Ph.D. in Electronic and Computer Engineering
Dept. of Electrical and Electronic Engineering
University of Cagliari

Agile methodologies and blockchain development

Simona Ibba

Advisors: Giorgio Giacinto and Michele Marchesi
Curriculum: ING-INF/05

Cycle XXXI
January 2019

Dedicated to my mother Luisanna and my grandmother Maria

Contents

List of Figures

List of Tables

Aknowledgments - Ringraziamenti

Fare ricerca è un po' come essere esploratori. Non ci sono terre emerse da scoprire. Esistono però mondi nuovi su cui investigare, idee da sperimentare, strade da inventare. Con impegno, con entusiasmo, ma anche con quel giusto piglio e quel sano buonumore che dà gusto al lavoro. Me lo hai insegnato tu, caro Michele. Grazie per il tuo incontenibile desiderio di conoscenza, per la tua lungimiranza umana e accademica, per la tua correttezza. Seguirti nelle tante sfide scientifiche di questi anni non è stato semplice. È stato sicuramente un bellissimo ed arricchente terreno di crescita in cui migliorarsi e rischiare con la certezza di avere in te un approdo sicuro a cui aggrapparsi nelle eccessive intemperanze.

Grazie a te Giulio per aver scommesso per primo su di me. In questi anni, davanti ai successi, ti immaginavo sorridente ed orgoglioso a contemplare i frutti di un lavoro che, fondato sulle tue prime decisive indicazioni, in qualche modo ti appartiene.

Grazie a tutti i colleghi dell'Agile Group. Si può vivere il lavoro in tanti modi: è stato bello condividere con voi la passione per lo studio, l'impegno, le piccole battaglie di ogni giorno. Essere partecipi degli stessi progetti, abitare gli stessi ambienti per tante ore, sono i segni di una piccola comunità che cresce con la sua ricchezza e le sue fragilità, ma anche con la capacità di poter portare sempre competenza e valore in ogni confronto tecnico e scientifico. Grazie in particolare a Roberto per aver condiviso e guidato l'ultimo tratto di questo percorso, a Filippo per avermi valorizzato in ogni contesto e per la sua costante, affidabile e briosa presenza, ad Ilaria perché nella sua semplicità, nel suo senso di responsabilità, nella sua generosità c'è un'immensa grandezza. Grazie ad Andrea, fratello e amico nelle tante battaglie per le premure e le delicate attenzioni. Grazie a Gavina per i passi vissuti insieme che hanno impreziosito il cammino comune. Grazie a Michaela per la sua costante e disarmante disponibilità.

Grazie alla mia famiglia che con discrezione, orgoglio e fiducia ha custodito i miei passi a volte incomprensibili.

Lo studio e la ricerca hanno come alleati la costanza, la pazienza, una sana curiosità e quel trepidare che ti regala il guizzo di osare su strade che ancora non

immagini. Capita quindi che, mentre ti cimenti in nuove tecnologie, tu possa scoprire che la ricerca più importante, che comprende tutte le altre, è la ricerca di sé e della Verità. Le strade si aprono all'impossibile, si intessono di un Mistero inesauribile che dimora in te nonostante l'inquietudine. Si spalanca il desiderio di metterti in cammino verso l'Infinito e di lasciarsi sorprendere e avvolgere dalla Sua gratuità. Grazie allora alle Monache Agostiniane di Pennabilli con cui condivido, fra sorelle e amiche, questa ricerca in comunione con loro e con l'umanità. Grazie per aver prudentemente e discretamente atteso i miei passi di libertà. È bello scrivere qua, nella nostra casa, le ultime righe di questo lavoro di tesi. Si aprono nuove pagine e nuovi scenari. Non so quali scoperte si potranno fare insieme. So che la ricerca continua. Ed è Bellezza. E stupore. Grazie!

Chapter 1

Introduction

The blockchain is one of the most interesting technologies developed in recent years and it is known as the first native digital medium for value, as well as Internet was the first digital medium for information. Blockchain is a ledger, a register, a decentralized and public shared database, reachable by accessing a peer-to-peer network. Each unit of the register represents a block. The blocks are linked together in the same order in which they were created and are connected using cryptographic algorithms that make them and the information stored therein not modifiable.

In recent years, therefore, research centers, software houses and, above all, many startups started using blockchain as a new form of technological democracy, really decentralized and able to guarantee everyone the opportunity to make, verify and control, in full transparency, any type of transaction. Blockchain technology therefore allows new forms of distributed software architectures whose components do not require a central control authority. Software development needs to be supported by architectural models or blockchain oriented meta-models. The introduction of smart contracts [21], small computer programs stored inside the Ethereum[105] blockchain, modifies the typical software engineering methodologies. The distributed applications developed on blockchain technologies, can in fact follow rules different from used for the development of the normal centralized applications. For example, traditional software development models assume that all components of the technology are flexible and can be modified as wanted. The blockchain is instead based on a totally different assumption: the information stored on the blockchain and therefore the smart contracts can not be modified or deleted without compromising the integrity of the blockchain itself. Moreover the smart contracts run in an isolated environment and their results must be the same whatever node they run in. They cannot access the external world that changes with time, but can access and send messages to the blockchain itself that is immutable.

Such an innovative context entails new challenges in software engineering. Greater emphasis on specific security and testing practices, new development tools, specific modeling languages and new metrics to evaluate software quality are needed. These new metrics are required to measure complexity, communication, decentralized systems, but also resource consumption (e.g. the so-called gas in the ethereum system).

In this scenario Agile methodologies, that are suited to system whose requirements are not completely understood, or tend to change, could be a good strategy of software development for blockchain applications. Some of the characteristics of the Agile methodologies are in fact present in decentralized applications (DApps). They are very innovative applications, that must be written correctly and as fast as possible in order to be launched in the market. Agile is iterative and incremental with short iterations, and is suited to deliver quickly and often and therefore it is appropriate in the context of DApp development. Agile methods moreover are suited for small, self-organizing teams working together, as is the case for many DApp teams. One of the most interesting applications based on blockchain are the Initial Coin Offerings (ICOs). An ICO is an innovative way to raise funds and launch a startup. It is also an opportunity to take part in a project, or in a DAO (Decentralized Autonomous Organization). The use of ICOs is a global phenomenon that involves many nations and several business categories: ICOs collected over 5.2 billion dollars in 2017. The success of an ICO is based on the credibility and innovativeness of project proposals. This fund-raising tool contains however some critical issues, such as the use of tokens that have no intrinsic value and do not generate direct liquidity, and the role of investors in the startup's management. The Lean software development, an approach that maximizes the value given to the customer, and aims to eliminate waste and optimize across organizations, could be helpful to face these critical aspects.

The ICOs and, more generally the decentralized applications are based on smart contracts. By means of blockchain explorers like Etherscan.io, it is possible find the deployed smart contracts for all decentralized applications. Considering the technological breakthrough introduced by blockchain it is important to know characteristics and measures and quality parameters of the development process and its phases, as well as metrics to help ensure that the development process is under control to meet the product's quality goals. This thesis analyzes the software development process of blockchain-based applications. In particular this work wants to study the following aspects.

- Understanding the main characteristics of ICOs, investigate software engineering activities related to ICOs, recognize the ICOs developed using Agile methods and make a comparison between the characteristics of

ICOs and those of Agile ICOs. In addition, we perform a deeper analysis of Agile ICOs concerning project planning, software development and code quality.

- Evaluating the lean startup approach as a methodology for the implementation of an ICO based on the collaboration between all stakeholders and founded on a continuous iteration process that allows investors to be an integral part in the startup's development and therefore to interact continuously with the executive team and product development team.

- Analizyng smart contracts deployed on the Ethereum blockchain performing a comprehensive exploratory study. The objective of the analysis is to provide empirical results about smart contracts features, their interaction with the blockchain, the role of the development community, and the source code characteristics.

- Presenting some applications based on blockchain developed using Agile methodologies.

1.1 Thesis overview

The thesis is organized as follows.

Chapter 2 describes blockchain technology and its evolution over time. It provides the essential elements for the understanding of the functioning of the Initial Coin Offerings (ICOs) as a way, as an opportunity to take part in a software project, in a Decentralized autonomous organization (DAO). It introduces a particular typology of software programs called smart contracts the EVM on the Ethereum blockchain environment, called *Solidity* whose main characteristics are described in next sections.

Chapter 3 evaluates the lean startup approach as a methodology for the implementation of an ICO based on the collaboration between all stakeholders and founded on a continuous iteration process that allows investors to be an integral part in the software startup's development and therefore to interact continuously with the executive team and product development team.

Chapter 4 aims to understand if and how agile practices are used in ICOs in response to such high technological, process and market variability. To do this, we take into account the principles of Agile Software Development and we study if and how Agile methodologies and practices are used in the development of ICOs. It conduces an analysis of smart contracts source codes of the Agile ICOs in terms of code metrics and use of test tools. It aims also to provide a

deep analysis of the Agile ICOs in terms of their project planning and software development.

Chapter 5 analyses software features and metrics of smart contracts, in order to measure progress and performance and to provide food for thought for improvement of these software artifacts. We performed an empirical study collecting the dataset of all smart contracts source codes available from Etherscan.io up to December 2017. Moreover it computed several software metrics on the entire dataset and we identified the twenty most used smart contracts, representing a reduced set on which we performed a systematic and more detailed analysis, in terms of both functionality and development history.

Chapter 6 describes some applications we designed and developed with Agile methodologies.

Chapter 7 finally draws the conclusions of this thesis highlighting results and contributions and presenting futures works.

Chapter 2

Blockchain technology

Blockchain technology is a data structure of the Decentralized Ledger Technology (DLT) family that generates trust and ensures data integrity without the need of a trusted third part. A blockchain is shared in the nodes of a peer-to-peer network and each node stores a copy of it. Data recorded inside the blockchain is obtained through transactions. From a general and high-level point of view, a transaction is a valid message between two authorized accounts. In particular, each blockchain implementation includes validation algorithms, message protocols, and a data-upgrading strategy. Nowadays, the first and most important blockchain implementation is the bitcoin system [64]. This system introduced the blockchain as the solution to the "double spending" problem in digital money transfer.

In this blockchain, transaction data is organized in blocks. Each block is digitally signed (by means of a hashing operation) and linked to the previous one. This makes data immutable and defends integrity. Block creation is a task entrusted to network nodes that listen for transaction requests waiting to be validated. Transactions are composed of a list of sender accounts, a list of receiver accounts, and of an amount of digital coins (called bitcoins). The transaction is valid only if each sender wants to transfer coins that he/she has previously received (by means a transaction already recorded in the blockchain). In bitcoins, the block creation passes through a proof-of-work, a cryptographic puzzle for which the difficulty is proportional to the total computing power of the network. Those which first solve the cryptographic puzzle are called "miners" and obtain a prize in bitcoins, that can be spent with a transaction.

The second step in DLT evolution consists in the advent of the blockchain-based Decentralized Platform for Applications. In this case, the most important implementation is the Ethereum system [17]. Ethereum introduced a new typology of account, a robotic account that lives inside the blockchain. This type of account is known as a contract (or smart contract, as coined by Szabo [94]

in 1997). Like normal accounts, a contract is able to both receive and send transactions. In addition, a contract contains a computer program that makes it able to compute and store data automatically.

In particular, smart contract programs contain a set of functions that other accounts can call by means of a specific transaction. Indeed, Ethereum transactions are messages that concern not only coin transfer.

The Ethereum blockchain is a state machine and each change in stored data (i.e., account balances, smart contract data, etc.) passes through a state transition. A new state depends on the data recorded in the previous state and on transaction bodies.

Because of the energy effort of network nodes, each blokchain operation has a cost. In the bitcoin system, users can pay a voluntary fee to the network for each transaction. In the Ethereum system, the fee mechanism revolves around the gas. In particular, the execution of each transaction and of each smart contract instruction requires a certain amount of gas. Operations have different gas costs. For example, storing data is more expensive than executing an arithmetic sum. Conversely, reading stored data is free. The gas price is not fixed. Users can decide how much pay it. However, users have to take in account that if the chosen gas price is too low, the network can ignore their requests.

Smart contracts purposes range from crowdfunding campaigns and money control to supply chain management and transparency [6]. They also allow the creation of decentralized organizations in which rules are written inside the code, in a programming language. Permission management, data availability, and validation and checking operations are typical features that smart contracts can provide through the decentralized platform.

2.1 Initial Coin Offerings

We can describe an ICO both as a way, not regulated by an authority, to raise funds and launch a startup, and as an opportunity to take part in a project, in a DAO or even in an economic system.

2.1.1 The main characteristics of ICOs

The idea of ICO is very similar to the well-known concept of Initial Public Offering (IPO), where a company decides to place its shares on the stock exchange, to open its capital to new shareholders. In this way, new listed companies enter the stock market and consequently increase their capital. We can therefore define ICOs as investments that provide "crypto objects" to investors. These are commonly named *tokens*. Tokens are also considered to be coins offered during

an ICO, and as such they can be considered equivalent to the shares purchased under an IPO. Note also that the vast majority of ICOs issue tokens in exchange for cryptocurrencies convertible into real money; this allows investors to access the functionality of a particular project. Moreover, ICOs in general remain open for a period of a few weeks, up to a maximum of one or two months. In the following we indicate the main features of an ICO.

- ICO prices are set by the creators of the startup or by the person who designed the project;
- the investor who owns the tokens issued by a startup in the phase of capital raising does not always have the right to express an opinion or to be part of decisions about the project, even if it remains one of the available options;
- the first investors will probably have greater advantages included in their tokens as incentives. The creators of a startup, to thank investors and to improve their loyalty, often offers them a variable bonus percentage that is proportional to the amount of cryptocurrency that the investor chooses to put in that token, and then in that startup;
- after the conclusion of an ICO, its tokens are traded on some cryptocurrency exchange, which is a website where digital currencies can be traded against each others, and against legal money, so that they can be traded very soon with respect to other kinds of startup financing;
- the startups that collect capital through ICOs are not subject to taxation (at least by now).

2.1.2 How does an ICO work?

A startup initiates the ICO process by establishing, first of all, three aspects: the blockchain underlying the system, its protocols and rules. Subsequently the ICO's creators define and make available the tokens that will be sold. In addition, in order to evoke the greatest possible interest, startups announce their ICO in several ways. The most used are represented by social media and ICO websites in which ICO's creators describe their business project.

The new token issued during the ICO will also need to be traded in an exchange, in a similar way of trading in the stock exchange after an Initial Public Offering (IPO). ICOs active or about to be activated can be traced through different websites, whereas the sale of tokens against cryptocurrencies is performed through selected exchange platforms (the most famous being Bittrex, Kraken, Poloniex, Livecoin, SpaceBTC and Bitlish). In order to buy tokens, the investors

must possess a virtual wallet holding the needed cryptocurrencies, that can in turn be bought in an exchange using traditional money. Investors can buy ICO tokens very easily and directly, starting from the startup website. So, investors eager to invest in promising startups through their ICOs have to explore thoroughly the various exchange platforms and the social media dealing with ICOs. In this way, they find and evaluate the active and forthcoming ICOs, and can make their choice, buying the chosen tokens.

2.2 Smart contracts and Solidity

Our analysis takes into account a particular typology of software programs called smart contracts, written in a programming language specific for running in the EVM on the Ethereum blockchain environment, called *Solidity*.

2.2.1 Smart Contracts

A smart contract is a computer program that aims to implement a logical sequence of steps according to some clauses and rules. In a conceptual level, smart contracts consist of three parts [94]:

- the code of a program that becomes the expression of a contractual logic;
- the set of messages which the program can receive, and which represent the events that activate the contract;
- the set of methods which activate the reactions foreseen by the contractual logic.

Smart Contracts run in a blockchain where contract transactions can be permanently recorded in a transparent environment and are immutable. Once the smart contract is deployed into the blockchain its code cannot be modified and the clauses introduced by the parties in the contract will obligatorily be respected because of the computational nature of the system, as for the execution of any software program.

There are different blockchain able to run programs implementing smart contracts. Even the Bitcoin blockchain supports a limited amount of software code that can be deployed using transaction in a blockchain address [5]. Other examples are Hyperledger Fabric [18], the Qtum platform [27] and the Achain platform [26].

Among all, the most popular is the Ethereum platform, the first blockchain specifically conceived to run smart contracts, and the most popular programming language for smart contracts is “Solidity”. In this platform it is possible to

read some information that characterize each Ethereum transaction. In particular, smart contracts are activated by messages, that are ethereum transactions executed by the message sender.

Currently the Ethereum platform hosts the large majority of smart contracts.

The process to deploy a smart contract into the Ethereum blockchain is composed of three phases. The first phase consists in the code writing in solidity language; the second consists in the code compiling, that can be executed in a local environment (i.e. the remix environment[1]) to convert the script in the EVM *bytecode* [92]; and finally the last phase consists in creating a transaction in the blockchain, that actually deploys the contract. At the moment of the deployment, the blockchain assigns an address to the smart contract. Accessing to that address it is possible to visualize some useful data of the smart contract like its address, its balance, and its Application Binary Interface (ABI).

In order to avoid the possibility of the EVM overload, the execution of smart contract functions (when they involve a changing of the blockchain data) lead to a cost in terms of cryptocurrency. In particular, to each low level operation is associated a computational cost (defined in units of *Gas*)[105]. The price in Ether of a unit of Gas is not fixed but follows the free market rules.

2.2.2 Solidity

Solidity is a *contract-oriented*, high-level language whose definition was influenced by Object Oriented (OO) languages like Python, C++, and especially by JavaScript.

It is a typed programming language and supports traditional types such as integer, string, array, as well as structures, associative arrays, and enumerations. Moreover, Solidity has a specific type, the *address*, that identifies users and other contracts. Each contract variable can be interpreted as a record of a database which can be queried and modified by calling functions of the code that manages the database. The set of variables and their associated values represent the state of the contract. Smart Contracts functions can be externally called by means of blockchain transactions. In order to make the development more modular, specific function *modifiers* can be defined and associated to different functions, for instance to perform checks in a declarative way.

Recently, appeared some mainstream integrated development environments (IDE's) supporting solidity code development, as for example IntelliJ IDEA, developed by JetBrains. We used the Intellij-Solidity plugin[2] to read and compare contracts source codes.

[1] Available online at https://remix.ethereum.org/

[2] https://plugins.jetbrains.com/plugin/9475-intellij-solidity

On the contrary there is still a lack of tools specific for analyzing Solidity source code metrics, so that we recurred to the similarity of Solidity with Javascript and C++ for the analysis of Solidity source codes metrics. In fact, an exploratory evaluation of the features of smart contracts source codes can be performed using metrics and methodologies obtained adapting the existing ones.

Chapter 3

ICOs and Lean Startup Methodology

An Initial Coin Offering (ICO) is an innovative way to raise funds and launch a startup. It is also an opportunity to take part in a project, or in a DAO (Decentralized Autonomous Organization). The use of ICO is a global phenomenon that involves many nations and several business categories: ICOs collected over 5.2 billion dollars only in 2017. The success of an ICO is based on the credibility and innovativeness of project proposals.

ICOs are the new trend in the cryptocurrencies field. The technology to create a new cryptocurrency is cheap: in a short time and without large investments any company can present itself to the market with its fundraising and the related token. With these premises, an ICO is the most innovative solution to finance themselves outside the traditional channels, especially for startups.

In fact, a good source of funding is essential to launch a startup. At first, it is possible to apply for local or international institutional funding, that generally does not provide for the repayment of the grant, but which also involves very long waiting times and a very complex bureaucracy. Even traditional funding operations that involve venture capitalists (VCs) or business angels have long waiting times. The risk is also that a traditional VC could acquire a high percentage of shares and become prevailing in the key decisions of the company. On the other hand, a typical fundraiser needs a good marketing campaign, with many supporters participating with small amounts of money. Even a financial partner can be very risky, especially if the partner is a very experienced person who want to steal the business idea. The creation of an ICO therefore represents a valid way to collect initial capital for startups.

The success of an ICO is fundamentally based on three key elements: reliability of team members, evaluation of the project and of its white paper, and comments from other investors. Analyzing these three factors, investors should be able to answer two simple questions: "What novelty and what value does this project bring to the world?" And consequently: "Does it make sense to invest

in this project?" The questions arising from this premise are therefore the following: "Can an investor monitor the evolution of the startup based on an ICO and actively collaborate on the success of this startup?". This work evaluates the lean startup approach as a methodology for the implementation of an ICO based on the collaboration between all the stakeholders involved and founded on a continuous iteration process that allows investors to be an integral part in the startup's development and therefore to interact continuously with the executive team and product development team. The chapter is structured as follows. Section 3.1 presents the related works. Section 3.2 proposes an ICOs overview which includes phenomenon statistics, a taxonomy, and a description of critical aspects. In section 3.3 we show ICOs as Lean Startups and discuss about some study cases. Finally, in section 3.4 we present the conclusions.

3.1 Background

To date, because of its novelty, literature hardly addresses this topic. In October 2017 Flool et al.[37], analyzing the history of the blockchain technology and of cryptocurrencies, presented the ICO phenomenon as a realization of an anarcho-capitalists system, made trusty by the underlying technology. Authors reported results of their studies related to key elements which make *good* an ICO, stating that the *crucial element* is trust (generated by the technology and by the ICO features). The initial coin offering process has also been studied by Kaal et. al[48] in November 2017. They described ICOs and the related environment. In addition, they underline the similarities and differences between ICOs and the IPOs of the stocks market, focusing the attention on risks and bad practices which could compromise investments and the general trust in the ICO system. Trust creation can not ignore the legal aspects of the ICO funding mechanism. Barsan[4] gave particular attention to this aspect. He highlights regulator organisms are well equipped to apply existing regulation to virtual currencies and ICOs. He also provides a legal classification of ICOs, distinguishing the currency-like tokens from the security-like ones. In order to evaluate risks and actual value of an ICO, Venegas [100] proposed an empirical approach based on the correlation analysis of the network activity. Adhami et al. [3] too, focused the attention on empirical evaluation of ICOs, classifying them in accomplished and failed. Very recently, Fenu et al. analyzed 1387 ICOs, assessing the factors that were critical to theirs success [35] using a statistical analysis, whereas Hartmann et al. analysed 28 ICO websites to reveal the state of the practice in terms of ICO evaluation [42].

ICOs are a startup funding method that has similarity with the crowdfunding. In 2014, Mollick presented results of his empirical analyses on the dynamics of crowdfunding [58] and factors that influence the performances. Recently, Wang

et al. studied the effects of the interaction between creators and backers on crowdfunding success [102], basing on the sentiment analysis of the comments. On the other hand, several works focused the attention on the lean startup development and their funding opportunity. Poppendieck et al. described lean startup concept and its key elements in their tutorial [77] in 2012. In 2013, Bosch et al. proposed a early stage startup development framework [14] in which all stages which a startup team has to accomplish during the first phases of their business initiative, starting from the idea generation to the validation of the Minimum Viable Product (MVP), are described.

3.2 ICOs: overview

3.2.1 Overview of ICOs phenomenon statistics

In this section, in order to figure out the dimension of the ICO phenomenon, we provide some statistics. We analyzed from the 1th of December up to the 12th January 2017 specialized websites[1] which collect ICOs and their details. We can state that 2017 was the year of ICOs. According to *icowatchlist.com* data, during that year ICO raised over 3.3 billion dollars. By comparison, in 2016 ICOs raised a total of 106 million dollars[2]. Exploring ICOs we realize that they represent a global phenomenon. In particular, 88 nations presented at least one ICO. Despite this reality, it must be said that four countries raised over the 54% of the total. They are Switzerland (21%), United States (19.1%), Israel (7.6%) and Singapore (6.7%). As regards the number of ICOs per nation, USA, Russia, UK and Singapore are the most active nations. Table 3.1, summarizes the first ten nations per total raised amount.

By the end of 2017 the *icobench.com* website listed 1259 ICOs, referring to a heterogeneous set of projects. About 50% of ICO projects are ICOs already ended. 33% are ongoing ICO and the remaining 17% are upcoming ICOs.

In order to understand the ICO trend we decided to categorize them by industrial sector. In this regard, we pinpointed all relevant data from the aforementioned ICO websites. However, each website presents information by con-

[1] ICO data are extracted from the following websites:
http://www.icobench.com,
http://www.coinschedule.com,
http://www.icowatchlist.com,
http://www.coingecko.com,
http://www.icoalert.com,
http://www.icostats.com,
http://www.icodrops.com

[2] https://www.coindesk.com/2016-ico-blockchain-replace-traditional-vc/

Country	Total Raised	% of Total	ICO Projects
Switzerland	463,775,825	21.02%	51
United States	421,402,100	19.10%	248
Israel	167,370,000	7.59%	15
Singapore	148,780,000	6.74%	79
Russian Federation	81,174,361	3.68%	202
France	78,050,000	3.54%	15
United Kingdom	61,050,000	2.77%	106
Serbia	53,070,000	2.41%	4
Gibraltar	27,480,000	1.25%	14
Spain	26,660,000	1.21%	10

Table 3.1: The first ten nation involved in the ICO phenomenon spread

sidering different criteria and perspective, and only few of them propose a classification. In general, an ICO is described by: name, logo, token, start date, end date, description, website, white paper, social links, accepted cryptocurrency, development platform, ICO price, min and max target amount to raise, country, upcoming, ongoing, ended, and so on.

Merging and cross-referencing the analyzed data, we built the taxonomy shown in Table 3.2. To identify the taxonomy dimensions, we made a list of categories already identified by the various websites, using as labels the most used ones. In some cases we joined some of them. In total, we identified 24 dimensions which represent the category of industrial ICO sectors. Afterward, we populated the taxonomy considering both the number of projects developed and the amount of funds raised in each specific sector. In this way, we were able to understand the ICO sector trend and the investors' interest towards projects. We represent results in percentage terms. Table 3.2 shows that projects in *Blockchain Platform & Services* are the most popular: 20% of projects has been launched in this sector. We can also see that these projects are the most heavily funded, having received 25% of the total raised amount. The second most funded category is *Network/Communication/Storage* with 20% of funds raised. Therefore, we notice that nearly half of all investors are interested in the two above mentioned categories.

Since ICO funding is an ever changing phenomenon, the proposed classification should not be considered as definitive, but as a starting point on a path toward a more exhaustive categorization.

We show in Table 3.3 the ten most funded ICOs in 2017, reporting also their category, according to the taxonomy shown in Table 3.2.

Category	% Projects per Category	% Fund raised per Category
Blockchain Platform&Services	20,00%	25,00%
Finance	12,00%	7,00%
Trading & Investing	10,00%	8,50%
Commerce/Retail	8,00%	3,00%
Payments/Wallets/Cryptocurrency	8,00%	9,00%
Gaming/VR	6,00%	4,00%
Funding/VC	5,00%	1,20%
Network/Communication/Storage	5,00%	20,00%
Betting/Gambling	3,00%	2,00%
Data/Artificial Intelligence/Machine Learning	3,00%	2,00%
Media/Content	3,00%	0,50%
Healthcare	2,00%	7,00%
Real estate	2,00%	0,80%
Security/Identity	2,00%	2,00%
Social Network	2,00%	3,00%
Energy/Utilities	1,50%	0,40%
Education	1,00%	0,01%
Industry/Logistics	1,00%	0,20%
Insurance	1,00%	0,20%
Mining	1,00%	0,30%
Transportation	0,70%	0,20%
Tourism	0,40%	0,10%
Legal	0,05%	0,40%
Other	2,35%	3,19%

Table 3.2: An industrial sector taxonomy of ICOs

ICO Dataset The dataset has been populated using the API provided by icobench.com website[3]. On date 16th January 2018, we updated the ICO dataset, holding on that date information regarding 1542 ICOs. In particular, we used the POST request

https://icobench.com/api/v1/ico/{id}

where {id} is a progressive number that uniquely identifies an ICO. This request provided comprehensive information about each ICO stored in the website database. The data were extracted using a script written in R language, which includes the *httr* [4] library developed by Wickham.

