., scientific literature or white literature), and they do not include "grey literature," which is produced outside academic forums. Examples of "grey literature" are white papers, magazines, news articles, videos, websites, GitHub pages, Wiki articles, blogs, emails, and tweets. Literature reviews that include both white and grey literature are termed as Multivocal. Thus, we can have Multivocal Literature Reviews and Multivocal Mapping Studies, which, as input, present not only peer-reviewed primary studies but also include other sources from grey literature. Note that the Multivocal nature is important in practitioner-oriented and application-oriented fields to include the vision of practitioners and to identify emerging research topics [14].

3 RELATED WORK

The increasing interest in blockchain, smart contracts, and cryptocurrencies has also led to a growing number of tertiary studies that answer various research questions related to blockchain in

[1]Note that this is an informal definition which does not refer to a legal binding contract. Although the term might be a misnomer, it is commonly employed in the blockchain arena.

Table 1. Summary of Related Work

Type of Study	ID	Ref.	Year	Title
Mapping study	SMS1	[4]	2017	A systematic mapping study on current research topics in smart contracts
	SMS2	[3]	2017	Blockchain-based smart contracts: A systematic mapping study
	SMS3	[28]	2018	Smart contract applications within blockchain technology: A systematic mapping study
	SMS4	[39]	2019	Use of blockchain smart contracts in software engineering: A systematic mapping
SLR	SLR1	[31]	2019	A review paper on smart contracts in blockchain network-based applications
	SLR2	[19]	2019	A comparison of approaches for visualizing blockchains and smart contracts
	SLR3	[37]	2020	Blockchain smart contracts formalization: Approaches and challenges to address vulnerabilities
Survey	SUR1	[36]	2016	Scripting smart contracts for distributed ledger technology
	SUR2	[5]	2017	A survey of attacks on Ethereum smart contracts SoK
	SUR3	[17]	2018	On legal contracts, imperative and declarative smart contracts, and blockchain systems
	SUR4	[43]	2018	An overview of smart contract: Architecture, applications, and future trends
	SUR5	[29]	2018	Smart contracts and opportunities for formal methods
	SUR6	[20]	2018	Towards safer smart contracts: A survey of languages and verification methods
	SUR7	[49]	2019	Smart contract development: Challenges and opportunities
	SUR8	[1]	2019	Blockchain and smart contracts
	SUR9	[48]	2020	An overview on smart contracts: Challenges, advances, and platforms
	SUR10	[30]	2020	Empirical evaluation of blockchain smart contracts

general, and smart contracts in particular. Furthermore, depending on their aim, the studies have been conducted as mapping studies, systematic literature reviews, or surveys. The tertiary studies we have considered as related work are those focused on smart contracts, and are listed and classified in terms of the type of tertiary study on Table 1.

The following subsections introduce the related work grouped in terms of the type of tertiary study: mapping study, systematic literature review, and survey.

3.1 Mapping Studies

This group is the most closely related to our study, in the sense that we also present a mapping study. In general, none of the studies in this group tackles the identification of smart contract languages. Furthermore, it should be borne in mind that the number of papers extracted to conduct those mappings is lower than the selected primary studies of our mapping. However, the most important difference with respect to our study is that none of them takes grey literature into account. Table 2 lists the set of mapping studies found as related work as well as the number of primary studies (cf., Num) they have included and the research questions they have addressed.

The mapping study by Alharby et al. [3, 4] aims to identify current research topics, applications, and open challenges for future studies in smart contracts from a technical point of view, that is, it is not focused on languages, but on different aspects such as contract source code, security, privacy, and performance. They identify 24 primary studies in scientific (white) literature and the conclusions are related to general aspects such as scalability and performance issues, among others, but no conclusion is drawn related to smart contract languages themselves.

Table 2. Research Questions of Mapping Studies Considered as Related Work

ID	Ref.	Num	Research Questions
MS1	[4]	24	RQ1. What are the current research topics on smart contracts?
MS2	[3]		RQ2. What are the current smart contract applications?
			RQ3. What are the research gaps that need to be addressed in future studies?
MS3	[28]	64	RQ1. Which publication channels are the main targets for research within the field of blockchain-based smart contracts?
			RQ2. How has the frequency of identified publication channels related to the field of blockchain-based smart contracts changed over time?
			RQ3. What are the research types found in the identified papers related to the field of blockchain-based smart contracts?
			RQ4. Are the identified papers empirically validated?
			RQ5. What are the reported approaches in blockchain-based smart contract research that address the problems identified in RQ6?
			RQ6. What are the problems that relate to the applications of smart contracts within the blockchain-based platforms?
			RQ7. What are the corresponding solutions to the problems identified in RQ6?
MS4	[39]	8	RQ1. What are the challenges or benefits of using blockchain-based smart contracts in software engineering?
			RQ2. What are the most reported uses of blockchain-based smart contract in software engineering?
			RQ3. What aspects of software engineering are most affected by initiatives in blockchain-based smart contracts?

Macrinici et al. [28] focus on smart contract applications within the blockchain in general, but no mention is made of any particular smart contract language. They identify 64 primary studies in the scientific literature, and do not cover grey literature. Their research questions range from demographic questions to questions related to the approaches, the problems found, and their solutions. They categorise the problems found in smart contracts in problems related to the blockchain mechanism, to the virtual machine, and to the contract source code.

Finally, Tarig et al. [39] carry out a mapping study to identify the challenges or the benefits of using smart contracts in software engineering. Their research questions are general questions oriented towards ascertaining how software engineering could benefit from smart contracts, and are not focused on smart contract languages. They identify eight primary studies in the scientific literature. Their conclusions range from the lack of professionals with skills in both blockchain and software engineering, to security issues.

3.2 Systematic Literature Reviews

The research questions of systematic literature reviews are more focused compared to the questions proposed in mapping studies. Several systematic literature reviews are related to general blockchains [8, 9, 18, 21, 23, 31, 40, 47]. They turn a blind eye to smart contracts and focus on different aspects of blockchain.

The reviews whose goal is related to smart contracts are found in References [19, 31, 37]. Naaz et al. [31] focus on applications of smart contracts and their use cases, and not on the languages themselves. Furthermore, a systematic approach has not been followed, as a consequence, the research questions are not explicitly specified in the paper nor the set of identified papers. Härer and Fill [19] focus exclusively on visual modelling of smart contracts and also fail to follow a systematic approach. Singh et al. [37] focus on smart formalisation and they review only 35 primary studies, and as a systematic literature review their research questions are more specific than ours. Finally, none of the reviews includes grey literature.

3.3 Surveys

Surveys are another form of tertiary studies that, in general, are less rigorous than systematic literature reviews. We have found a set of surveys related to smart contracts (cf., Table 1). Many of the studies that belong to this group have not been conducted systematically, and as a consequence, neither the research questions nor the identified studies are explicit and they are difficult to replicate. Furthermore, none of these surveys includes grey literature.

Seijas et al. [36] give an overview of the scripting languages used in existing cryptocurrencies, but only review the details of Nxt and Ethereum in the context of a high-level overview of Distributed Ledger Technology and cryptocurrencies. Atzei et al. [5] survey security vulnerabilities, but only those of Ethereum smart contracts. Governatori et al. [17] survey the suitability of smart contract programming languages to facilitate legal interactions through legal contracts, but focus on the programming paradigm (imperative or declarative) and from a more theoretical point of view. Wang et al. [43] give an overview of the recent advances of smart contracts and present their future development trends, but focus only on the Ethereum and Hyperledger platforms. Miller et al. [29] survey the existing smart contract ecosystems and the existing tools for the analysis of smart contracts. Dwivedi et al. [11] address whether current smart contract languages are appropriate for guiding business collaboration in a legally and relevant way. Zou et al. [49] focus their survey on determining the differences between traditional software development and smart contract software development from the developers point of view, and focus mainly on Ethereum. Abdelhamid&Hassan's survey [1] aims to assist developers in their decision regarding the suitability of the blockchain and smart contract technology to a specific application area by giving an overview of the blockchain technology. They are not focused on smart contract languages. Mokdad and Hewahi [30] propose an evaluation framework to assess smart contracts obtained as a result of the survey. The framework is focused on infrastructure and development criteria. Their evaluation only covers the most prominent smart contract platforms: Ethereum, EOS, and Stellar blockchains. The authors define the prominence according to the market capital.

Finally, the most closely related surveys to our mapping study are References [20, 48]. Harz and Knottenbelt [20] identify 26 smart contract languages and present different verification methods. The main differences are that we provide an overview of the research field (since we conduct a mapping study), and we have followed a systematic approach. Their goal is to characterise the languages, and our goal is to characterise the research field. Zheng et al. [48] focus on the challenges and recent technical advances in smart contracts, compare smart contract platforms, and give a categorisation of smart contract applications along with several representative examples. The main difference with our approach is that their survey is focused on implementation languages, and they do not cover specification languages nor deal with grey literature.

4 METHOD

The main purpose of our study is to observe, document, analyse and characterise the state-of-the-art-and-practice related to smart contract languages, both in academy and industry. Our goal is to identify the current smart contract languages, their main features, and their technological background.

Since blockchain is an emerging and disruptive technology that has a strong link with industry, this study has been performed as a multivocal mapping study that takes the grey literature into account. In this review, the best practices and guidelines were followed for conducting SLRs in software engineering, as proposed by Kitchenham et al. in References [25–27], the guidelines for multivocal literature reviews proposed by Garousi et al. in Reference [14], and the guidelines for

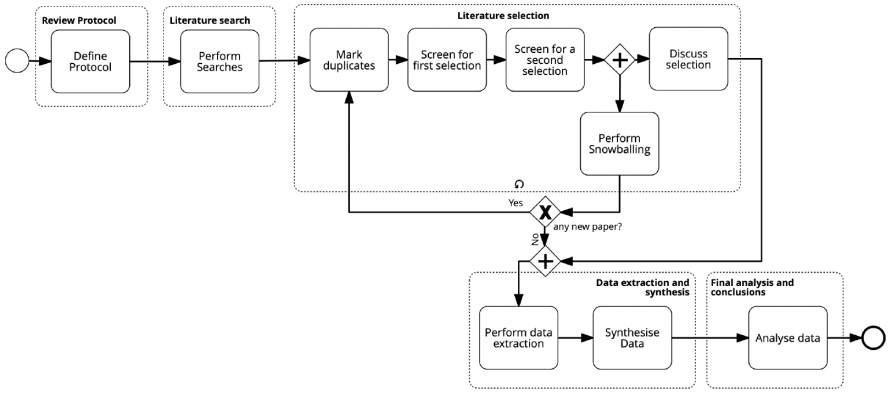


Fig. 1. Process followed to conduct the review.

snowballing proposed by Wohlin [45]. To support the review process, we have used StArt[2] [12], a software tool that gives support to the SLR protocol proposed by Kitchenham et al. [25].

In accordance with Kitchenham guidelines, a review is composed of three stages: planning, conducting and reporting. The protocol is developed during the planning stage. As a result of this stage, not only a plan where all the details about the review protocol is devised, but the responsibilities of the reviewers and the task assignments to conduct the study are also defined. The review protocol includes research questions, search and evaluation strategies, inclusion/exclusion criteria, quality assessment, data collection forms, and methods of analysis. The conducting stage (or execution stage, as it is labelled in StArt) consists of the execution of the different steps of the plan defined in the previous stage. Finally, the reporting stage (or summarisation, as it is labelled in StArt) produces a report as a final result. Figure 1 depicts a process with the different activities executed while conducting the review.

The main differences between the method used in this study in comparison with the method proposed by Kitchenham et al. include:

- We used a broad automated search with a later manual snowballing rather than a restricted manual search process.
- The studies were collected at three different moments in time, one at the beginning of the process, another one just before the data synthesis, and during a second round for updating the searches.
- We have included grey literature throughout the snowballing process. The aim of including grey literature with snowballing is twofold: to narrow the searches, if we compare it with a general search on Google; and to have references that are verified, as they have been included in white literature.

4.1 The Research Questions

The definition of the research questions constitutes a critical activity of planning the mapping study or SLR, because these questions drive the whole review process. Certain research questions are broad and concerned with classifying the literature in some way (as it is usually carried out in mapping studies), and other research questions are concerned with a qualitative evaluation of smart contract languages.

[2] http://lapes.dc.ufscar.br/tools/start_tool.

This review aims to analyse the contributions and current research efforts related to the implementation of languages for the definition of smart contract languages, both in academia and industry. To achieve these goals, we defined the following research questions:

- **RQ1:** Which contributions were made over the years?
- **RQ2:** Which are the main researchers in the area of smart contract languages and where are they from?
- **RQ3:** Which are the most influential studies in the area?
- **RQ4:** Which are the top venues?
- **RQ5:** What languages are there in the literature for specifying or implementing smart contracts?
- **RQ6:** Are the languages supported by industry or academia?
- **RQ7:** Which features do these languages have?
  - **RQ7.1:** Is the language supported by one of the current blockchain platforms? Which platform/s supports the language?
  - **RQ7.2:** Are the languages stand-alone or are they defined as an extension of another existing language?
  - **RQ7.3:** Which is the main focus of the research in relation to the smart contract language?
  - **RQ7.4:** Which are the challenges the approach should face?
  - **RQ7.5:** Which are the reported use cases?

Note that RQ1–RQ4 are demographic questions [24], RQ5 is a question that belongs the "Current trends" category, and RQ6 and RQ7 questions are more related to the classification or categorisation of primary studies.

4.2 The Search Process

The search process has been conducted in four phases: two involve automatic searches, the third consists of a manual backwards snowballing approach of the studies selected in the previous phases, while the last phase involves a discussion of the selected papers. A second round has been carried out to update the searches. Hence, in a way, the search process is intermingled with the selection process, because the backwards snowballing is executed only for those studies that are accepted for extraction. Consequently, the backwards snowballing approach is the method employed to incorporate grey literature into our review. Note that this field is practitioner-oriented and practitioners seldom disseminate their results in academic channels. In fact, we have found that the majority of the selected academic primary studies include citations to grey literature, such as white papers, websites, GitHub pages, and blog posts. The search process is composed of the following phases:

- **Phase 1. Initial search:** The initial search was executed in July 2018, and is composed of the following stages:

  (1) **Search:** In this stage, the digital libraries and search engines are queried, and the results are loaded in the StArt tool.
  (2) **Mark duplicates:** This stage includes an automatic step and a manual step. StArt automatically detects and marks references as duplicated. However, StArt does not find all the duplicated references and, as a consequence, a later manual screening is necessary.
  (3) **First selection process:** This stage involves a screening to reject the irrelevant papers. The screening is based on title, abstract, and keywords.

(4) **Second selection process:** The final stage involves a thorough review of the studies that were selected in the previous stage. To this end, the full text of the studies is analysed. This stage produces a set of studies that are ready for extraction.

- **Phase 2. Second search:** The second search was executed in July 2019, and is composed of the following stages:

  (1) **Second search:** During this stage, the digital libraries and the search engines are queried again, that is, the same queries are executed but filtered by year. The aim is to incorporate only the studies published between July 2018 and July 2019.
  (2) **Remove and mark duplicates:** This stage involves a data-cleaning process, because some of the queried search engines do not support the filtering by year or filtering by a specific date. This data cleaning consists of removing from the dataset those studies that were obtained as a result of the same query executed in Phase 1. The resulting studies are then loaded into the StArt tool, and the same automatic and manual steps for marking duplicates are executed. Note that in this step the duplicates refers to studies that have been found by querying a different digital library or search engine.
  (3) **First selection process:** This stage is similar to that in Phase 1, that is, the new studies incorporated in Phase 2.2 are screened based on title, abstract, and keywords.
  (4) **Second selection process:** The thorough review of the papers selected in the previous stage is done, and as a result, a new set of studies ready for extraction is obtained.

- **Phase 3. Backwards snowballing:** This phase is executed in parallel with Stage 4 of Phases 1 and 2. We execute a backwards snowballing process [26], that is, we use the reference list of the accepted paper of extraction obtained as a result of the previous phases. The reference list of each accepted paper is screened, and a set of new studies is incorporated into StArt. The details of the backwards snowballing are explained in Section 4.3.
- **Phase 4. Discussion:** This phase is executed once the paper has been selected for extraction. The two researchers that conducted the review have a meeting to discuss whether the selected paper should be included in the final set of primary studies.
- **Second Round. Update searches:** A second round is executed to update the searches to incorporate the studies published between July 2019 and June 2020. The two researchers that conduct the review re-execute the Phases 1,2,3 and 4 to gather and include the new studies corresponding to the July 2019 to June 2020 period.

During this process, both researchers are responsible for the searches involved in Phase 1. In this phase, Reina-Quintero searched three digital libraries and Varela-Vaca searched another three digital libraries. Reina-Quintero is responsible for the repetition of the searches during Phase 2 and the second round. Finally, the snowballing process as well as the discussion phase are executed by both researchers.

To develop proper search strings, we follow a "Trail-and-error Search" approach. As a result, we have defined fairly simple search strings, as recommended in Reference [26]. Essentially, the following terms were sought (note that Hyperledger Fabric users often use the terms smart contract and chaincode interchangeably):

"smart contract programming model"	"smart-contract programming model"	"smartcontract programming model"
"smart contract language"	"smart-contract language"	"smartcontract language"
"smart contract chain code"	"smart-contract chain code"	"smartcontract chain code"
"smart contract chaincode"	"smart-contract chaincode"	"smartcontract chaincode"

Table 3. Digital Libraries and Search Engines Employed to Collect Data

Search Engine	Type	URL
ACM DL	S	http://dl.acm.org
Springer	S	http://www.springer.com
IEEE Xplore	S	http://ieeexplore.ieee.org
ScienceDirect	S	http://www.sciencedirect.com
Google Scholar	G	http://scholar.google.es
Scopus	G	http://www.scopus.com

Table 4. Query Strings Used in the Different Search Engines

Source	SearchIDs	Search string
ACM DL	S3, S43, S86	("smart-contract language" "smart contract language" "smartcontract language")
	S4, S44, S87	(+("smart contract" "smartcontract" "smart-contract") +"language")
	S88[3]	("smart contract" OR "smartcontract" OR"smart-contract") AND "programming model")
	S89	("smart contract" OR "smartcontract" OR"smart-contract") AND ("chain code" OR "chaincode")
Springer	S0, S45,S80	language AND "smart contract"
	S1, S46, S80	language AND "smartcontract"
	S3[4]	language AND "smart-contract"
	S82	"programming model" AND "smart contract"
	S83	"chain code" AND "smart contract"
	S84	"chain code" AND "smartcontract"
	S85	"chaincode" AND "smart contract"
IEEE Xplore	S5, S47	(("smart contract language") OR "smart-contract language") OR "smartcontract language")
	S6, S48	((("smart contract") OR "smart-contract") OR "smartcontract") AND "language")
	S90[5]	(((((("smart contract") OR "smart-contract") OR "smartcontract")) AND language)))
	S91	(((((("smart contract") OR "smart-contract") OR "smartcontract")) AND "programming model"))))
	S92	(((((("smart contract") OR "smart-contract") OR "smartcontract")) AND ("chaincode" OR "chain code")))))
Science Direct	S7, S49, S93	(("smart contract language") OR "smart-contract language") OR "smartcontract language")
	S8, S50, S94	((("smart contract") OR "smart-contract") OR "smartcontract") AND "language")
	S95	((("smart contract") OR "smart-contract") OR "smartcontract") AND "programming model")
	S96	((("smart contract") OR "smart-contract") OR "smartcontract") AND ("chaincode" OR "chain code")
Google Scholar	S11, S53, S101[6]	"smart contract" or "smart-contract" or "smartcontract" "language"
Scopus	S8, S51, S97	(("smart contract language") OR "smart-contract language") OR "smartcontract language")
	S10, S52, S98	((("smart contract") OR "smart-contract") OR "smartcontract") AND "language")
	S99	(((("smart contract") OR "smart-contract") OR "smartcontract") AND "programming model"))
	S100	(("smart contract" OR "smart-contract" OR "smartcontract") AND ("chaincode" OR "chain code"))

The digital libraries and search engines used as data sources during the search process are listed in Table 3. The Type column indicates whether the search engine is specific (S) or a more general indexing service (G).

The previous search strings are transformed into the query strings depicted in Table 4. Note that there are various search IDs (cf., SearchIDs) related to every search string, because each search has been executed at least three times: the first time during Phase 1, the second time during Phase 2,

[3]Note that the ACM DL search engine changed the syntax for the queries during the update phase.
[4]S0 returns the same results as in S3, and hence this query is not included in the second phase.
[5]The IEEE Xplore search engine changed during the update in the second round.
[6]The searches with the terms "programming model" and "chaincode" do not produce results.

Table 5. Number of Studies Obtained in the Different Phases

	Initial Search	Second Search	Updated Searches	Total
ACM	18	24	216	258
Springer	557	225	726	1,508
IEEE	148	25	45	218
Science Direct	73	135	239	447
Scopus	55	105	145	305
Google Scholar	962	69	48	1,079
Number of studies with automatic searches	1,813	583	1,419	3,815
Number of studies with snowballing		192	108	300
Number of studies incorporated manually		4	0	4
Number of studies				**4,119**

Table 6. Exclusion and Inclusion Criteria

Criteria	Description
Exclusion	The study is not written in English.
	The study is not related to a smart contract language.
	The study is a tutorial or poster summary only.
	The study occurs multiple times in the result set.
	The study is published before 1991.
	The study is not accessible electronically.
	The full text of the study is not available for downloading.
	It is a patent.
Inclusion	Title, keyword list, and abstract make explicit that the paper is related to a smart contract language.

and the third during the update in the second round. Note that some of the studies obtained as a result of the second phase could be obtained as a result of the first search. As a consequence, in Phase 2.2, we have executed a data-cleaning process to prevent the storage of duplicated studies in StArt.

As a result of the search process, 4,119 studies were selected. Table 5 summarises the number of studies obtained as a result of the queries executed in the different search engines during the initial search, the second search, and the second round, as well as the number of studies resulting from the snowballing process and the manual process. Finally, it deserves to be mentioned that the searches in Google Scholar have been executed using *Publish or Perish*,[7] which returns a maximum of 1,000 results.

4.3 Study Selection

The initial screening of the 4,119 papers found is mainly undertaken by Reina-Quintero, although Varela-Vaca also contributes, especially with the screening of the studies obtained in Phase 1 and the snowballing. The screening is based on title, abstract, and keywords. Note that this screening involves the scientific literature and, as a result, the studies that are irrelevant or duplicates are excluded. The inclusion and exclusion criteria used in the review are listed in Table 6. It should be borne in mind that we have used some of the standard exclusion criteria proposed in Reference

[7] https://harzing.com/resources/publish-or-perish.

[27]. As a result of this screening, 284 studies have been selected. The thorough review of the studies is made mainly by Varela-Vaca. As a result, 109 studies are selected for extraction.

Once a study has been selected for extraction, a backward snowballing process of the reference list included in the paper can be made. The process consists of selecting the references found in the Related Work, Introduction and/or Conclusion sections. To maintain the traceability of the process, we have used a shared spreadsheet to annotate the ID of the study, the number of extracted references (as numbered in the original paper), the section or sections in which the reference appeared, and the references finally selected. It should be borne in mind that the native tool that provides the StArt tool does not work properly with references obtained through snowballing, and hence a procedure is set up to add the studies obtained through this process. The procedure consists of including the selected references in a BibTex or a RIS file to load those files in the StArt tool as a new search.

4.4 Quality Assessment

In our Multivocal Literature Review (MLR), each publication is assessed for quality at the same time as the publication data extraction process is performed. The quality assessment allows us to rate and validate the study according to the extracted data. The result of the quality assessment can be used as a criterion to finally accept or reject a study.

To assess the quality of each study the Information Quality framework defined in Reference [42] is used. This framework defines four dimensions of Information Quality (IQ): Intrinsic, Contextual, Representational, and Accessibility. This IQ can be measured regarding one dimension or several depending on the attributes extracted from the data. In our case, Representational and Accessibility dimensions are skipped, since the format of the data and the access to the information issues are not relevant in our context. Thus, to measure IQ, the focus is on the following two dimensions:

- **Intrinsic IQ:** This dimension measures the accuracy, objectivity, believability, and reputation of data for the task at hand.
- **Contextual IQ:** This dimension measures whether contextual parameters of data are appropriate for the task at hand.

The assessment and measurements used for the aforementioned IQ dimensions are defined as follows:

- **Intrinsic IQ:** We measure the believability and reputation based on the type of publication, the forum where it is published, and the ranking of the forum. Thus, a study is classified in one of the categories J1, J2, J3, C1, C2, C3, O1, O2, O3, GL1, GL2, GL3. The category depends on its type and ranking, and it is obtained as described in (1). Note that to categorise the grey literature,the taxonomy proposed in Reference [2] is employed.

$$
study = \begin{cases}
\textit{White Literature} \begin{cases}
\textit{Journal} \begin{cases}
J1: \textit{Journal indexed as Q1 or Q2 in the ISI WoK and/or Scimago Journal rankings.} \\
J2: \textit{Journal indexed as Q3 or Q4 in the ISI WoK and/or Scimago Journal rankings.} \\
J3: \textit{Journal not indexed in the ISI WoK nor Scimago Journal rankings.}
\end{cases} \\
\textit{Conference/Workshop} \begin{cases}
C1: \textit{Conference indexed as Class 1 or Class 2 in the SCIE Conference ranking.} \\
C2: \textit{Conference indexed as Class 3 in the SCIE Conference ranking.} \\
C3: \textit{Conference not indexed in the SCIE Conference ranking.}
\end{cases} \\
\textit{Others} \begin{cases}
O1: \textit{Book chapters.} \\
O2: \textit{Encyclopedia, arXiv papers.} \\
O3: \textit{Technical report, and others.}
\end{cases}
\end{cases} \\
\textit{Grey Literature} \begin{cases}
GL1: \textit{Books, magazines, government reports, white papers.} \\
GL2: \textit{Annual reports, news articles, presentations, videos, websites, Q/A sites (such as StackOverflow), wiki articles.} \\
GL3: \textit{Blogs, emails, tweets.}
\end{cases}
\end{cases} \tag{1}
$$

Table 7. Assessment of Intrinsic IQ

Type of Pub.	Category	Intrinsic IQ Output
White literature	J1 ∨ C1	HIGH
White literature	J2 ∨ C2	MEDIUM
Grey literature	GL1 ∨ GL2	MEDIUM
White literature	J3 ∨ C3 ∨ O1 ∨ O2 ∨ O3	MEDIUM
Grey literature	GL3	LOW

According to the categories outlined above, Intrinsic IQ is obtained based on a three-point Likert scale with HIGH, MEDIUM, and LOW values, in which HIGH is the best result and LOW is the worst result. The Intrinsic IQ assessment is carried out according to the criterion specified in Table 7.

- **Contextual IQ:** The amount of data and completeness of the extracted data is measuered for each study. We have defined the questionnaire below to extract information regarding the studies:
  - **QA1** Could the name of the language be extracted?
  - **QA2** Could the language paradigm be extracted?
  - **QA3** Could we extract whether the language is supported by one of the current blockchain platforms?
  - **QA4** Could we extract the DSL type?
  - **QA5** Could we extract the focus?
  - **QA6** Could we extract the institution name?
  - **QA7** Could we extract the institution origin?
  - **QA8** Could we extract the challenges?
  - **QA9** Could we extract the use cases?
  - **QA10** Could we extract the kind of language?

  The metric employed to measure the amount of data and completeness of the extracted data is defined on a scale of [0-1] as follows:

$$m_{Completeness} = \frac{|Number\ of\ answered\ questions|}{|Total\ number\ of\ questions|} \tag{2}$$

  The Contextual IQ is defined based on a three-point Likert scale with HIGH, MEDIUM, and LOW values, in which HIGH is the best result and LOW is the worst result. The Contextual IQ assessment is carried out according to the criterion specified in Table 8.

The quality of a study is determined based on the level of quality obtained in the two dimensions, that is, intrinsic and contextual. We consider a study acceptable when the HIGH or MEDIUM quality level is obtained in both dimensions. The study is rejected when either quality dimension has a LOW value.