In order to analyze a temporally homogeneous set of ICOs, we selected the ICOs started and ended during 2017. This set includes 690 ICOs. The sum

[3]https://github.com/ICObench/data-api for references
[4]https://cran.r-project.org/web/packages/httr/httr.pdf

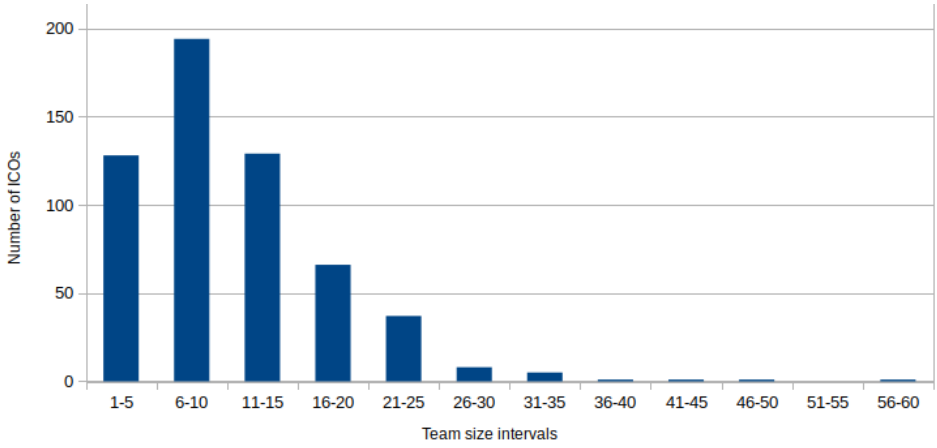
Name	Total Raised (USD M.)	Category	Start Date	Duration	Team (Advisors)	Nation
HDAC	258	BC Platform &Services	27/11/17	25	17 (7)	Switzerland
FileCoin	257	Network/ Communication/ Storage	10/08/17	31	13 (0)	USA
Tezos	232	BC Platform &Services	01/07/17	12	11 (3)	USA
EOS	185	BC Platform &Services	11/06/17	15	4 (0)	USA
Paragon Coin	183	BC Platform &Sevices	15/09/17	30	12 (0)	Russia
Sirin Lab	158	Commerce/ Retail	12/12/17	14	42 (7)	Switzerland
Bancor	153	BC Platform &Services	12/06/17	31	8 (10+5)	Israel
Polkadot	145	BC Platform &Services	15/10/17	12	NA	Singapore
QASH	105	Trading &Investing	606/11/17	02	9 (9)	Singapore
Status	102	Other	20/06/17	31	7 (0)	Switzerland

Table 3.3: The ten most important ICOs of 2017

of the raised amounts by these ICOs during 2017 is about 5.20 billion dollars. Considering only ICOs with non-zero raised amount, the average value of these amounts is about 17.21 million dollars, whereas the median is 7.30 million dollars. To focus the attention on the magnitude of the raised amounts, we considered the raised amount in log_{10} scale. This value is included in the range 2-9. In addition, to describe each ended ICO, we extracted four static key features from the dataset: the ICO Duration in days, the Rate (a rating score provided by icobench.com that summarize the overall quality of the ICO), the total Team size, the number of advisors, and the total raised amount. We then excluded 119 ICOs having zero team members or whose total raised amount was not-available. We investigated if and how key features influence the final raised amount computing, at first, the correlation factor between each key element and the raised amount for each ICO. In table 3.4 we summarize the four key features and their values. It is interesting to note that the ICO duration and the raised amount have a negative correlation. We focused the attention on the team size, considering all people registered in the dataset, including developers, advisors and supporters of the ICO. The average number of team members is 10.9, with a standard deviation equal to 7.1. In fig. 3.1 the distribution of the team size is provided.

The correlation between the time size and the raised amount of the ICO in log_{10} scale is equal to 0.32. To investigate the relation between team size and

Figure 3.1: Team size distribution

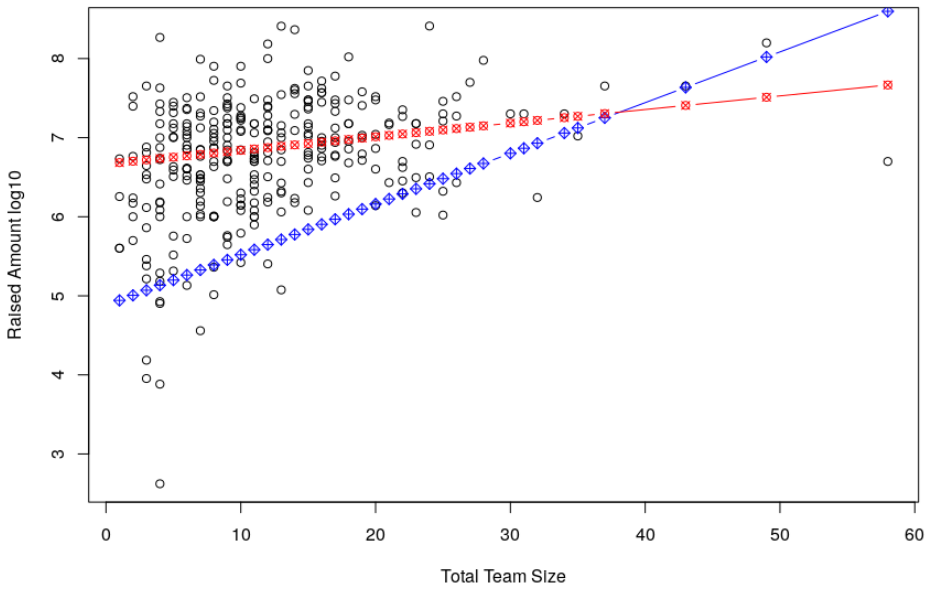


	Duration	Team Size	Advisors	Rating
Max Value	112 days	58	17	4.9
Average	29.65 days	10.87	2.17	3.18
Standard Deviation	18.09	7.05	3.37	0.80
Correlation*	-0.28	0.32	0.22	0.34

Table 3.4: Summary of the four key elements selected to investigate how they affect the total raised amount in terms of correlation coefficient.

ICO success, we computed the average raised amount per team size (AR), and the minimum raised amount per team size (MR). Results show that both these data are more correlated with the team size than the original data. The AR and the team size have a correlation coefficient equal to 0.51, whereas MR and team size have a correlation coefficient equal to 0.76. To describe the proportionality of these results with the team size, we computed the linear regression $y = m(x) + q$, where and x is the team size and y is the log_{10} of the amount. The AR function has parameters m = 0.017 (with standard error 0.11) and q = 6.67 (with standard error 0.12). The MR function has parameters m = 0.064 (with standard error 0.009) and q = 4.88 (with standard error 0.23) Fig. 3.2 shows these two functions. Blue diamond dots represent the linear regression function of the minimum raised amount per team size. Red squared dots represent the linear regression function of the average raised amount per team size.

Figure 3.2: Raised amount per team size.



3.2.2 ICOs' critical aspects

An ICO is based on the assumption that investors will buy the ICO token in order to obtain a future return on investment (ROI). In particular, an investor will buy the token at the ICO selling price with the aim of selling it after ICO ends, at a higher price. For this reason, an ICO must be organized to be attractive to investors. In short, an ICO must first of all be credible. ICO general information, the product information, the team composition, and the vision of the proposing startup, are key elements in the eyes of investors during the evaluation of investment opportunity.

In traditional VC rounds, investors acquire an ownership percentage, after a business evaluation. Conversely, ICO investors do not enter in the business ownership. Investors aim to obtain a profit on what they are buying, i.e. the token. Actually, token are something that will allow the access to some services after the startup idea will be realized. Investors wish to buy tokens whose value will increase after the startup business will launch its product. The first investment performance indicator is the ROI. As of writing, the vast majority of closed ICOs are characterized by a positive ROI, and several cases present a very high increase of the token value (see for instance the ROI of Stratis and NEO, characterized by a return on January 2018, greater than one thousand percent!). In few cases, investors lost theirs money, as in the case of Paragon ICO, one of the ICOs which raised most money, that currently has a negative

ROI (-44%). Another important aspect is that ICO investment can be liquidated just by selling the bought tokens (the equivalent of the exit operation in venture capital). Tokens, however, are not directly payable in fiat currency. They have to be sold in specialized exchange websites, at the market price. This price is typically highly volatile, thus presenting a high risk. Summarizing, critical aspects of ICOs are:

- ICO project must be credible for investors (feasibility of the project, etc.);
- Token should have an intrinsic value: an ICO does not generate direct liquidity, but the value is given by its token;
- Risk of low or negative ROI;
- The investors, who are used to risk, play the role of the controller.
- The ICO tool is highly innovative: it is not possible to carry out historical analyzes or analytical forecasts. The key element of success is based on management flexibility.

The critical aspects of an ICO also can partially or totally match the typical crucial aspects of a startup firm, which operates in conditions of extreme vulnerability and faces many challenges. According to several authors [12] [81], the high failure rate of startups can be mainly attributed to the way in which the startup is managed and not only to typical market factors such as competition. The main risk of an ICO, and consequently of a startup [81], is therefore to spend time, money and energy in the development of a service or a product which people are not interested in.

What would be needed to reduce the identified problems?

After highlighting the limitations of an ICO and the challenges that a startup faces, in the followings we point out what are the elements that can contribute to the success of an ICO.

- Investor involvement not only in the fundraising phase, but also in the subsequent phases. Business risk is therefore shared, and investors are called upon to invest only in projects they really believe in and where they can make a significant contribution also in terms of ideas. In this way, the risk of speculation is limited.
- The design idea must be manageable through a token.
- The business model must be feasible and therefore concrete, sufficiently detailed, but at the same time must be flexible.

- A complex project can be divided into phases: the first steps, if the startup project is innovative, are the most critical.
- It is good to test the project idea right away by analyzing feedback from a small number of users.

The elements highlighted above, designed to increase the probability of success of a startup, are supported by numerous studies [11] [61][89] [57], and are typical of lean startup methodology in which the focus is on the customer, the decision-making process is based on the facts and pivoting and agile/lean thinking is fundamental.

3.3 ICOs as Lean Startups

In order to create a successful ICO it could be helpful the use of The Lean methodology and Value Proposition Canvas. These strategies in fact can be used to ensure that the market of designed product actually exists and that the idea can be considered good. In the context of an ICO, moreover, the activities can not be exclusively focused on reaching a solution, but it is necessary to examine the problem in detail before proceeding with any elaboration [62]. At the start of an ICO, in fact, both the problem and the solution are not generally well understood by investors and often also by the development team. In this context of uncertainty, the typical elements of lean startup methodology such as prototyping, execution of experiments [60], validation of initial business hypotheses and continuous learning can be easily applied as elements of greater security [81] [11]. We outline below some aspects of this methodology that can be easily applied to the management of an ICO.

1. **The Pivot**. It is a change of direction during the development of the project. All changes are based on what is learnt in the previous stages. If you reduce the time between the pivots you increase chances of success and you spend less money. The pivot is connected to the concept of feedback cycle formed by the three phases Build-Measure-Learn (BML) and to the Minimum Viable Product (MVP). A chance of success is proportional to the minimum time it takes to get through the BML loop, and then to the minimum time between pivots. With this approach, you start with an idea of product or startup, and the end result can be something else. The direct feedback and the tests by potential users of the product could therefore induce to change market segment, customer type, costs, partners, strategies, while maintaining the same vision of the startup. In an ICO, given that initial investors back the team more than the idea, the pivoting should not be a problem.

2. **Validated learning**. This process should apply to an ICO that works in an area of extreme uncertainty in order to verify the progress of the project [81]. A positive marker of an ICO in fact cannot be just the revenue. An iterative validated learning process allows an evaluation of the hypothesis (that could be valid or invalid) by running experiments and by the analysis of information that leads to the formulation of new ideas. Identifying a very clear use case that requires the decentralized approach typical of blockchain technology could be the fist step of this process.

3. **Testing**. The Lean startup methodology highlights the importance of test cycles. It allows to verify concretely if the need really exists, if it is perceived by the identified target, and if it is strong enough to be satisfied. Testing speeds up learning and create a competitive value. When a stakeholder analyses an ICO, one of the most relevant questions is if the idea and the team are good in that specific context. According to Lean Startup methodology, the success of an ICO could be connected to testing the product in each phase, to verify the need and the use of the product. In accordance with the decentralized nature of the blockchain, the use of tests applied in a decentralized way can be useful.

3.3.1 Three different case studies

We aim to analyze the ICO phenomenon based on the Lean startup methodology. We examined those ICOs in which the proposer team states explicitly that a Lean startup approach is used. We examined three different case studies, each with a different application of this methodology. The first ICO uses the principles of modularity, simplicity and scalability typical of Lean startup methodology to develop a platform to build decentralized applications; the second, according to Lean startup methodology, focuses its attention on feedback from users. Finally, the third ICO designs a platform that, using the Lean startup methodology, aims to address the problem of lack of interaction between investors and development team of ICOs.

Lisk - Blockchain Application Platform. Lisk[5] is one of the oldest ICOs and is a Lean startup. It was registered in Switzerland by Max Kordek, Oliver Beddows and Guido Schmitz-Krummacher on 22 February 2016 and raised money in bitcoins. The platform was born from a fork of Crypti's blockchain and its price, as well as that of most tokens, peaked in 2017. At present, Lisk is one of the most solid startups financed by an ICO. Lisk has raised over 14,000 Bitcoins or about $ 9 million at the time of the campaign, and has now a market cap of more than one

[5]https://lisk.io/

$ billion. Every month, on the ICO website a monthly report is published on the activities of the startup and on its financial evolution. Lisk spends around 76,000 CHF for its running costs per month. The daily volume traded on exchanges is of several tens of million CHF. Lisk is based on the principles of modularity, simplicity and scalability typical of Lean startup methodology, and provides a platform for the construction and distribution of decentralized apps. Developers have the ability to build decentralized applications (DApp) with some mainstream programming languages such as JavaScript and Node.js. Therefore, developers do not need to learn the Solidity language, as in the Ethereum blockchain. Unlike what happens to the DApp on Ethereum, the applications developed on Lisk will be built on a parallel blockchain (sidechain), so as not to create problems for the main blockchain, especially in the case of bugs. A modular SDK allows developers to take advantage of a series of libraries, modules, algorithms and third-party tools that make the development environment user-friendly and customizable, and therefore suitable for creating blockchain applications.

Galactikka - A social networking project. Galactikka[6] is another ICO in which the proposing team declares to use the Lean startup methodology. Galactikka is an innovative social network that allows authors to promote their original content and to earn money with their posts, photos and video materials when they are published and shared. The platform integrates a community, blogs and a system for Q&A. The goal of Galactikka is therefore to help amateur authors to make themselves known and to profit from their creativity. Galactikka was designed in Russia, so its main language is Russian. Galactikka uses the approach of phases and interactions typical of the Lean Startup methodology, giving great value to the feedback provided by the users. For this reason, in the first instance, the team prefers to use only the Russian language, because it is the language best known to them. In the first phase also the contents inserted by the users will have to be in Russian language. According to the Lean startup method, it is in fact convenient to test the application on a small group of users. In this way, the development team intends to concentrate initially on a limited user target, whose language is fully understood, in order to avoid wasting energy and resources on a global audience that is too large. In this way, it is possible to increase the speed of project development.

doGood - Blockchain-fueled social platform for lean startup. doGood[7] aims to get through one of the main limitations of an ICO: the lack of tools that can allow investors to provide feedback during the development phases of the project idea related to a startup. With a Lean startup approach, doGood wants

[6] http://galactikka.com/
[7] https://dogood.io/

to offer funders the opportunity to monitor the team's progress and to provide direct guidance at all stages of the project. The Lean startup methodology is needed, given the uncertainty in the evolution of the project, and in order to ensure that the proponent team provides the promised results, thus determining an increase in the value of the token. Using the Lean startup methodology, the doGood ICO seeks to improve interactions between the team and other stakeholders. Smart contracts help decision making and reduce the cost and the time-to-market. In this way, it is possible to increase token value and reduce the risks involved in these ventures. doGood is therefore a web platform that stems from the idea that it is necessary to improve interaction between people by proposing a democratic method to solve complex problems based on open innovation principles, design thinking and especially on Lean startup philosophies. Every person involved in the project, and therefore also every investor, in a decentralized way and from any part of the world can indeed perform a series of activities and be totally protagonist of the success of the startup. Incentives and governance system are based on the Ethereum blockchain, aiming to a better identification of solutions to problems, and to the ability of proposing arrangements in a decentralized and large-scale manner. The system is designed with the hybrid use of two architectural paradigms: a client-server architecture (centralized), and a client-server architecture based on blockchain technology (decentralized). This ICO merges the use of smart contracts with the Lean startup methodology, gaining a double advantage for investors – they have greater visibility within the project and the related startup, and can provide relevant and appropriate information on the construction of the system. The token is called just GOOD. A smart contract system, in application of the Lean startup methodology, is connected to the various decision-making milestones of the project's evolution. A GOOD token is assigned to a project in exchange for the VOTE tokens. VOTE-type tokens are used by investors, proportional to the amount of GOOD Token held, to be able to cast their votes in the decision-making stages of the project. In this way, the Product Development Team can understand unequivocally, as a result of a democratic operation, what are the wishes of the investors. The use of the blockchain is useful for its intrinsic properties that guarantee authenticity and security of the vote of the stakeholders.

3.4 Conclusions

Startups based on a ICO are playing a fundamental role in creating the market of blockchain applications. An ICO can be a valuable tool for those teams that want to quickly obtain financing, but it also has several limitations, due essentially to the immaturity of the technological system and to the risk of financial spec-

ulation. The Lean startup approach can be useful to solve some of ICO issues. The tokenization nature of an ICO proposal needs a form of sustainable and regulated token sale event, that can be built on an MVP. The concepts of *pivot* and *validated learning* can be very useful, but also the investors' goals must be taken into account. They can be directed exclusively to immediate gain and not to company growth, strategic planning or operational work. A Lean startup methodology could be useful in order to respond to a tokenization that gives rise to new business models and new products or services that must effectively address customer needs. Many iterations and the direct involvement of all the stakeholders can further improve and help to market the original idea.

The following chapter presents the development of ICOs with Agili methodologies.

Chapter 4

Initial Coin Offerings and Agile Methods

As reported in the previous chapter, an ICO is a new way to perform crowdfunding campaigns, based on blockchain technology [64, 93, 49]. It allows to finance startups working on blockchain technology or applications on a global scale, directly and without intermediaries.

With the aim of involving as many investors as possible, a startup will create and distributes tokens. The token is developed through a smart contract, a computer program running on a public blockchain [17]. Smart contracts are the base for the development of decentralized applications (DApps). Most smart contracts used for ICOs run on Ethereum blockchain, and are usually written in a programming language called Solidity.

Although startups performing ICOs share many characteristics with software startups not based on blockchain technology, several factors make the software development context of ICOs unique. We want therefore study this specificity. Consequently, the research presented in this chapter aims primarily at understanding the main software development characteristics of ICO startups, from the planning phase to the testing phase, taking into consideration also the quality of code.

To this purpose, we analyzed the whole set of ICOs gathered from ICObench[1] and registered until February 20, 2018. We focused on their main features, such as: the team size and roles in order to understand if its composition is consistent with the best known software development methodologies; the rating provided by ICObench in order to investigate the projects quality; the use of social media in order to discover if ICOs teams believe that the communication with investors and their feedback are important; some financial aspects related to motivating

[1] http://icobench.com

the people involved in an ICO.

The market of ICOs is extremely volatile and complex. ICO teams and investors should pay particular attention to the speed of changes and to the technological risks that characterize its evolution.

Furthermore, software startups in general, and even more so, startups founded through an ICO, operate in conditions of great uncertainty and the team's ability to manage changes is crucial. Software startups, therefore, need to follow the-market trends, and to adapt to ever-changing business activities and risks [23]. For this reason, in this work we investigated the use of Agile methodologies in ICOs, as a mean of managing uncertainty.

In fact, according to many studies [65, 43, 44], Agile methodologies are optimal for projects that, like ICOs, demonstrate high variability in the development process, in the abilities of the team or stakeholders, and in the technology being used. In particular, Agile development is especially appropriate for products or services providing high value for the customers, and for all involved stakeholders [43]. Considering the decentralized nature of the blockchain, an ICO can in fact have investors, customers and other stakeholders from all over the world. We aim to understand if and how agile practices are used in ICOs in response to such high technological, process and market variability. To do this, we take into account the principles of Agile Software Development and we study if and how Agile methodologies and practices are used in the development of ICOs.

First, for each ICO registered on ICObench until February 20, 2018, we carried out a textual analysis of its online available documentation, in order to detect any typical Agile keywords. In this way, we found a subset of ICOs developed with an Agile approach (for the sake of simplicity named Agile ICOs in the following). We also made a comparison between the characteristics of Agile ICOs and the properties of the whole dataset in terms of team composition, rating, financial aspects and use of social media.

In addition, we analyzed more in depth the set of Agile ICOs, specifically focusing on their roadmap and their project development. We conducted an analysis of smart contracts[52] source codes of the Agile ICOs in terms of code metrics and use of test tools.

To summarize, the goals of this chapter are understanding the main characteristics of ICOs, investigating software engineering activities related to ICOs, recognizing the ICOs developed using Agile methods, and making a comparison between the characteristics of ICOs and those of Agile ICOs. This works aims also to provide a deep analysis of the Agile ICOs in terms of their project planning, software development, and source codes.

The remaining of this chapter is organized as follows. Section 4.1 describes the most relevant related work on ICOs and on Agile methodologies. Section 4.2 presents the research method. This section describes the steps which al-

lowed to define our dataset, to find the Agile ICOs, and to perform our analysis. Section 4.3 describes the results of the analysis of the ICOs dataset, including the comparison between the results obtained for the ICOs in general, and the results obtained for the Agile ICOs. These results include the team composition, the rating, the financial aspects. Section 4.4 analyzes in depth Agile ICOs. This section includes the results of the analysis of the ICO roadmaps, of the software projects and of the smart contracts. Section 4.5 provides a discussion about the outcomes of our analysis. Finally, Section 4.6 concludes the chapter.

4.1 Background

Research literature on blockchain in general, and on ICOs in particular, is limited to the last few years.

[57] shows that the Agile and Lean Startup methodologies can be compatible and complementary. Agile methodologies drive software development, whereas the Lean Startup methodology is more oriented towards the development and management of the business and of the product. Agile methodologies were used in Sabrix, Inc [13], a startup software company that, to accommodate the pressing demands of users, exploited the urgency as the main engine for the development of the product, and allowed the startup to switch from a initial chaotic management to the correct implementation of a software product – team and product progressed simultaneously. In order to verify if agile practices are applied in software startups, [73] performed a survey involving 1526 software startups, with questions related to five agile methodologies, including regular refactoring, agile planning, frequent release and daily standup meeting. The survey was focused on the relationship between velocity and quality in agile practices, and they discovered that the software startups favored velocity over quality. [40] claims that startups run the risk of failure and of being quickly out of business, if some engineering practices are not used. They studied the software development strategies employed by startups, and point out that it is necessary to reduce time-to-market, speeding up the development of product using users' feedback. [74] provides an exploration of the state-of-art on software startup research and specifies which software engineering practices must be chosen to increase the ability of startups to survive under highly uncertain conditions. Recently, [39] proposes a unified and multidimensional framework to represent together the role, active or passive, of digital startups with respect to change, and to the level of dynamism of the environment. According to the results of this exploratory study, Lean Startup Approaches (LSA) are strongly related to the use of Agile development methodologies. In addition, startups oriented to have an active role in determining changes use the approach called Business model

innovation.

The blockchain technology is an invention that led to a high dynamism in several business areas [93]. It gave rise to the definition of a new branch of software engineering called "Blockchain-Oriented Software Engineering" [78]. According to [19], the blockchain technology and in particular the invention of digital tokens, has the potential to create a *new entrepreneurial landscape*, representing the opportunity to invest in early-stage projects, and, on the other hand, the opportunity for startups to fund their projects in a more democratic way. These opportunities represents the core of the Initial Coin Offerings (ICOs) phenomenon, subject of this study.

[50] presents a model that rationalizes the use an ICO for the launch of a peer-to-peer platform that still needs to be built. This work highlights two strategies that can generate value: a coordination model among many subjects involved in a peer to peer network and the use of "popular wisdom", that is, the analysis of information available on the web and posted by users or by other stakeholders that describe the quality of the platform.

The possibility of using an ICO as a fundraising tool to finance business and technology initiatives directly and without intermediaries was analyzed by [45]. In this work, they analyzed the Lean Startup methodology as a tool to face the main critical aspects of a startup and examined some ICOs based on lean startup approach.

A first overview about ICOs was made by [35]. They examined all ICOs, published on 2017 on icobench.com website in order to evaluate their quality and software development management and to discover the features that can influence the ICO success. A similar analysis is described in [3]. According to this work, the success factors of an ICO originate in the process behind the organization of the ICO. Another success factor is the quality of the services provided to the investors who buy the tokens.

The main problem of an ICO is the capability of investors to make a distinction between a genuine fundraising activity and a scam. [42] analyzed this phenomenon, collected information from specific websites, and categorized the ICOs in order to identify their key success factors. Other issues are related to the legal aspects of an ICO. Another aspect to be managed is the possibility of changes in ICO legal regulation. The rapid explosion of the ICOs phenomenon has generated some legislative loopholes. In [29] it is pointed out the lack of clear rules for the accounting of ICO funds in the company balance sheet. In [33] it is described how ICOs are regulated in different countries. At the moment, there are no uniform ICO regulation standards.

4.2 Research Method

The phenomenon of crowdfunding based on cryptocurrency is very recent. This study is the first in literature about this subject, and has an exploratory nature. We analyzed all ICOs records on ICObench[2] until February 20, 2018. This is a free ICO rating platform that collects data about thousands of ICO. It provides the ICObench Data API[3], allowing developers to get the information stored in the platform, including ICO listings, ratings, and stats. The research approach is composed of two main parts: data collection and analysis of all ICOs registered on ICObench until February 20, 2018, and the choice, through appropriate keywords, of the ICOs to be specifically analyzed, because they are managed with an Agile approach.

4.2.1 Data Collection Steps

We divided the data acquisition process into different steps, as described below. To collect the data related to ICOs we used ICObench Data API[4] - ICObench Data API was introduced on December 12, 2017[5] - to study the list of all ICOs, the list of ICOs by search parameters and filters, the list of all ICOs ratings, all information in the ICO profile and other statistics. ICObench Data API has already been used in other major studies [59, 41, 20]. We called the extraction of this data "Step 1".

To identify which ICOs exhibit an agile approach, we defined some search keywords and searched these words within the white papers of ICOs downloaded through ICObench in Step 1. A white paper is a comprehensive technical report describing the product or service of the ICO. We define this process "Step 2"

Step 1

We collected the ICOs' data from the specialized website called *icobench.com*. To date, this website provides an useful API for developers and data scientists who want to create new applications or to analyze the ICO phenomenon[6].

In particular, this API needs user authentication and uses the HMAC method with SHA384 algorithm to authenticate the query. The data provided are in JSON format. In order to acquire the full available data, we used the POST request

[2] https://icobench.com
[3] https://github.com/ICObench/data-api
[4] application programming interface
[5] https://medium.com/@ICObench/icobench-2017-in-numbers-d987b0a280d0
[6] For the API specification see https://icobench.com/developers, accessed on July 16, 2018.

called "ICO - Profile", which we sent for all ICOs' "id" *https://icobench.com/api/v1/ico/{id|url}*

We implemented in R [80] the procedure to automatize the connection to the API, to call the request, to organize and save the ICO data in the R list data type. We collected the data of the first 1.952 ICOs recorded in the icobench.com database until February 20, 2018. We discovered that 115 of them had no available data. The data of the acquired ICOs occupy about 50 MB of memory.

Each list item describes an ICO with up to 25 named sublists, that group hundreds of named values. We focused on five sublists: team, rating, finance, dates, links. The team includes name, country, title, link to socials media, and so on for each team member. The rating provides the icobench ICO evaluation vote. The Dates includes the date related to the timing of the ICO (opening, closing). Links contains the URLs of the ICO official website and the link to the ICO white paper. Because of the importance of the ICOs' white papers, we wrote a R script to download all the available documents. We collected a total of 1.144 readable PDF files. The ICOs white papers pdf files occupy 4.3 GB of memory.

Step 2

We identified the ICOs that apply agile practices by searching for keywords in their white papers. We chose the keywords to look for in the following way.

- We used the Google Keyword Planner tool and we obtained all the keywords associated with the main keyword: *"agile methodology"*. In this way we got over 700 keywords.
- We selected all keywords that have an average monthly number of searches on Google above 1000.
- We included keywords consisting of single words (for example "*scrum*") and their specifications covering at most another word. For example we selected the keyword "*scrum programming*" and we excluded the keyword "*scrum programming development*" because implicitly included in the previous one.

Eventually, we obtained 90 keywords.We performed a textual analysis, by means of a R script, to verify in which white papers at least one of the 90 selected keywords was present. We analyzed all 1.144 readable white papers. The script converts the pdf content in a text string. Subsequently, for each white paper it counts the number of occurrences of each keyword in a case-insensitive mode. As a result, the script returns the table of keywords occurrences per ICO white paper. We identified 55 ICOs in which at least one of the 90 selected keywords is

present. We then manually verified that each of these 55 white papers actually referred to an agile software development mode. The obtained subset is about the 5% of the total analyzed ICOs. For simplicity, in the rest of the document, we will call this subset *Agile ICOs*. Fig. 4.1 schematically describes the process that allowed us to identify the 55 ICOs.

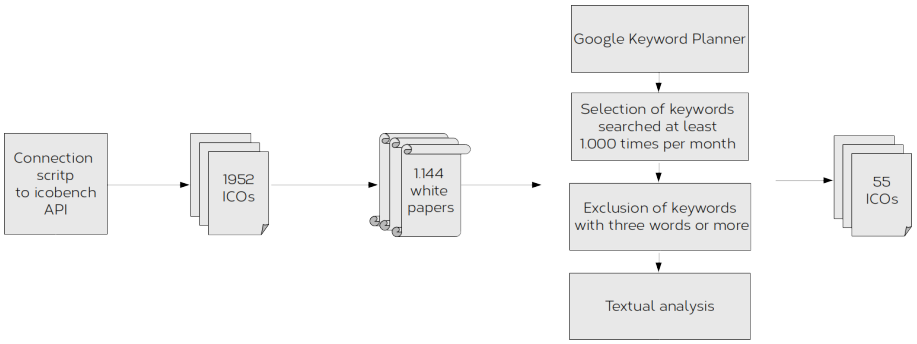


Figure 4.1: Flow diagram describing the selection process of the Agile ICOs

4.2.2 Analysis Setup

After the creation of the set of Agile ICOs, we compared them with the overall set of ICOs. We implemented specific R scripts to collect and analyze the information available in the ICO dataset. We collected data about the size of the team, and its composition in terms of roles and gender. We obtained the gender of team members by means of a names database realized by Mark Kantrowitz[7]. We used R to classify the team members by gender, to perform statistical and comparative analysis, and to collect and analyze some of the business and financial information available in the ICO dataset, also including the rating and the use of social media.

4.3 Data Analysis

In this section we report the results of the analysis. As described previously, we organized the analysis in steps. In the followings we present the numbers, the distributions and the statistical values that characterize the acquired ICOs, and in particular the Agile ICOs.

[7] available from http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/nlp/corpora/names/

4.3.1 Analysis of ICOs teams

According to [87], one of the main factors that impact the sustained usage of agile methods is the team composition, that should have a right balance in terms of technical skills, domain knowledge, team size, and gender, race and culture.

Team size and composition

We analyzed a total of 18,699 people involved in ICO founding teams. We firstly analyzed the size of the team that develops each ICO. We consider 1646 ICOs that declare at least one team member. We found that the mean size of the ICO team amounts to 11.4 people. The maximum team size includes 67 people. These results are higher than the results reported in [45], where the average team size was 10.87, and the maximum team size was 57.

When we computed these statistics on the subset of 55 Agile ICOs we found that ICOs in this typology have larger teams. In facts, the average size is 14.9 people, despite the fact that the larger team includes 43 people. This data has its intrinsic importance. In an ICO, a very small and anonymous team increases the investment risk and may be a scam. It is therefore important for investors to be able to consult the information relating to each team member, both on the ICO website and on LinkedIn or Twitter. The greater number of people involved, with a detailed description of each person supported by the link to the related social pages, the lower the probability that the ICO may be a scam.

An advisor is a domain expert (for instance an academic), or an investor, or a consultant. We found that 4.461 people involved in ICOs are advisors. As shown in Tab 4.1, advisors represents 18.9% of team composition, or in other terms there is nearly an advisor every five people.

ICO Stats	Avg. Size	Max size	% Advisors	% Women
All	11.4	67	18.90	16.30
Agile	14.9	43	23.04	15.00

Table 4.1: Summary of statistics on ICO team. The average percentage of advisors and female people per team are computed on teams of at least one person

In Agile ICOs, there are more advisors. On average, they represents over the 23% of the team. According to [35], in a very large team sometimes there are many advisors who contribute suggestions, but are not really involved in the ICO operations. We can consider advisors as a group of individuals with experience and able to provide credibility and value to the project, as long as they specifically work in the ICO and, create added value. We can see in fact that some advisors have their names in over 100 projects, each of which spans overlapping

periods of time. Such an advisor cannot physically allocate enough time to a project, to provide true value to their customers. We can compare the roles inside an ICO team with the typical roles described in the SCRUM methodology. In particular, in the SCRUM methodology the world is divided into "pigs" and "chickens" [86]. The former, during the development of the project, bring into play "the skin". All other stakeholders are spectators (chickens). The chickens may also be strongly interested in the project, but they do not work in a strict and direct way like the pigs. By analogy, in an ICO we can define team members as pigs, while advisors can be considered as chickens. In fact, the advisors are consultants who can support the team if it is needed; often they are selected for marketing reasons, and do not work directly to the development of the project.

4.3.2 Gender heterogeneity

We investigated the gender heterogeneity in ICO teams. It is well known that a global gender gap exists in entrepreneurship, and in particular in the technological startup founding teams. Female presence in startup teams is typically below 30%, which is the highest percentage recorded in the Chicago startup ecosystem [9]. For example, in a recent survey [73] that examined about 1526 software startups, they discovered that in their teams a very small percentage (8%) are females, in comparison to the percentage of males (76%). Note that 16% of team members did not reveal gender information.

Also ICO teams consist of people predominantly male. Our algorithm classified 9776 people; we found that the female presence is equal to 16.3%. Considering the average of the number of men and women per ICO founding team, we found that the number of men is about 5 times larger than the number of women. Considering the Agile ICO set, we noticed that the female presence is slightly lower than the presence computed in the total dataset. In facts, these teams have 15% of women, being on average composed by 6.5 men per 1 women. There is no case of team composed only by women.

In the ICOs, however, the presence of women is twice the presence detected by [73] in software startups not based on an ICO. [69] shows that gender diversity plays a significant role when considering productivity and collaboration within a software development team. These results could also be applied to ICO teams. In addition, also in relation to investors, according to [82], the number of women interested in investments in cryptocurrencies represents currently the 13% (one in eight women) of the total. This research also suggests that women invest very differently than men: the former take a much more strategic approach, and suffer less the "Fear of Missing Out" [79] than their male counterpart. The study also shows that women investors tend to collaborate much more than men, consulting family and friends about their investment before proceeding.

4.3.3 ICO Rating

The statistical analysis of the ICO Rating (a score parameter provided by ICObench that summarize the ICO reliability) allow us to compare the overall results with the subset of Agile ICO. The Rating is a real number which ranges from zero to five computed by ICObench, using a weighted average of two distinct evaluations. The first is computed by a proprietary assessment algorithm which considers the team composition, ICO information, product presentation, the marketing campaign and the presence on & social media. The second is assigned by experts who evaluate from 1 to 5 the ICO for team, vision, and product[8]. Actually, we found that ICOs Ratings vary from 0.4 to 4.9. On average, ICOs have a Rating equal to 3.1. Agile ICOs have, on average a better Rating score. In fact, the average Rating value of the 55 ICOs is 3.6. Considering the diversity of samples analyzed - the first with 1646 elements, the second with 55 elements - we performed a statistical analysis on the significance of the differences. We first verified that the values of the ICO rating do not follow a normal distribution. In facts, the Anderson-Darling normality test [9] produces p-value <2.2e-16 (the distribution is normal if p-value > 0.05 [97]). We then performed the Wilcoxon test[38] which is not based on distributional assumptions and therefore provides reliable results even when data do not have a Gaussian distribution. Through the test carried out[10] we can see that the differences are significant. The p-value = 2.476e-05 is in fact lower than the threshold value $\alpha = 0.05$. Empirical evidence is strongly contrary to the null hypothesis and the observed data are statistically significant. The minimum value of Agile ICOs is 2.1.

4.3.4 Social Media

We analyzed the use social media in order to understand how ICOs use this channel to communicate with investors and customers. 1810 ICOs out of 1952 use at least one social media. 1769 ICOs have at least one Twitter account, 1528 have a Facebook page. Telegram is used by 1231 ICOs, Youtube by 1112, Medium by 1069, Reddit by 812, GitHub by 796, Slack by 555 and Discord by 46 ICOs. All Agile ICOs communicate with investors and customers by means of social media. All Agile ICOs in fact have a Twitter account and 51 out of 55 have at least one Facebook page. Telegram is used by 42 Agile ICOs, Youtube by 39, Medium by 38, Reddit by 32, GitHub by 32, Slack by 14 and Discord by 3 ICOs. Given the decentralized nature of ICOs, the developers have to create a strong and active virtual community to support projects. As reported in [91], social media became

[8] https://icobench.com/ratings

[9] computed in R using the library *nortest*

[10] computed using the implementation provided by R

part of the standard communication tools in recent years.Telegram groups and Slack channels are used as tools to which interested parties can ask questions about the ICO. On the other hand, social networks are used by the team to share information and news, and to raise awareness on their cryptocurrency and ICO. According to [28], the communities of cryptocurrency users require transparent and reactive communication. According to Agile Methods, a software product is a constantly evolving project in which the initial idea could be modified and adapted, using the feedback provided continuously by users. External feedback at each stage allows the development of true competitive value of the product and services offered, promoting quality, efficiency and trust. Given the decentralized nature of the blockchain, social media are the only way, for an ICO, to communicate with users and investors. It is therefore not surprising that the use of social media is greater in Agile ICOs than in non-Agile ones.

4.3.5 Financial aspects

A token is a digital asset that in addition to having an exchange value, has an intrinsic value that derives from its use. ICObench describes the way in which the tokens are sold. The parameters reported on the website are: the number of tokens for sale, the percentage of token to be sold during the ICO, the hard and the soft capitalization, namely the goal of the ICO offer expressed in a reference currency, and the minimum selling target to be reached to develop the product. The number of ICOs which provide this parameter is 1053. We discovered that over 60% of such ICOs provides more than 50% of their tokens to investors. The remaining tokens are managed by the team. In particular, about 44% of ICOs choose to distribute more than 60% of its tokens to investors during the crowdfunding. This set is not characterized by a normal distribution of values. Applying the Anderson-Darling normality test we obtained a p-value equal to 6.404e-12, lower than the threshold α equal to 0.05.

Agile ICOs differ with respect to the general statistics. The 38 out of 55 Agile ICOs which provide this parameter, on average distribute a lower percentage of tokens to the investors during the crowdfunding. In particular, only 50% of ICOs distribute more than 50% of tokens during the crowdfunding, and only 23% of Agile ICOs assign more than 60% of tokens to investors. The remaining tokens are managed by the ICO team. According to the results of the Anderson-Darling normality test, we can consider these values as sample of a normal distribution of values (p-value = 0.06406 > 0.05). We hypothesize that a greater availability of tokens for the ICO team is consistent with the principles underlying the Agile Manifesto [7], in which the projects are based on motivated individuals who must have all the support needed to complete the work. The result of the Wilcoxon test allows us to consider significant the differences between the two

sets of data. The p-value is equal to 0.0262 and is lower than 0.05. So the null hypotesis is rejected, and the two samples can be considered as taken from different populations.

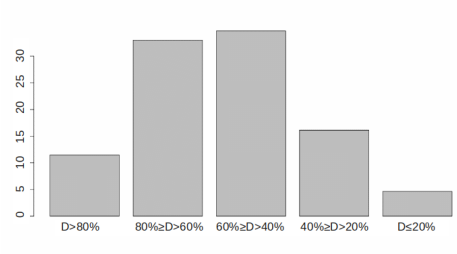


Figure 4.2: All ICOs

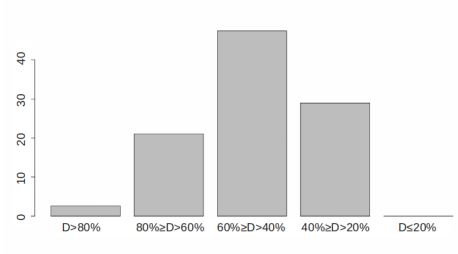


Figure 4.3: Agile ICOs

Histogram of the number of ICOs per percentage of Token to be sold during the ICO

Ico market capitalization

The market cap is the value of tokens expressed in a reference currency. An ICO, through its *hard cap* (hard capitalization), sets the limit of how much money will be accepted to finance the development of the product. The excess money received is returned to the investors. The *soft cap* (soft capitalization) is the minimum amount required by the project in order to continue its development. If this amount is not reached, investors can withdraw their contribution. A crowdsale that reaches the soft cap is considered successful.

We found that 469 ICOs provide both the hard cap and the soft cap.Among these, the soft cap is on average 19% of the hard cap, so a crowdsale that reaches 19% of the hard cap is considered successful. Analyzing in particular Agile ICOs, 25 out 55 provide both hard cap and soft cap. In this case, on average, the soft cap is 25% of the hard cap, so in Agile ICOs only a crowdsale that reaches 25% of the hard cap is considered successful. Therefore, Agile ICOs needs a initial capital proportionally higher than the non-Agile ICOs to develop the project. No Agile IC reached 100% of the hard cap (the maximum percentage is 85%).

In this case, the differences found between Agile ICOs and other ICOs are not significant. The results of the Wilcoxon test suggest that the two set of data can be considered as elements of the same population. The resulting p-value is equal to 0.05878, greater than the threshold value, α, equal to 0.05, and therefore the null hypothesis can be considered valid.

The distribution of the ratio (in percentage values) between *softcap* and *hardcap* is shown in Fig. 4.4 and 4.5.

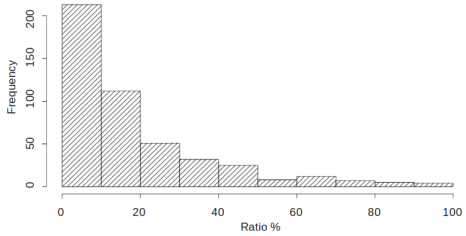


Figure 4.4: All ICO

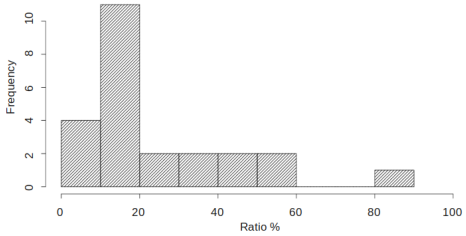


Figure 4.5: Agile ICO

Histogram of the percentage of the Hard Cap per ICO to be reached to consider the ICO as a success.

4.4 Analysis of Agile ICOs projects

In this section we focus on the analysis of software projects of Agile ICOs. We studied in particular two aspects: the first is the ICO *roadmap*, the typical step by step description with which the ICO proposers declare their development program of the proposed product or service. The second is the software development project, that is the available repository of the ICO source files. This second aspect includes the analysis of development tools and testing practices used by the developers of the ICO.

4.4.1 Roadmap and ICO state

As we said in section 4.3, a team describes, usually in the white paper, the roadmap of activities after the crowdfounding, and outlines the actions which they aim to achieve during the product development. Generally, this description is a simple graphica overview of the project's goals and deliverables, and of the related timeline. Investors look the roadmap to know when and how the business idea will be operative and profitable, to understand the development phases of the product. We chose to identify the starting-point of a roadmap as the time when the ICO crowfunding closes, and the team gathers the money needed to develop the product. In the subset of 55 Agile ICOs, only 9 ICOs don't provide a roadmap. In the remaining 46 ICOs the roadmap is also called milestone, timeline or highlights. In these, the time period that roadmaps cover, ranges from few month to over five years, as shownin Fig. 4.6 and summarized in the following:

- only 18 ICOs present a roadmap longer than one year;
- the longest roadmap extends 72 months after the end of the ICO;

- the average duration of roadmaps is 16 months after the ICO end;
- in 2 cases, the roadmap is concluded with the closing of the ICO.

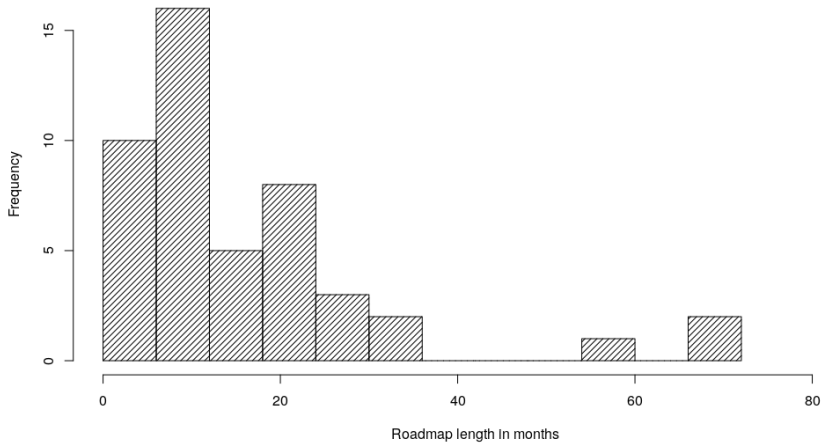


Figure 4.6: Histogram of the number of months of development described in the Agile ICO roadmaps

We define the time of post-ico state as the number of month passed after the ICO period. Fig. 4.7 shows the distribution of the state of ICOs in terms of number of month passed after the end of the ICO selling phase.

The roadmaps therefore concern the future of the ICO and provide a realistic plan on the use of the funds in view of the objectives set. The content of the roadmap helps investors to understand when and how they are involved in the project, and provides them a view on possible changes. A too detailed roadmap is typical of the plan-driven methodologies and contradicts one of the fundamental principles of Agile methodologies, that aim to respond to change more than to follow a predetermined plan[22]. In a roadmap developed with an Agile approach, the possible difficulties that the team can meet and the way to deal within possible obstacles should be taken into consideration. A roadmap developed with an Agile approach should therefore be compared not only to the future features of a project, that is difficult to accurately detail and predict, but should also show the daily work and progress of the team, with a special focus on feedback and opinions of the people involved. The roadmap is therefore a useful tool to promote the transparency of development, also in order to manage customer expectations. Focusing on specific features diverts attention from the general vision of the project. An Agile roadmap is therefore able to embrace the

inevitable changes, to communicate a short-term plan, but it must also include a flexibility that allows this plan to be adjusted to the customer's value or changes imposed by the market. According to [85, 1] we define below some guidelines that characterize a roadmap designed with an Agile approach.

- The roadmap must be oriented towards objectives much more than towards the features be developed, so that everyone involved can understand the evolution of the product.
- The creation of Agile roadmaps requires continuous communication within the team, and with investors. It also needs to harness the effort of all the parts involved.
- In an ICO, it is essential to respond adequately to the needs of investors. Responding promptly to customers' needs through continuous dialogue is one of the essential characteristics of the Agile approach. The roadmap should therefore take into account all investors' feedback for possible improvements. The new ideas, evaluated through a score, must be included in a future release backlog. The ideas of investors and customers should therefore guide the definition of future priorities.
- The roadmap should be changed quite frequently (from one to three months) in order to adapt the plans with the obtained feedback. Updating the roadmap can help a project to face changes without diverting attention from long-term goals.

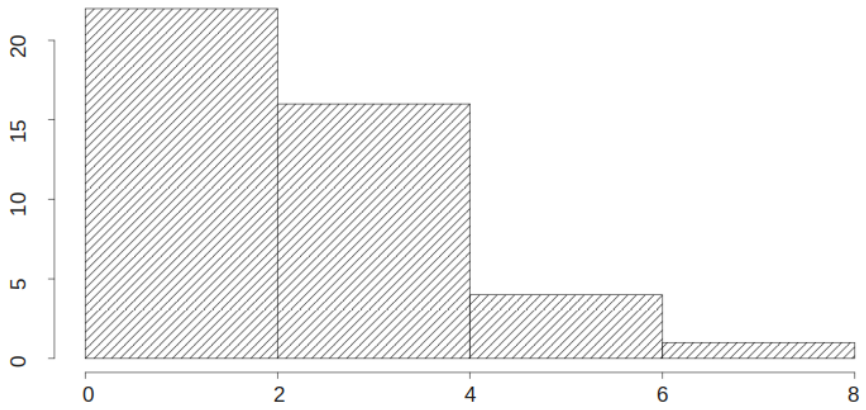


Figure 4.7: Histogram of the number of months passed after the end of the ICO

4.4.2 Software development

In order to better understand the process of software development of ICOs, for each of the 55 ICO projects selected in step 2 we examined the documentation to find the availability of software project repositories in which the development team stores and manages the software under development. We found that:

- 36 ICOs have a software project publicly available on the Github platform.
- 12 of the remaining ICOs published in the Ethereum blockchain explorer *Etherscan*[11] the solidity code of the smart contract used to implement the token selling.
- 7 ICOs do not have a publicly available software project nor provide any smart contract solidity file.

Summarizing, 48 out of 55 Agile ICOs provide at least the smart contracts used to develop the token of the ICO. These smart contracts are written in Solidity [2], which is the most popular high-level language for implementing smart contracts. It was influenced by C++, Python and JavaScript and is designed specifically for the Ethereum platform. Solidity files have extension *.sol*.

We analyzed the 32 Agile ICOs software projects available on Github. In particular, we counted the number of *repositories* (the folders of the project), the typology of files, and the number of solidity files.

In summary, the 32 Github projects contain a total of 14.199 files. The total number of folders is about 2800. On average, each project contains 5.8 *repositories*, and the maximum number of *repositories* per project is 38.

Regarding the contents, source code files represents the most of the files present in ICO software projects. In particular, *js* files (javascript) dominate the scene with 5.015 files, equal to 35.32% of the total. The projects contain 786 smart contracts written in solidity (.sol), equal to 5.54% of the total. Graphics file formats (like png and svg formats) represent the second most frequent kind of files. Tab. 4.2 summarizes the number of occurrences of the ten most common file types found in Agile ICO software projects.

To better understand the process underlying the development of Agile ICOs, we have verified for each project the use of specific development frameworks and in particular the use of *Truffle*. Truffle is a popular development framework for Ethereum that includes built-in smart contract compilation, linking, deployment, binary management, and automated testing[12].

[11] https://etherscan.io/

[12] Truffle is available at https://truffleframework.com

Extension	# Files	Percentage
js	5015	35.32%
png	1043	7.35%
sol	786	5.54%
cs	680	4.79%
md	594	4.18%
hpp	533	3.75%
json	485	3.42%
cpp	467	3.29%
go	436	3.07%
svg	418	2.94%
others	3742	26.35%

Table 4.2: The ten most common file extensions in Agile ICO projects.

Smart Contract code metrics

In this section we report the results of the analysis of the smart contracts used to implement the Agile ICOs. In particular, in order to characterize the content of these solidity files (with extension *.sol*), we applied a selection of code metrics. We also report the comparison between our results with the results provided by Tonelli et. al [99], related to more than 12.000 smart contracts published on the blockchain explorer Etherscan and deployed on the Ethereum blockchain until January 2018.

In our analysis we examined 502 solidity files found in Agile ICOs github projects, which could be referred directly to the ICO developers. We excluded from our analysis the files copied (or *forked*) from other projects (including templates taken from development framework like *Openzeppelin*[13]).

We added to the analysis the 12 smart contracts published only in Etherscan, as described before. In total, we examined 514 smart contract files. For each of them we applied the software metrics defined in Table 4.3, that include volume metrics and complexity metrics In the following, we will use the term *contract* to refer to a specific type of object of the Solidity language [2] The declaration of an object *contract* is similar to the declaration of a *class* of object oriented languages. In a contract definition it is possible declare functions and variables, that can be modified to be private, public or internal to the contract. A solidity contract can also inherit from other contracts.

Table 4.4 reports the summary of the statistics of the computed volume source code metrics. We considered "function" the declaration, through the

[13]https://openzeppelin.org/

LOC	Lines of Code: number of lines containing executable code
CPL	Comment per Line: number of lines containing comments divided by the number of lines of code
NDC	Number of Declared Contracts in the solidity code file
NDF	Number of Declared Functions in the solidity code file
FPC	Functions Per Contract (number of functions in the source code file divided by the number of contracts)
AFL	Average function length in a solidity code file, in terms of lines of code
Acyclo	Average McCabe Cyclomatic Complexity of the functions among a solidity file
Mcyclo	The maximum McCabe Cyclomatic Complexity computed among all functions of the solidity file
SumCyclo	Sum of the single McCabe Cyclomatic for all functions complexity of the solidity file.

Table 4.3: Definition of computed source code metrics.

keyword *function*, of the executable units of code within a contract. We did not considered as functions the definition of *function modifiers*. In solidity a modifier is a short portions of code defined through the keyword *modifier* that can be called and incorporated in functions. They can be used to easily change the behavior of functions.

In this table we also provide, for each metric, the value of the First Quartile that separates the lowest 25% of values from the highest 75%, and the Third Quartile that separates the lowest 75% of values from the highest 25%. All the reported statistics represent an overview of the distribution of metrics values. All statistical analysis were performed using R. Results show that smart contracts in Agile ICO projects have on average 65.59 lines of code. The maximum number of LOC is 808 and the minimum is 2. For comparison, results of [99] report that the mean number of LOC is 183.8. We found that smart contracts present in Agile projects are characterized by a lower number of LOC.

The mean and the median value of CPL allows us to state that examined files are well commented. As reported, there is expected about one line of comments every two lines of code.

The examined solidity files declare, on average, only two contracts. This number can be considered low, in relation to the value of 9.2 reported in [99]. The maximum number of contract declarations is 66 and the minimum is 1.

In total, the mean number of functions (NDF) declared in a contract is about 6.6, the maximum value is 71 and the minimum is zero. Also in this case, the

mean number is lower than the results of [99] (25.9 functions per file). The mean values of NDC and NDF can be considered related to the value of the LOC metric. Solidity files are shorter and consequently less functions and contracts are declared. Contracts having no function declarations can be used to define variables and data structures to be inherited by other contracts. In general, if there is no constructor, the contract will assume the default constructor.

The metric Function Per Contract (FPC) represents the equivalent of the number of *method per class* in object oriented languages. Considering each contract in the files, we found that, on average, each of them declare 4.3 functions. The maximum number of declared functions per contract is 28 and the minimum is zero. On average, each contract has functions that are long 7.8 lines (AFL). The maximum average length is 172.5 The related distribution is characterized by a high standard deviation.

From these results we can deduce that the smart contracts of Agile ICOs tend to be short programs, with a limited number of elements (contracts and functions) and with short functions. This favors easier reuse and maintenance of the code [84].

In general, the development of the ICO token pass through the implementation of a standard interface called ERC20[14]. Given the availability of already implemented ERC20 tokens, the reuse of code is commonly adopted during the creation of new tokens. The high values of standard deviation show that smart contracts are very different from each other; these results are typical of long-tail distributions, whose tails collect the highest values.

Metric	LOC	CPL	NDC	NDF	FPC	AFL
Mean	65.59	0.453	1.961	6.661	4.291	7.785
Median	28	0.317	1	3	3	5.667
St. Dev	88.47	0.48	4.87	8.96	4.50	12.19
Max.	808	2.727	66	71	28	172.5
Min.	2	0	1	0	0	1
1st Qu.	16	0.067	1	2	2	3
3rd Qu.	81.75	0.717	1	8	5	9

Table 4.4: Volume metrics of smart contracts belonging to Agile ICO projects.

For each solidity file, we computed the McCabe cyclomatic complexity [56] of all the functions implemented in it. The cyclomatic complexity measures the number of linearly independent paths through a function. We used a commercial software[15]. Table 4.5 summarizes the results related to the average, the

[14] https://theethereum.wiki/w/index.php/ERC20_Token_Standard

[15] We computed the cyclomatic metrics using *Understand*, by *scitools*. Cyclomatic metrics are

maximum and the sum of the cyclomatic complexity of all the functions defined in each solidity file belonging to Agile ICO projects. The minimum values of these metrics are equal to zero due to the presence of contracts that do not implement any function.

We found that the average cyclomatic complexity (*ACyclo*) has a value equal to 1.2. The maximum value of the average cyclomatic complexity is 7.

The maximum cyclomatic complexity (MaxCyclo) is, for each contract, the the maximum value of McCabe cyclomatic complexity among the functions of the contract. Its mean value is 1.83, and its highest value is equal to 17.

Contracts are characterized by a limited sum of the cyclomatic complexity (SumCyclo) computed for each function in their solidity files. The mean value of the sum is 7.97, lower than the value reported in [99], due to the fact that contracts of Agile ICO projects are shorter in terms of LOC. Values of this metric are characterized by a standard deviation equal to 12.27, and a maximum value equal to 134. These value are typical of long tail distributions.

Metric	ACyclo	MaxCyclo	SumCyclo
Mean	1.21	1.833	7.969
Median	1	1	4
St. Dev	0.59	1.70	12.27
Min.	0.00	0	0
Max.	7	17	134
1st Qu.	1	1	2
3rd Qu.	1	2	9

Table 4.5: Cyclomatic metrics computed in the solidity files belonging to Agile ICO projects.

Testing

In the Ethereum platform, each smart contract deployed in the blockchain both the data related to the transactions, and the code that implements the logic to allow the sending of transactions between two or more actors. Therefore, data and logic that compose a smart contract are stored irreversibly. Given the principle of the immutability of the blockchain, once a smart contract gets deployed, its code cannot be changed.

If a developer finds a bug or wants to correct an error, s/he has to develop a new smart contract, deploy it on the blockchain and transfer all the existing data to the new contract. The deployment of a smart contract includes an

described in https://scitools.com/support/cyclomatic-complexity/

Ethereum transaction that requires the payment of a fee, which depends on the size of the Smart Contract. For this reason, the test phase before the deployment is very important and it should be managed appropriately also through the adoption of best practices and specific tools for continuous testing, typical of Agile methodologies.