Table 8. Assessment of Contextual IQ

Metric	Values	Contextual IQ Output
$m_{Completeness}$	>0.5	HIGH
$m_{Completeness}$	(0.2, 0.5]	MEDIUM
$m_{Completeness}$	0.2≤	LOW

Table 9. Alignment of Extracted Data and Research Questions

Form	Extracted data	Research question
Publication form	Publication title	RQ1, RQ2, RQ3
	Publication authors	RQ1, RQ2, RQ3
	First author's institution	RQ2
	Publication Venue	RQ4
	Publication Year	RQ1
	Citations of the publication	RQ3
Extraction form	Name	RQ5
	Paradigm	RQ7
	Supported by blockchain platform	RQ7.1
	Blockchain name	RQ7.1
	DSL Type	RQ7.2
	Focus	RQ7.3
	Institution name	RQ6
	Institution origin	RQ6
	Challenges	RQ7.4
	Use cases	RQ7.5
	Kind	RQ5

4.5 Data Extraction Process

The data extraction process is divided into two stages: (1) the data extraction itself, and; (2) the revision of the extracted data. Varela-Vaca is responsible for the data extraction, while Reina-Quintero is responsible for the revision. The extraction is performed by filling in a form created in the StArt tool. The form is composed of the following fields:

- *Name*: String representing the name of the language.
- *Paradigm*: Imperative, declarative, symbolic, graphic, undetermined.
- *Blockchain-dependent implementation*: yes, no.
- *Blockchain name*: The name of the blockchain that supports the language.
- *DSL Type*: Stand-alone, extension.
- *Focus*: The main focus of the research in relation to the smart contract language.
- *Institution name*: The name of the institution that creates/supports the language.
- *Institution origin*: Academia, industry.
- *Challenges*: The challenges the approach should face.
- *Use cases*: Reported use cases of the language.
- *Kind*: Implementation, specification, unknown.

Furthermore, the extracted data helps us to answer the research questions addressed by the study. Table 9 shows how the extracted data and the research questions are aligned. While questions RQ1–RQ4 are answered by using publication data, questions RQ5–RQ7 are answered by using data from the extraction form. Note that, since StArt provides a general form for each publication, data, such as the title of the publication or the authors, does not have to be added to the extraction form.

4.6 Replication Package

To strengthen the replicability of our review, we have published a bundle with all the artefacts and the final results of our study in http://www.idea.us.es/smart-contract-languages/. The selected

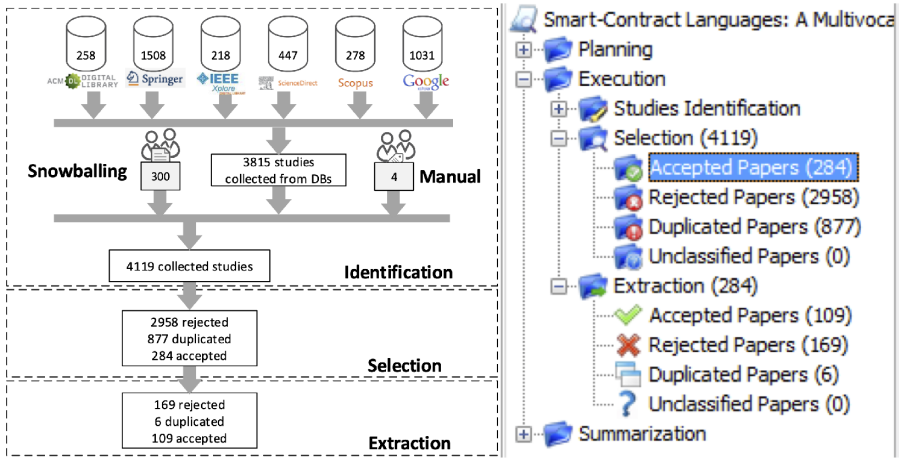


Fig. 2. Summary of the results of the different phases in StArt.

studies can be accessed online by following the bibliographic information reported in Appendix A[8] and in the reference list.

5 DATA EXTRACTION RESULTS

The literature review process was conducted from July 2018 to June 2020. The process lasted one year, in which we developed the protocol, identified and selected the primary studies from the scientific and grey literature, performed the data extraction and the synthesis processes, and reported the results of our study. Since the process took so long, data sources were queried at two different moments in time, at the beginning of the process and the end. The two researchers were involved in the whole process, as detailed in the previous section. Figure 2 shows a diagram and a StArt screenshot that summarise the results obtained during the three stages of the review execution: identification, selection, and extraction.

The primary studies retrieved from both the white literature and the grey literature are listed in the references section and in Appendix A.[9] Once the primary studies were recovered, we used a word cloud created using keyword lists to check whether we had selected some papers from outside the relevant scope.

In the following subsections, we present the data synthesis and the results of the mapping study. The data synthesis has been achieved by using the methods proposed in Reference [26] for mapping study analysis and qualitative synthesis.

5.1 Primary Studies

In accordance with Kitchenham et al. [26], the analysis methods used to summarise results from mapping studies are straightforward. Our set of primary studies is composed of 109 primary studies, 70 of them are obtained from the scientific literature and 39 of which from the grey literature. Table 1 in Appendix A[10] lists the studies selected from the scientific literature and shows the ID, the reference, the authors, and the venue. Furthermore, Table 2 (also in Appendix A) lists the studies selected from the grey literature, and shows the ID, the reference, the title, the authors, the type

[8] see http://www.idea.us.es/smart-contract-languages/.
[9] see http://www.idea.us.es/smart-contract-languages/.
[10] cf. http://www.idea.us.es/smart-contract-languages/.

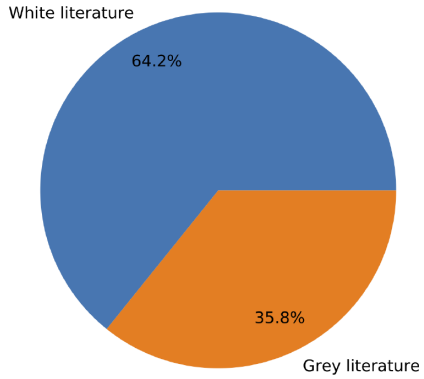


Fig. 3. Grey literature vs. white literature.

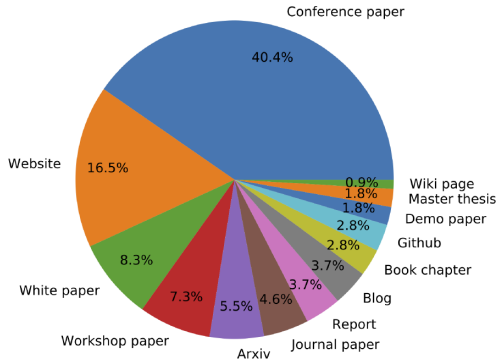


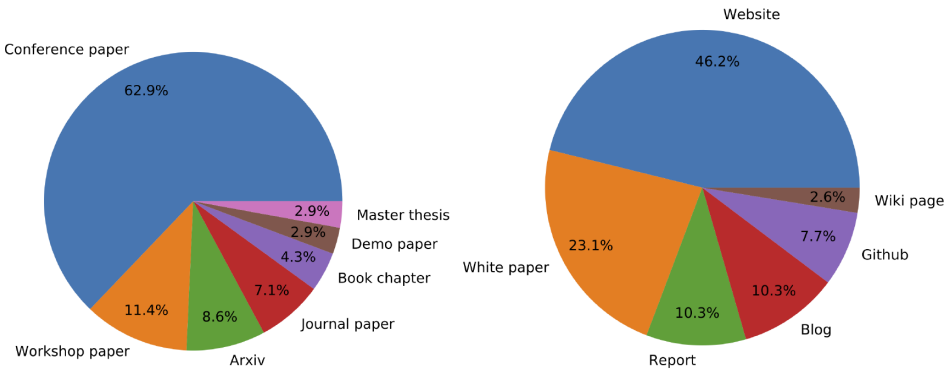
Fig. 4. Percentage of publications per type.

of contribution, and the year. Note that the ID of scientific literature studies starts with S, while the id of grey literature studies starts with G. Figure 3 shows the percentage of the primary studies retrieved from the scientific (white) and grey literature.

Regarding the type of publications, Figure 4 shows the percentage of studies per type of publication. As can be observed, 40.4% of the studies have been published in conferences, and only 4.6% have been published in journals. This demonstrates that this is a young area of research. If the types of publication are separated according to the type of literature (white or grey), then it can be observed that 62.9% of papers (44 out of 70) have been published in conferences, and only 5 out of 70 have been published in journals (cf., Figure 5(a)). In relation to the grey literature, the most popular type of publication is that of websites with 46.2% of the studies (18 out of 39) (cf., Figure 5(b)).

5.2 Study Quality Assessment

Each primary study has been assessed for quality by calculating the Intrinsic and Contextual IQ measurements (see Section 4.4). A summary of the quality measurements are introduced in Figure 6 by means of a bubble chart. The figure shows the two dimensions (one in each axis) and the values each dimension can have (i.e., HIGH, MEDIUM, and LOW). A bubble represents the number of studies with the two values of the dimensions. For instance, 84 primary studies obtained a Medium Intrinsic IQ and a High Contextual IQ. There are 19 out of 109 primary studies with High Intrinsic and Contextual IQ. Finally, six studies have Low Intrinsic IQ and High Contextual IQ. Considering



(a) White literature. (b) Grey literature.

Fig. 5. Percentage of publications per type grouped by type of literature (white or grey).

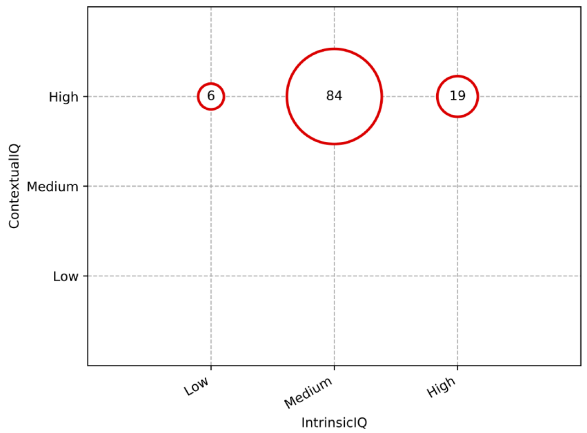


Fig. 6. Quality assessment.

the quality conditions established in Section 4.4, we can state that all the primary studies have an acceptable level of quality for their consideration in the current work.

6 DISCUSSION OF RESEARCH QUESTIONS

This section addresses our research questions and strives to answer each of them.

6.1 RQ1. Which Contributions were Made over the Years?

The goal of this question is to provide information regarding publication quantity and frequency, in both the white (scientific) and grey literature, because the identification of the existing primary studies over time could help towards analysing trends, such as the maturity of the domain. Thus, Figure 7 shows the number of studies of the white literature and the number of studies of the grey literature per year. It should be borne in mind that our review only covers six months of 2020, and as can be observed, the majority of primary studies retrieved are published between 2017 and 2019. Furthermore, the number of studies published in 2018 is double the number of studies published in 2017. Indeed, the number of studies of the grey literature is greater than the number in the white literature during 2017, that is, the industry contribution is more relevant than the academic contribution that year, but the academic contribution gains in relevance in the subsequent years,

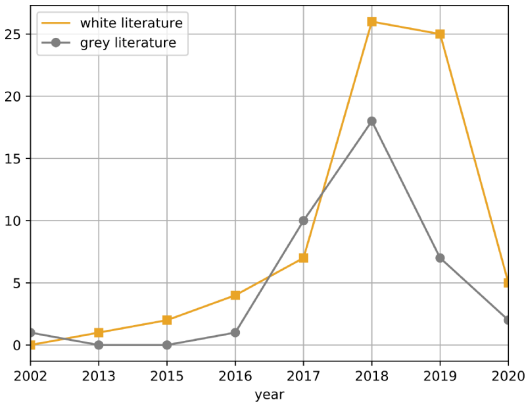


Fig. 7. Number of studies of white and grey literature per year.

Table 10. Authors with More Than One Study

Author	No. of studies	Institution/Corporation
Massimo Bartoletti	4	University of Cagliari
Dwight Guth	3	Runtime Verification, Inc.
Roberto Zunino	3	University of Trento
Accord Project	2	The Linux Foundation Projects
Anastasia Mavridou	2	NASA Ames Research Center
Andrew Miller	2	University of Illinois at Urbana-Champaign
Aron Laszka	2	University of Houston
Ethereum	2	Ethereum foundation
Florian Daniel	2	Politecnico di Milano
Hans Weigand	2	Tilburg University
Ilya Sergey	2	Yale-NUS College and NUS School of Computing
Jack Pettersson	2	Rchain.coop
Joost de Kruijff	2	Tilburg University
Naoto Sato	2	IBM Research Japan Ltd.
Ralph Johnson	2	Runtime Verification, Inc.
Richard Hull	2	IBM Research Japan Ltd.
Russell O'Connor	2	BlockStream
Takaaki Tateishi	2	IBM Research Japan Ltd.
Theodoros Kasampalis	2	Runtime Verification, Inc. and University of Illinois at Urbana-Champaign
Traian F. Serbanuta	2	Runtime Verification, Inc.
Virgil Serbanuta	2	Runtime Verification, Inc.
Yan Zhu	2	Beijing University of Science and Technology

and this growing trend is maintained during 2019–2020. This implies that smart contract language was an emerging research topic in 2016–2019 and has since become a mature research topic in which not only academia but also industry is interested.

6.2 RQ2. Which are the Top Researchers in the Area of Smart Contract Languages and Where are they from?

This is a demographic research question whose goal is to determine the characteristics of the studies in the area, in this case, concerning the authors. Table 10 lists the authors/organisations that have contributions in more than one study. We have found 326 different authors of the 109 selected primary studies, which makes an average of 3.3 authors per study. Only 6.74% of the

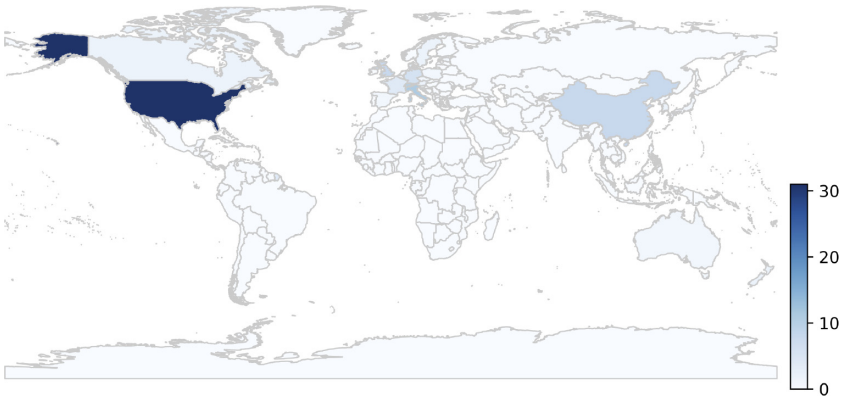


Fig. 8. Publications per country.

authors (22 out of 326) have participated in more than one primary study, which shows, again, the youthfulness of the area. Table 10 lists the authors with more than one study together with the institution or the corporation name to which the authors belong.