We therefore analyzed the use of development tools and practices for Smart Contracts. In particular we look for the use of the Truffle suite for testing practices. We found that Truffle is commonly used in Agile ICOs. 19 out of 36 projects include the typical Truffle elements (the file *truffle.js* and the directories build, contracts, and migration). Using Truffle, developers can take advantage of several development tools included in this suite. One of the most relevant is the possibility to create a Test suite. Tests can be written both in solidity and in javascript. We found that 16 of the 19 Agile ICO projects that use Truffle include a test suite. In addition, we found the presence of other kinds of testing code developed to test other components of the project. The use of Truffle, which provides an automated testing framework to test smart contract before the deployment on blockchain, is fully consistent with the application of Agile methodologies.

4.5 Discussion

The results of this study provide an overview of the world of ICOs, which can be described as is a new blockchain based fundraising mechanism in which a startup sells its tokens in exchange for Bitcoins or Ethers. The ICOs, therefore, offer important new possibilities related to the intrinsic properties of blockchain technology. In order to make the most of this potential, in such recent and innovative context, it is necessary to employ appropriate software development methodologies. We believe that the Agile methodologies, which are suitable for the management of innovative startups because they allow to easily face the changes, can also be useful when applied to the ICOs. Agile methods are suited to develop system whose requirements are not completely understood, or tend to change. These characteristics are present in ICOs because they are typically very innovative applications and often there is a run to launch an ICO to be the first on the market. An Agile approach is based on iterative and incremental development with short iterations, and is suited to deliver quickly and to deliver often. In addition, Agile methodologies are suited for small, self-organizing teams working together, as is the case for many ICO teams. In our analysis we found that in Agile ICOs the team is on average made up of 14.9 people, of which 23.04% on average are advisors. This means that the development team is on average made up of 10 people. Among them, we include 6 or 7 developers, but

also other professionals, like testers or UI designers. In SCRUM methodology, for example, a development team consists of 3-9 people. For larger projects, Scrum provides a mechanism called Scrum of Scrums that assign the large project to several teams [63]. On the other hand, we need to take into account some important factors inherent in decentralized technologies that need to be carefully controlled in applying Agile methodologies. An ICO is based on the use of smart contracts that record data transactions and manage the specific functionalities of the project. The data stored on the blockchain are immutable, so in order to modify a smart contract we have to deploy a new smart contract. Agile development, on the contrary, insists on continuous feedback from users to foster innovation and it also strongly highlights the necessity to continually iterate on the product. This is the part of the process that can create issues for blockchain products. Another aspect that must be taken into great consideration is the one related to project planning. The planning of an agile project is based on the split of functionalities in user stories, which are fragments of functionalities that give value to the user. The analysis of the requirements leads to the definition of a certain number of stories, each having an intrinsic value to the project. A user story must be measurable, and must be developed in a limited time. A roadmap developed with an agile approach should therefore pay particular attention to the feedback and wishes of all involved stakeholders. The roadmap is therefore a useful tool to promote the transparency of the development, also in order to manage customer expectations. It must not be too detailed, and must allow the implementation of the project as a set of user stories or features. The implementation flow could take place in Sprints, and each user story should be provided with one or more Acceptance Tests [55].

Regarding the quality of the code, the results obtained by us are not consistent with those found by [31], that made a comparison between software metrics in a software system (written in Python and in Java) and developed using agile methodologies and in systems developed using plan-driven methodologies. They assert that metrics distribution generated from Agile methodologies are not related to better quality of software. Unlike ours for example they found that the LOC distribution does not demonstrate major differences. In our analysis smart contracts present in Agile projects are characterized by a lower number of LOC in comparison with ICOs that not use agile methodologies. In general all smart contracts tend to be short with a limited number of contracts and functions. It is also connected with the fact that to deploy a file on the blockchain it is necessary to pay a fee proportional to the size of the file. The simplicity of the code is perfectly consistent with one of the five fundamental values of the XP [8].

4.6 Conclusions

The chapter proposed an analysis of Initial Coin Offerings, a new phenomenon that has recently become a relevant topic of study within the blockchain community. In order to better understand this phenomenon, operating in an uncertain and constantly changing context, we investigated all engineering activities related to ICOs, from the planning phase to the testing phase. We analyzed the whole set of ICOs registered in ICObench until the February 20, 2018 in order to discover the team composition, the communication channels with investors distributed around the world and the financial aspects. We therefore studied the use of agile practices in ICOs as a method to cope with change. We have selected and analyzed in detail a subset of ICOs specifically developed with agile methodologies relatively to their roadmap, project development, and source code quality. Overall, about the 5% of the examined ICOs apply Agile practices. In addition we conducted an analysis of smart contracts of Agile ICOs in terms of code metrics, language versions and use of test tools. We discovered that the agile methodologies are suited to develop ICOs because these are highly innovative projects, whose requirements are not completely understood or tend to change. The agile approach is iterative and incremental with short iterations and is suited to deliver quickly and to deliver often. This property is very useful in the context of ICOs. On the other hand the necessity to continually iterate on the product, typical of agile methodologies, can create issues for ICOs. The immutability of the blockchain must be taken into consideration. In fact, smart contracts can not be updated once they have been loaded. To face this difficult the Test Driven Development is very useful. Also the practice of Collective code ownership is guaranteed in ICOs by the transparency of smart contracts in the blockchain. Another practice of Agile development that are applied in ICOs is the use of Coding Standards. The too detailed roadmaps are instead typical of the plan-driven methodologies. Finally we can say that the smart contracts of Agile ICOs have good metrics of the software because their source codes are very short and simple.

Limitations. We summarize below the main limitations of study. The first issue is related to the novelty of work. The typology of analysis made is therefore pioneering. We focused part of our study on the collection of Agile Keywords inside the white paper, in order to recognize which kind of ICOs funded project used Agile practices. We did not analyze therefore ICOs in which no agile keywords are ufficially stated in the white paper, but the development could be anyway Agile based. We are indeed aware that not all ICOs they may have declared whithin its own documentation Agile keywords even if they use Agile Methods for developing projects. We collected 55 Agile ICOs and we calculated the software metrics only on the smart contracts of these ICOs (for a total of 514

smart contracts). We made a comparison between this subset and the code metrics related to all smart contracts stored on Ethereum blockchain in 2017 (around 12000 smart contracts). Our subset may be small but the results obtained can serve for a future generalization considering a consistent dataset.

Future works. In the next iteration of this empirical study we want examine more aspects of ICOs to recognize the use of Agile practices in a more comprehensive way taking into consideration an analysis related to each single agile methods. In the future will be possible use the results of this work as a snapshot dated February 2018 of ICOs characteristics, and use it for a time-based comparison focusing mainly on the rate of adoption of Agile pratices. In addition, a future work should provide a correlation analysis between the usage of Agile practices and the ICOs success, both in terms of capitalization and in terms of the development of their projects.

A further study related to smart contracts is presented in the next chapter.

Chapter 5

Ethereum Smart Contracts

The publication of the Ethereum white paper in 2014 [17] and the implementation of the Ethereum platform moved the blockchain technology [93] to the second generation. In fact, what this platform for decentralized applications proposed, was new and disruptive, that is a blockchain-based programmable Turing complete virtual machine to run software code written specifically for the blockchain environment [105]. Such software was originally conceived to take advantage of the blockchain features in order to automatically implement the constraints two parties can agree upon when they sign a contract in a trustless environment, so that the software code was named "smart contract". Nowadays, the initial concept has been largely extended so that smart contracts can be considered as general purpose software programs.

Smart Contracts (SCs for short) are small computer programs stored inside the Ethereum public ledger and associated to a particular blockchain address which references the SC software code. Ethereum smart contracts are mainly written in Solidity, a programming language derived from Javascript, Python and C++, which allows to run programs on the blockchain infrastructure as decentralized applications. The smart contracts code is compiled and the corresponding *opcode* is loaded into a blockchain address and run by the Ethereum Virtual Machine (EVM). Virtually, SCs can perform any computational task standard programs can perform, but there are specific constraints that must be respected due to the decentralized structure of the blockchain and to the consensus protocol adopted by Ethereum, so that SCs display specific features and issues which are unknown in traditional software development. A typical example is the extraction of a pseudo-random number which should be replicated in all the blockchain nodes in order to obtain the same result [52].

Due to these specificities, this technology is having a great success and has paved the way for a new set of applications, yet to be fully defined. Ethereum is the most important blockchain based platform in terms of number of trans-

actions. At time of writing the number of accounts stored in the blockchain is just over twenty-five million. The number of contract accounts activated in the Ethereum platform is over four million five hundred thousand (about 18% of the total)[1]. Contract accounts are used both to create decentralized applications and to create new digital tokens, looking to new business opportunities and to an easier way of funding (the ICO phenomenon [45, 35]). The byte-codes of contracts are always available, because they are recorded in the blockchain. However, byte-codes are not intelligible; in order to increase the trust of users, developers of decentralized applications may provide the source code of their contracts.Furthermore, third party websites, like Etherscan.io, offer a verification service that makes smart contracts public. The overall success of decentralized applications presents practitioners and software engineers with new and specific challenges and blockchain oriented software should be managed according to their specificity [78]. Furthermore, in the scenario of a wide diffusion of the blockchain technology, smart contracts could represent the backbone for several future software development decentralized applications [93, 46, 54, 53]. Since blockchain is a newborn technology, the development of new decentralized applications could take advantage of a thorough analysis of what has been created up to now, with the aim of correcting errors of the past and improving software development best practices.

By the end of 2017 the amount of smart contract source code freely available and the number of related transactions on the Ethereum blockchain reached a size which allows a systematic empirical and statistical study. This study considers the source code features and some smart contracts code measures, the evolution of the Solidity language, and other features relating smart contract source code to the transactions performed on the Ethereum blockchain. Such an empirical analysis would have been an impossible task just a few months before the time of our study because of the scarcity of smart contracts source code available deployed on the blockchain and for the contemporary scarcity of statistics related to the operations and interactions among smart contracts and the blockchain.

This study aims at understanding software features and metrics of smart contracts, in order to measure progress and performance and to provide food for thought for improvement of these software artifacts.

We performed an empirical study collecting the dataset of all smart contracts source codes available from Etherscan.io up to December 2017. We computed several software metrics on the entire dataset and we identified the twenty most used smart contracts, representing a reduced set on which we performed a systematic and more detailed analysis, in terms of both functionality and

[1] data from https://www.etherchain.org/contracts

development history. We identified some empirical indicators useful to describe smart contracts from a statistical point of view. By means of these indicators we studied the usage of smart contracts in the Ethereum blockchain and their evolution over time.

Results lead us to observe an active developer community that constantly follows the evolution of the language, develops more and more specialized smart contracts, and improves contracts already developed. Code measures shows that smart contracts have a limited number of line of code, but are well commented and implements specifics functionalities.

The remaining of chapter is organized as follows: Section 5.1 provides a selection of related work in the field of smart contract analysis and metric applied to specific software categories. Section 5.1 provides a description of the solidity language and of the Ethereum environment. Section 5.2 describes the dataset and the results of the analysis in terms of contract name, compiler version, balance and transactions, and of the measure of source codes, such as the number of line of code, the number of contract declarations and the related size of the bytecodes. Section 5.3 analyzes twenty smart contracts, selected form the dataset by the highest number of transactions. Firstly provides a description of each contract, then describes the interaction of the development community in terms of number of versions and of reuse of code. Finally reports the results of the code analysis performed by means volumetric and complexity code metrics. Section 5.4 summarizes results of code metrics applied on the source codes. Section 5.5 discusses the findings of this work, summarizing results and providing some considerations derived from them. Section 5.6 concludes the chapter.

5.1 Background

Research literature on blockchain in general and on smart contracts in particular, from a software development perspective is limited to the last few years. The development and the diffusion of “Solidity” as programming language for writing smart contracts on the Ethereum platform started very recently and the definition and implementation of the language and of its Virtual Machine on Ethereum (EVM) is still ongoing.

In this section we provide an overview of the more recent findings in the field with a glimpse on the specific domain of smart contracts programming and related topics already published in software literature.

Only very recently, the research on software engineering and computer science paid particular attention to the blockchain technology and its specificities. In 2017, Porru et al [78] underline the need of a new branch of software engineering, and coined the term *BOSE* (Blockchain-oriented software engineering).

In this context, authors highlighted the need of new professional roles, new specialized metrics and new modeling languages in order to ensure security and reliability. They designed possible solutions proposing the directions for future specific steps of the BOSE.

Bartoletti et al. [6] conducted a survey of smart contracts by studying their usage, development platforms and design patterns. Furthermore, they categorized the contracts by their application domain in order to understand the best convenient investment.

Tonelli et. Al [99] analyzed more than 12000 certified smart contracts provided by Etherscan, along with Bytecode and ABI. They results reports that metrics are less variable than in traditional software systems because of the specificity of the domain. Furthermore in smart contract software metrics there are no large variations from the mean. All values are generally within a range of few standard deviations from the mean.

In order to define a specific Blockchain Software Engineering, Destefanis et al. [30] argue that smart contracts have a non-standard software life-cycle and therefore applications can hardly be updated or it is more difficult to release a new version of the software.

Wan et al. [101], in order to design efficient tools to detect and prevent bugs within the blockchain, performed an empirical study to understand the blockchain bug characteristics. They investigated the bugs frequency distribution manually examining 1108 bugs in eight open source blockchain.

Bragagnolo et al. in [15] presented *SmartInspect*, a tool able to debug smart contract code, addressing the lack of inspectability of a deployed code. In fact, once a smart contract is deployed, data are encoded and the source code can not be redeployed. Authors proposed a solution by analyzing the contract state through a decompilation techniques and a mirror-based architecture without redeployed it.

Rocha et al. [83] implemented a tool to handle smart contract written in Solidity language, the solution is specifically designed for Pharo (a live programming environment based on Smalltalk code language).

Norvill et al. [67] used *Etherscan.io* in order to explore smart contracts and to analyze bytecode level metrics or to identify similarities between compiled pieces of code. They focused their attention on contracts compiled code, source code, and metadata such as the contract name.

The smart contracts are the basis for Initial Coin Offerings (ICO), the new means of crowdfunding centered around cryptocurrency in the blockchain development area. In this regard Fenu et al. [35] analysed the quality and the software development management of 1388 ICOs in the 2017. Ibba et al. [45] investigated on the ICO process analyzing a dataset obtained collecting data from specialized websites. They emphasized the advantages which Lean

methodologies could lead both to the team organization and to involve the stakeholders.

Eventually [70] Ortu et al. proposed the usage of micro patterns to evaluate the software quality, their results suggest that this approach could be a useful way to monitor the software evolution and to trace refactoring operations.

In general the literature on smart contracts software and in particular on the Solidity programming language is still limited and a comprehensive empirical analysis on a dataset of thousands smart contracts source codes and the metrics representing and characterizing their interaction and usage within the Ethereum blockchain has not been performed yet.

5.2 Analysis of the Smart Contracts dataset

We performed an exploratory empirical study on 10174 smart contracts, deployed in the Ethereum blockchain and validated using the Etherscan validation service. Our dataset includes all smart contracts uploaded until the end of the year 2017. The analysis considers two sets of information at different levels. The first set characterizes the contract with respect to the blockchain environment and to the interactions with it. It is a set of parameters associated to, and defining the contract state, which can be time varying. It consists of a list containing descriptive information of each smart contract. In particular, it contains the Ethereum address, the contract name, the number of transactions, the compiler version and the balance of each smart contract verified in Etherscan. We extracted all the information from both the source code and by browsing the Ethereum blockchain transactions related to each contract, starting from the list of verified smart contract source codes provided by etherscan.io[2].

The second set characterizes software code, is fixed, and can be viewed as independent from the blockchain environment. It consists of a collection of 10174 ".sol" files containing the contracts source code as extracted from the Etherscan website. In fact, Etherscan provides a descriptive page for each contract as well as the source code in separated frames. We extracted the source code from the contract page implementing an R script[3]. Given a contract address, the script loads the HTML code of the contract page, recognizes the start and the end of the source code, extracts and saves it in plain text. The size of the source codes dataset is about 100 MB.

Our exploratory study first examines the two sets independently, then compares the information collected on both.

[2]List available at https://etherscan.io/contractsVerified

[3]https://bitbucket.org/account/user/smartcontractsanalyzers/projects/CD

We first analyzed the parameters that characterize the smart contracts in the blockchain, aiming to provide statistical information of features like the name usage, the compiler version, the number of transactions, and the balance of contracts.

In the second part we characterized smart contracts source codes, also by means of a statistical analysis. In particular, we computed a set of code metrics for each smart contract in the dataset and present the statistics characterizing the entire dataset.

5.2.1 Smart Contracts parameters: analysis

We evaluated the main parameters and metadata that describe every smart contract in our dataset. Specifically, we focused our attention on the *contract name*, the compiler version, the number of transactions, and the contract balance in Ether.

We chose to analyze the list of contract names in order to evaluate if the ethereum developers community uses specific names for specific functionalities or whether the contract name does not have particular meaning.

The analysis of the dataset of the compiler versions allows us to understand if developers follow the continuously updating of the language specifics, released in order to fix bugs and to provide new and optimized functionalities.

The contract balances and the number of transactions are two series of values characterizing contracts in terms of usage, popularity, and in terms of funds inserted into that account we obtained both a snapshot of the interaction of each contract in the blockchain and an overall statistics on their values. The number of transactions is the total number of transaction that a contract receives and sends from *normal* accounts (owned by users). This number does not include transactions sent between contracts (called internal transactions).

All these data are public available for each smart contract deployed in the Ethereum blockchain and verified by Etherscan.

Contract Name

In the Etherscan platform, smart contracts are characterized by a Contract Name. According with the Etherscan specification, the Contract Name must match the *ContractName* in the source code that is deployed into the blockchain. See for instance the contract Crowdsale in Appendix 5.4.1 or the contract KittyCore in appendix 5.4.3.

So we refer to *Contract Name* either as the name of Etherscan which identifies the solidity file containing the source code or to the keyword inside the solidity file where, for each file, there may be different contracts. In facts the language

syntax, the keyword *contract* substitutes the keyword *class*, but a contract has features similar of a class. In facts a contract can be represented as a structure that includes a set of variables and a set of functions, that can be public or private. But the similarity is hardly complete. For example, class code can be called from other classes in standard software and methods can be called using methods and class names. Classes can be statically coupled when a class resources to code of another class in the system. Class names are also chosen according to good programming practices where the name reflects also class functionalities and purpose (eg. the "rectangle" class, the "point" class). On the contrary, some of these features are lost in smart contracts and so does the semantic of the name. The contract name looses any "architectural" design meaning and its methods or functions, its functionalities, are called by mean of blockchain transactions.

As a consequence different smart contracts may hold the same name and contain completely different code, or may be two slightly different versions of a same contract, o may be the very same contract deployed many times for testing purposes, or again part of code existing in one project and reused in another (eg the "token" contract, ERC20 compliant contracts) and so on. So it is of particular interest the analysis of contract name occurrences.

In our study we analyzed the collection of Contract Names in our dataset and we found that among the 10174 contracts, only 6205 names differ. More specifically, we found that:

- 4980 smart contracts have a unique Contract Name and are deployed only one time on the blockchain: there is no other address holding a contract with the same name.Therefore there is no ambiguity, the contract is identified by the name.

- 1225 Contract Names are used more than once (from 2 to 213 times). So that there are very popular names where different blockchain addresses register many contracts with identical names, but also the same contract (with the same solidity code) multiple times.

Tab. 5.1 reports the list of the ten most used contract names and shows that some contract names (eg. *crowdsale*, *token*, *ECR20Token*) occur more than one hundred times.

The occurrence of the same contract name multiple times is due to at least three possibilities: contracts codes are identical and the very same contract is used many times in different accounts; contracts codes are similar for functionalities and code metrics, but the codes differ slightly, so that are one a modification or adaptation of the other; contracts are completely different in code and metrics and they only share the same name, because semantic has still a limited role in smart contracts software development.

Contract Name	Number
Crowdsale	213
Token	143
ERC20Token	138
ApprovedTokenDone	82
Presale	81
MyToken	66
TokenERC20	63
PreICOProxyBuyer	60
CrowdsaleToken	56
MultiSigWallet	54

Table 5.1: The 10 most used contract names

A typical example of contracts sharing common names are contracts associated to ICOs[35]. The contract "Crowdsale" belongs to this category, since its code manages token crowdsales with different purposes and may be easily reused.

In general smart contracts with the same contract name, although belonging to different projects, have very similar functionalities and metrics.

Among the 213 smart contracts called *Crowdsale*, we found that six source codes are deployed at least twice. One of these codes [4] has 4 duplicates. This is a smart contract with the same bytecode and identical metrics that were subsequently memorized in the blockchain.

Compiler Version

According to [83] any smart contract written in Solidity has a grammar that starts with the SourceUnit rule which contains instances of a pragma directive that declares the source file compiler version. It starts with the keyword "pragma" followed by an identifier, and then any combination of one or more characters until a semicolon terminates the row (see Fig. 5.1).

This declaration ensures that the contract does not suddenly behave differently with a new compiler version. In our dataset, the latest version of the compiler is the v0.4.20 and the most used version is the v0.4.18.

In fact Solidity is fast evolving and new features or functionalities of the language are introduced from time to time, rendering unstable the behavior of the code under different versions. Versions may be updated when a bug is discovered or new language constructs are needed and so on.

Fig. 5.2 reports the histogram of the number of verified contracts per compiler version. There are some specific cases that we consider useful to mention

[4] See for instance the source code of the address 0xa1877c74562821ff59ffc0bc999e6a2e164f4d87

Figure 5.1: Example of definition of the pragma version. In the first row is specified that in the following will be used the version 0.4.18 of solidity

```
pragma solidity ^0.4.18;

contract Simple {
        ...
        }
...
```

for our analysis. The only smart contract with compiler v0.1.6 is developed by Piper Merriam, the creator of Ethereum Alarm Clock (ECM) that allows users to schedule a contract call for a specified future block [5].

There is only one contract[6] that use the version v0.1.7. It is a smart contract developed by Gavin Wood, one of the Ethereum founders and the inventor of Solidity v0.1.6 and v0.1.7 have been introduced in October 2015 and November respectively. Five versions have the first transaction verified on 24.03.2016, when the Etherscan service was launched.

In order to understand how fast the developers acknowledge the updating of the language, we collected the date of release of the documentation (generally available on Github) related to a new version of the pragma and we compared it with the date of first transaction that involves a contract with the same version of pragma. In most cases, given a compiler version, the first transaction related to the smart contract (or the first activation) has been executed the same day or a few days after the release of that version in Github (23 out of 34).

In the remaining cases, the documentation of the pragma version has been made available after the date of the first usage.

Fig. 5.3 shows the history of compiler versions and the dates of the releases of the compiler new versions (red dots) and the first transaction to a smart contract characterized by the same compiler version. The figure shows a net growth reflecting the growth in use of smart contracts in 2016 and 2017.

Balances and transactions

Focusing on the smart contract balance, we found that a very few smart contracts collect the majority of the total balance of all smart contracts. In fact, the total balance among the 10175 smart contracts is about 4.64 millions Ether, but 80% of the total balance belongs to 10 smart contract alone, namely to less then 0.1% of the contracts accounts. In general smart contracts do not aim to collect Ethers, except in the case they are *wallets*. A wallet is a smart contract realized to securely collect Ethers and could implements some functions such as the

[5] with address 0x07307d0b136a79bac718f43388aed706389c4588

[6] with address 0xbF35fAA9C265bAf50C9CFF8c389C363B05753275 and contract name *wallet*

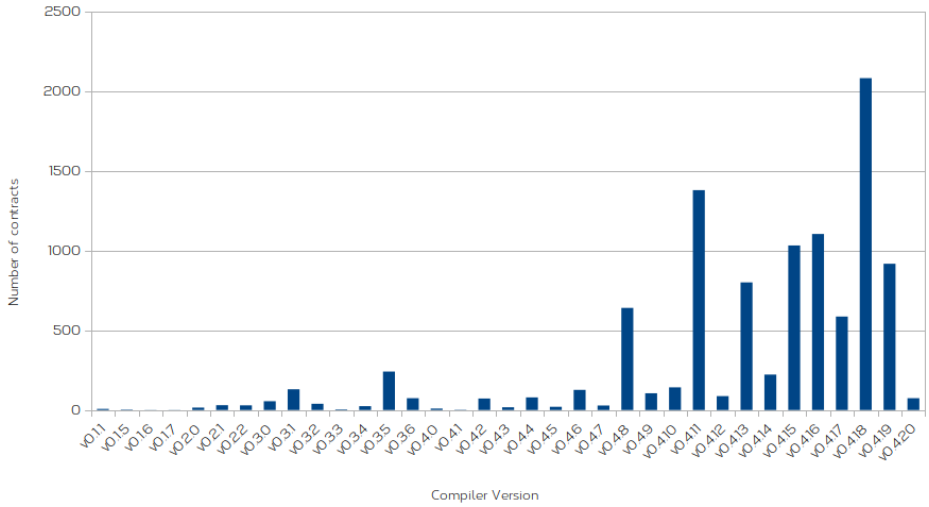


Figure 5.2: Histogram of the number of verified contracts per compiler version

multiple ownership or the escrow. Tab. 5.2 summarizes the information about these contracts.

Considering the contract name in this table, the most of them can be recognized as being wallets. In order to investigate on the distribution of the wealth, we represented in Fig 5.4 the distribution of the balance of the contracts in our dataset. The figure shows the Complementary Cumulative Distribution Function (CCDF) of the balance. The plot is in log-log scale and axes tags are in normal scale.

Fig. 5.5 shows the CCDF of the total number of transactions of smart contracts. This second distribution follows a power-law-like behavior until the values around 10^4 transactions.

We computed the correlation among the two datasets. The resulting correlation coefficient is 0.026 stating that there is no correlation between the number of transaction of a smart contract and its balance.

Despite the two distributions display similar features and show a flat-tail, there is no simple general relationship among smart contract balance and the number of transactions. In facts, as reported in Tab. 5.2, smart contracts whit high balance may hold a low number of transactions and vice-versa.

5.2.2 Measures on Smart Contracts source codes

In this paragraph we describe the analysis performed on the contracts source codes, discuss the parameters under investigation and provide the results of

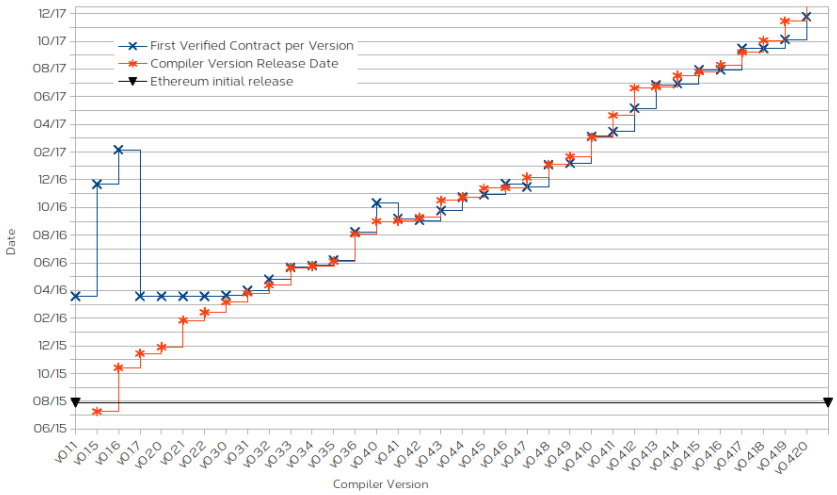


Figure 5.3: Date of release of compiler versions and the date of the first contract activation per compiler version.

Address	Balance	ContractName	TxCount	% Tot balance
0xab7c74abC0C4d48d1bdad5DCB26153FC8780f83E	1500000,00	Wallet	243	32,33%
0xf0160428a8552ac9bb7e050d90eeade4ddd52843	466648,15	TokenSales	3512	10,06%
0x7da82c7ab4771ff031b66538d2fb9b0b047f6cf9	369023,14	MultiSigWallet	226	7,95%
0xa646e29877d52b9e2de457eca09c724ff16d0a2b	269419,35	MultiSigWallet	37	5,81%
0xcafe1a77e84698c83ca8931f54a755176ef75f2c	263522,66	MultiSigWallet	335	5,68%
0xbf4ed7b27f1d666546e30d74d50d173d20bca754	232267,28	WithdrawDAO	18771	5,01%
0x851b7f3ab81bd8df354f0d7640efcd7288553419	218474,70	MultiSigWalletWithDailyLimit	72	4,71%
0xB62EF4c58F3997424B0CCeaB28811633201706Bc	203467,99	Fundraiser	75	4,39%
0x16a0772b17ae004e6645e0e95bf50ad69498a34e	91780,96	MultiSigWallet	90	1,98%
0xa4dDd3977920796BFb14cA8d0FB97491fA72a11d	79431,47	RefundVault	26	1,71%

Table 5.2: Smart Contract balance

the source code analysis. In order to analyze the contracts source code, we computed the values of the following code metrics, that can be divided in two groups. The first group represents the **Volume metrics**. The second group includes **Contract oriented metrics** which describe the logical size of the source code.

Volume metrics

M1, *Line of Code* (LoC) is the number of line of code excluding comments and blank lines. For comparison, we computed also the total number of code lines (including blanks and comments).

M2, *Comments per line* (CpL) is the ratio between lines of comment and lines of code.

Contract oriented metrics

M3, *Number of Declared Contract* (NDC) is the number of contracts (the

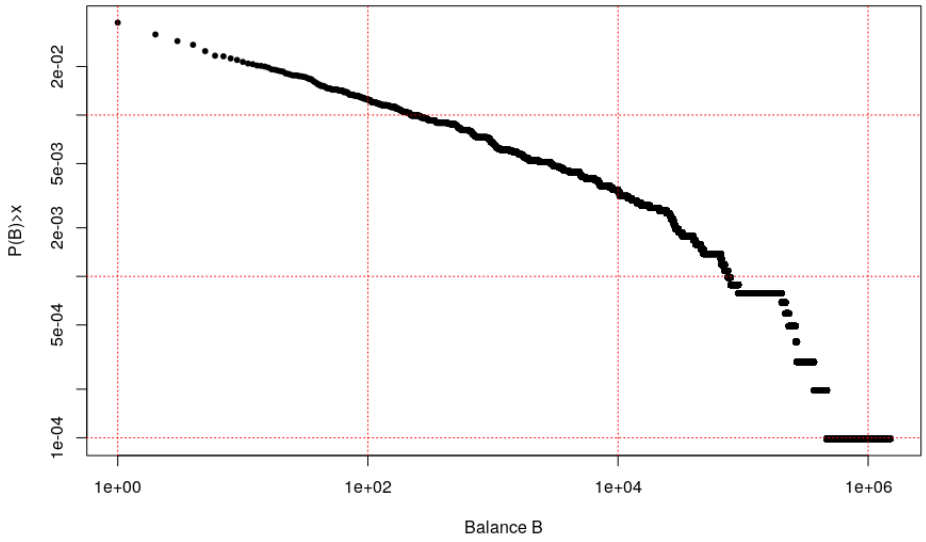


Figure 5.4: CCDF of the balance in Ethereum per contract

equivalent of classes in OO languages) declared in the source code. In solidity the declaration of a contract type is defined with the keyword *contract*. A contract can inherits from other contracts declared in the source code and can instantiate contracts, as described in Appendix 5.4.1. Fig.5.6 shows the declaration of two contracts. The contract *Derived* inherits functions and variables from the contract *Base*. Note that despite the source code can contain several contract declarations, the deployment regards only one of them (that can use or inherit the other contracts in the source code).

M4, *Number of Declared Functions* (NDF) is the number of functions declared in the source code.

Furthermore, we measure the length of the bytecode of each contract. The bytecode is the result of the compiling operation and its length depends on the content of the source code, on the version of the compiler and on the compiling optimizations.

The results are reported in Tab. 5.3. The table reports different statistics: the averages, variances, standard deviations, medians, minima, and maxima values for each metric.

All metrics displays the features typical of a flat tail distributions. They have high dispersion around the mean, with values of standard deviation comparable or even higher than the median, except for the number of lines. Such

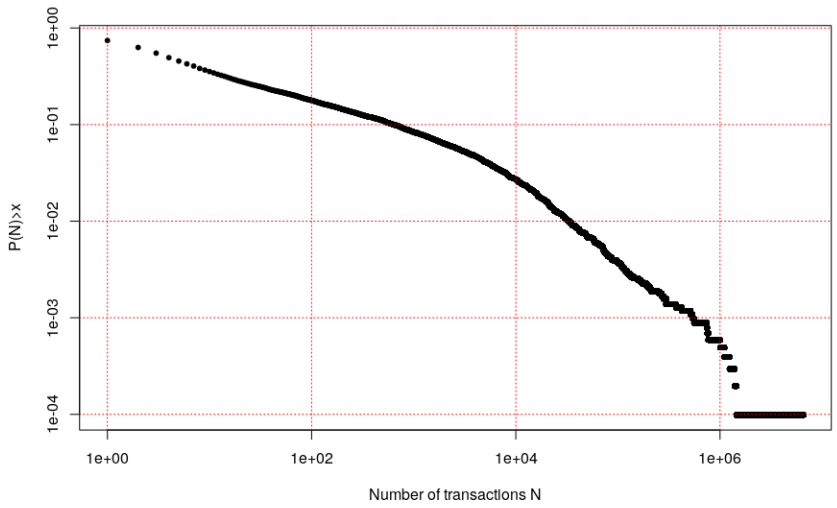


Figure 5.5: CCDF of the number of transactions per contract

```
contract Base {
        //...
}
contract Derived is Base() {
        //..
}
```

Figure 5.6: Contracts declaration in solidity

phenomenon is typically related to the presence of statistical units with very large values of the metric which contribute to rise the value of the average with respect to the median. The maximum values are an order of magnitude larger than the average, indicating the presence of outliers. The shortest bytecode has a length of 57 bytes. Considering the maximum values, the longest source code has a length over 10 times the average value in the dataset. The same can be said for metrics M1 (LoC) and M3 (NDC). The longest Bytecode is about five times the average. Max values of M2 (CpL) and M4 (NDF) are much higher than the average value.

In order to represent the distribution of metrics values in the dataset we plot the histograms for the number of lines, of the number of contract declaration per file (NDC) and of the length of the bytecode.

Fig. 5.7 shows the histogram of the number of lines. Each bin is large one hundred units. The mode of the distribution is between 100 and 200 lines. Fig. 5.8 shows the number of occurrences of the discrete values of the NDC, i.e the number of contract declarations per source code file. In this case the bin size is set to one.

The mode of the number of contract declarations per file is 1 since source codes with more than 15 contract declarations are rare. These two graphs show a fast decreasing trend, characterized by a long tail.

Fig. 5.9 provides the histogram of the length of contracts bytecodes. Each bin is large 1000 bytes. This graph presents a normal-like distribution. The mode is between 6000 and 7000 byte.

In order to investigate if and how code metrics influence each other, we computed a cross correlation matrix. Tab 5.4 reports the results of the cross correlation coefficients between code metrics, adding in the analysis the length of the bytecode and the number of transactions of each contract, that will be discussed later. The highest correlation coefficient is between the metric M1 (LoC) and the total number of lines. Also the M4 (NDF) has a high correlation coefficient with the LoC and the total number of lines. The M2 (CpL) is not correlated with the length of the code or with the M4 (NDF). This means that the number of comments on the code is heterogeneous and, in general, not proportional to the length of the source code.

The length of the bytecode is only moderately correlated both with the code length and with the number of declared functions. In addition, the number of transaction that involve a smart contract is not correlated with any code metric. This means that, for instance, highly used smart contracts have very different source codes. In the following we will confirm this results by means a deeply analyze the contracts counting the highest number of transactions.

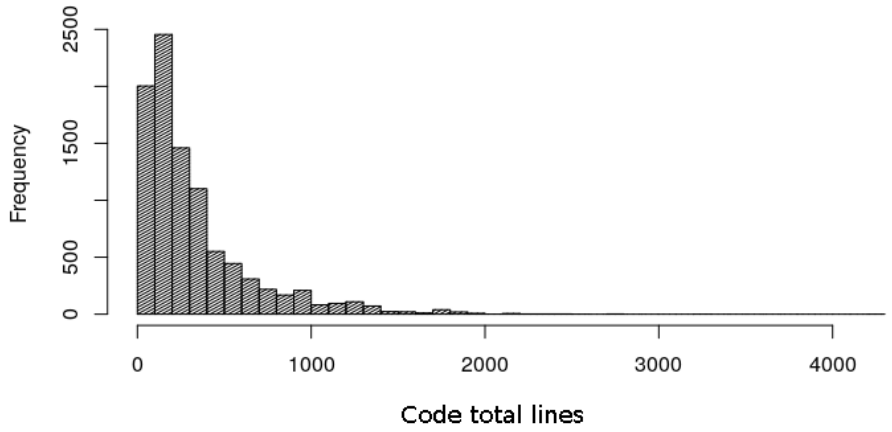


Figure 5.7: Histogram of the number of line per source code

Figure 5.8: Histogram of the number of contract declaration per source code

	Lines	M1: LoC	M2: CpL	M3: NDC	M4: NDF	Bytecode
Average	321.81	180.01	0.48	4.39	5.30	9448.99
Variance	110453.33	35452.37	0.19	13.92	49.33	47139522.84
Standard Dev.	332.35	188.29	0.44	3.73	7.02	6865.82
Median	206.00	117.00	0.42	3.00	3	7967.00
Min	0.00	0.00	0.00	0.00	0	57.00
Max	4240.00	2294.00	10.07	36.00	125.00	50607.00

Table 5.3: Statistics on code metrics computed among 10174 contract source codes

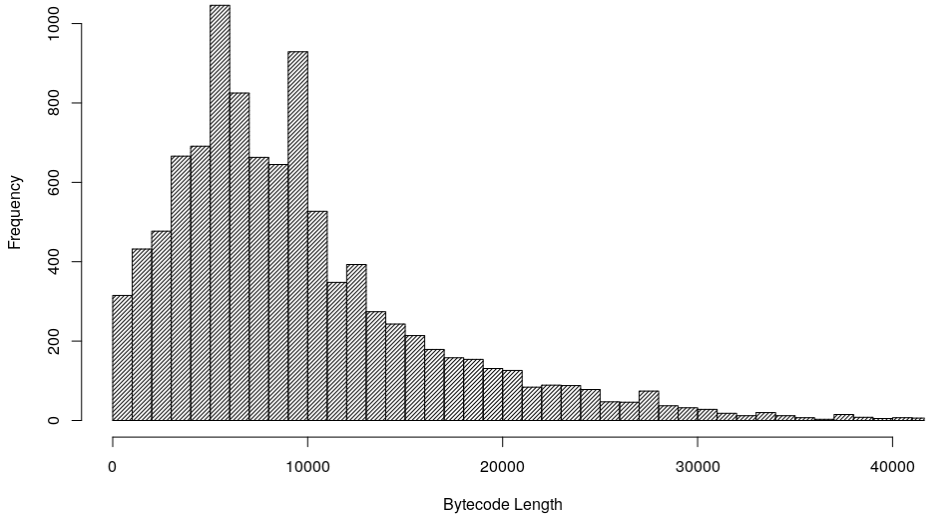


Figure 5.9: Histogram of the length of bytecodes in byte

5.3 Detailed analysis on the top 20 used Smart Contracts

We present a detailed analysis of the twenty smart contracts with the largest number of transactions (Tx count). Tab. 5.5 lists these contracts. Such contracts can be classified according to their typology [6] in five categories: Wallet, Financial, Game, Library, and Notary. Wallet contracts are characterized to be deposits of ether and they usually have a high balance. Financial contracts aim to provide functions useful to manage financial goods such as tokens. Game contracts implement lotteries and digital collections. Library contracts are developed and deployed to provide functionality useful for other contracts (i.e maths libraries). Finally, Notary contracts take advantage on the blockchain characteristics to

	Line Total	Bytecode	M1: LoC	M2: CpL	M4: NDF	Tx Count
Line Total	1	0,52	0,95	0,22	0,91	0,02
Bytecode	0,52	1	0,56	0,03	0,52	0,02
LoC	0,95	0,56	1	0,02	0,91	0,02
CpL	0,22	0,03	0,02	1	0,12	-0,01
NDF	0,91	0,52	0,91	0,12	1	0,02
Tx Count	0,02	0,02	0,02	-0,01	0,02	1

Table 5.4: Matrix of the cross-correlation coefficients between metrics and indicators computed among 10174 contracts

record the agreements between parts.

We selected the top twenty smart contracts in order of number of transactions (Tx count), see Tab. 5.5.

In the following we provide a short description for each of them.

#	Address	Tx count	Contract name	Project	Typology	Standard
1	0x8d12a197cb00d4747a1fe03395095ce2a5cc6819	6562254	Etherdelta	Ether Delta	Wallet	
2	0x03df4c372a29376d2c8df33a1b5f001cd8d68b0e	1450979	Bitocinereum	Bitcoinereum	Finantial	ERC-20
3	0x06012c8cf97bead5deae237070f9587f8e7a266d	1390301	KittyCore	CryptoKitties	Game	ERC-721
4	0xE94b04a0FeD112f3664e45adb2B8915693dD5FF3	1241476	ReplaySafeSplit	BitTrex	Library	
5	0x6090a6e47849629b7245dfa1ca21d94cd15878ef	1103826	Registrar	Ethereum Name Service	Notary	
6	0x86fa049857e0209aa7d9e616f7eb3b3b78ecfdb0	1005345	Dstoken	EOS	Financial	ERC-20
7	0xa3c1e324ca1ce40db73ed6026c4a177f099b5770	768834	Controller	BitTrex	Library	
8	0xd26114cd6ee289accf82350c8d8487fedb8a0c07	750295	OMGToken	OmiseGO	Financial	ERC-20
9	0xf230b790e05390fc8295f4d3f60332c93bed42e2	742060	TronToken	TRON	Financial	ERC-20
10	0x93e682107d1e9defb0b5ee701c71707a4b2e46bc	557685	MCAP	MCAP Labs	Financial	ERC-20
11	0xa74476443119a942de498590fe1f2454d7d4ac0d	535935	GolemNetworkToken	Golem Network	Financial	ERC-20
12	0xb1690c08e213a35ed9bab7b318de14420fb57d8c	514167	SaleClockAuction	CryptoKitties	Game	
13	0xaBbb6bEbFA05aA13e908EaA492Bd7a8343760477	421810	ReplaySafeSplit		Library	
14	0xd0a6e6c54dbc68db5db3a091b171a77407ff7ccf	370266	EOSSale	EOS	Financial	ERC-20
15	0x744d70fdbe2ba4cf95131626614a1763df805b9e	296699	SNT	Status Network	Financial	ERC-20
16	0x9a642d6b3368ddc662CA244bAdf32cDA716005BC	296623	HumanStandardToken	QTUM	Financial	ERC-20
17	0xb97048628db6b661d4c2aa833e95dbe1a905b280	279107	PayToken	TenXPay	Financial	ERC-20
18	0xece701c76bd00d1c3f96410a0c69ea8dfcf5f34e	269043	Etheroll	Dice	Game	ERC-20
19	0xaa1a6e3e6ef20068f7f8d8c835d2d22fd5116444	253559	ReplaySafeSplit		Library	
20	0x0D8775F648430679A709E98d2b0Cb6250d2887EF	213364	BAToken	Brave Browser	Financial	ERC-20

Table 5.5: List of the twenty smart contracts under examination.

5.3.1 Smart Contracts description

EtherDelta. It is tagged as etherdelta_2 on Etherscan and is the smart contract executed to store and transfer tokens with Ethereum wallets, in the cryptocurrency exchange EtherDelta[7]. EtherDelta is in fact one of the most used decentralized trading platform for Ethereum and manages ERC20 compatible tokens. In order to trade on EtherDelta an user must create a wallet or use an existing wallet which interacts with this smart contract.

[7]https://etherdelta.com

Bitcoinereum[8] is the fist Bitcoin-like mineable Ethereum ERC20 Token and, through the Bitcoin Supply mechanism, enables a bitcoin-like currency to run on the ethereum blockchain. To bring the Bitcoin supply mechanism into Ethereum, Bitcoins enter Ethereum blockchain in form of ERC20 tokens.

KittyCore and **SaleClockAuction** are two smart contracts belonging to one of the most popular applications of Ethereum blockchain, CryptoKitties, the game in which users can buy, sell, and breed cartoon kittens. The application was launched on November 28 and in a little more than a month these two contracts (out of a total of 17 smart contracts developed in this project) have been responsible for 6,2% of all transactions on the ethereum network.

ReplaySafeSplit. In the set of 20 smart contracts, the contract name *ReplaySafeSplit* appears three times. The functionality of these three smart contracts are very similar: they are used to split Ether funds in several addresses and protect against replay attacks between Ethereum Classic (ETC) and Ethereum (ETH). As a result of the hard fork of the Ethereum network (on July 20th, 2016), holders of an ETH fund prior to the 1920000 block ended up with two funds on the same address and therefore found themselves having ETH and ETC in equal quantities: the ETHs on the network *support-dao-fork* network and ETC on the *oppose-dao-fork* network. The two coins are still linked to each other: a move of the ETHs move also ETC and vice versa. ReplaySafeSplit is used to separate ETH pre-forks on two new and different addresses, one specific for ETH post-fork and another one specific for ETC. ReplaySafeSplit recalls the fork oracle smart contract[9]. A specific version (labeled Bittrex_2[10] on Etherscan) is used on the Digital Currency Exchange Bittrex (https://bittrex.com/) with the same capabilities.

Registrar. It is one of two smart contracts that compose the core of the Ethereum Name Service[11] (ENS), an extensible naming system based on the Ethereum blockchain. *Registrar* owns a domain and, according to the rules written in the contract, issues subdomains of that domain to users. For each domain and subdomain *Registrar* memorizes the owner (an external account, typically an user or an other smart contract), the resolver and the time-to-live for all records.

DSToken (labeled EOSTokenContract) and **EOSSale** (labeled EOSCrowdsale) are smart contracts of the famous Infrastructure for Decentralized Applications EOS[12] that introduces a blockchain architecture designed to allow the vertical and horizontal scaling of decentralized applications. EOSTokenContract is

[8] http://www.bitcoinereum.com/

[9] Having address *0x2bd2326c993dfaef84f696526064ff22eba5b362*

[10] Having address *0xE94b04a0FeD112f3664e45adb2B8915693dD5FF3*

[11] https://ens.domains/

[12] https://eos.io/

in fact the token of the EOS ICO that aims to finance *block.one*, the platform that, based on scalability, flexibility and usability criteria, intends to make the blockchain technology accessible to businesses which, in this way, can memorize smart contracts on blockchain. EOS tokens are ERC-20 compatible tokens distributed on the Ethereum blockchain under a related ERC-20 smart contract. EOSTokenContract handles all the logic of ownership and transfers; Instead, EOSCrowdsale manages all the logic of contributions, periods and claiming.

Controller. It is one of the two smart contracts that implements the core of Bittrex (the other one is ReplaySafeSplit, as previously described) and manages the exchange of cryptocurrency. The main function of Controller is MakeWallet that is used to create ETH wallets and has control functions of owner and destination.

OMGToken (labeled OmiseGoToken). It is the token of OmiseGO (OMG)[13], currently one of the most famous cryptocurrencies of the ICO market which aims to simplify and make cryptocurrency transactions almost instantaneous. OMG is a public Ethereum-based financial technology for use in mainstream digital wallets. At the same time it is an e-wallet and payment platform acting through assets and crypotocurrencies. The advisors of OMG are almost all from the Ethereum foundation. OMGToken is an ERC20 basic token on Ethereum. Once the OMG blockchain is created, the OMG tokens are transferred to this new blockchain.

TronToken (TRX) is the token of the TRON ecosystem[14] is the blockchain-based decentralized protocol and open-source platform that aims to construct a global free content entertainment system and provides funtions of credit sharing and payment for many services such as online casinos, mobile games, live shows, social networks. It is based on an ICO and is a ERC20 standard Ethereum token. Starting from December 2017 it is the second most used token with market capitalization that rose from $477 million to $3 billion just within 5 days (from December 13 to December 18).

MCAP[15] uses the ERC 20 protocol for peer-to-peer transactions and is the token of MCAP Labs ecosystem. Its ICO was launched by BitcoinGrowthFund (BGF) with the aim to invest in the mining of cryptocurrencies, especially Bitcoin. The algorithms developed by BGF identifies which cryptocurrency must be mined at any time to maximise profit. The smart contract has five functions: *mcap* to initialize contracts with initial supply tokens to the creator of the contract; *transfert* that sends coins; *approve* which allows another contract to spend some tokens in the owner behalf; *approveAndCall* that in a single transaction

[13]https://omisego.network/
[14]https://tronlab.com/en.html
[15]https://bitcoingrowthfund.com/mcap)

approves and communicates the approved contract and finally *transfer*, called from a contract that attempts to obtain the coins.

GolemNetworkToken (GNT). It is the token of the Golem Network project[16], a decentralized distributed network of computers in which users can sell and buy computing power. Through Golem Network users can decentralize all the tasks thanks to the computer of another user connected to the network, or sell the computing power of their own computer to help those who need it. The GNT Token is partially-ERC20-compliant because it does not implement the *approve*, *allowance*, and *transferFrom* functions, and the *Approval* event. On the Ethereum blockchain, the crowdfunding start block is 2607800 and it was launched in the 11th November 2016. The main functions of this smart contract are: management of payments for resource usage and remuneration for software developers; submitting of deposits by providers and software developers and participation in the process of software validation and certification.

SNT (labeled StatusTokenContract) is the token of Status Network[17], an open source messaging platform that includes a DApps browser, a messenger, a wallet, and can be described as a mobile operating system to access Ethereum from anywhere. It is therefore a peer-to-peer messaging app without central server to store private data or conversation. Status Network aims, through the use of blockchain technology, to remove centralized third-party applications or middlemen in the people communications. The entire project combines 10 smart contracts. SNT is a ERC20-compliant token and derives from the MiniMe Token[18] that allows for token cloning (forking). SNT has a modular architecture and is used to power the Status Client, including some fundamental utilities such as a Decentralized Push Notification Market, the Governance of the Status client, Username Registration using ENS, and so on.

HumanStandardToken (labeled QtumTokenContract) is the token of the Qtum project[19], a Value Transfer Protocol (VTP) blockchain. Qtum is therefore a smart contract ecosystem for businesses that want to run decentalized apps blockchain-based, executable on mobile devices. The aims is to turn any human-readable agreements into a smart contract. Qtum uses Bitcoin's UTXO model in order to allow the contact execution also on mobile devices. HumanStandardToken is a ERC20-compliant and includes 3 contracts called *Token*, *StandardToken* and *HumanStandardToken*. The contract Token modifies ERC20 base standard in the totalSupply function because a getter function for the totalSupply is automatically created.

[16] https://golem.network/
[17] https://blog.status.im
[18] https://github.com/Giveth/minime
[19] https://qtum.org/

PayToken (labeled TenXContract) is the token of the TenXPay (TENX)[20] project that aims to solve one of the major problems of the cryptocurrency market: how you spend cryptocurrencies in the real world. It is a portfolio-bank based on cryptographic assets with a debit card. With an encryption-protected off-line multi-asset instant transaction network, the service supports unlimited cryptographic assets (initially only supports ETH, ERC20, DASH and BTC). Users can choose which cryptographic asset to use for payment by debit card and ATM withdrawals. The contract calls a function named MakeWallet. PayToken is a ERC20-compliant token. Users can store PayToken in any ERC20-enabled wallet.

Etheroll (labeled *Etheroll_old_3*) is a smart contract of the Ethereum Dice game project and is used to place bets on dice games using Ethers with no deposits or sign-ups. The dice rolls are random and cryptographied in a secure way, thanks to the Ethereum blockchain. In order to obtain the final results of dices, the Etheroll smart-contract invokes the API of Random.org[21], performs sha3() encryption on its result and on IPFS address of the TLSNotary proof.In the following we will provide more detailed information of this Smart Contract.

BAToken (labeled BatTokenContract - BAT) is the token of the new Brave browser, created by Brendan Eich, creator of Javascript and cofounder of Mozilla. Users are paid in digital currency to view advertising or to click on the advertising banners. BAT is ERC20-compliant.

Most of the smart contracts listed in Tab.5.5 are financial contracts, and the description highlight the economic interest behind the contract. We found that the several of the described projects makes use of an ICO to fund, and consequently promote, the business idea. These projects are Etherdelta, EOS, OmiseGo, TRON, MCAP, Golem, Status, Qtum, TENX, Etheroll, and Brave Browser. One of the success factors of an ICO is the team size and its composition[35, 45]. So, projects which resort to an ICO are more likely supported by a convincing and well-formed development team.

5.3.2 Smart Contracts usage indicators

In this section we define empirical indicators useful to describe smart contracts usage from a statistical point of view. We identified various *usage indicators* characterizing how and to which extent smart contracts code is called or used in the application of the Ethereum blockchain. We divided the usage indicators in two groups. A first group, characterizing ***blockchain interaction***, describes the occurrences in the blockchain of contract-related operations. It contains the following indicators.

[20] https://www.tenx.tech/

[21] https://api.random.org/json-rpc/1/invoke

I1, *Number of transactions* (Tx Count): the overall number of transactions (both in input and in output) involving the contract.

I2, *Transaction per day* (Tx/day): the number of transactions normalized with respect to the days of activity (DoA) namely the elapsed time in days between the contract creation and its last transaction.

The selected indicators can be easily extracted from the blockchain data and offer a snapshot of the impact that the contract had on the blockchain.

A second group, ***developers' interaction***, includes indicators describing the evolution of a contract in terms of its development history and of its reuse to create new contracts. It contains the following indicators.

I3, *Number of Deployments*: counts the total number of contract versions deployed in the Ethereum blockchain and verified using the etherscan service. Consider that each deploy involves a cost in Ether. We compared this indicator with the total number of contracts having the same name.

I4, *Number of versions*: counts the number of versions of a smart contract which are used within the same project. This indicator consider only versions of the contract that have been active in a certain period of time and it does not count contracts with a low number of transactions (less than 100). It indicates a continuous activity of development.

I5, *Number of code reuse*: counts the number of new contracts created reusing another smart contract source code belonging to a different project. As the previous indicator, we excluded from this analysis contracts having a low number of transactions (less than 100).

We also take into account the balance of the smart contracts (i.e. the amount in Ether associated to the contract address). But we don't consider it as a good usage indicator because it increases and decreases over time, and, furthermore, only few contracts are used as a deposit of Ether, see subsection 5.2.1.

Tab 5.6 reports the value of the usage indicators for the twenty smart contracts analyzed together with the compiler version. Results show that these contracts are characterized to be involved every day in a big number of transactions, and to have a null balance in the most of the cases. On the other hand, the indicators describes the heterogeneity in the usage of these smart contracts in terms developers' interactions.

Blockchain interaction

The twenty smart contracts chosen have the highest value of I1 (TxCount), namely the total number of transactions. A transaction that involves a smart contract is also called *message* and contains the instructions needed to execute a function of the contract. It involves a change of blockchain data (i.e its state). Consider that every blockchain change has a cost, that depends on the computa-

#	Contract name	I1: Tx count	I2: Tx/day	Balance	DoA	I3: NoD / Tot	I4: NoV	I5: RoC	Compiler
1	Etherdelta	6562254	20191.55	4.61e+14	325	8 / 8	5	0	v0.4.9
2	Bitcoinereum	1450979	17694.87	0	82	1 / 1	1	0	v0.4.17
3	KittyCore	1390301	42130.33	70.9	33	1 / 1	1	0	v0.4.18
4	ReplaySafeSplit	1241476	2369.23	0	524	6/6	1	0	v0.3.5
5	Registrar	1103826	4580.19	0,	241	3 / 11	2	0	v0.4.10
6	DSToken	1005345	5376.18	1373.27	187	10 / 10	1	6	v0.4.11
7	Controller	768834	5571.26	0	138	4 / 16	2	0	v0.4.11
8	OMGToken	750295	4191.59	0	179	1 / 1	1	0	v0.4.11
9	TronToken	742060	5936.48	0	125	1 / 1	1	0	v0.4.16
10	MCAP	557685	2523.46	0	221	3 / 4	1	0	v0.4.11
11	GolemNetworkToken	535935	1201.65	0	446	3 / 3	1	0	v0.4.4
12	SaleClockAuction	514167	15580.82	7.09	33	1 / 1	1	0	v0.4.18
13	ReplaySafeSplit	421810	876.94	0	481	6 / 6	1	0	v0.3.5
14	EOSSale	370266	1969.50	0	188	1 / 1	1	0	v0.4.11
15	SNT	296699	1521.53	0	195	1 / 1	1	0	v0.4.11
16	HumanStandardToken	296623	1694,99	0	175	44 / 44	1	11	v0.4.10
17	PayToken	279107	1409.63	0	198	1 / 1	1	0	v0.4.11
18	Etheroll	269043	1724.63	0.79	156	20 / 21	4	0	v0.4.10
19	ReplaySafeSplit	253559	479.32	0	529	6 / 6	1	2	v0.3.5
20	BAToken	213364	987.80	0	216	1 / 1	1	0	v0.4.10

Table 5.6: Contract usage indicators

tional effort needed to execute the transaction. This selected contracts are those that have involved many changes of state of the Ethereum blockchain.