Furthermore, Figure 8 depicts a map with the country of origin of the studies. It should be borne in mind that we considered that the country of origin of a study is determined by the country of origin of the institution of the first author (if there is more than one author). The top 3 countries are the USA, Italy, and Switzerland, whereby the USA, with 31 studies, is far above Italy and Switzerland with 11 and 10 studies, respectively.

6.3 RQ3. Which are the Most Influential Studies in the Area?

This research question is also a demographic question whose goal is to identify the most influential primary studies in the area. This question is especially interesting for PhD students or beginners, because it gives an idea of the most influential studies. Thus, Table 11 lists the top 10 primary studies in the scientific literature ordered in terms of the number of citations according to Google Scholar. The Scopus and Web of Science citations are included in the table to enable more exhaustive and analytic work to be carried out. As a complement to traditional metrics, we have included the altmetrics [33].[11] The altmetrics are calculated for each study as the aggregation of the metrics of usage (e.g., number of abstracts views, clicks, downloads), captures (e.g., when end users add a bookmark, a favourite, become a reader, become a watcher), mentions (e.g., blog posts, comments, reviews, and Wikipedia links), and social media (e.g., number of likes, shares, ratings).

It can be observed that S06, the paper about Hawk, is the most cited with 1,250 citations. The second and third positions are occupied by S19 and S05, respectively. Regarding the altmetrics, the most relevant paper is still S06, followed by S05 and S03. For the reader interested in the classification of all the papers, Appendix B[12] contains a table that lists all the selected scientific studies in order of the number of citations.

6.4 RQ4. Which are the Top Venues?

The goal of this question is to help researchers in this area to choose where to publish by ascertaining where other researchers have published scientific literature previously. We have found 45 different venues, including journals, conferences, workshops, and others. Only 11.1% of the venues

[11] The PlumX tool was used to gather the altmetrics.
[12] http://www.idea.us.es/smart-contract-languages/.

Table 11. Top-10 of Papers Ordered by the Citations in White Literature

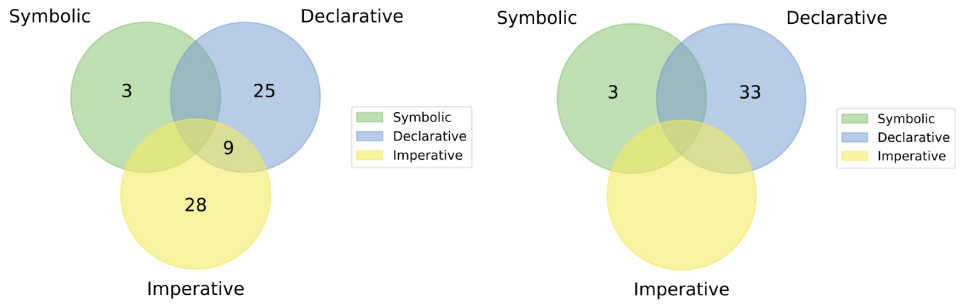
Rank	ID	Title	Google Scholar	Scopus	Web of Science	Altimetrics
1	S06	Hawk: The blockchain model of cryptography and privacy-preserving smart contract	1,250	601	314	3,311
2	S19	Ekiden: A platform for confientiality-preserving, trustworthy, and performant smart contract execution	111	-	-	-
3	S05	From institutions to code: Towards automated generation of smart contracts	94	50	23	601
4	S18	Scilla: a smart contract intermediate-level language	80	-	-	-
5	S03	Towards a shared ledger business collaboration language based on data-aware processes	73	31	13	357
6	S08	Designing secure Ethereum smart contracts: A finite state machine-based approach	73	4	0	-
7	S11	Automated execution of financial contracts on blockchains	58	28	20	321
8	S10	Simplicity: A new language for blockchains	58	19	-	192
9	S02	eContractual choreography-language properties towards cross-organizational business collaboration	42	36	23	131
10	S07	Obsidian: A safer blockchain programming language	40	17	11	255

Table 12. Top-11 Venues Ordered by the Number of Studies Published

Rank	Venue	Type	No. of papers
1	CoRR	arxiv	6
2	International Symposium on Leveraging Applications of Formal Methods (ISoLA)	conference	3
3	International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)	conference	3
4	International Conference on Financial Cryptography and Data Security (FC)	conference	3
5	International Workshop on Value Modeling and Business Ontologies (VMBO)	workshop	2
6	International Conference on Service-Oriented Computing (ICSOC)	conference	2
7	International Conference on New Technologies, Mobility and Security (NTMS)	conference	2
8	International Conference on Decentralized Applications and Infrastructures (DAPPCON)	conference	2
9	International Conference on Blockchain and Cryptocurrency (ICBC)	conference	2
10	European Symposium on Programming (ESOP)	conference	2
11	Conference on Computer and Communications Security (CCS)	conference	2

(5 out of 45) are journals; almost three quarters of the venues (32 out of 45) are conferences; 7 out of 45 are workshops. Table 12 lists the top 11 venues ordered in terms of the number of studies published in that venue. Note that the winner is the Computing Research Repository (CoRR) at arXiv, which is gaining popularity among researchers due to its rapid dissemination of results. For the reader interested in the classification of all the venues, Appendix B[13] can be consulted.

[13] http://www.idea.us.es/smart-contract-languages/.



(a) Implementation (b) Specification

Fig. 9. Number of smart contract languages grouped by paradigm.

6.5 RQ5. What Languages are There in the Literature for Specifying/Implementing Smart Contracts?

This question aims to identify the existing smart contract languages within the scope of this review. To answer this question, we have collected the name of the language in the extraction form. Since several approaches fail to provide a name for the language, we refer to those languages with the paper *ID*; for example, the language proposed in Reference [S11] has no name in the original approach, and therefore we refer to it as S11, the ID of the study. We have found 101 different smart contract languages. Figure 9 shows a summary of the languages grouped by paradigm. For the sake of clarity, the figure shows the languages grouped by their purpose, implementation (Figure 9(a)) or specification (Figure 9(b)). In every group, the languages have been classified into the following programming paradigms (note that there are languages that can have imperative and declarative features):

- **Imperative language**. This is a language based on an imperative programming paradigm, that is, a language whose statements change the state program [35].
- **Declarative language**. This is a language based on a declarative programming paradigm, that is, whose statements are stateless [35].
- **Symbolic language**. This is a language based on a programming paradigm in which the program can treat itself as data [10].

As the purpose of the language may influence the paradigm, Figure 9 separates the figures of implementation languages and specification languages (Figures 9(a) and 9(b), respectively). As can be observed, there are nine implementation languages that follow a mixed (declarative-imperative) paradigm. Furthermore, there is no imperative language with specification purposes.

6.6 RQ6. Are the Languages Supported by Industry or Academia?

The goal of this question is to ascertain whether a language is supported by industry or by academia. This is interesting, because industry usually better supports languages than does academia and provides more stable languages. We consider that a language is supported by industry if it is introduced by an organisation (e.g., BlockStream and Accord Project) or if any of the authors belongs to a company (i.e., IBM or Runtime Verification Inc.). Figure 10 shows the distribution of languages supported by industry and academia grouped by the type of language. Nearly 40% of languages are supported by industry (41 out of 101), while the rest are supported by academia. It is interesting to remark that most of the languages supported by industry (35 out of

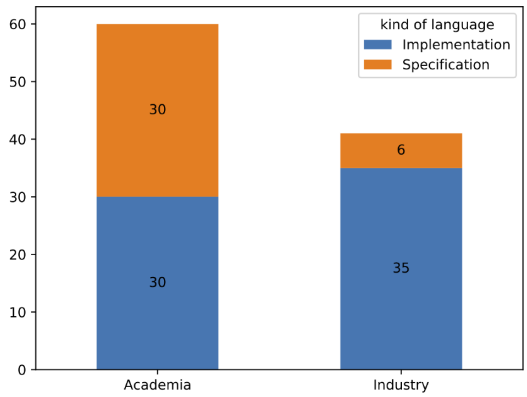


Fig. 10. Academia vs. industry.

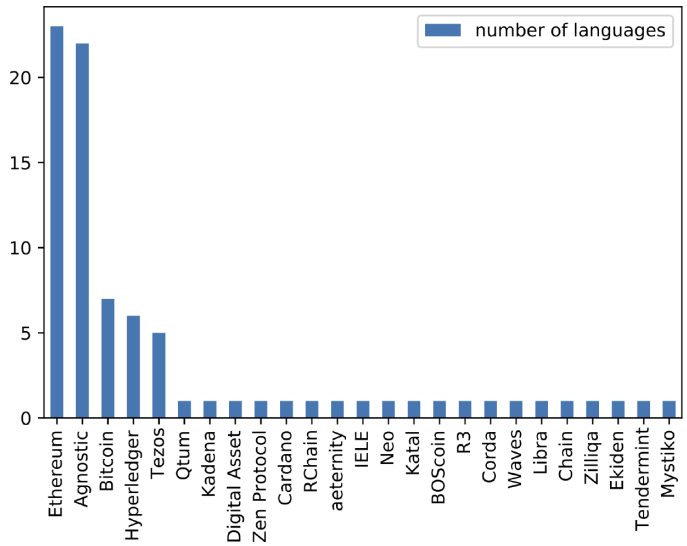


Fig. 11. Number of languages supported by each blockchain platform.

41) are implementation languages, while half of the languages supported by academia are specification languages.

6.7 RQ7. Which Features do These Languages Have?

The goal of this question is to characterise the languages according to certain features that could be interesting for researchers and practitioners. To this end, RQ7 is divided into five sub-questions.

6.7.1 RQ7.1. Is the Language Supported by One of the Current Blockchain Platforms? Which Platform/s Supports the Language? Sometimes smart contract languages are designed to be executed on a concrete blockchain platform, while in other cases, languages are prototypes that are currently unsupported by any blockchain platform, or they could even be designed in terms of an agnostic blockchain platform. The goal of this question is twofold: (1) to identify the ecosystem of blockchain platforms that support the identified languages, and; (2) given a smart contract

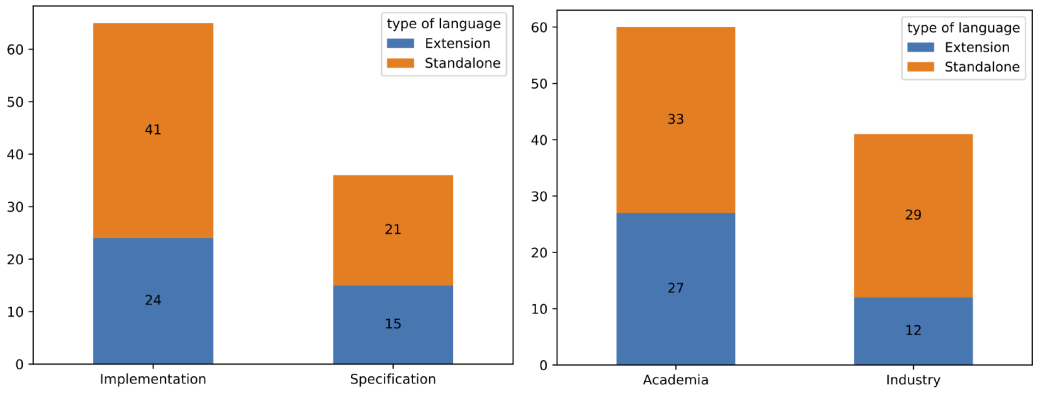
Table 13. Languages per Blockchain Platform

Blockchain platform	Languages
Ethereum	ADICO-Solidity, Babbage, Bamboo, Blockly, Bounty Contract, Flint, G03, Hawk, Idris, Proof-Carrying Smart Contracts (PCSC), Pyramid, Rule-based Representation (RBR), S11, S22, S47, S64, S65, S66, Solidity, Vyper, Yul, Zether smart contract (ZSC), scl
Agnostic	Abstract Smart Contracts(ASC), Cicero, Codius, Contract Modeling Language (CML), Das Contract, Ergo, FSolidM, Findel, Logic-SC, Marlowe, Nomos, OpenLaw, Plurality, Porthos, S60, SmaCoNat, Smart Contract Description Language (SCDL), UML Profile for Smart Contracts, Unibright Contract Interface, ZoKrates, eDSL, zkay
n/a	CommitRuleML, G04, Reaction RuleML, S36, SPECS, Takamaka, eSourcing Markup Language (eSML)
Bitcoin	Balzac, BitML, Ivy, Script, Simplicity, Typecoin, koa
Hyperledger	Business Collaboration Rules Language (BCRL), Contract Specification Language (CSL), DSL, DSL4ASC, Obsidian, S42
Tezos	Archetype, LIGO, Liquidity, Michelson, fi
Qtum	Qtum Smart-Contract Language (QSCL) ontology-based language
Kadena	Pact
Digital Asset	DAML
Zen Protocol	ZF*
Cardano	Plutus
RChain	Rholang
aeternity	Sophia
IELE	IELE
Neo	NEO smart contracts
Katal	ACTUS
BOScoin	Trust Contract
Corda	Contract Specification Language (CSL)
R3	Contract Specification Language (CSL)
Waves	RIDE
Libra	Move
Chain	Chain Core
Zilliqa	Smart contract intermediate-level language (Scilla)
Ekiden	Ekiden contracts
Tendermint	Tenderfone
Mystiko	Aplos

language, to identify the specific platforms that support it. Thus, Figure 11 shows the number of languages supported by each blockchain platform.

We have found 24 different blockchain platforms. The most commonly used platform is that of Ethereum, which supports 23 out of 101 languages. There are 22 languages that are platform-agnostic, that is, they could be executed on any blockchain platform. Far from Ethereum, in the third and fourth positions of the ranking, are Bitcoin and Hyperledger, with 7 and 6 smart contract languages, respectively.

Table 13 lists the blockchain platforms and the smart contract languages. The third row of the table shows a set of languages that are theoretical and are yet to be implemented or that are specification languages that cannot be deployed on a blockchain platform. Finally, there are 13 languages without a blockchain platform, because the information could not be extracted from



(a) Implementation vs. specification (b) Academia vs. industry

Fig. 12. Stand-alone smart contract languages vs. extension languages.

the papers: BIP Framework, Beagle, Business Collaboration Language (BCL), Dr. SES, Mandala, Rainfall, S16, S53, SMPC, TRiC Descriptors, TinySol, dSLAC, and lambda-smart. For the sake of clarity, these languages have not been represented in Figure 11 nor in Table 13.