The average number of transactions for the entire dataset of smart contracts downloaded from etherscan.io is 3019

On Tab 5.6 the contract *BAToken*, in position twenty, has a number of transactions over seventy times higher than the average value of the complete dataset. The first contract *Etherdelta* has been involved in a transaction twenty thousand times more than the average usage. In total, these contracts are about the 0.2% of the total set but are involved in about the 61.3% of the total number of transactions. These numbers are in line with the distribution of the number of transaction previously reported in Fig 5.5. The enormously larger usage of this subset of smart contracts, explains the presence of a strong flat-tail in the statistical distribution reported in Fig 5.5 and justifies our choice of examining in detail the most used smart contracts.

The values of I2 (Tx/day) are a normalization of the values of the indicator I1, obtained dividing them by the effective usage time. This allows us to compare smart contracts at terms of frequency of interactions, although them have been deployed in the blockchain in different times.

The number of usage days is the number of days between the first and the last transaction of the contract. All the contracts under examination are characterized by a high value of I2, from a minimum of 479.32 up to 42130.33 transactions per day. Contracts with a high value of I2 can be considered either needful contracts in the Ethereum ecosystem, or contracts that have had a extraordinary popularity in their activity period.

It is relevant the case of the contract *KittyCore*, that, as described before, is a game. Considering the value of the indicator I2 of that contract, that is the highest value of transaction rate among the twenty selected contracts, we found that this contract is involved in about 30 transactions per minute. It is associated to the contract *SaleClockAuction* that also has a very high value of I2.

In Tab. 5.6 we reported the Days of activity (DoA) for each contract. The contract *KittyCore* counts only 33 DoA. The longest-running contracts are the *ReplaySafeSplit* family (all exist and have been used for more than a year), followed by *Etherdelta* and *Registrar*.

For what concerns the balances, only five out of twenty smart contracts have non null balances and only two are significantly high. In particular, the two contracts are *Etherdelta* and *Dstoken*. As already described, the first is a popular wallet. The second is a financial token born to fund a nascent blockchain. The analysis shows that the most of the twenty smart contracts does not aim to collect ether inside.

Developers' interactions: versions and reuse of code

We examined next the interaction of developers with smart contracts through the blockchain. According to the indicators defined, we analyze the number of deployments (I3), the number of versions (I4) and the number of times of code reuse (I5) for each smart contract.

One of the objectives of our exploratory study is to investigate if smart contracts have been implemented passing through a code development process. For this reason, we checked the history of each contract, examining the presence of past versions and if improved versions have been deployed into the Ethereum blockchain.

We started our investigation filtering the dataset by the *contract name*, and then, since different smart contracts can have the same name, by means of an accurate analysis of the lines of code, we extracted the set of contracts referable to the same development history. The analysis of the source code allows to identify different contracts holding the same name. These contracts have been analyzed as different contracts. For computing the indicator I3 (Number of Deployments), we consider the number of contracts referable to the same source code. In Tab 5.6 we reported the I3 indicator (NoD) together with the number of contracts with the same name (Tot).

We defined a new *version* of a smart contract each new smart contract that once uploaded in the blockchain replaces the previous one in terms of blockchain interactions. The new version could contain code changes. We reported the values of the usage indicator I4 (Number of Version) in Tab 5.6 as NoV.

Finally, for the indicator of Reuse of Code (I5) we considered *reuse* of the code of a smart contract the cases where the source code of the contract is used to implement a very similar contract that has the same name but is referable to a different project (for instance to implement a new token). The number of reuse of code is reported in Tab 5.6 as RoC.

Evaluating version or reuse of code we did not consider those smart contracts which have a low number of transactions (less than one hundred), because they are rarely operational and could be only tests. So, they do not represents properly a new version or a reuse.

In the following we analyze some contracts, focusing on the developers interaction, namely in terms of versions and reuse of code.

EtherDelta has eight contracts with the same name. It has five different version, and the last one created is the most active. Two of the old versions are still used but they have a low number of transactions, about one per week. We notice that both of them have a lower amount of Ethers than the first one, therefore we can suppose that these smart contracts were used only by the contract developers and not by final users.

ReplaySafeSplit has three different smart contracts in the top twenty and all of them are active, and are involved, on average, in a transaction every five minutes. By analyzing the code and the project's history, we found that they have a different usage for different projects. We classified them as reuse of smart contract code for the smart contract, as reported in the row 19 in Table 5.7. The other two (rows 4 and 13) are the examples of the reuse of code of the aforementioned smart contract and they do not have new versions and are not reused.

We found eleven contracts named *Registrar*. Analyzing these contracts we found that only 3 source codes can be evaluated in terms of the indicator I3, and the remaining have a completely different code. In addition, the discarded smart contracts have been involved in less than 10 transaction and these were probably tests. Only one out of three is active and belongs to the top twenty. We found one old version of this contract, and no reuse of code.

DSToken has ten records and we found six reuse of code, each of which derives from the smart contract in table 5.5, line 6. To confirm this, we checked on Etherscan that different labels are associated to these (still active) six addresses.

By considering a transaction number higher than one hundred and with reference to Etherscan, we found for *HumanStandardToken* 11 documented reuse of code. The remaining contracts are not evaluated in terms of reuse or versions of code given the low number of transactions (in the order of units). However the code in these contracts have the same functionalities. We can state that, among those investigated, these two smart contracts are undoubtedly the most popular in terms of reuse of code because they were used as a reference

for different projects.

The *Controller* has two versions related to the project Bittrex. We detected a situation similar to *Registrar*: only 4 smart contracts belong to the project analyzed. The remaining 12 have a number of transactions in the order of units and a source code completely different. To be more precise we respectively found: three couples of smart contracts and 6 smart contracts with the same source code or a different version of this.

Considering the records of *MCAP* we did not find different versions or reuse of code. One of the records has a different code and the other two are probably tests because of the low number of transactions (in the order of units). Similarly, *GolemNetworkToken* has no new versions or reuse of code.

Finally, all but one of the smart contracts named *Etheroll* found in the dataset are related to the same project. We considered four of these as different versions and the remaining as tests because of the low number of transactions. Actually, the previous versions are not still used. This phenomena, in terms of source code improvement, is similar to *EtherDelta* case. Referring to Table 5.6 the *Etheroll* smart contract in line 18 is no longer used and it has been replaced by the current active version[22].

We also analyzed the declared pragma version. We found that the in cases of different version of the smart contract, the pragma version of new versions is generally updated with respect to the previous one, but not always correpsonds to the most updated version of the language.There is only one case, the Etheroll Smart Contract[23], that does not update the version with respect to the previous one (v04.10) even if the next version (v04.11) has already been released. In all cases of Smart Contract updating, the developers have deployed the new version in the blockchain, supporting the related costs.

5.3.3 Code metrics

For every smart contract source code listed in Tab. 5.5 we computed the code metrics described in Section 5.2.2 and the following additional code metrics.

M5, *Line code per Function* (LpF): it is the average number of lines of code written to implement a function.

M6, *Max cyclomatic complexity* (MCC): it is the max value of the McCabe cyclomatic complexity among the cyclomatic complexities of all functions in the contract.

M7, *Sum of cyclomatic complexities* (SCC): it is the sum of the McCabe complexity of each function in the source code. That value depends on the number

[22] Having address 0xD91E45416bfbBEc6e2D1ae4aC83b788A21Acf583

[23] Having address 0xece701c76bd00d1c3f96410a0c69ea8dfcf5f34e

of function in the contract. The average cyclomatic complexity among a smart contract source code is equal to the division between its values of M7 and M4.

The last two metrics are **Complexity Metrics**. In particular, the cyclomatic complexity measures the number of linearly independent paths through a function in the source code. We computed the cyclomatic complexity according to McCabe definition[56] and using a commercial software[24].

We report in Tab. 5.7 the resulting values for the metrics from M1 to M7 computed for each smart cSontract source code belonging to the selected set.

Contract Name	M1: LoC	M2: CpL	M5: LpF	M3: NDC	M4: NDF	M6: MCC	M7: SCC
Etherdelta	232	0.18	5	7	32	4	60
Bitcoinereum	132	0.03	7	1	17	5	23
KittyCore	971	0.74	8	16	69	7	82
ReplaySafeSplit	20	0.2	7	2	2	3	4
Registrar	324	0.64	6	3	29	5	58
Dstoken	376	0.12	4	9	51	4	64
Controller	170	0.01	4	6	23	4	32
OMGToken	185	0.71	7	10	25	2	36
TronToken	131	0.03	7	1	17	5	23
MCAP	62	0.4	8	2	6	5	15
GolemNetworkToken	161	0.49	11	3	11	6	42
SaleClockAuction	300	0.68	7	6	21	2	23
ReplaySafeSplit	28	0.43	4	3	2	5	6
EOSSale	584	0.13	5	11	68	4	87
SNT	850	0.8	9	14	57	4	94
HumanStandardToken	70	0.93	6	3	8	2	11
PayToken	253	0.87	6	10	31	3	39
Etheroll	1112	0.41	4	5	52	3	64
ReplaySafeSplit	17	0.24	3	2	2	3	4
BAToken	129	0.31	8	4	12	7	30
Average Values	305.35	0.4175	6.3	5.9	26.7	3.85	39.5

Table 5.7: Code metrics results in the twenty selected source codes

Results in Tab. 5.7 allow us to compare the value of metrics from M1 to M4 for the overall set of contracts presented in Tab. 5.3 with those of the top twenty contracts having the higher number of transactions and representing the contracts having the larger interaction in the blockchain.

We found that the source codes of the top twenty contracts have, on average, a value of M1 (LoC) equal to 305.35, that is higher, in average, than the value of M1 computed on the full dataset (180.01 lines). In particular, exactly half of the source codes have a value of M1 higher than 180. Results confirms that the number of transaction and number of line of code are not correlated.

Analyzing M2 in Tab 5.7, namely the number of comments per line of code , we can observe a high variability of the results. Values ranges from one line of

[24]We computed the cyclomatic metrics using *Understand*, that is a *scitools* software. These cyclomatic metrics are described in https://scitools.com/support/cyclomatic-complexity/

comments every one hundred lines of code to about one line of comments per one line of code. On average, there are 0.41 comments per line of code and this number is a little lower than the average value of the full dataset (0.49).

Considering the number of declared contracts measured by M3, and the number of declared functions measured by M4, we can observe, on average, higher values of declarations in comparison with the global results. In particular the average value of M3 is 5.90 (the average value of the full dataset is equal to 4.39) and M3 values range between a minimum of 1 to a maximum of 16, and half of the smart contracts have M3 greater than 4. High values of M3 means that source codes of smart contracts are written exploiting the inheritance mechanism. The source code of KittyCore (having the maximum number of declared contracts) is a typical example of systematic use of the inheritance. The structure of this contract is reported in Appendix 5.4.3.

The average value of M4 is 26.70, and it is about five times the average value of the full dataset (that is equal to 5.30). This means that, on average, the twenty selected contracts implements an higher number of functions. In facts, 17 out of 20 declare more than 5 functions. Values of M4 have minimum 1 and maximum 69. The highest values of declarations is related to the contract in third position, namely *KittyCore* that implements a large number of functionalities. High values of declarations characterize also some tokens (i.e *Dstoken, EOSSale, SNT*. These contracts improve the functionalities defined in the ERC-20 standard, by adding specific and customized features.

Analyzing the results of the metric M5 (Number of lines per function), computed only for the set of twenty contracts, we can observe that the functions have, on average, 6.30 lines. The contract GolemNetworkToken has the larger value. Considering the ERC-20 compliant contracts, the variability of the functions length suggests that tokens are not all implemented in the same way.

Considering the cyclomatic complexity metrics, M6 and M7,we can observe that the majority of the source codes has a maximum complexity (M6, MCC) lower than four (or in other words, it is hard to find functions with cyclomatic complexity greater that 4).

Both the source code of *KittyCore* and the source code of *BAToken* have a function characterized by the highest value of cyclomatic complexity equal to seven. See in Appendix 5.4.3 for the function of *KttyCore* which has the maximum cyclomatic complexity.

The lower value of M6 is equal to 2 and characterizes the contracts *OMGToken, SaleClockAuction* and *HumanStandardToken*.

Considering M7, namely the sum of the cyclomatic complexity of each function declared in the source code, the three most complex contracts belong to *SNT, EOSSale, and KittyCore*. The three versions of *ReplaySafeSplit* are characterized by very low value of M7 because the low number of function. The codes

reported in the 5.4.2 shows that this contract has only two functions.

5.3.4 Analysis of results

	M1 LoC	M2 CpL	M5 LpF	M3 NDC	M4 NDF	M6 MCC	M7 SCC
M1 LoC	1.00	0.25	0.03	0.68	0.88	0.14	0.82
M2 CpL	0.25	1.00	0.34	0.40	0.12	-0.17	0.15
M5 LpF	0.03	0.34	1.00	0.09	-0.04	0.59	0.13
M4 NDC	0.68	0.40	0.09	1.00	0.83	0.13	0.78
M5 NDF	0.88	0.12	-0.04	0.83	1.00	0.23	0.94
M6 MCC	0.14	-0.17	0.59	0.13	0.23	1.00	0.36
M7 SCC	0.82	0.15	0.13	0.78	0.94	0.36	1.00

Table 5.8: Cross Correlation Matrix of source code metrics

Tab 5.8 shows the correlation matrix between metrics computed among the twenty selected contracts. As we can expect, metrics values of M1, M3, M4 and M7 are mutually correlated and this means that the longer is the code, the more complex is the program. In particular, the sum of the cyclomatic complexity (M7) and the number of declare function (M4) have a correlation factor that represent a strong linearity of the ratio between the two metrics. The values of M5 and M6 have a average-high value of correlation factor. The metric M2 does not present particular cases of correlation.

In order to analyze the relationship between source codes and the use of the contracts, we analyzed if results of the applied metrics are correlated with the usage indicators. We studied if and how the two analysis are correlated computing the correlation matrix in Tab. 5.9 that reports the correlation factors computed for the twenty selected contract.

	I1 Tx count	I2 Tx/day	I3 NoU
M1 LoC	-0.05	0.35	0.00
M2 CpL	-0.25	0.05	0.26
M5 LpF	-0.11	0.14	-0.28
M3 NDC	0.06	0.41	-0.21
M4 NDF	0.08	0.39	-0.13
M6 MCC	0.13	0.39	-0.40
M7 SCC	0.18	0.31	-0.18

Table 5.9: Correlation coefficients between usage indicators and code metrics

We discovered that the two analysis are few correlated. In particular, the indicator I1 (Tx count) is weakly correlated with all the metrics. The indica-

tor I2 (Tx/day) shows a interesting moderate correlation with the metrics that describes the size and complexity of the source code (M1, M3, M4, M6 and M7).

5.4 Sample of Smart Contracts source codes

5.4.1 Crowdsale

A portion of the source code of the contract Crowdsale deployed at the address: 0xa1877c74562821ff59ffc0bc999e6a2e164f4d87. This smart contract is named "Crowdsale".

The source code includes two contract definitions. The first contract is *token* and the second is *Crowdsale*. The contract token is an interface. In an interface, all functions are only declared but not implemented. The contract *Crowdsale* declare an "instance" of the the contract *token* called *tokenReward* and assign to it the contents of an already deployed smart contract. In the source code, the instance of a contract can be used to execute its functionalities.

```
pragma solidity ^0.4.8;
contract token {
    function transfer(address receiver, uint amount){ }
    }

contract Crowdsale {
    uint public amountRaised;
    uint public resAmount;
    uint public soldTokens;
    mapping(address => uint256) public balanceOf;

    // initialization
    ...
    token public tokenReward =
      token(0x2Fd8019ce2AAc3bf9DB18D851A57EFe1a6151BBF);
    /*addressOfTokenUsedAsReward*/
        ...
   }
```

5.4.2 ReplaySafeSplit

The original source code of ReplaySafeSplit. This code has the lowest sum of cyclomatic complexity belong the set of the twenty most used smart contracts.

Its source code is available on etherscan.io at the address: 0xE94b04a0FeD112f3664e45adb2B8915693dD5FF3

```
contract AmIOnTheFork {
```

```
    function forked() constant returns(bool);
}
contract ReplaySafeSplit {
    // Fork oracle to use
    AmIOnTheFork amIOnTheFork =
        AmIOnTheFork(0x2bd2326c993dfaef84f696526064ff22eba5b362);
    event e(address a);
    // Splits the funds into 2 addresses
    function split(address targetFork, address targetNoFork) returns(bool) {
        if (amIOnTheFork.forked() && targetFork.send(msg.value)) {
                        e(targetFork);
            return true;
        } else if (!amIOnTheFork.forked() && targetNoFork.send(msg.value)) {
                        e(targetNoFork);
            return true;
        }
        throw; // don't accept value transfer, otherwise it would be trapped.
    }

    // Reject value transfers.
    function() {
        throw;
    }
}
```

5.4.3 KittyCore

Contract declaration and the listing of the function *isValidMatingPair*. This function has the higher cyclomatic complexity. The contract name of the deployed contract corresponds with the name of the last contract declaration. The last contract inherits most of the contracts declared above. The function *isValidMatingPair* is the function which has the highest cyclomatic complexity among the contract. The complete source code is available on etherscan.io at the address:
0x06012c8cf97BEaD5deAe237070F9587f8E7A266d

```
pragma solidity ^0.4.11;

contract Ownable {...}
contract ERC721 {...}
contract GeneScienceInterface {...}
contract KittyAccessControl {...}
contract KittyBase is KittyAccessControl {...}
contract ERC721Metadata {...}
contract KittyOwnership is KittyBase, ERC721 {...}
contract KittyBreeding is KittyOwnership {
```

```
  ...
   function _isValidMatingPair(
        Kitty storage _matron,
        uint256 _matronId,
        Kitty storage _sire,
        uint256 _sireId
        )
        private
        view
        returns(bool)
        {
         // A Kitty can't breed with itself!
         if (_matronId == _sireId) {
               return false;
         }

         // Kitties can't breed with their parents.
         if (_matron.matronId == _sireId || _matron.sireId == _sireId) {
               return false;
         }
         if (_sire.matronId == _matronId || _sire.sireId == _matronId) {
               return false;
         }

         // We can short circuit the sibling check (below) if either cat is
         // gen zero (has a matron ID of zero).
         if (_sire.matronId == 0 || _matron.matronId == 0) {
               return true;
         }

         // Kitties can't breed with full or half siblings.
         if (_sire.matronId == _matron.matronId || _sire.matronId == _matron.sireId) {
               return false;
         }
         if (_sire.sireId == _matron.matronId || _sire.sireId == _matron.sireId) {
               return false;
         }


        return true;
  }
        ...
}

contract ClockAuctionBase {...}
contract Pausable is Ownable {...}
contract ClockAuction is Pausable, ClockAuctionBase {...}
contract SiringClockAuction is ClockAuction {...}
contract SaleClockAuction is ClockAuction  {...}
```

```
contract KittyAuction is KittyBreeding {...}
contract KittyMinting is KittyAuction {...}
contract KittyCore is KittyMinting {...}
```

5.5 Discussion

Results of this exploratory study provide us a very interesting overview of the world of smart contracts. This world can be described as very active in the usage of the blockchain, heterogeneous in the typologies and in the code features, and supported by an interactive and reactive development community.

We found that the smart contract developers' community follows constantly the evolution of the smart contract programming language, Solidity. This can find reasons in the necessity to develop, already from the start, efficient and secure smart contracts. In facts, the update of a smart contract for bug fixing or for adding new functionalities consists in the deployment of a new one Smart Contract in the blockchain and, in parallel, on the disposal of the old one. Creation of a Smart Contract leads to a cost in Ether that depends on the dimension of its bytecode (that is the results of the compiling).

Focusing on the purpose of smart contracts we found that developers have overtaken the concept of "parties' agreements" that characterizes the first idea of smart contract. In facts they created several typology of decentralized applications, ranging from games to utility tokens. We can also underline the importance of the reuse of the code. Thanks to the availability of thousands smart contracts source codes, developers can start from already implemented contracts to create new and more efficient applications, or updated and customized versions of old smart contracts. In addition, source codes are generally well commented, and this helps new developers to understand their contents. The "contract name" of a deployed smart contract could cause some misunderstanding. We discover that some specific names are very used despite in general are associated with very different source codes. So, the name is not representative of the contract.

Analyzing the interaction of deployed smart contracts with the blockchain, we discover that the number of transactions and the balance of the corresponding addresses follows power-law distributions. For what concerns the balance, we discover that the distribution of the wealth overtakes the Pareto law because the wealth is strongly centralized on very few contracts (about the 90% belongs to twenty out of over ten thousands deployed smart contracts). This is related to the variety of typology of smart contracts. In particular, the most of the wealth belongs to contracts of the *wallet* type, responsible of the management and protection of high amount of cryptocurrency.

The results of the analysis of the Source Codes give us a picture of a heterogeneous collection of scripts. This is characterized by code metrics that on average don't assume high values (for instance, the average number of line of code is about one hundred and eighty lines), but have high variances. Source codes present, on average, four contract declaration, revealing the use of the inheritance or the recalling of already deployed contracts. The cross correlation analysis shows us that the smart contract bytecode (that can be seen as the payload of the transaction with which the Smart Contract is deployed) is mildly correlated with the number of lines of code and with the number of declared functions. Anyhow, we found that the number of transactions is not correlated with the source code characteristics.

We analyzed in detail a subset of twenty smart contracts, namely those that have the highest number of transactions. We discover that they are mainly financial smart contracts (that implement a token compatible with the standard ERC20). We found also wallet, library, notary, and game Contracts. Most of the contract in this subset belongs to a project funded using an ICO, fact that emphasize the role of the development team and justify the use of ERC20 tokens.

For better describe the activity behind these twenty smart contracts we define five usage indicators. These characterize both the interaction with the blockchain in terms of number of transaction and number of transaction per day, and of the activity of the development community in terms of number of uploads, number of versions and number of reuse of code. We discovered that the game contract "KittyCore " has had, from the beginning, a high interaction with the blockchain. We also found that some contracts are still active in the blockchain after years, as is the case of the contracts called ReplaySafeSplit. Regarding the number of uploads and the number of "reuses of code", we discovered that eleven contracts have a development story behind it. In facts, these contracts are the results of the continuously improving and the related replacement of the old versions. In four cases we can report the release of a new version of the contract. We found that the source code of four contracts was reused to develop new smart contracts.

Finally, we analyzed the source code metrics of these twenty contracts. Findings show us that statistics of this subset of very used smart contracts differs from statistics computed belong the total set. In particular, these contracts are, on average, longer (about three hundred lines) and define five times more functions. In addition, we discover that these two metrics are strongly correlated with the sum of the McCabe cyclomatic complexity computed for the functions in the source codes. Finally, we computed the correlation coefficient between source code metrics and usage indicators. Results reveal that the number of transactions per day (that represents the frequency of usage of a smart contract) is moderately correlated with the number of lines of code, with the number of

declared functions, and with the cyclomatic complexity of the source code.

5.6 Conclusion

This chapter presented the setup, the analysis and the results of an exploratory study that take in exam Ethereum smart contracts and their source code. We aimed to discover characteristics of source codes and compare them with the use of the smart contract in the Ethereum Platform. We acquired a dataset of 10174 source codes, published by the 31st of December 2017, and we analyzed this dataset in order to provide an empirical description. We conduct an exploratory study that examined the dataset from several points of view.

In our opinion, this exploratory study offers food for thought that should be the starting point for further analysis. Future works should consider an higher number of smart contracts (taking into account the remaining of smart contracts without available source code) and further and specific code metrics (i.e to evaluate eventual code optimization in order to limit the Ethereum *gas* consumption or to measure the use of libraries and the interaction with already deployed contracts), other usage indicators (such us the internal transactions and the interaction between deployed contracts) and a widely analysis of correlation.

Chapter 6

Agile methods for blockchain applications

The application of blockchain technology certainly does not end with the ICOs. In this chapter some applications of the blockchain developed with Agile methodologies are reported.

6.1 A blockchain-based system for employment contracts

Temporary work contracts play a critical role in the current world economic and social context. Increasing international competition, slow economic growth and high unemployment rates have lead to the creation of greater job flexibility in many countries and institutions. The diffusion of non-standard contractual arrangements is also largely facilitated by technological innovations. In a global economic context, the competitiveness of companies is closely linked to the ability to adapt rapidly to new challenges and changes. Temporary employment could be an important and flexible business tool, in order to react to the market fluctuations, affected by economic policies and some seasonal conditions. As a result, according to International Labour Organization [1] atypical contractual arrangements are a feature of the contemporary world of work.

However non-standard work contracts are often characterized by a lack of workers' guarantees, insecurity, low wages, limited growth prospects, lack of vocational training and less access to social security systems. Moreover young people, regardless of the level of education they have and the skills they possess, are engaged in non-standard jobs more frequently than other groups of the

[1] http://www.ilo.org/global/about-the-ilo/newsroom/news/WCMS_534122

population.

In addition to this labor flexibility is also often associated with the absence of a development model combining the social quality and sustainability of new forms of work to economic growth and enabling transition from one job to another and more generally, a full acquisition of human rights. It is therefore necessary to build social protection instruments that do not only protect workers from the risk, but also from the charm of flexibility.

We want to show how blockchain technology can be used to address the fundamental problems that occur around temporary employment, in order to protect employees and to prevent that the competition being distorted in the benefit of those companies that wish grow on the backs of exploited illegal workers. In particular, we answer if the blockchain technology can simplify the management of temporary employment relationship.

In fact both businesses and employees need the recognition of the value of their work, and, in this complex scenario of non-standard work contracts, the use of blockchain technology of may be an excellent solution in order to ensure reliability, transparency and security. Blockchain technology indeed is based on a decentralized technical database to efficiently manage transactions. It stores these transactions in a Peer-to-Peer network. Blockchain technology is also a public registry: transactions consist of encrypted data that are verified and approved by the nodes participating in the network, and, subsequently, added in a block and recorded in the blockchain. The blockchain is shared between all nodes of the network. The same information are present on all nodes and therefore becomes unmodifiable unless through an operation that requires the approval of the majority of the nodes in the network. In any case, it will not change the history of these same information. Therefore this technology introduces a new level of transparency and efficiency, by allowing the network to manage the transactions and creating confident transactions in an untrustworthy environment. Focusing on the system implementation, we want to discover if smart contracts can implement a temporary employment relationship model.

Blockchain technology allows to quickly register work contracts with full protection of the rights both of the worker and of the employer, in compliance with the legislation. Smart contracts are immutably saved on the blockchain and can be observed and checked for compliance at any time by the competent authorities.

6.1.1 Background

Blockchain technology can be used in all contexts where a decentralized system is necessary in order to ensure the involvement of many actors in the same network and guarantee a full transparency and reliability between people who

do not know each other. Therefore blockchain technology is not only useful for creating digital currencies or new financial technologies, but can be applied for a wide variety of applications, such as protection systems of digital identity, provenance of documents, organizational data management, digital and physical assets. An important research is that carried out by [34]. They, designing an interdisciplinary approach, analyze legal aspects and consequences of the use of blockchain for job organizations that want to challenge the law and the labor market. [66] instead examines the use of smart contracts, combined with intelligent multi-agent systems and Internet-of-Things devices, in order to deliver self-aware contracts with a high degree of automation for peer-to-peer collaborations. They apply a smart contract, mapped onto an automated protocol, for initiating and terminating a rental contract.

An innovative regulation of labour hours and the associated payments, is proposed by Chronobank Team[2]. They apply recent advancements in blockchain and cryptographic technologies in order to develop a non-volatile and inflationary resistant digital asset. The transfer system considers the average hourly rate of human labour as the most fundamental unit of economic value.

An interesting research about is that of Wang et al. [103]. In their work the technology blockchain is used to certify the human resource documentations and to connect the information with the documentation. In this case the decentralized mechanism provides a low cost and high efficiency of data transfer and gets a high-performance work system in the human resource management of enterprises. In addition, [68] shows the potential of blockchain technology in governmental tasks such as secure document handling and digital ID management and provides proof of its compliance for authenticating of persistent documents.

The Blockchain Research Lab [3] developed a prototype to manage a smart contract between agency, manufacturer and worker. They focus on the case of the necessity to create a valid contract between the agency and the worker considering that the agency needs a leasing permission and the manufacturer needs enough funds to pay the agency. The system, through a smart contract, checks the coexistence of all these requirements.

The blockchain technology is also a promising technology for the implementation of several typology of decentralized systems. In particular, in the field of a public and collaborative smart city system, blockchain represents a smart solution. In [46] thanks to the use of smart contracts and an Agile Involvement of Citizens, is proposed a shared and public database of eviormental signals.

[2]https://chronobank.io/files/whitepaper.pdf

[3]https://blog.hacking.law/blockchain-technology-and-the-future-of-temporary-employment-51ff7062b6

6.1.2 The Decentralized Employment System

In this section we describe the technical aspect of the proposed Decentralized Employment System (D-ES).

The system is a solution that makes transparent and traceable any employment relationship, established for the temporary work. The system simplifies the recruitment procedure, and it is a useful tool to prevent the black labour.

In order to model the system, we first identify the actors which will be considered in the development. The model considers four typology of actors. Two typology of actors represent human users. The first is the Employer, who creates the work activity and announces the availability of vacant posts. The second is the Worker, who applies for a temporary job.

The other two typologies of actor are components of the system. Because the technical difficulty of putting directly the hands in the blockchain, the DES will provide a simplified user interface, in the form of web platform. The Platform is the actor that makes users (Employers and Workers) able to create new works, to apply for them, and to access further information about the employment relationship. And finally, the last actor is the blockchain. The system will be based on the blockchain technology because its capability to provide trust and security, and for the possibility of developing decentralized application we will exploit to implement the D-ES. In the system, the blockchain has the double role: the role of ledger, public and unchangeable, and the role of control system which safeguards workers and prevents scams. In fact, thanks the availability of a decentralized virtual machine, the blockchain is not only a database but also a computing resource.

6.1.3 The D-ES state system

We model an employment relationship as a state system in which states describes the current phase of the relationship.

Taking in account the need of a legal recognition and authorization, we suppose actors of the system able to create legal employment relationships. For instance, the Employer should have the legal rights to hire people. At the same time, a Worker should be legally recognized to be able for a specific job. We represent this point defining a background system representing a generic *Central Authority*. The Central Authority, in order to supervise the employment relationships, could take advantage of the transparency property of the blockchain. In the section 6.1.6 provide further details of this aspect.

In order to describe the model, we first specify states, the events that change the state, and the role of the actors. Fig.6.1 shows the D-ES state diagram. The initial state, S0, is an idle state. The D-ES is ready for the creation of a new

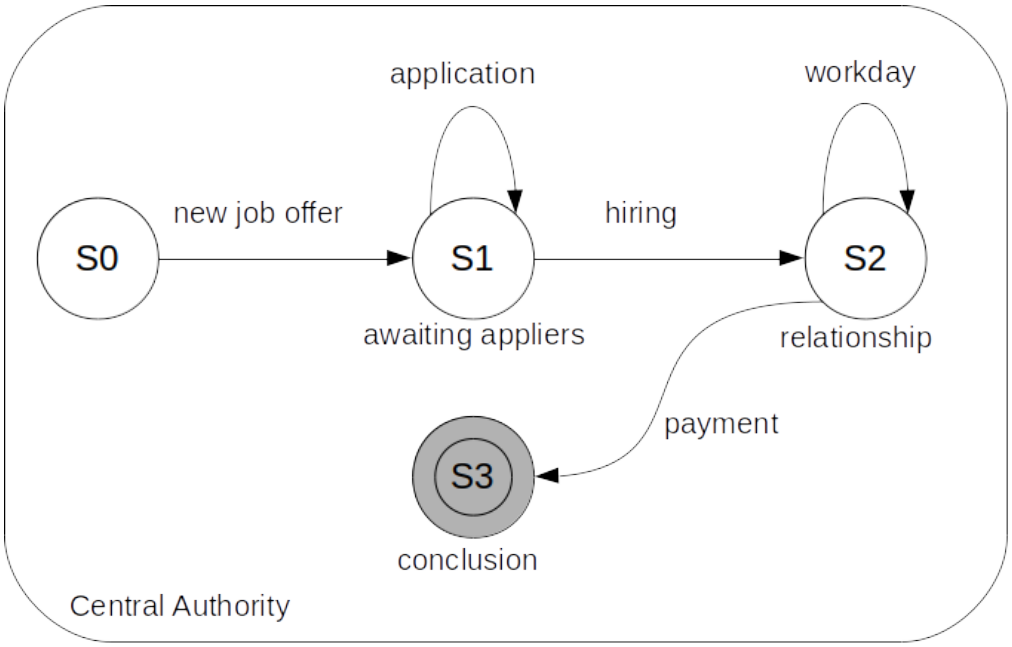


Figure 6.1: The state diagram describing the employment phases controlled by the D-ES.

employment relationship.

The first event is called "*new job offer*" and it consists in the creation of a new job offer. It happens when a Employer completes the procedure to setting up a job offer, aided by the platform. The Employer uses the interface provided by the platform in order to specify all the properties of the new open position: the number of working hours, the time wage, the job title, etc. In addition, the Employer deposits the amount of digital asset in order to cover the value of the wage. When all is done, the platform creates and sends to the Blokchain some ready-made message. Those messages include all the information required to create the set of smart contracts with which the D-ES will control the employment relationship. Technical detail of smart contracts will be discussed in the following.

The "*new job offer*" event changes the state from the initial state to the "*awaiting appliers*" state. In this state, the D-ES is configured to accept the application of new workers. Now, workers can apply for the open position. In this state, the platform shows the job offer to the workers. An internal event "*application*" describes when a worker applies for the job offer. The worker has to send a message to the blokchcain, or precisely, to a smart contract. That smart contract is charged to acquire the application request, and to compute

and return to the Worker, an applicant's identification code.

When an applicant Worker meets the Employer, they can give rise the "*hiring*" event. The Employer now being in posses of the applicant identifier. He sends a message to the blockchain in order to start the employment relationship.

Now the system is arrived to the "*relationship*" state. In this state, the Worker can check, at any time, its working situation and verify the number of the matured working hours. An internal event "*workday*" describes the maturation of a daily number of working hours, and occurs when the Employer sends a message to the *blockchain* in order to certify that the worker has completed a work day.

Automatically, when the stipulated working hours are over, the contract declares the end of the relationship and occurs the "*payment*" event. During this event, the system moves the wage deposited by the Employer in the account of the Worker. This event moves from the "*relationship*" to the "*conclusion*" state.

6.1.4 Implementation of the decentralized system

The blockchain has an active role and it is a real actor of the system. Summarizing, the blockchain will identify the Employers, identify the Workers, record every employment relationship, control and compute the evolution of the employment relationship, and, finally, compute and transfer the wage from Employers to Workers.

All these actions will be done by means the execution of a decentralized computer programs called smart contracts. The D-ES works through a decentralized ecosystem of three typologies of smart contract. They are named: "sc_deposit", "sc_application", "sc_relationship". In order to automate the creation of new employment relationship, the three typologies of smart contract will be recorded ready-made in the Platform system.

According with the system state described before, the Platform customizes the three smart contracts with the job description data provided by the Employer. At the event "new job offer", the three smart contracts are created, configured and written inside the blockchain. Each one of the three smart contracts knows the address of the others two.

The contract type "sc_deposit" implements a token deposit. Token and coin deposits are a popular application in the Ethereum system. In our system, the sc_deposit includes a *payment* function programmed to transfer the deposited wage to a specified Worker's address. This function can be called only by the sc_relationship contract which address is recorded in the sc_deposit memory.

Each contract type "sc_application" provides the *application* function. This function is programmed to be called by Workers' addresses and returns a cryptographic identification code, valid only for the employment relationship that it

represents. In addition, that smart contract records and provides information about the job offer.

Finally, a contract type "sc_relationship" provides the *hiring* function. This function is programmed to create an employment relationship. It receives messages from the Employer address, containing the blockchain address of the Worker and his identification code as produced by the associated "sc_application". The contract checks the validity of the identification code. In addition, the contract stores the current number of working hours matured by Workers. This contract is also able to ask "sc_application" for the agreed number of working hours. A second function *workday* updates the number of work hours when receives the appropriate message from the Employer. This function automatically computes the end of the employment relationship. In that case, in order to call the *payment* function, it sends a message to the "sc_deposit", specifying the address of the Worker.

6.1.5 The Platform

We anticipate the need for a Platform (that can be seen as an user interface), responsive and easy to use. This Platform provides forms and instructions that makes simple the creation of the blokchain system and the control the employment relationship. The Platform creates the three smart contract. It simplifies the creation of a new job offer and provide a friendly interface for the applicant workers. Furthermore it provides the visualization of the state of the employment relationship.

6.1.6 Discussion

This section proposes a blockchain based system able to simplify the management, control and supervise the temporary employment relationship. The blockchain resulted perfectly adapted to simplify and automate the temporary employment system. Our system provides high-performance in managing temporary contracts by addressing some of the key aspects that make them unsuitable for being applied to the the present working context. It aims to protect the rights of workers and enterprises at the same time, but also to ensure full control of the competent authorities during the verification of the necessary requirements for signing of the contract (the conditions related to employer and worker must be fulfilled at the same time) and during the verification of proper contract's performance. In fact, the competent authorities are not always able to detect illegal actions in terms of taxation and protection of workers in real time, and generally do not have the capacity to carry out constant and complete monitoring. The benefits of applying our system can thus be summarized as

follows. In the temporary work contracts based on blockchain and smart contract, employee and business data and agreements between them are analyzed automatically in order to facilitate the processing of contracts, make this procedure fully automated, increase the accuracy and processing speeds and allow full compliance with existing contractual law. It can cancel the time to verify the contract correctness by the competent authorities: if a contract was concluded, the requirements of the employer and the employee were corrected. In addition, the competent authority can carry out constant checks in real time by simply accessing data recorded on blockchain. Contracts may also be dispatched automatically to the competent authorities. The immutability of the data saved on blockchain makes the payments that are consistent with what is stated in the contract terms. Contractual terms must match payment execution and must be based on hours worked.

6.2 Smart Contracts as Blockchain-oriented Microservices

The paradigms of microservices and of smart contracts share many similarities. Microservices are small applications developed as set of autonomous services, decomposing a monolithic architecture in independently deployed isolated services with a clear and well defined purpose [96, 95]. They naturally implement a modular structure and changes to a part of the application only requires changes and redeployment of a limited number of services.

Smart Contracts as well represent and well defined, usually isolated, programs, typically implementing simple and autonomous tasks with a well defined purpose, which can be considered as services provided by the Contract. Blockchain features naturally implement a modular structure where changes to a small part of the aplication only involves changes and redeployment of a limited number of smart contracts.

More recent approaches to microservices require or suggest the execution of the services on the edge, and in the IoT framework services can be provided by any processing unit. Smart Contracts run on a blockchain and interaction occurs into the blockchain by mean of transactions, implemented by all blockchain-oriented systems[78]. Decentralization is common to both paradigms and in the case of smart contracts it is implemented by the blockchain nodes.

In particular, smart contracts are computer programs which are executed inside blockchains of second generation. This represents the main innovation brought by the Ethereum platform [17]. Several are the case of studies concerning the application of smart contracts in scenario different from the financial

applications, in which are exploited the property of decentralization of data and computation, and the intrinsic data availability typical of blockchain based architecture [46, 53, 75, 54, 76]. We have to take into account that in *public* blockchains the validation of blocks occurs in a competitive manner, so that every operation (i.e. smart contract code deployment, data insertion or request, information exchange and so on) involves a non negligible cost. Furthermore smart contracts deployed on public blockchain display criticalities that must be taken into account when providing services [32, 104, 16, 30].

To implement a microservice architecture, we propose the use of a private Ethereum blockchain. This avoids operative costs that characterize public blockchains and ensures a better control of privacy. In blockchain systems, each message is recorded and time-marked. By using a private blockchain it is possible to obtain a unique timing for all (micro-) services provided by the system, solving the possible problem of different timestamps associated with a public, competitive validating blockchain where each node can validate the transactions at different times.

Other private blockchains are based on the Bitcoin system. Hyperledger [4] is an umbrella of open source project hosted by The Linux Foundation, it is composed by a various of enterprise-ready solutions such as Fabric, Sawtooth and Iroha. One of the feature of Hyperledger is the blockchain permissioned. All nodes within the network are authenticated and authorized to partecipate reducing security risks. Only involved parties can see information about the related transactions that are not visible on the whole network such as in public blockchain like Bitcoin or Ethereum.

Generalizing the concept of blockchain we can take in consideration all the DLTs (Decentralized Ledger Technology). For instance, we can take in exam the solution of *IOTA* to the problem of transaction costs. In this architecture, blocks are reduced to only one and free transaction, that when is created must validate two previous transactions, in order to create a structure called *The Tangle*[5]. In the microservices scenario, the Tangle should be used to send costless service requests. But this DLT focuses only on the data management and transparency, and does not provide a decentralized platform to run smart contracts we want describe as services.

In the following, we describe a blockchain-oriented microservices architecture based on smart contracts, for the case study of an e-commerce application.

[4] https://www.hyperledger.org/
[5] https://iota.org

6.2.1 Model

We present a dummy but paradigmatic example of an e-commerce application [6] whose microservice architecture is reshaped in services provided by smart contracts running onto a private blockchain. Fig. 6.2 shows the conceptual scheme of the e-commerce system. The architecture of the system is composed of two layers.

The first layer is the interface between applications and the blockchain. It provides the ABI (The Ethereum Application Binary Interface) for the development of software applications, and the user interface in which ABI are embedded. In particular, a software application can use smart contract's ABI to compose and make requests of services. The contract's ABI allows to specify which function in the contract to invoke, as well as guarantees that the contract will return data in the expected format.

The second layer is composed of the set of smart contracts running in the blockchain. In our model, each microservice is implemented by means of an atomic smart contract. Communication between layers takes place with remote procedure calls (RPC), through the Web3 Ethereum library[7] that allows to write javascript programs able to execute blockchain transactions, contract calls and so on.

Specificity of blockchain can be used to easily implement services. For instance, each user is uniquely identified by an Ethereum address. The Account service records and manages this information. In addition, depending on the client profile, the system enables different functionalities or services. Once registered, data are stored permanently within the blockchain and all nodes have a copy of the blockchain.

The user registration and the login are conceived as different microservices autonomously managed by a dedicated smart contract. Inventory service records product information inside the blockchain and returns these information to the storefront web page, or to the application. Some of its functionality are available only for enabled users. Likewise it can be said for the Shipping Service.

6.3 CitySense: blockchain-oriented Smart Cities

A Smart City is an urban system where some requirements exist simultaneously: a functional management model of traffic and public transport, a context in which citizens can work remotely, without moving, in a lot of activities, the

[6] http://microservices.io/patterns/microservices.html
[7] https://web3js.readthedocs.io/en/1.0/getting-started.html

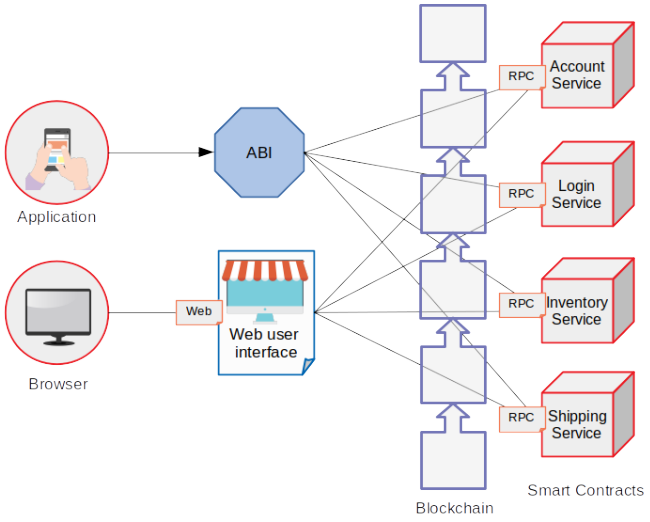


Figure 6.2: E-commerce system as a composition of smart contract-based microservices

opportunity to live in safety thanks to the use of innovative solutions of public surveillance, assistance and the application of appropriate technology for energy saving and to reduce the environmental impact. Linked to these technical aspects we want to identify some political and human issues which are equally important. In fact, a Smart City is able to create new ways of participation in public life and calls on people to become active citizens in order to enhance a continuous and two-way relationship with local government [106].

In this complex scenario the implementation of an urban IoT may be an excellent solution for real optimization and management of public services. Several kinds of data, collected by sensors in a IoT system [47], can be used to increase the openness of public government and political choices, to improve the people's awareness about well-being of the city and to encourage the involvement of citizens in the drive for sustainable development. In order to achieve these goals, we have chosen blockchain technology as public available shared database and then as instrument of reliability, transparency and security. Moreover, we are going to apply SCRUM methodology to build this system blockchain based and we are going to involve citizens in the software development. In this way, we can make a real open source community.

6.3.1 Background

Smart Cities projects are focused on viability, digital services, payments and environment, but it should be improved in the following key areas: provide better ground coverage of the services, as every item can be both client and server, and distribute the processing tasks (for power, money, time, etc.) among a big number of calculators capable of simple operations. Modern smartphones are up, in performance, to some of the mainframe servers deployed 20 years ago in the ISP industry [71]. Tailored software and simple computation tasks can be written to be executed by such small devices with good performances. Across Europe, several cities have been engaged in environment safeguard plans, starting from the 2008 SETIS Plan to reduce CO2 emission that widespread over 12 countries and 200 mid-sized towns, up to single town's projects to reduce power consumption and invest on new energy markets while the industrial ones that relies on technologies and sources from the last century are slowly fading away. Emphasizing on the social approach of a smart city plan would be the key for raising the interest of the citizens who will became the main actors in this project, for contributing to the mass effect of self knowledge of the environment they live in and for keeping it as wealthy as possible. This social approach can be interpreted correctly by the blockchain technology. In fact, after its introduction, the blockchain technology has interested more and more scientists who study potentialities and applications in a multidisciplinary world of topics. For instance, researchers studied aspects related to the network [76], economy [90], and software engineering [78]. We reported the most recent publications, which show the state of art in the field of blockchain and in smart contracts applications to smart cities and Internet of Things. The feasibility of a blockchain-oriented smart city is an in-progress study. Looking in the web, is easy to find related initiatives, like that of the Dubai government which is programming to create a blockchain based smart city [8], and debates about potentiality of this application [9]. Sharma et al. [88] studied an application of blockchain technology to build a vehicle network which takes in account several problematics that are the mobility of nodes (which represent vehicles), the confidentiality and security of data. Security aspects are also discussed by Biswas et al. in their work [10]. We take this work into particular consideration since it work discusses the need of a specific security framework.
Such framework is composed of four layers: the physical layer (which includes sensors), the communication layer (in which is considered the blockchain), the database layer and the interface layer (that considers all applications). Considering the IoT a key element for the smart cities development, we briefly discuss IoT

[8] http://www.coindesk.com/dubais-museum-future-sees-blockchain-smart-cities/
[9] https://dcebrief.com/blockchain-powers-new-smart-city-initiative/

related applications of the blockchain technology. It is an enabling technology for empowering the potentiality of the IoT. The work of Quaddah et al.[72] provides a well-defined framework named FairAccess to enable the communication between nodes by means of some blockchain based mechanisms (i.e smart contracts and transactions). Thanks to the blockchain, this framework provides a stronger and transparent access control tool. E-business aspects of IoT technology are discussed by Zhang [107]. He studied a blockchain application which implements a seller-buyer model describing business operations between two or more devices. Christidis and Devetsikiotis [21] provide a discussion based on the literature, proving that smart contracts and blockchain applied to the IoT can be pretty powerful. They also provide an interesting section about the blockchain taxonomy. IoT (RFID based) and blockchain can also enable products traceability in a supply chain, as discussed by Tian [98], and can also enable the control of remote robots, as discussed by Ferrer [36]. Recent studies [51] have shown that a smart city needs to have a smart local council and a smart methodology in order to develop an efficient software. A good way to achieve this goal is the use of SCRUM process with some changes with respect to the original approach.

6.3.2 The system

Networks of Things acquire and share information on what is happening of interest to the citizens and can be used within the city to build a smart system of services.

The Smart Object network will be able to receive and store information from the surrounding environment through the use of sensors, in order to measure the values of the phenomena of interest, and through the use of actuators that act on the context. Our idea is to develop an IoT application, based on blockchain, to investigate some aspects of urban setting, to acquire relevant data and to process the information received.

We use SCRUM methodology in order to implement this innovative blockchain-based system. We chose to apply a SCRUM process because of its capabilities of being flexible, adaptive and iterative. These aspects are perfectly in line with the intrinsic properties of blockchain, like decentralization and reliability, and especially like the capability of managing data, transactions and all changes in a much faster and more efficient way. Therefore, we believe that Scrum may constitute a suitable methodology in order to increase the communication between users who send useful data recorded on the blockchain, and in order to exploit the continuous evolution of software and the collective accountability.

According to SCRUM methodology, in order to verify the correctness of system developed, the client's role is extremely important in several aspects at

all the stages of the process. The contribution of customers impacts directly on product quality. For this reason, the clients must be involved in any decision adopted and shall collaborate closely with the development team. In our work, we consider as clients all citizens who wish to contribute to well-being of the city through the transmission of data from mobile devices by means a specific application. The same application will allow the transfer of information that comes from sensors. It will also be able to provide other functionalities, like sending messages, providing reports and user feedbacks. The data can be recorded on blockchain by means of a connection to the peer-to-peer network. The Product Owner will be responsible for analyzing information provided by users through the mobile applications, for interpreting consumer expectations and requirements, for filtering communications transmitted, for identifying priorities and for distributing tasks within the development team. The local government will be involved by the Product Owner in the analysis of information communicated by users in order to provide feedback to the developers and take decisions. The city council then, through its decisions, shall participate in all phases of work and for each sprint from preparation of the public tender to monitoring of activities performed. This method favours effective application of an entirely cooperative approach.

6.3.3 CitySense

In order to conduct our study, we have identified some measurable phenomena and their characterizing quantities (noise, concentration of hydrogen, methane, carbon monoxide and microparticulates in the air, temperature, humidity and light). These quantities are related to environmental pollution issues in which the public opinion is most involved because they have a direct impact on people's lives.

The increasing diffusion of small programmable embedded devices that are easily connected to networks via wireless technologies had increased the number of actors eligible to participate to IoT networks that can share information. From this perspective, peculiar to the IoT world, the creation of an ecosystem of small interconnected "Smart Objects", that can be considered as an extension to the human body (nowadays we think about smartphone, smartwatches, wristbands and portable gadgets) would bring to life a complex and intelligent network of distributed systems that can interfere, both in a good and in a bad way, with the surrounding environment.

If we apply this schema to a geographic area, such as a town or a county, we can imagine a full coverage of the ground made by portable devices that collect data and send them back to a central collection endpoint.

The data collection mechanism works by combining the measurements acquired by the mobile devices with the validation process made with a specific mining software that process the data.

Mobile phones are the best devices to be used for this task as they provide a quite good computational capability, moderate battery life, wireless connectivity to other near devices, internet access and they are easily programmable to run generic purpose software on top. To connect to hardware-level machine, like electronic devices and sensors, a HW/SW connection layer should be deployed. As Today, a lot of such programmable tiny operational boards equipped with AVR processing units could be used for this task. Miniaturized boards with small footprint up to 30mmx20mm [1S] can be connected to small power sensors such as DTH-22 for temperature and humidity, TEMT6000 for light, GP2Y1010AU0F for PM8 detection, due to their low current requirements and MUXable digital I/Os, as the number of analogic inputs is low on this miniaturized boards. Bigger sensors, such as C0, Hydrogen, Methane, require a huge amount (1 Amp x 5/9 Volts) of current in comparison to the previous digital ones, as they need to heat a filament (just like happened in vacuum tubes) to be able to operate.

This current consumption has an impact also on portability of the complete kit, as the power requirements would be too much for the desired target (people on the move). On the other hand, employing small sensors like the one mentioned above, in conjunction with a BLE (Low Power Bluetooth 4.0 connectivity) can open up a wide variety of solutions, like the possibility to attach such device to a small solar panel, the size of a pocket mirror, or to be powered from a dynamo connected to a bicycle wheel. Also, the number of sensors deployed could be larger as the software elaborates the results in a more sophisticated way. Computational power on this task in not a real big issue as digital multiplexing on data is quite easy for this scenario. All the sensor package can be powered via a very common and cheap 3.7V LiPo battery, and even be powered from the USB OTG phone (if capable), that can boost up to 5V to 450 mA.

6.3.4 The blockchain solution

The blockchain is an enabling technology which allows to obtain the satisfaction of security and availability of data, and provides the computational power that makes it able to control the communication between nodes.

In order to develop the CitySense system, we use the Ethereum platform to record measurements arriving from the IoT network of sensors. In our system, sensors are IoT devices, programmed to be connected with the blockchain and able to send messages. As in [10], CitySense is structured by layers as is shown in Fig. 6.3.

Layers of the system will be implemented in parallel. Each layer has specific

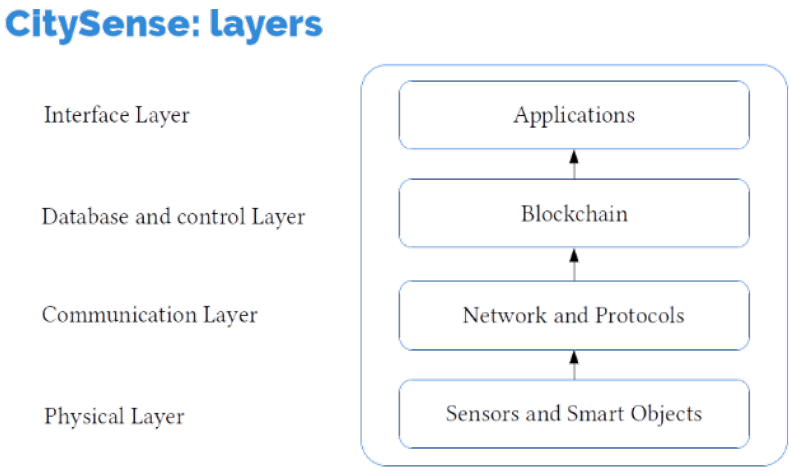


Figure 6.3: Layers of structure

and distinctive requirements and will be implemented in an iterative process, with the advantages of the SCRUM methodology. The set of sensors composes the physical layer. In our system, sensors are carried around the city by their owners. For this reason, each measurement must be associated with the geographical position, in addition to the typology of measurement and to the timestamp. Considering the network layer, in our system sensors are programmed to send measurements to the blockchain through the peer-to-peer network by means of a light version of existent clients such as geth. Actually, sensors can be connected to a mobile device (for instance a smartphone) to run, or can be autonomous systems able to connect to the peer-to-peer network. Different communication mechanisms, such as WiFi, 3G or Bluetooth, can be used. Specific communication protocols should be defined considering the cost of a transaction in the Ethereum system (which is proportional with the payload size of the message), and designed according to a convenient data format. In CitySense the blockchain consists in the database layer in which measurements are organized and stored. In our system we use contracts to control and save data. A first contract is designed to be the receiver of messages coming from sensors. We call it acquisition and sorting contract (ASC). Depending on the geographical position, the ASC sends the measurement to one of the set of specialized contracts that we call geographic contracts (GC). Inside each GC only measurements coming from a specific geographic area are stored. With specific implemented functions, GCs can be queried to provide measurements organized in several ways (for instance measurement can be organized by their typology, by timestamp, by value).

The variety of applications enabled by the CitySense system composes the application layer. Thanks to the collaborative nature of our system, the same people who participate to the creation of the CitySense are also able to take advantage of the information that is made available. For instance, a CitySense mobile application could be useful to people who want to know the environment quality of a specific area in the city. Furthermore, web applications could be used by a public administration to check if and where the city has pollution problems.

6.3.5 Discussion

In our vision, blockchain represents the disruptive technology which will drive the development of future smart-cities related applications. Key features, such as to be a shared, transparent, distributed, secure, available and smart technology, make the blockchain an opportunity to improve potentialities of IoT and smart-cities development. As it is known, blockchain data are publicly available. In a smart-city, transparency makes citizen aware and able to know the contribution of each of them and how public governments use data. In case of need (i.s. sensitive or personal data) it is always possible to encrypt data before they are stored inside the blockchain.

In a blockchain based service, nodes participate to the objective in a democratic way, under the constraint of the consensus rule. Therefore, blockchain enables trust-free transactions without the need of a central control authority. IoT devices participate in the peer-to-peer network sending messages to it, as nodes of the system and under the blockchain rules. In a blockchain system, smart contracts, computer programs located and working in the blockchain, can automatically acquire data from IoT devices and produce computed outputs. Because of the distributed nature, smart contracts cannot be modified or interrupted. For this reason, their usage could improve the reliability of a smart-city system.

IoT, SCRUM methodology and blockchain form the basis of digital transformation. A trust distributed technology ensure privacy, scalability, transparency and reliability. In a Smart City the number of linked objects is set to increase and will produce higher and higher operating costs. The blockchain will be a key element for the cost-cutting in the tracking and coordination of physical devices and will solve failure problems of traditional networks. Currently all IoT systems depend on client and server communication protocols, such as SSL and TLS and on cryptographic mechanisms such as the Public Key system, used in order to make the communication system verifiable and to authenticate the nodes of the network. Such architecture will soon have to face several problems (delay transmission for example) caused by data traffic congestion. Therefore, the

decentralization, which is an intrinsic property of blockchain, could be the right solution in order to increase the network efficiency and reduce management costs. CitySense will allow a direct communication between smart devices and will verify the transactions without a central server.