6.7.2 RQ7.2. Are the Languages Stand-alone or are They Defined as an Extension of Another Existing Language? The aim of this question is to ascertain whether the language has been developed from scratch (as a stand-alone) or it is an extension of another language. Figure 12 shows the number of stand-alone languages versus the percentage of languages that extend other languages. Stand-alone languages constitute 61.38% of the languages (62 out of 101), and the rest are defined as extensions. For instance, the ZF* smart contract language (cf., G11 and G22) is considered an extension, because it is built on the basis of the F* language. Additionally, Vyper (cf., G18), is considered to be stand-alone, because it defines a domain-specific language to implement Ethereum secure smart contracts.

Furthermore, Figure 12(a) represents the number of stand-alone languages that are classified as implementation versus those that are classified as specification. Thus, 65% of stand-alone languages are implementation languages (41 out of 63) and 61.5% of extension languages are implementation languages. While Figure 12(b) represents the number of stand-alone and extension languages that are supported by academia versus those that are supported by industry. While 52.38% of stand-alone languages are supported by academia (33 out of 63) and 69.23% of extension languages are supported by academia.

6.7.3 RQ7.3. Which is the Main Focus of the Research in Relation to the Smart Contract Language? The goal of this question is to describe the main focus of the research concerning the smart contract language. The focus categories have been obtained by clustering the keywords and abstracting them into different categories for the white literature; for the grey literature and those studies without keywords or abstract, we exhaustively read about the language in other accepted papers and in related work. Figure 13 shows a ranking of the categories that include more than one language.

The top 4 categories are related to verification, financial, security, and legal issues. The languages included in the verification category focus on the verification of smart contracts at different levels of abstraction. The verification is focused on analysing or providing a mechanism to verify properties such as completeness, correctness, and soundness. The languages in this category are: BIP Framework, Balzac, FSolidM, Findel, G04, IELE, Idris, LIGO, Liquidity, Obsidian, Proof-Carrying

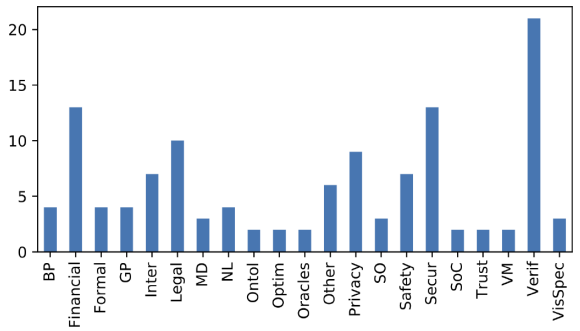


Fig. 13. Number of languages by focus. BP = Business Process; GP = General Purpose; Inter = Interactions; MD = Model-Driven; NL = Natural Language; Ontol = Ontology; Optim = Optimization; SO = Service-oriented; Secur = Security; SoC = Separation of Concerns; VM = Virtual Machine; Verif = Verification; VisSpec = Visual Specification.

Smart Contracts (PCSC), Rule-based Representation (RBR), S16, S22, Simplicity, Smart contract intermediate-level language (Scilla), ZF*, ZoKrates, dSLAC, lambda-smart, and scl.

Financial contracts, in accordance with Reference [G27], are "legal agreements between two (or more) counter-parties on the exchange of future cash flows." The languages in this category are focused on providing languages that are both easy to comprehend by non-technical users and can be managed on distributed ledgers. This category includes ACTUS, DAML, Findel, G04, Marlowe, Move, NEO smart contracts, Plutus, Porthos, PCSC, S11, Trust Contract, and ZF*. Note that this category includes Neo smart contracts, which also belongs to the legal contracts category. This overlap is due to the proximity of these two categories and there are approaches that cover both domains, legal and finance.

The research on languages in the security category [6] is focused on improving the security of the languages by providing mechanisms, for instance, to prevent attacks. This category includes the following languages: Archetype, Bamboo, Dr. SES, Ekiden, FSolidM, Flint, Obsidian, RIDE, S32, S60, Vyper, Zether smart contract (ZSC), and koa.

The languages included in the legal contract category are focused on the modelling of legal agreements to make them accessible to non-computer experts, and to bridge the gap between developers and lawyers. This category includes the languages ADICO-Solidity, Babbage, Beagle, Blockly, CommitRuleML, Ergo, NEO smart contracts, Nomos, OpenLaw, and Reaction RuleML.

The rest of the categories, ordered in terms of the number of languages, include[14]:

- Interactions cover a set of approaches focused on the calling among smart contracts and concurrent access to other resources from different points of view ranging from smart contract negotiation to the collaborative development of smart contracts.
- Business process, whose focus is the use of smart contracts in different aspects of business processes, such as defining business logic, the business process collaboration or integration.
- Formalisation, whose focus is to define a formalisation for specifying semantic aspects or reasoning about the smart contracts or to extend the characteristics of the language such as including temporal properties.
- General purpose, whose focus is to describe the operations and functions of the language to be applied in a broad spectrum of cases.

[14]The list of languages enclosed in those categories can be consulted in the Appendix see http://www.idea.us.es/smart-contract-languages/.

- Model-driven, which is focused on defining smart contracts as high-level models and generating the code needed for specific platforms.
- Natural language, which includes approaches that pursue the specification of contracts in human-readable or natural language.
- Service-oriented, with languages that are focused on integrating smart contracts in service-oriented contexts.
- Visual specification, with languages that strive to define smart contracts visually.
- Ontology, which includes approaches for formalising smart contracts with the tools of the semantic technological space.
- Optimisation, which include languages whose focus is the optimisation of a certain aspect of the blockchain, such as optimising the code generation.
- Oracles include languages whose focus is the introduction of oracles into smart contracts.
- Separation of concerns, whose focus is the improvement of the modularity of smart contracts by applying the principle of separation of concerns, such as using aspect-oriented approaches.
- Virtual machine, whose focus is to define, generate, analyse, or improve aspects related to runtime execution, virtual machine, and bytecode.

6.7.4 RQ7.4. Which are the Challenges the Approach Should Face? The aim of this question is to put down in black and white the main challenges of the field. As there are 101 languages, and for the sake of clarity, we have grouped the challenges in terms of the focus previously identified for the languages, since we assume that languages that share the same focus should have similar challenges. As a consequence, the challenges are organised and listed in terms of the focus. The challenges have been obtained by reading the papers carefully and grouping them by the focus.

Verification. The challenges in this category encompass the improvement of the modelling and analysis of properties for verification [S4, S20], for example, by including properties such as: the number of miners or gas expenses; the generation of scenarios automatically based on formal models to detect problems; the co-evolution of the EVM specification and the EVM itself; the formalisation of anti-patterns; and the improvement of development tools to prevent developers' mistakes.

Financial contracts. The challenges that financial contracts should face are twofold: technical [S11, S12] and legal. From the technical point of view, the challenges are related to: improving the implementation of the languages and the architecture of frameworks; evaluating architectures in which certain components are off-ledger; evaluating different kinds of contract managers; dealing with multi-party contracts; dealing with trusted off-chain information (oracle problem); and adjusting contracts to real situations by splitting the contract into one automatic part and one non-automatic part that mediates. From the legal point of view, the challenges are related to carrying out an extensive analysis of the legal implications of smart contracts, and to creating a legal framework that enables legal contracts and smart contracts to merge.

Security, Privacy, and Safety. Since security, privacy, and safety are closely related, in this section the challenges of the languages whose focus is *Security*, *Safety*, and *Privacy* are included. The main open problems that have to be faced by the languages in this category are:

- The analysis of vulnerabilities, such as replay attacks, re-entrancy problems, infinity loops, and code injection [S39, G18].
- In relation to vulnerabilities, the management of the various sets of the sources of vulnerabilities, the extraction of the high number of features involved in vulnerabilities, and the inference of the relation between the features while analysing the vulnerabilities.

- Security testing according to the inferred knowledge.
- The improvement of data privacy and security on execution [S32], by preserving, for instance, privacy on transactions or assets.
- The improvement of the expressiveness of some of the languages [S17], that is, some of them fail to include certain operations to avoid vulnerabilities, and they could be improved by adding these operations.
- The lack of authorisation mechanisms to control the smart contract execution [S25].
- The lack of audit mechanisms to enable the code analysis of smart contracts to discover bugs, violations, and other issues [S35].
- The scalability and privacy when moving computations from the blockchain to external nodes [S34].
- The verification of security and safety properties during the creation of the smart contract [S23].
- The implementation of the language and the definition of formal grammar and semantics (in approaches that are still immature [S18]).
- The inclusion of information from external data sources [S38] (which is known in the blockchain arena as the oracle problem).
- The conduction of user studies to evaluate certain aspects of the language [S7].

Legal agreements. The challenges of this category are also twofold: technical and legal. The technical challenges [S31] comprise: the inclusion of a mechanism to reverse transactions; the inclusion of the right of withdrawal; the inclusion of a dunning process; the oracle problem; a mechanism to incorporate external legal documents; the verification of events that are produced off-chain; the improvement of the implementation to incorporate more types of contracts and the creation of tools; and, finally, endowing lawyers to build smart contracts in an enforceable and understandable way. The legal challenges [S44] are related to dealing with the volatility of cryptocurrencies and the international acceptation of smart contracts from a legal point of view.

Interactions. The challenges of the languages in this category range from common challenges that have appeared in the previous categories, such as the scalability, the incorporation of case studies of real industrial contexts and the improvement of the expressiveness of the language to challenges [S24], that are more specific to this category, such as the definition of approaches capable of specifying multi-chain smart contracts in a single specification [S63], the smart contract interactions, and the autonomous configuration of negotiation parameters that are currently defined manually.

Business process. The challenges in this category include: the integration of blockchain into current IT ecosystems and business integration approaches [S2]; the reconciliation of the data-aware style of coordination, enabled by blockchain, and the process-centric style of business process implementations, when striving to adapt legacy business processes to blockchain [S3]; the determination of the kind of tools to develop business collaboration in the blockchain; and, finally, the lack of tools for the discovery of smart contracts in a context in which there are libraries of smart contracts based on business artifacts [S3].

Formalisation. The challenges of the languages in this category are common to some of the previous categories, such as dealing with multiple smarts contract interacting with each other [S33] or the avoidance of code repetition [S37].

Natural language. The challenges of the languages in this category include dealing with the ambiguity of terms [S53]; increasing the level of abstraction on specifying smart contract interactions, contract conditions, and so on; and multilingual contracts [S30].

Rest of categories. As the rest of the categories include no more than three languages, we summarise their challenges in this section. The list of challenges are:

- The lack of scalability-control and latency-control mechanisms [S58].
- The need for improvement in the modularisation of smart contract languages to separate concerns and improve the maintainability [S66].
- The oracles problem, which incorporates external information into smart contracts.
- The need for improvement in reasoning capabilities [S69], for instance, the application of event analysis using temporal aspects via complex-event processing [S42].
- Provision of indexers and registries for solving the unambiguous identification, discovery, and reuse of smart contracts in SOA environments [S57, S64, S65].
- Verification of the validity of alterations to the modelling of proof-carrying smart contracts [S41] and the integration of proof into blockchain consensus in the context of smart contract alterations.

6.7.5 RQ7.5. Which are the Reported Use Cases? The goal of this question is to describe the reported uses cases of smart contract languages. Conceptually, smart contract languages are commonly linked to cryptocurrency and transaction aspects. However, the identification of use cases can help to show the usefulness and flexibility of the language in various scenarios.

To answer this question, we have identified the use cases, examples, and applications reported in the studies. Each has been classified into a category. We have extended the taxonomy proposed in Reference [7] with the categories introduced in Reference [48] and two extra categories that are not included in any of them. Bartoletti et al. [7] classify smart contracts according to their intended domain in the categories: *Financial*, *Notary*, *Game*, *Wallet*, and *Library*. Since there are currently more domains, we have complemented those categories with some of those used by Zheng et al. [48] to classify smart contract applications, namely, *Internet of Things*, *Distributed Systems Security*, *Data Provenance*, *Sharing Economy*, and *Public Sector*. Furthermore, and to carry out a more precise categorisation, we have added two more categories: *Legal contracts*, since there are studies whose focus is the specification of legal contracts; and *Others* if the study case cannot be categorised into any of the previous existing categories. The final list of categories used for the classification of use cases is as follows:

- *Financial*. In accordance with with References [7] and [48], financial contracts enable money, banking products (e.g., deposits, credits, and mortgages), and insurance to be managed.
- *Game* [7]. If the use case is related to the implementation of games of chance (e.g., Lottery, Tic tac toe).
- *Wallet* [7]. If the use case involves handling keys, sending transactions, or managing contracts.
- *Library* [7]. If the use case is related to the definition of general-purpose operations, such as mathematical operations.
- *Sharing economy*. In accordance with Reference [48], this category covers the use cases of smart contracts in automatic payment systems, item sharing, and currency exchange platforms.
- *Public sector* [48]. This involves use cases of smart contracts for e-voting systems, personal reputation systems, and smart property exchange.
- *Notary* [7]. This category includes the use cases related to data persistence, certification, and provenance.

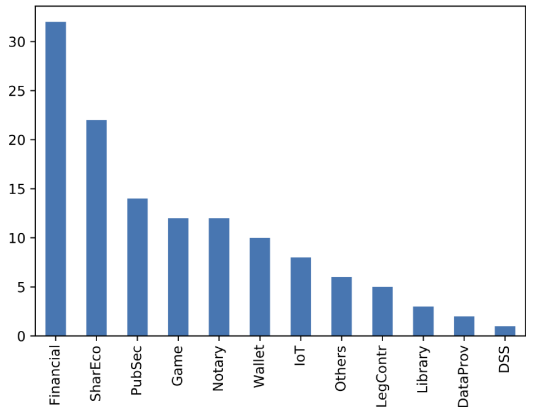


Fig. 14. Impact of use cases, examples, and applications per categories. SharEco = Sharing Economy; Pubsec = Public Sector; IoT = Internet of Things; LegContr = Legal Contracts; DataProv = Data Provenance; DSS = Distributed Systems Security.

- *Distributed system security* [48]. This includes use cases related to the improvement of cloud computing infrastructures and the security of distributed systems.
- *Internet of things (IoT)* [48]. This includes use cases related to the control of industrial manufacturing, processes of production, supply chains, and other citizen services (e.g., waste collection) that involve the use of sensors.
- *Data provenance* [48]. This involves the use cases related to ensuring information quality (e.g., in scientific or health activities).
- *Legal contract*. It includes the use cases related to enforcing agreements between two or more parties.

When use cases, examples, or applications are detected, they are recorded and categorised based on the previous categories. For instance, a use case related to a credit system is categorised as *Financial* and another related to crowdfunding is classified as *Sharing Economy*. In the case when a paper provides various use cases, examples, and/or applications, it is included in multiple categories. Figure 14 shows a ranking of the identified categories.

The top three prominent categories are *Financial*, *Sharing economy*, and *Public sector*. The *Financial* category includes the greatest number of use cases with 32 occurrences. The second category of the ranking is Sharing economy with 22 use cases, and the third is Public sector with 14 use cases. The remaining categories have 12 or less use cases each.

7 THREATS TO VALIDITY

The assessment of threats to validity is critical to assure the quality of a study. In accordance with Wohlin et al. [46], four aspects of validity should be taken into account:

- *Construct validity:* This aspect is related to the degree to which the application of constructs to phenomena is warranted concerning the research goals and questions [44]. Examples of threats to validity that could be included in this group are the suitability of the research questions and the categorisation scheme used for the data extraction. To mitigate these threats: we followed the guidelines proposed in References [26, 27] to design our research questions; we have used standard inclusion and exclusion criteria and an information quality framework to assess the quality of the studies regarding the information they provide; we have used taxonomies published in relevant references to classify languages and answer

some of the research questions; and, finally, we have carefully documented all the venues and databases, by using the StArt tool and by writing this report. One limitation is the lack of empirical evidence in the primary studies that come from the grey literature, but that does not mean that the result is incorrect, as Garousi et al. [15] state in their study.

- *Internal validity:* This aspect is related to the trustworthiness of the result of a study in terms of how well the study is conducted, as therefore this aspect highly depends on the procedures of the study and how rigorously they are executed. To encourage the internal validity of our results, we have followed a systematic approach [14, 25]. To mitigate the risk of not finding all relevant studies, we execute a formal search using defined keywords, and a backwards snowballing that allows us to recover the studies from the grey literature. In relation to the bias that could be introduced by applying the inclusion/exclusion criteria, it has been partially mitigated, because both researchers search the literature, one of them comes out the first selection, and the other re-tests the selected studies to select those ready for extraction.
- *External validity:* This aspect is related to the possible generalisation of the findings and the interest of other people outside the review. To encourage the external validity of our results (1) we have clearly stated the inclusion/exclusion criteria, and (2) we have prepared a bundle with all the datasets and artefacts produced during our analysis (see Section 4.6). Hence, data is available for those researchers or practitioners who wish to replicate or extend our review. In spite of the aforementioned access, one thread of external validity that remains in this study is that of access to the digital libraries. Our institution maintains subscriptions to all the digital libraries used in this study, but there could be certain researchers who have no access at all, although it deserves to be mentioned that we have used the most popular databases in the scientific community.
- *Conclusion validity:* This aspect is concerned with reaching appropriate conclusions through rigorous and repeatable treatment. By following the systematic approach and accessing the material we have published, this study avoids the possible bias that may threaten conclusion validity.

8 CONCLUSIONS

Blockchain and smart contracts are receiving substantial mainstream attention from academia and industry. As a consequence, a growing number of studies and languages for the specification and/or implementation of smart contracts have been published since 2002. The number of studies and languages together with the period, of almost 20 years, stresses the need for imposing certain order in the field and synthesise not only the state-of-the-art but also the state-of-practice. However, although various mapping studies, systematic literature reviews and surveys have been published regarding smart contract languages (cf., Section 3), none of them have incorporated the vision of practitioners.

To bridge this gap, one of the main aims of this review is to identify and categorise the state-of-the-art-and-practice related to smart contract languages, in terms of existing languages and their main features. To this end, we have conducted a systematic and exhaustive multivocal mapping study related to smart contract languages in which a huge number of papers (109) were studied, including both white and grey literature. Almost 36% of the studies have been collected from grey literature, which proves the importance of the industry in this field, and that the vision of the field is incomplete without grey literature. Furthermore, one hundred and one smart contract languages have been extracted from the selected primary studies, which have been analysed and classified in terms of several characteristics. The most interesting may be the identification of current challenges, because puts down in black and white the open problems, which include the

need to improve the developer coding experience by providing tools to write smart contracts as human-readable as possible, and the need to deal with oracles and trusted off-chain information and processes, among others. In addition, it deserves to be mentioned that challenges not only involve technical staff, but the collaboration with experts from law and finances is also mandatory to define a legal framework that permits smart contracts to be used with certain legal guarantees.

In addition to the conclusions obtained by answering the research questions, this mapping study has also helped us to unleash some interesting findings that reveal several gaps. First, there are a large number of smart contract languages and blockchain platforms, making the whole ecosystem rather chaotic. However, there is no common or standard language to specify smart contracts that are valid regardless of the blockchain platform. Although there have been several efforts towards the specification of smart contracts in any programming language and their execution into a specific technology, no common agreement has yet been made. Therefore, it would be interesting to work towards the standardisation of a smart contract language that could be executed for any technology. Second, within this jungle of languages and platforms, companies are strictly bound to a platform once it is selected. For example, many problems arise when the company implements smart contracts that are executed on a specific blockchain platform and the company subsequently wishes to transfer the smart contract to another blockchain platform. In this respect, there is a need for tools that could translate and move smart contracts from one blockchain platform to another. Finally, it is of major interest that the majority of the studies have been published in conferences and not in journals, and that just two of these journals have a high impact factor. This is a clear symptom of the immaturity of the field.

To conclude, this mapping study provides a snapshot of the smart contract languages field that serves as a baseline for future work. In this respect, this exhaustive classification of smart contract languages can be seen as a tool for future surveys or future literature reviews in which particular issues or aspects such as security and privacy might be studied in further detail.

ACKNOWLEDGMENTS

We express our sincere gratitude to Prof. Rafael M. Gasca for introducing us to the field of smart contracts, and to Ph.D. María Teresa Gómez-López for her support during the whole process of conducting this review.

REFERENCES

[1] Manar Abdelhamid and Ghada Hassan. 2019. Blockchain and smart contracts. In *Proceedings of the ICSIE*. ACM, 91–95. DOI: https://doi.org/10.1145/3328833.3328857

[2] Richard J. Adams, Palie Smart, and Anne Sigismund Huff. 2017. Shades of grey: Guidelines for working with the grey literature in systematic reviews for management and organizational studies. *Int. J. Manage. Rev.* 19, 4 (2017), 432–454. DOI: https://doi.org/10.1111/ijmr.12102

[3] Maher Alharby and Aad van Moorsel. 2017. Blockchain-based smart contracts: A systematic mapping study. In *Proceedings of the CS&IT*. Academy & Industry Research Collaboration Center, 125–140. DOI: https://doi.org/10.5121/csit.2017.71011

[4] Maher Alharby and Aad van Moorsel. 2017. A systematic mapping study on current research topics in smart contracts. *Int. J. Comput. Sci. Info. Technol.* 9, 5 (Oct. 2017), 151–164. DOI: https://doi.org/10.5121/ijcsit.2017.9511

[5] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. 2017. A survey of attacks on Ethereum smart contracts (SoK). In *Proceedings of the POST (LNCS)*, Vol. 10204. Springer, 164–186.

[6] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl E. Landwehr. 2004. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Depend. Sec. Comput.* 1, 1 (2004), 11–33. DOI: https://doi.org/10.1109/TDSC.2004.2

[7] Massimo Bartoletti and Livio Pompianu. 2017. An empirical analysis of smart contracts: Platforms, applications, and design patterns. Retrieved from https://arxiv:cs.CR/1703.06322.

[8] Davide Calvaresi, Alevtina Dubovitskaya, Jean-Paul Calbimonte, Kuldar Taveter, and Michael Schumacher. 2018. Multi-agent systems and blockchain: Results from a systematic literature review. In *Proceedings of the PAAMS (LNCS)*, Vol. 10978. Springer, 110–126.

[9] Fran Casino, Thomas K. Dasaklis, and Constantinos Patsakis. 2019. A systematic literature review of blockchain-based applications: Current status, classification, and open issues. *Telemat. Informat.* 36 (2019), 55–81. DOI: https://doi.org/10.1016/j.tele.2018.11.006

[10] Michael A. Covington. 2010. First lecture on symbolic programming and Lisp. Retrieved from http://www.covingtoninnovations.com/mc/LispNotes/FirstLectureOnSymbolicProgramming.pdf.

[11] Vimal Dwivedi, Vipin Deval, Abhishek Dixit, and Alex Norta. 2019. Formal-verification of smart-contract languages: A survey. In *Advances in Computing and Data Sciences*. Springer, Singapore, 738–747. DOI: https://doi.org/10.1007/978-981-13-9942-8_68

[12] Sandra Fabbri, Elis Montoro Hernandes, André Di Thommazo, Anderson Belgamo, Augusto Zamboni, and Cleiton Silva. 2012. Using information visualization and text mining to facilitate the conduction of systematic literature reviews. In *Proceedings of the ICEIS (LNBIP)*, Vol. 141. Springer, 243–256.

[13] Vahid Garousi, Michael Felderer, and Mika V. Mäntylä. 2016. The need for multivocal literature reviews in software engineering: Complementing systematic literature reviews with grey literature. In *Proceedings of the EASE*. ACM, 26:1–26:6.

[14] Vahid Garousi, Michael Felderer, and Mika V. Mäntylä. 2019. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Info. Softw. Technol.* 106 (2019), 101–121. DOI: https://doi.org/10.1016/j.infsof.2018.09.006

[15] Vahid Garousi and Mika V. Mäntylä. 2016. When and what to automate in software testing? A multi-vocal literature review. *Info. Softw. Technol.* 76 (2016), 92–117. DOI: https://doi.org/10.1016/j.infsof.2016.04.015

[16] Gartner. 2020. *Gartner Predicts that Organizations Using Blockchain Smart Contracts Will Increase Overall Data Quality by 50%*. Technical Report. Gartner. Retrieved from https://www.gartner.com/en/newsroom/press-releases/2020-01-30-gartner-predicts-that-organizations-using-blockchain.

[17] Guido Governatori, Florian Idelberger, Zoran Milosevic, Régis Riveret, Giovanni Sartor, and Xiwei Xu. 2018. On legal contracts, imperative and declarative smart contracts, and blockchain systems. *Artific. Intell. Law* 26, 4 (2018), 377–409. DOI: https://doi.org/10.1007/s10506-018-9223-3

[18] Purva Grover, Arpan Kumar Kar, and P. Vigneswara Ilavarasan. 2018. Blockchain for businesses: A systematic literature review. In *Proceedings of the I3E (LNCS)*, Vol. 11195. Springer, 325–336.

[19] Felix Härer and Hans-Georg Fill. 2019. A comparison of approaches for visualizing blockchains and smart contracts, in: Jusletter IT 21. DOI: https://doi.org/10.5281/zenodo.2585575

[20] Dominik Harz and William J. Knottenbelt. 2018. Towards safer smart contracts: A survey of languages and verification methods. Retrieved from http://arxiv.org/abs/1809.09805.

[21] Florian Hawlitschek, Benedikt Notheisen, and Timm Teubner. 2018. The limits of trust-free systems: A literature review on blockchain technology and trust in the sharing economy. *Electronic Comm. Res. Appl.* 29 (2018), 50–63. DOI: https://doi.org/10.1016/j.elerap.2018.03.005

[22] Syed Akhter Hossain. 2017. Blockchain computing: Prospects and challenges for digital transformation. In *Proceedings of the ICRITO*. 61–65.

[23] Mubashar Iqbal and Raimundas Matulevicius. 2019. Blockchain-based application security risks: A systematic literature review. In *Proceedings of the CAiSE (LNBIP)*, Vol. 349. Springer, 176–188.

[24] Muhammad Uzair Khan, Salman Sherin, Muhammad Zohaib Iqbal, and Rubab Zahid. 2019. Landscaping systematic mapping studies in software engineering: A tertiary study. *J. Syst. Softw.* 149 (2019), 396–436. DOI: https://doi.org/10.1016/j.jss.2018.12.018

[25] Barbara Kitchenham and Stuart Charters. 2007. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*. Technical Report EBSE-2007-01. School of Computer Science and Mathematics, Keele University.

[26] Barbara Ann Kitchenham, David Budgen, and Pearl Brereton. 2015. *Evidence-based Software Engineering and Systematic Reviews* (1 ed.). CRC Press. Retrieved from https://www.crcpress.com/Evidence-Based-Software-Engineering-and-Systematic-Reviews/Kitchenham-Budgen-Brereton/p/book/9781482228656.

[27] Marco Kuhrmann, Daniel Méndez Fernández, and Maya Daneva. 2017. On the pragmatic design of literature studies in software engineering: An experience-based guideline. *Empir. Softw. Eng.* 22, 6 (2017), 2852–2891. DOI: https://doi.org/10.1007/s10664-016-9492-y

[28] Daniel Macrinici, Cristian Cartofeanu, and Shang Gao. 2018. Smart contract applications within blockchain technology: A systematic mapping study. *Telemat. Informat.* 35, 8 (2018), 2337–2354. DOI: https://doi.org/10.1016/j.tele.2018.10.004

[29] Andrew Miller, Zhicheng Cai, and Somesh Jha. 2018. Smart contracts and opportunities for formal methods. In *Proceedings of the ISoLA (4) (LNCS)*, Vol. 11247. Springer, 280–299.

[30] Imane Mokdad and Nabil M. Hewahi. 2020. *Empirical Evaluation of Blockchain Smart Contracts*. Springer International Publishing, Cham, 45–71. DOI: https://doi.org/10.1007/978-3-030-38677-1_3
[31] Rohaila Naaz, Neha Shrivastav, and Deepak Kaler. 2019. A review paper on smart contracts in blockchain network-based applications. In *Proceedings of the ICAC*. 130–134.
[32] Satoshi Nakamoto. 2009. Bitcoin: A peer-to-peer electronic cash system. Retrieved from http://www.bitcoin.org/bitcoin.pdf.
[33] Jason Priem, Dario Taraborelli, Paul Groth, and Cameron Neylon. [n.d.]. Altmetrics: A manifesto,. http://altmetrics.org/manifesto/.
[34] John Ream, Yang Chu, and David Schatsky. 2016. Upgrading blockchains smart contract use cases in industry. *Deloitte Insights* (June 2016). Retrieved from https://www2.deloitte.com/us/en/insights/focus/signals-for-strategists/using-blockchain-for-smart-contracts.html#.
[35] Peter Van Roy and Seif Haridi. 2004. *Concepts, Techniques, and Models of Computer Programming*. MIT Press. Retrieved from http://www.info.ucl.ac.be/people/PVR/book.html.
[36] Pablo Lamela Seijas, Simon J. Thompson, and Darryl McAdams. 2016. Scripting smart contracts for distributed ledger technology. *IACR Cryptol. ePrint Arch.* 2016 (2016), 1156. Retrieved from http://eprint.iacr.org/2016/1156.
[37] Amritraj Singh, Reza M. Parizi, Qi Zhang, Kim-Kwang Raymond Choo, and Ali Dehghantanha. 2020. Blockchain smart contracts formalization: Approaches and challenges to address vulnerabilities. *Comput. Secur.* 88 (2020). DOI: https://doi.org/10.1016/j.cose.2019.101654
[38] Nick Szabo. 1997. Smart contracts: Formalizing and securing relationships on public networks. *First Monday* 2, 9 (1997). Retrieved from http://dblp.uni-trier.de/db/journals/firstmonday/firstmonday2.html#Szabo97.
[39] Faizan Tariq and Ricardo Colomo Palacios. 2019. Use of blockchain smart contracts in software engineering: A systematic mapping. In *Proceedings of the ICCSA (5) (LNCS)*, Vol. 11623. Springer, 327–337.
[40] Paul J. Taylor, Tooska Dargahi, Ali Dehghantanha, Reza M. Parizi, and Kim-Kwang Raymond Choo. 2020. A systematic literature review of blockchain cyber security. *Dig. Commun. Netw.* 6, 2 (May 2020), 147–156. DOI: https://doi.org/10.1016/j.dcan.2019.01.005
[41] Horst Treiblmaier and Roman Beck. 2019. *Business Transformation Through Blockchain*. Springer.
[42] Richard Y. Wang and Diane M. Strong. 1996. Beyond accuracy: What data quality means to data consumers. *J. Manage. Info. Syst.* 12, 4 (1996), 5–33. DOI: https://doi.org/10.1080/07421222.1996.11518099
[43] Shuai Wang, Yong Yuan, Xiao Wang, Juanjuan Li, Rui Qin, and Fei-Yue Wang. 2018. An overview of smart contract: Architecture, applications, and future trends. In *Proceedings of the IV*. IEEE, 108–113.
[44] Roel J. Wieringa. 2014. *Design Science Methodology for Information Systems and Software Engineering*. Springer. DOI: https://doi.org/10.1007/978-3-662-43839-8
[45] Claes Wohlin. 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the EASE*, Martin J. Shepperd, Tracy Hall, and Ingunn Myrtveit (Eds.). ACM, 38:1–38:10. DOI: https://doi.org/10.1145/2601248.2601268
[46] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, and Björn Regnell. 2012. *Experimentation in Software Engineering*. Springer. DOI: https://doi.org/10.1007/978-3-642-29044-2
[47] Min Xu, Xingtong Chen, and Gang Kou. 2019. A systematic review of blockchain. *Financ. Innov.* 5, 1 (Dec. 2019). DOI: https://doi.org/10.1186/s40854-019-0147-z
[48] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Weili Chen, Xiangping Chen, Jian Weng, and Muhammad Imran. 2020. An overview on smart contracts: Challenges, advances and platforms. *Future Gener. Comput. Syst.* 105 (2020), 475–491. DOI: https://doi.org/10.1016/j.future.2019.12.019
[49] Weiqin Zou, David Lo, Pavneet Singh Kochhar, Xuan-Bach Dinh Le, Xin Xia, Yang Feng, Zhenyu Chen, and Baowen Xu. 2019. Smart contract development: Challenges and opportunities. *IEEE Trans. Software Eng.* (2019), 1–1. DOI: https://doi.org/10.1109/TSE.2019.2942301

SELECTED WHITE LITERATURE

[S1] Mark S. Miller, Tom Van Cutsem, and Bill Tulloh. 2013. Distributed electronic rights in JavaScript. In *Proceedings of the ESOP (Lecture Notes in Computer Science)*, Vol. 7792. Springer, 1–20.
[S2] Alex Norta, Lixin Ma, Yucong Duan, Addi Rull, Merit Kõlvart, and Kuldar Taveter. 2015. eContractual choreography-language properties towards cross-organizational business collaboration. *J. Internet Serv. Appl.* 6, 1 (2015), 8:1–8:23.
[S3] Richard Hull, Vishal S. Batra, Yi-Min Chen, Alin Deutsch, Fenno F. Terry Heath III, and Victor Vianu. 2016. Towards a shared ledger business collaboration language based on data-aware processes. In *Proceedings of the ICSOC (Lecture Notes in Computer Science)*, Vol. 9936. Springer, 18–36.
[S4] Jack Pettersson and Robert Edström. 2006. *Safer Smart Contracts Through Type-driven Development. Using Dependent and Polymorphic Types for Safer Development of Smart Contracts*. Master's thesis. Chalmers University of Technology. University of Gothenburg. Department of Computer Science and Engineering. Göteborg, Sweden, February 2016.

[S5] Christopher Frantz and Mariusz Nowostawski. 2016. From institutions to code: Towards automated generation of smart contracts. In *Proceedings of the FAS*W@SASO/ICCAC*. IEEE, 210–215.

[S6] Ahmed E. Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. 2016. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE Computer Society, 839–858.

[S7] Michael J. Coblenz. 2017. Obsidian: A safer blockchain programming language. In *Proceedings of the ICSE (Companion Volume)*. IEEE Computer Society, 97–99.

[S8] Anastasia Mavridou and Aron Laszka. 2017. Designing secure Ethereum smart contracts: A finite state machine-based approach. Retrieved from https://arXiv:1708.03778.

[S9] Luis Daniel Ibáñez and Elena Simperl. 2017. TRiC: Terms, RIghts and conditions semantic descriptors for smart contracts. In *Proceedings of the ESWC (Satellite Events) (Lecture Notes in Computer Science)*, Vol. 10577. Springer, 317–326.

[S10] Russell O'Connor. 2017. Simplicity: A new language for blockchains. In *Proceedings of the PLAS@CCS*. ACM, 107–120.

[S11] Benjamin Egelund-Müller, Martin Elsman, Fritz Henglein, and Omri Ross. 2017. Automated execution of financial contracts on blockchains. *Bus. Inf. Syst. Eng.* 59, 6 (2017), 457–467.

[S12] Alex Biryukov, Dmitry Khovratovich, and Sergei Tikhomirov. 2017. Findel: Secure derivative contracts for Ethereum. In *Proceedings of the Financial Cryptography Workshops (Lecture Notes in Computer Science)*, Vol. 10323. Springer, 453–467.

[S13] Vikram Dhillon, David Metcalf, and Max Hooper. 2017. Recent developments in blockchain. *Blockchain Enabled Applications*. 151–181.

[S14] Vincenzo Scoca, Rafael Brundo Uriarte, and Rocco De Nicola. 2017. Smart contract negotiation in cloud computing. In *Proceedings of the CLOUD*. IEEE Computer Society, 592–599.

[S15] Rui Yuan, Yubin Xia, Haibo Chen, Binyu Zang, and Jan Xie. 2018. ShadowEth: Private smart contract on public blockchain. *J. Comput. Sci. Technol.* 33, 3 (2018), 542–556.

[S16] Xiaomin Bai, Zijing Cheng, Zhangbo Duan, and Kai Hu. 2018. Formal modeling and verification of smart contracts. In *Proceedings of the ICSCA*. ACM, 322–326.

[S17] Massimo Bartoletti, Tiziana Cimoli, and Roberto Zunino. 2018. Fun with bitcoin smart contracts. In *Proceedings of the ISoLA (4) (Lecture Notes in Computer Science)*, Vol. 11247. Springer, 432–449.

[S18] Ilya Sergey, Amrit Kumar, and Aquinas Hobor. 2018. Scilla: A smart contract intermediate-level language. *CoRR* abs/1804.05141.

[S19] Raymond Cheng, Fan Zhang, Jernej Kos, Warren He, Nicholas Hynes, Noah M. Johnson, Ari Juels, Andrew Miller, and Dawn Song. 2018. Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contract execution. *CoRR* abs/1804.05141.

[S20] Tesnim Abdellatif and Kei-Léo Brousmiche. 2018. Formal verification of smart contracts based on users and blockchain behaviors models. In *Proceedings of the NTMS*. IEEE, 1–5.

[S21] Massimo Bartoletti and Roberto Zunino. 2018. BitML: A calculus for bitcoin smart contracts. In *Proceedings of the ACM Conference on Computer and Communications Security*. ACM, 83–100.

[S22] Krishnendu Chatterjee, Amir Kafshdar Goharshady, and Yaron Velner. 2018. Quantitative analysis of smart contracts. In *Proceedings of the ESOP (Lecture Notes in Computer Science)*, Vol. 10801. Springer, 739–767.

[S23] Anastasia Mavridou and Aron Laszka. 2018. Tool demonstration: FSolidM for designing secure Ethereum smart contracts. In *Proceedings of the POST (Lecture Notes in Computer Science)*, Vol. 10804. Springer, 270–277.

[S24] Xiao He, Bohan Qin, Yan Zhu, Xing Chen, and Yi Liu. 2018. SPESC: A specification language for smart contracts. In *Proceedings of the COMPSAC (1)*. IEEE Computer Society, 132–137.

[S25] Franklin Schrans, Susan Eisenbach, and Sophia Drossopoulou. 2018. Writing safe smart contracts in Flint. In *Programming*. ACM, 218–219.

[S26] Joost T. de Kruijff and Hans Weigand. 2018. An introduction to commitment-based smart contracts using ReactionRuleML. In *Proceedings of the VMBO (CEUR Workshop Proceedings)*, Vol. 2239. CEUR-WS.org, 149–157.

[S27] Jingwen Hu and Yong Zhong. 2018. A method of logic-based smart contracts for blockchain system. In *Proceedings of the ICDPA*. ACM, 58–61.

[S28] Pablo Lamela Seijas and Simon J. Thompson. 2018. Marlowe: Financial contracts on blockchain. In *Proceedings of the ISoLA (4) (Lecture Notes in Computer Science)*, Vol. 11247. Springer, 356–375.

[S29] Tara Astigarraga, Xiaoyan Chen, Yaoliang Chen, Jingxiao Gu, Richard Hull, Limei Jiao, Yuliang Li, and Petr Novotný. 2018. Empowering business-level blockchain users with a rules framework for smart contracts. In *Proceedings of the ICSOC (Lecture Notes in Computer Science)*, Vol. 11236. Springer, 111–128.

[S30] Emanuel Regnath and Sebastian Steinhorst. 2018. SmaCoNat: Smart contracts in natural language. In *Proceedings of the FDL*. IEEE, 5–16.

[S31] Tim Weingärtner, Rahul Rao, Jasmin Ettlin, Patrick Suter, and Philipp Dublanc. 2018. Smart contracts using Blockly: Representing a purchase agreement using a graphical programming language. In *Proceedings of the CVCBT*. IEEE, 55–64.

[S32] Yan Zhu, Xiaoxu Song, Shuai Yang, Yao Qin, and Qiong Zhou. 2018. Secure smart contract system built on SMPC over blockchain. In *Proceedings of the iThings/GreenCom/CPSCom/SmartData*. IEEE, 1539–1544.

[S33] Naoto Sato, Takaaki Tateishi, and Shunichi Amano. 2018. Formal requirement enforcement on smart contracts based on linear dynamic logic. In *Proceedings of the iThings/GreenCom/CPSCom/SmartData*. IEEE, 945–954.

[S34] Jacob Eberhardt and Stefan Tai. 2018. ZoKrates - Scalable privacy-preserving off-chain computations. In *Proceedings of the iThings/GreenCom/CPSCom/SmartData*. IEEE, 1084–1091.

[S35] Markus Knecht. 2019. Mandala: A smart contract programming language. *CoRR* abs/1911.11376.

[S36] 2018. Conditional formalization of smart contract using semantic web rule language. *J. Eng. Appl. Sci.* 13 (2018), 8716–8721.

[S37] Nachiappan Valliappan, Solène Mirliaz, Elisabet Lobo Vesga, and Alejandro Russo. 2018. Towards adding variety to simplicity. In *Proceedings of the ISoLA (4) (Lecture Notes in Computer Science)*, Vol. 11247. Springer, 414–431.

[S38] Zaynah Dargaye, Antonella Del Pozzo, and Sara Tucci Piergiovanni. 2018. Pluralize: A trustworthy framework for high-level smart contract-draft. *CoRR* abs/1812.05444.

[S39] Junhui Kim and Joongheon Kim. 2019. Light-weight programming language for blockchain. In *Proceedings of the MobiSys*. ACM, 653–654.

[S40] Giovanni Ciatto, Alfredo Maffi, Stefano Mariani, and Andrea Omicini. 2019. Towards agent-oriented blockchains: Autonomous smart contracts. In *Proceedings of the PAAMS (Lecture Notes in Computer Science)*, Vol. 11523. Springer, 29–41.

[S41] Thomas D. Dickerson, Paul Gazzillo, Maurice Herlihy, Vikram Saraph, and Eric Koskinen. 2018. Proof-carrying smart contracts. In *Proceedings of the Financial Cryptography Workshops (Lecture Notes in Computer Science)*, Vol. 10958. Springer, 325–338.

[S42] Fabiana Fournier and Inna Skarbovsky. 2019. Enriching smart contracts with temporal aspects. In *Proceedings of the ICBC (Lecture Notes in Computer Science)*, Vol. 11521. Springer, 126–141.

[S43] Takaaki Tateishi, Sachiko Yoshihama, Naoto Sato, and Shin Saito. 2019. Automatic smart contract generation using controlled natural language and template. *IBM J. Res. Dev.* 63, 2/3 (2019), 6:1–6:12.

[S44] Wei-Tek Tsai, Ning Ge, Jiaying Jiang, Kevin Feng, and Juan He. 2019. Beagle: A new framework for smart contracts taking account of law. In *Proceedings of the 13th IEEE International Conference on Service-Oriented System Engineering, (SOSE'19)*. IEEE, 134–13411. DOI: http://dx.doi.org/10.1109/SOSE.2019.00028

[S45] Joost T. de Kruijff and Hans Weigand. 2019. Introducing CommitRuleML for smart contracts. In *Proceedings of the VMBO (CEUR Workshop Proceedings)*, Vol. 2383. CEUR-WS.org.

[S46] Bruno França, Sophie Radermacher, and Reto Trinkler. 2019. Katallassos: A standard framework for finance. *CoRR* abs/1903.01600.

[S47] Kees Boogaard. 2019. *A Model-Driven Approach to Smart Contract Development*. Master's thesis. Business Informatics. Utrecht University.

[S48] Elvira Albert, Pablo Gordillo, Benjamin Livshits, Albert Rubio, and Ilya Sergey. 2018. EthIR: A framework for high-level analysis of Ethereum bytecode. In *Proceedings of the ATVA (Lecture Notes in Computer Science)*, Vol. 11138. Springer, 513–520.

[S49] Karl Crary and Michael J. Sullivan. 2015. Peer-to-peer affine commitment using bitcoin. In *Proceedings of the PLDI*. ACM, 479–488.

[S50] Theodoros Kasampalis, Dwight Guth, Brandon M. Moore, Traian-Florin Serbanuta, Yi Zhang, Daniele Filaretti, Virgil Nicolae Serbanuta, Ralph Johnson, and Grigore Rosu. 2019. IELE: A rigorously designed language and tool ecosystem for the blockchain. In *Proceedings of the FM (Lecture Notes in Computer Science)*, Vol. 11800. Springer, 593–610.

[S51] Massimo Bartoletti, Letterio Galletta, and Maurizio Murgia. 2019. A minimal core calculus for solidity contracts. In *Proceedings of the DPM/CBT@ESORICS (Lecture Notes in Computer Science)*, Vol. 11737. Springer, 233–243.

[S52] Ada Bagozi, Devis Bianchini, Valeria De Antonellis, Massimiliano Garda, and Michele Melchiori. 2019. A three-layered approach for designing smart contracts in collaborative processes. In *Proceedings of the OTM Conferences (Lecture Notes in Computer Science)*, Vol. 11877. Springer, 440–457.

[S53] Peng Qin, Jingzhi Guo, Bingqing Shen, and Quanyi Hu. 2019. Towards self-automatable and unambiguous smart contracts: Machine natural language. In *Proceedings of the ICEBE (Lecture Notes on Data Engineering and Communications Technologies)*, Vol. 41. Springer, 479–491.

[S54] Marek Skotnica and Robert Pergl. 2019. Das contract - A visual domain specific language for modeling blockchain smart contracts. In *Proceedings of the EEWC (Lecture Notes in Business Information Processing)*, Vol. 374. Springer, 149–166.

[S55] Fausto Spoto. 2019. A Java framework for smart contracts. In *Financial Cryptography Workshops (Lecture Notes in Computer Science)*, Vol. 11599. Springer, 122–137.
[S56] Tomasz Górski and Jakub Bednarski. 2019. Modeling of smart contracts in blockchain solution for renewable energy grid. In *Proceedings of the EUROCAST (1) (Lecture Notes in Computer Science)*, Vol. 12013. Springer, 507–514.
[S57] Andrea Lamparelli, Ghareeb Falazi, Uwe Breitenbücher, Florian Daniel, and Frank Leymann. 2019. Smart contract locator (SCL) and smart contract description language (SCDL). In *Proceedings of the ICSOC Workshops (Lecture Notes in Computer Science)*, Vol. 12019. Springer, 195–210.
[S58] Eranga Bandara, Wee Keong Ng, Nalin Ranasinghe, and Kasun De Zoysa. 2019. Aplos: Smart contracts made smart. In *Proceedings of the BlockSys (Communications in Computer and Information Science)*, Vol. 1156. Springer, 431–445.
[S59] Cosimo Laneve, Claudio Sacerdoti Coen, and Adele Veschetti. 2019. On the prediction of smart contracts' behaviours. In *From Software Engineering to Formal Methods and Tools, and Back (Lecture Notes in Computer Science)*, Vol. 11865. Springer, 397–415.
[S60] Nejc Zupan, Prabhakaran Kasinathan, Jorge Cuellar, and Markus Sauer. 2020. *Secure Smart Contract Generation Based on Petri Nets*. Springer Singapore, Singapore, 73–98.
[S61] Danil Annenkov, Jakob Botsch Nielsen, and Bas Spitters. 2020. ConCert: A smart contract certification framework in Coq. In *Proceedings of the CPP*. ACM, 215–228.
[S62] Ben Lippmeier, Amos Robinson, and Andrae Muys. 2019. Smart contracts as authorized production rules. In *Proceedings of the PPDP*. ACM, 14:1–14:14.
[S63] Adrian Mizzi, Joshua Ellul, and Gordon J. Pace. 2019. Porthos: Macroprogramming blockchain systems. In *Proceedings of the NTMS*. IEEE, 1–5.
[S64] Hamza Baqa, Nguyen Binh Truong, Noël Crespi, Gyu Myoung Lee, and Franck Le Gall. 2019. Semantic smart contracts for blockchain-based services in the Internet of Things. In *Proceedings of the NCA*. IEEE, 1–5.
[S65] Luca Guida and Florian Daniel. 2019. Supporting reuse of smart contracts through service orientation and assisted development. In *Proceedings of the DAPPCON*. IEEE, 59–68.
[S66] Chien-Che Hung, Kung Chen, and Chun-Feng Liao. 2019. Modularizing cross-cutting concerns with aspect-oriented extensions for solidity. In *Proceedings of the DAPPCON*. IEEE, 176–181.
[S67] Samuel Steffen, Benjamin Bichsel, Mario Gersbach, Noa Melchior, Petar Tsankov, and Martin T. Vechev. 2019. zkay: Specifying and enforcing data privacy in smart contracts. In *Proceedings of the ACM Conference on Computer and Communications Security*. ACM, 1759–1776.
[S68] Ankush Das, Stephanie Balzer, Jan Hoffmann, and Frank Pfenning. 2019. Resource-aware session types for digital contracts. *CoRR* abs/1902.06056.
[S69] Maximilian Wöhrer and Uwe Zdun. 2020. Domain specific language for smart contract development. In *Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency*. IEEE.
[S70] Ilya Sergey, Vaivaswatha Nagaraj, Jacob Johannsen, Amrit Kumar, Anton Trunov, and Ken Chan Guan Hao. 2019. Safer smart contract programming with Scilla. *Proc. ACM Program. Lang.* 3, OOPSLA (2019), 185:1–185:30.

SELECTED GREY LITERATURE

[G1] Patrick Dai, Neil Mahi, Jordan Earls, and Alex Norta. 2017. *Smart-contract Value-transfer Protocols on a Distributed Mobile Application Platform*. Technical Report. Qtum Foundation.
[G2] Stuart Popejoy. 2017. *The Pact Smart Contract Language*. Technical Report. Kadena.
[G3] Nick Roberts. 2018. *A Modern Programming Language for Smart Contracts*. Technical Report. University Carnegie-Mellon.
[G4] Nick Szabo. 2002. A Formal Language for Analyzing Contracts. Retrieved from https://nakamotoinstitute.org/contract-language/.
[G5] Roconnor. 2016. Script. *bitcoinwiki*. Retrieved from https://en.bitcoin.it/w/index.php?title=Script&oldid=61707.
[G6] Nicola Atzei, Massimo Bartoletti, Tiziana Cimoli, Stefano Lande, and Roberto Zunino. 2018. *BALZaC. Bitcoin Abstract Language, analyZer and Compiler*. Technical Report. Blockchain@Unica group. University of Cagliari. Retrieved from https://blockchain.unica.it/balzac/docs/.
[G7] Digital Asset. 2017. DAML SDK Documentation. *White paper*. Retrieved from https://docs.daml.com/_downloads/DigitalAssetSDK.pdf.
[G8] Chain. 2017. Ivy for Bitcoin: A smart contract language that compiles to Bitcoin Script. *blog*. Retrieved from https://blog.chain.com/ivy-for-bitcoin-a-smart-contract-language-that-compiles-to-bitcoin-script-bec06377141a.
[G9] Cornell Blockchain. 2017. Bamboo: A language for morphing smart contracts. *GitHub page*. Retrieved from https://github.com/pirapira/bamboo.
[G10] Christian. 2017. Babbage—A mechanical smart contract language. *blog page*. (2017). Retrieved from https://medium.com/@chriseth/babbage-a-mechanical-smart-contract-language-5c8329ec5a0e.

[G11] Asher Manning. 2017. Zen Protocol's Smart Contract Paradigm-Zen Protocol. Retrieved from https://blog.zenprotocol.com/zen-protocols-smart-contract-paradigm-a6e54a187d84.

[G12] Stephen Andrews. 2018. fi-smart coding for smart contracts. *website*. Retrieved from https://learn.fi-code.com/.

[G13] Evan Schwartz and Stefan Thomas. 2018. *Codius - White Paper*. Technical Report. codius. Retrieved from https://github.com/codius/codius-wiki/wiki/White-Paper.

[G14] Fabrice Le Fessant, Alain Mebsout, David Declerk, and Adrien Champion. 2018. The Liquidity Language for smart contracts. Retrieved from http://www.liquidity-lang.org/.

[G15] Nomadic Labs. 2018. Michelson: The language of smart contracts in Tezos. *website*. Retrieved from https://tezos.gitlab.io/master/whitedoc/michelson.html.

[G16] Michael Peyton Jones. 2018. Plutus Platform. *GitHub page*. Retrieved from https://github.com/input-output-hk/plutus.

[G17] Lucius Gregory Meredith, Jack Pettersson, Gary Stephenson, Michael Stay, Kent Shikama, and Joseph Denman. 2018. *Contracts, Composition, and Scaling. The Rholang Specification*. Technical Report. rchain. Retrieved from https://developer.rchain.coop/assets/rholang-spec-0.2.pdf.

[G18] Vitalik Buterin. 2018. Vyper-Vyper documentation. *website*. Retrieved from https://viper.readthedocs.io/en/latest/.

[G19] Stefan Schmidt, Marten Jung, Thomas Schmidt, Ingo Sterzinger, Günter Schmidt, Moritz Gomm, Klaus Tschirschke, Tapio Reisinger, Fabian Schlarb, Daniel Benkenstein, and Bastian Emig. 2018. *Unibright-the Unified Framework for Blockchain-based Business Integration*. Technical Report. unibright.io. Retrieved from https://www.unibright.io/download/Unibright_Whitepaper.pdf.

[G20] Ethereum Foundation. 2019. Solidity Documentation. *website*. Retrieved from https://solidity.readthedocs.io/en/latest/.

[G21] Gabriel Alfour. 2019. *LIGO*. Technical Report. Marigold. Retrieved from https://ligolang.org/blog/.

[G22] Asher Manning. 2017. ZF*. *website*. Retrieved from https://docs.zenprotocol.com/zf.

[G23] Aethernity workgroup. 2018. Aethernity documentation. *website*. Retrieved from http://aeternity.com/documentation-hub/aesophia/.

[G24] Everett Hildenbrandt and Dwight Guth. 2018. IELE Semantics. *GitHub page*. Retrieved from https://github.com/runtimeverification/iele-semantics.

[G25] Fábio César Canesin, Yak Jun Xiang, Jonathan Lim, Ethan Fast, J. Lowenthal, Alan Fong, and Erik van den Brink. 2018. NEO White paper. Retrieved from https://docs.neo.org/docs/en-us/basic/whitepaper.html.

[G26] Accord Project Technology Working Group. 2018. Cicero. *GitHub page*. Retrieved from https://github.com/accordproject/cicero.

[G27] Nils Bundi. 2018. ACTUS: The algorithmic representation of financial contracts. *whitepaper*. Retrieved from https://docs.wixstatic.com/ugd/3df5e2_eceb16e5f7f14d11903a6412aebb9e4a.pdf.

[G28] Aaron Wright, David Roon, and ConsenSys AG. 2019. OpenLaw Documentation. *website*. Retrieved from https://docs.openlaw.io/

[G29] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. 2019. *Zether: Towards Privacy in a Smart Contract World*. Technical Report. Cryptology ePrint Archive. Retrieved from https://eprint.iacr.org/2019/191

[G30] Accord Project. 2019. Ergo Language Guide. *website*. Retrieved from https://docs.accordproject.org/docs/logic-ergo.html.

[G31] BOScoin. 2017. Smart contracts and trust contracts: Part 3. Retrieved from https://medium.com/@boscoin/smart-contracts-trust-contracts-part-3-6cf76bf5882e.

[G32] Michael Burge. 2017. Write your next Ethereum Contract in Pyramid Scheme. Retrieved from http://www.michaelburge.us/2017/11/28/write-your-next-ethereum-contract-in-pyramid-scheme.html.

[G33] Theodoros Kasampalis, Dwight Guth, Brandon Moore, Traian Serbanuta, Virgil Serbanuta, Daniele Filaretti, Grigore Rosu, and Ralph Johnson. 2018. *IELE: An Intermediate Level Blockchain Language Designed and Implemented Using Formal Semantics*. Technical Report. Department of Computer Science. College of Engineering.Retrieved from http://hdl.handle.net/2142/100320.

[G34] Deon Digital. 2019. Deon Digital CSL Language Guide Documentation. *website*. Retrieved from https://deondigital.com/docs/v0.30.0/cslguide.pdf.

[G35] Benoit Rognier. 2019. What is Archetype. *website*. Retrieved from https://docs.archetype-lang.org/.

[G36] Antonina Begicheva and Ilya Smagin. 2018. *RIDE: A Smart Contract Language for Waves*. Technical Report. wavesprotocol.org. Retrieved from https://wavesprotocol.org/files/docs/white_paper_waves_smart_contracts.pdf?cache=b.

[G37] Franklin Schrans. 2018. *Writing Safe Smart Contracts in Flint*. Technical Report. Imperial College. Faculty of Engineering. Retrieved from https://www.imperial.ac.uk/media/imperial-college/faculty-of-engineering/computing/public/1718-ug-projects/Franklin-Schrans-A-new-programming-language-for-safer-smart-contracts.pdf.

[G38] Sam Blackshear, Evan Cheng, David L. Dill, Victor Gao, Ben Maurer, Todd Nowacki, Alistair Pott, Shaz Qadeer, Rain, Dario Russi, Stephane Sezer, Tim Zakian, and Runtian Zhou. 2020. *Move: A Language with Programmable Resources*. Technical Report. Libra Association. Retrieved from https://developers.libra.org/docs/assets/papers/libra-move-a-language-with-programmable-resources/2020-05-26.pdf.

[G39] Ethereum Foundation. 2020. Yul. *website*. (2020). Retrieved from https://solidity.readthedocs.io/en/latest/yul.html.