Scrum methodology will be also used in order to ensure software quality, reduce the time-to-market, enhance the support of the citizens and create an infrastructure that allows the transfer of data in real time using a sensor networks with a low energy consumption [24]. The support of the citizens is essential to make this project succeed, but at the same time implies more complexity. For this reason we estimated a duration of 30 days for every sprint: this choice improves the collaboration of stakeholders. We planned also unitary tests for each iteration [25].

Chapter 7

Conclusions

The three main objectives of the work described in this thesis were:

- to understand the main characteristics of ICOs and evaluate the lean startup approach as a methodology for the implementation of an ICO;
- to investigate software engineering activities related to ICOs, recognize the ICOs developed using Agile methods;
- to analyze and characterize the smart contracts deployed on the Ethereum and to provide empirical results about smart contracts features, their interaction with the blockchain, the role of the development community, and the source code characteristics.

In our work we analyzed the new and complex phenomenon of ICOs, an alternative means of financing startups based on the concept of token and on a decentralized blockchain approach. Startups based on a ICO are playing a fundamental role in creating the market of blockchain applications. ICOs provide a pre-sale of tokens what will be used to pay for a service to be launched on the market, or even the launch of a new cryptocurrency. In most cases, the same investors become consumers or users of the same service. All of this allows investors to buy crypto tokens at a discounted price, even if in reality their value will be dictated by the mechanism of supply and demand only after being placed on the market. An ICO can be a valuable tool for those teams that want to quickly obtain financing, but it also has several limitations, due essentially to the immaturity of the technological system and to the risk of financial speculation.

In chapter 3, we analyzed the ICO phenomenon starting from the available data provided by ICO datasets, performing various statistical computations to understand what affects the ICO success. Then, we tried to understand if the Lean startup approach can be useful to solve some of ICO issues. The tokenization nature of an ICO proposal needs a form of sustainable and regulated token

sale event, that can be built on an MVP. The concepts of *pivot* and *validated learning* can be very useful, but also the investors' goals must be taken into account. They can be directed exclusively to immediate gain and not to company growth, strategic planning or operational work. A Lean startup methodology could be useful in order to respond to a tokenization that gives rise to new business models and new products or services that must effectively address customer needs. Many iterations and the direct involvement of all the stakeholders can further improve and help to market the original idea.

In chapter 4 we investigated all engineering activities related to ICOs, from the planning phase to the testing phase. We analyzed the whole set of ICOs registered in ICObench until the February 20, 2018 in order to discover the team composition, the communication channels with investors distributed around the world and the financial aspects. We therefore studied the use of Agile practices in ICOs as a method to cope with change. We have selected and analyzed in detail a subset of ICOs specifically developed with Agile methodologies relatively to their roadmap, project development, and source code quality. Overall, about the 5% of the examined ICOs apply Agile practices. In addition we conducted an analysis of smart contacts of Agile ICOs in terms of code metrics, language versions and use of test tools. We discovered that the Agile methodologies are suited to develop ICOs because these are highly innovative projects, whose requirements are not completely understood or tend to change. The Agile approach is iterative and incremental with short iterations and is suited to deliver quickly and to deliver often. This property is very useful in the context of ICOs. On the other hand, the necessity to continually iterate on the product, typical of Agile methodologies, can create issues for ICOs. The immutability of the blockchain must be taken into consideration. In fact, smart contracts can not be updated once they have been loaded. To face this difficult the Test Driven Development is very useful. Also the practice of Collective code ownership is guaranteed in ICOs by the transparency of smart contracts in the blockchain. Another practice of Agile development that are applied in ICOs is the use of Coding Standards. The too detailed roadmaps are instead typical of the plan-driven methodologies. Finally we can say that the smart contracts of Agile ICOs have good metrics of the software because their source codes are very short and simple.

The chapter 5 presents the setup, the analysis and the results of an exploratory study that take in exam Ethereum smart contracts and their source code. We acquired a dataset of 10174 source codes, published by the 31 December 2017, and we analyzed this dataset in order to provide an empirical description. We conduct an exploratory study that examined the dataset from several points of view. In our opinion, this exploratory study offers food for thought that should be the starting point for further analysis. Future work should consider a higher number of smart contracts (taking into account the

remaining of smart contracts without available source code) and further and specific code metrics (i.e to evaluate eventual code optimization in order to limit the Ethereum *gas* consumption or to measure the use of libraries and the interaction with already deployed contracts), other usage indicators (such us the internal transactions and the interaction between deployed contracts) and a widely analysis of correlation.

List of Publications Related to the Thesis

- IBBA, S., et al. CitySense: blockchain-oriented smart cities. In: Proceedings of the XP2017 Scientific Workshops. ACM, 2017. p. 12.
- Mannaro, K., Baralla, G., Pinna, A., & IBBA, S. (2018). A Blockchain Approach Applied to a Teledermatology Platform in the Sardinian Region (Italy). Information, 9(2), 44.
- Pinna, A., & IBBA, S. (2018). A blockchain-based Decentralized System for proper handling of temporary Employment contracts. Intelligent Computing - Proceedings of the 2018 Computing Conference - Advances in Intelligent Systems and Computing - Springer - **Winner of the Best Student Paper Award**
- Tonelli, R., Pinna, A., Baralla, G., & IBBA, S. Ethereum Smart Contracts as Blockchain-oriented Microservices. In: Proceedings of the XP2018 Scientific Workshops. ACM, 2018
- IBBA, S., Pinna, A., Baralla, G., & Marchesi, M. (2018, May). ICOs Overview: Should Investors Choose an ICO Developed with the Lean Startup Methodology?. In International Conference on Agile Software Development (pp. 293-308). Springer, Cham.
- IBBA, S., Pinna, A., Lunesu, M.I., Marchesi, M., Tonelli, R. (2018). Initial Coin Offerings and Agile Practices. Future Internet - IN PRESS
- Pinna, A., IBBA, S., Baralla, G., Tonelli, R. Marchesi, M. A Massive Analysis of Ethereum Smart Contracts. Exploratory study and code metrics. Information and software technology - UNDER REVIEW

List of all Publications

- IBBA, S., Orrù, M., Pani, F. E., & Porru, S. (2015, November). Hashtag of Instagram: From Folksonomy to Complex Network. In KEOD (pp. 279-284).

- Pani, F. E., Porru, S., & IBBA, S. (2015, July). A Model for Digital Content Management. In Proceedings of 4th International Conference on Data Management Technologies and Applications (pp. 240-247). SCITEPRESS-Science and Technology Publications, Lda.

- Eros Pani, F., Porru, S., Orrù, M., & IBBA, S. (2015, November). A Complex Network Approach for Museum Services. In Proceedings of the International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (pp. 216-221). SCITEPRESS-Science and Technology Publications, Lda.

- IBBA, S., & Pani, F. E. (2016). Digital libraries: the challenge of integrating Instagram with a taxonomy for content management. Future Internet, 8(2), 16.

- Stocchi, M., Lunesu, I., IBBA, S., Baralla, G., & Marchesi, M. (2016). The Future of Bitcoin: a Synchrosqueezing Wavelet Transform to Predict Search Engine Query Trends. In KDWeb.

- Stocchi, M., Lunesu, I., IBBA, S., Baralla, G., & Marchesi, M. (2016). A Synchrosqueezed Wavelet Transform Assisted Machine Learning Framework for Time Series Forecasting. In KDWeb.

- IBBA, S., et al. CitySense: blockchain-oriented smart cities. In: Proceedings of the XP2017 Scientific Workshops. ACM, 2017. p. 12.

- IBBA, S., Pani, F. E., & Alberto Buschettu. (2016, November). Categorization and Matching for Drone-based Services. In KMIS (pp. 223-227).

- Mannaro, K., Baralla, G., IBBA, S., Pinna, A., & Garau, C. (2017, October). Towards a smart region: The case study of a teledermatology platform in sardinian region (Italy). In Wireless and Mobile Computing, Networking and Communications (WiMob), (pp. 370-377). IEEE.

- IBBA, S., Pani, F. E., Stockton, J. G., Barabino, G., Marchesi, M., & Tigano, D. (2017). Incidence of predatory journals in computer science literature. Library Review, 66(6/7), 505-522.

- Calderamo, M., IBBA, S., Pani, F. E., Piras, F., & Porru, S. An Innovative Web Application for Advanced Library Services. European Project Space on Computational Intelligence, Knowledge Discovery and Systems Engineering for Health and Sports, 3.

- Pani, F. E., Valcalda, B., IBBA, S., & Porru, S. Document Management. The Success of European Projects using New Information and Communication Technologies, 97.

- Calderamo, M., IBBA, S., Pani, F. E., Piras, F., & Porru, S. Advanced Museum Services. The Success of European Projects using New Information and Communication Technologies, 38.

- Mannaro, K., Baralla, G., Pinna, A., & IBBA, S. (2018). A Blockchain Approach Applied to a Teledermatology Platform in the Sardinian Region (Italy). Information, 9(2), 44.

- Baralla, G., IBBA, S., & Zenoni, R. (2017). Aposentu: A Social Semantic Platform for Hotels.

- Pinna, A., & IBBA, S. (2018). A blockchain-based Decentralized System for proper handling of temporary Employment contracts. Intelligent Computing - Proceedings of the 2018 Computing Conference - Advances in Intelligent Systems and Computing, Springer - **Winner of the Best Student Paper Award**

- Tonelli, R., Pinna, A., Baralla, G., & IBBA, S. Ethereum Smart Contracts as Blockchain-oriented Microservices. In: Proceedings of the XP2018 Scientific Workshops. ACM, 2018

- IBBA, S., Pinna, A., Baralla, G., & Marchesi, M. (2018, May). ICOs Overview: Should Investors Choose an ICO Developed with the Lean Startup Methodology?. In International Conference on Agile Software Development (pp. 293-308). Springer, Cham.

- IBBA, S., Pinna, A., Baralla, G.,Tonelli, R., & IBBA, S.Survey: how much the academic startups know and use Agile Software and Lean Startup methodologies? In: Proceedings of the XP2018 Scientific Workshops. ACM, 2018

- Baralla, G., IBBA, S. Marchesi, M., Tonelli, R., Missineo, S.,. A blockchain based system to ensure transparency and reliability in food supply. In Proceedings of the International Workshop on future perspective of decentralized applications - 24th International European Conference on Parallel and Distributed Computing, 2018

- IBBA, S., Pinna, A., Lunesu, M.I., Marchesi, M., Tonelli, R. (2018). Initial Coin Offerings and Agile Practices. Future Internet - IN PRESS

- Pinna, A., IBBA, S., Baralla, G., Tonelli, R. Marchesi, M. A Massive Analysis of Ethereum Smart Contracts. Exploratory study and code metrics. Information and software technology - UNDER REVIEW

Bibliography

[1] Jury.online tech development: Agile roadmap, 2018. [cited at p. 41]

[2] Solidity documentation, release 0.4.25, ethereum, 2018. [cited at p. 42, 43]

[3] Saman Adhami, Giancarlo Giudici, and Stefano Martinazzi. Why do businesses go crypto? an empirical analysis of initial coin offerings. *Journal of Economics and Business*, 2018. [cited at p. 14, 30]

[4] Iris Barsan. Legal challenges of initial coin offerings (icp). 2017. [cited at p. 14]

[5] Massimo Bartoletti, Tiziana Cimoli, and Roberto Zunino. Fun with bitcoin smart contracts. [cited at p. 10]

[6] Massimo Bartoletti and Livio Pompianu. An empirical analysis of smart contracts: platforms, applications, and design patterns. In *International Conference on Financial Cryptography and Data Security*, pages 494–509. Springer, 2017. [cited at p. 8, 54, 66]

[7] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. Manifesto for agile software development. 2001. [cited at p. 37]

[8] Kent Beck and Erich Gamma. *Extreme programming explained: embrace change*. addison-wesley professional, 2000. [cited at p. 48]

[9] Elisabeth SC Berger and Andreas Kuckertz. Female entrepreneurship in startup ecosystems worldwide. *Journal of Business Research*, 69(11):5163–5168, 2016. [cited at p. 35]

[10] K. Biswas and V. Muthukkumarasamy. Securing smart cities using blockchain technology. In *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1392–1393, Dec 2016. [cited at p. 98, 101]

[11] Jens Björk, Jens Ljungblad, and Jan Bosch. Lean product development in early stage startups. In *IW-LCSP@ ICSOB*, pages 19–32, 2013. [cited at p. 22]

[12] Steve Blank. *The four steps to the epiphany: successful strategies for products that win*. BookBaby, 2013. [cited at p. 21]

[13] Joseph A Blotner. Agile techniques to avoid firefighting at a start-up. In *OOPSLA 2002 Practitioners Reports*, pages 1–ff. ACM, 2002. [cited at p. 29]

[14] Jan Bosch, Helena Holmström Olsson, Jens Björk, and Jens Ljungblad. The early stage software startup development model: a framework for operationalizing lean principles in software startups. In *Lean Enterprise Software and Systems*, pages 1–15. Springer, 2013. [cited at p. 15]

[15] Santiago Bragagnolo, Henrique Rocha, Marcus Denker, and Stephane Ducasse. Smartinspect: Smart contract inspection technical report. *Inria Lille*, December 2017. [cited at p. 54]

[16] Santiago Bragagnolo, Henrique Rocha, Marcus Denker, and Stéphane Ducasse. Smartinspect: solidity smart contract inspector. In *2018 International Workshop on Blockchain Oriented Software Engineering, IWBOSE@SANER 2018, Campobasso, Italy, March 20, 2018*, pages 9–18, 2018. [cited at p. 95]

[17] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 2014. [cited at p. 7, 27, 51, 94]

[18] Christian Cachin. Architecture of the hyperledger blockchain fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*, 2016. [cited at p. 10]

[19] Yan Chen. Blockchain tokens and the potential democratization of entrepreneurship and innovation. *Business Horizons*, 61(4):567 – 575, 2018. [cited at p. 30]

[20] Jiri Chod and Evgeny Lyandres. A theory of icos: Diversification, agency, and information asymmetry. 2018. [cited at p. 31]

[21] Konstantinos Christidis and Michael Devetsikiotis. Blockchains and smart contracts for the internet of things. *Ieee Access*, 4:2292–2303, 2016. [cited at p. 3, 99]

[22] Alistair Cockburn and Jim Highsmith. Agile software development, the people factor. *Computer*, 34(11):131–133, 2001. [cited at p. 40]

[23] Gerry Coleman and Rory V O'Connor. An investigation into software development process formation in software start-ups. *Journal of Enterprise Information Management*, 21(6):633–648, 2008. [cited at p. 28]

[24] Giulio Concas, Giuseppe Destefanis, Michele Marchesi, Marco Ortu, and Roberto Tonelli. Micro patterns in agile software. In *International Conference on Agile Software Development*, pages 210–222. Springer, 2013. [cited at p. 104]

[25] Steve Counsell, Giuseppe Destefanis, Xiaohui Liu, Sigrid Eldh, Andreas Ermedahl, and Kenneth Andersson. Comparing test and production code quality in a large commercial multicore system. In *Software Engineering and Advanced Applications (SEAA), 2016 42th Euromicro Conference on*, pages 86–91. IEEE, 2016. [cited at p. 104]

[26] Tony; et al. Cui. Achain blockchain whitepaper, 2017. [cited at p. 10]

[27] Patrick Dai, Neil Mahi, Jordan Earls, and Alex Norta. Smart-contract value-transfer protocols on a distributed mobile application platform. *URL: https://qtum. org/uploads/files/cf6d69348ca50dd985b60425ccf282f3. pdf*, 2017. [cited at p. 10]

[28] Josh Davis. Do you have a communications strategy for your initial coin offering (ico)?, 2017. [cited at p. 37]

[29] D. S. Demidenko, E. D. Malevskaia-Malevich, and Y. A. Dubolazova. Iso as a real source of funding. pricing issues. In *2018 International Conference on Information Networking (ICOIN)*, pages 622–625, Jan 2018. [cited at p. 30]

[30] Giuseppe Destefanis, Andrea Bracciali, Michele Marchesi, Marco Ortu, Roberto Tonelli, and Robert Hierons. Smart contracts vulnerabilities: A call for blockchain software engineering? [cited at p. 54, 95]

[31] Giuseppe Destefanis, Steve Counsell, Giulio Concas, and Roberto Tonelli. Software metrics in agile software: An empirical study. In *International Conference on Agile Software Development*, pages 157–170. Springer, 2014. [cited at p. 48]

[32] Giuseppe Destefanis, Michele Marchesi, Marco Ortu, Roberto Tonelli, Andrea Bracciali, and Robert M. Hierons. Smart contracts vulnerabilities: a call for blockchain software engineering? In *2018 International Workshop on Blockchain Oriented Software Engineering, IWBOSE@SANER 2018, Campobasso, Italy, March 20, 2018*, pages 19–25, 2018. [cited at p. 95]

[33] SS Emtseva and NV Morozov. Comparative analysis of legal regulation of ico in selected countries. *KnE Social Sciences*, 3(2):77–84, 2018. [cited at p. 30]

[34] Michele Faioli, Emanuele Petrilli, and Donato Faioli. Blockchain, contratti e lavoro. la ri-rivoluzione del digitale nel mondo produttivo e nella pa. *Economia & lavoro*, 50(2):139–158, 2016. [cited at p. 89]

[35] Gianni Fenu, Lodovica Marchesi, Michele Marchesi, and Roberto Tonelli. The ico phenomenon and its relationships with ethereum smart contract environment. In *Blockchain Oriented Software Engineering (IWBOSE), 2018 International Workshop on*, pages 26–32. IEEE, 2018. [cited at p. 14, 30, 34, 52, 54, 58, 71]

[36] Eduardo Castello Ferrer. The blockchain: a new framework for robotic swarm systems. *CoRR*, abs/1608.00695, 2016. [cited at p. 99]

[37] John Flood and Lachlan Robb. Trust, anarcho-capitalism, blockchain and initial coin offerings. 2017. [cited at p. 14]

[38] Edmund A Gehan. A generalized wilcoxon test for comparing arbitrarily singly-censored samples. *Biometrika*, 52(1-2):203–224, 1965. [cited at p. 36]

[39] Antonio Ghezzi and Angelo Cavallo. Agile business model innovation in digital entrepreneurship: Lean startup approaches. *Journal of Business Research*, 2018. [cited at p. 29]

[40] Carmine Giardino, Nicolo Paternoster, Michael Unterkalmsteiner, Tony Gorschek, and Pekka Abrahamsson. Software development in startup companies: the greenfield startup model. *IEEE Transactions on Software Engineering*, 42(6):585–604, 2016. [cited at p. 29]

[41] Christopher Hahn, Adrian Wons, Christopher Hahn, and Adrian Wons. Umsetzung des icos, 2018. [cited at p. 31]

[42] Felix Hartmann, Xiaofeng Wang, and Maria Ilaria Lunesu. Evaluation of initial cryptoasset offerings: the state of the practice. In *Blockchain Oriented Software Engineering (IWBOSE), 2018 International Workshop on*, pages 33–39. IEEE, 2018. [cited at p. 14, 30]

[43] James A Highsmith and Jim Highsmith. *Agile software development ecosystems*, volume 13. Addison-Wesley Professional, 2002. [cited at p. 28]

[44] Jim Highsmith and Alistair Cockburn. Agile software development: The business of innovation. *Computer*, 34(9):120–127, 2001. [cited at p. 28]

[45] Simona Ibba, Andrea Pinna, Gavina Baralla, and Michele Marchesi. Icos overview: Should investors choose an ico developed with the lean startup methodology? In *International Conference on Agile Software Development*, pages 293–308. Springer, 2018. [cited at p. 30, 34, 52, 54, 71]

[46] Simona Ibba, Andrea Pinna, Matteo Seu, and Filippo Eros Pani. Citysense: Blockchain-oriented smart cities. In *Proceedings of the XP2017 Scientific Workshops*, XP '17, pages 12:1–12:5, New York, NY, USA, 2017. ACM. [cited at p. 52, 89, 95]

[47] Jiong Jin, Jayavardhana Gubbi, Slaven Marusic, and Marimuthu Palaniswami. An information framework for creating a smart city through internet of things. *IEEE Internet of Things Journal*, 1(2):112–121, 2014. [cited at p. 97]

[48] Wulf Kaal and Marco Dell'Erba. Initial coin offerings: Emerging practices, risk factors, and red flags. 2017. [cited at p. 14]

[49] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 839–858. IEEE, 2016. [cited at p. 27]

[50] Jiasun Li and William Mann. Initial coin offering and platform building. 2018. [cited at p. 30]

[51] Michal Lom, Ondrej Pribyl, and Tomas Zelinka. Hybrid-agile approach in smart cities procurement, 2016. [cited at p. 99]

[52] Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 254–269. ACM, 2016. [cited at p. 28, 51]

[53] Katiuscia Mannaro, Gavina Baralla, Andrea Pinna, and Simona Ibba. A blockchain approach applied to a teledermatology platform in the sardinian region (italy). *Information*, 9(2):44, 2018. [cited at p. 52, 95]

[54] Katiuscia Mannaro, Andrea Pinna, and Michele Marchesi. Crypto-trading: Blockchain-oriented energy market. In *AEIT International Annual Conference, 2017*, pages 1–5. IEEE, 2017. [cited at p. 52, 95]

[55] Robert C Martin. *Agile software development: principles, patterns, and practices*. Prentice Hall, 2002. [cited at p. 48]

[56] Thomas J McCabe. A complexity measure. *IEEE Transactions on software Engineering*, (4):308–320, 1976. [cited at p. 45, 77]

[57] Adnan Miski. Development of a mobile application using the lean startup methodology. *International Journal of Scientific & Engineering Research*, 5(1):1743–1748, 2014. [cited at p. 22, 29]

[58] Ethan Mollick. The dynamics of crowdfunding: An exploratory study. *Journal of business venturing*, 29(1):1–16, 2014. [cited at p. 14]

[59] Paul P Momtaz. Initial coin offerings. 2018. [cited at p. 31]

[60] Dobrila Rancic Moogk. Minimum viable product and the importance of experimentation in technology startups. *Technology Innovation Management Review*, 2(3), 2012. [cited at p. 22]

[61] Roland M Müller and Katja Thoring. Design thinking vs. lean startup: A comparison of two user-driven innovation strategies. *Leading through design*, 151, 2012. [cited at p. 22]

[62] John W Mullins, John Walker Mullins, John Mullins, and Randy Komisar. *Getting to plan B: Breaking through to a better business model*. Harvard Business Press, 2009. [cited at p. 22]

[63] Ashish Mundra, Sanjay Misra, and Chitra A Dhawale. Practical scrum-scrum team: Way to produce successful and quality software. In *Computational Science and Its Applications (ICCSA), 2013 13th International Conference on*, pages 119–123. IEEE, 2013. [cited at p. 48]

[64] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008. [cited at p. 7, 27]

[65] Sridhar Nerur, RadhaKanta Mahapatra, and George Mangalaraj. Challenges of migrating to agile methodologies. *Communications of the ACM*, 48(5):72–78, 2005. [cited at p. 28]

[66] Alex Norta, Anton Vedeshin, Hando Rand, Simon Tobies, Addi Rull, Margus Poola, and Teddi Rull. Self-aware agent-supported contract management on blockchains for legal accountability. *URL: http://whitepaper. agrello. org/Agrello_Self-Aware_ Whitepaper. pdf*, 2017. [cited at p. 89]

[67] Robert Norvill, Beltran Borja Fiz Pontiveros, Radu State, Irfan Awan, and Andrea Cullen. Automated labeling of unknown contracts in ethereum. In *Computer Communication and Networks (ICCCN), 2017 26th International Conference on*, pages 1–6. IEEE, 2017. [cited at p. 54]

[68] Svein Ølnes and Arild Jansen. Blockchain technology as s support infrastructure in e-government. In *International Conference on Electronic Government*, pages 215–227. Springer, 2017. [cited at p. 89]

[69] Marco Ortu, Giuseppe Destefanis, Steve Counsell, Stephen Swift, Michele Marchesi, and Roberto Tonelli. How diverse is your team? investigating gender and nationality diversity in github teams. *Peerj Preprints*, 2016. [cited at p. 35]

[70] Marco Ortu, Giuseppe Destefanis, Matteo Orru, Roberto Tonelli, and Michele L Marchesi. Could micro patterns be used as software stability indicator? In *Patterns Promotion and Anti-patterns Prevention (PPAP), 2015 IEEE 2nd Workshop on*, pages 11–12. IEEE, 2015. [cited at p. 55]

[71] Marco Ortu, Giuseppe Destefanis, Stephen Swift, and Michele Marchesi. Measuring high and low priority defects on traditional and mobile open source software. In *Proceedings of the 7th International Workshop on Emerging Trends in Software Metrics*, pages 1–7. ACM, 2016. [cited at p. 98]

[72] Aafaf Ouaddah, Anas Abou Elkalam, and Abdellah Ait Ouahman. Fairaccess: a new blockchain-based access control framework for the internet of things. *Security and Communication Networks*, pages n/a–n/a, 2017. SCN-16-0184. [cited at p. 99]

[73] Jevgenija Pantiuchina, Marco Mondini, Dron Khanna, Xiaofeng Wang, and Pekka Abrahamsson. Are software startups applying agile practices? the state of the practice from a large survey. In *International Conference on Agile Software Development*, pages 167–183. Springer, 2017. [cited at p. 29, 35]

[74] Nicolò Paternoster, Carmine Giardino, Michael Unterkalmsteiner, Tony Gorschek, and Pekka Abrahamsson. Software development in startup companies: A systematic mapping study. *Information and Software Technology*, 56(10):1200–1218, 2014. [cited at p. 29]

[75] Andrea Pinna and Simona Ibba. A blockchain-based decentralized system for proper handling of temporary employment contracts. *arXiv preprint arXiv:1711.09758*, 2017. [cited at p. 95]

[76] Andrea Pinna, Roberto Tonelli, Matteo Orru, and Michele Marchesi. A petri nets model for blockchain analysis. *The Computer Journal*, 2018. [cited at p. 95, 98]

[77] Mary Poppendieck and Michael A Cusumano. Lean software development: A tutorial. *IEEE software*, 29(5):26–32, 2012. [cited at p. 15]

[78] Simone Porru, Andrea Pinna, Michele Marchesi, and Roberto Tonelli. Blockchain-oriented software engineering: Challenges and new directions. In *Proceedings of the 39th International Conference on Software Engineering Companion*, ICSE-C '17, pages 169–171, Piscataway, NJ, USA, 2017. IEEE Press. [cited at p. 30, 52, 53, 94, 98]

[79] Andrew K Przybylski, Kou Murayama, Cody R DeHaan, and Valerie Gladwell. Motivational, emotional, and behavioral correlates of fear of missing out. *Computers in Human Behavior*, 29(4):1841–1848, 2013. [cited at p. 35]

[80] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2018. [cited at p. 32]

[81] Eric Ries. *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses*. Crown Books, 2011. [cited at p. 21, 22, 23]

[82] Torjus Roberg. Research suggests that women are now leading the market for crypto investment, 2018. [cited at p. 35]

[83] Henrique Rocha, Stéphane Ducasse, Marcus Denker, and Jason Lecerf. Solidity parsing using smacc: Challenges and irregularities. In *International Workshop on Smalltalk Technology IWST'17*, 2017. [cited at p. 54, 58]

[84] Cláudio Sant'Anna, Alessandro Garcia, Christina Chavez, Carlos Lucena, and Arndt Von Staa. On the reuse and maintenance of aspect-oriented software: An assessment framework. In *Proceedings of Brazilian symposium on software engineering*, pages 19–34, 2003. [cited at p. 45]

[85] Robbin Schuurman. Tips for agile product roadmaps and product roadmap examples, 2017. [cited at p. 41]

[86] Ken Schwaber. *Agile project management with Scrum*. Microsoft press, 2004. [cited at p. 35]

[87] Mali Senapathi and Ananth Srinivasan. Sustained agile usage: a systematic literature review. In *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*, pages 119–124. ACM, 2013. [cited at p. 34]

[88] Pradip Kumar Sharma, Seo Yeon Moon, and Jong Hyuk Park. Block-vn: A distributed blockchain based vehicular network architecture in smart city. *Journal of Information Processing Systems*, 13(1):84 195, 2017. [cited at p. 98]

[89] Sandra EP Silva, Robisom D Calado, Messias B Silva, and MA Nascimento. Lean startup applied in healthcare: A viable methodology for continuous improvement in the development of new products and services. *IFAC Proceedings Volumes*, 46(24):295–299, 2013. [cited at p. 22]

[90] M. Stocchi, I. Lunesu, S. Ibba, G. Baralla, and M. Marchesi. The future of bitcoin: a synchrosqueezing wavelet transform to predict search engine query trends. In CEUR Workshop Proceedings, editor, *2nd international workshop on Knowledge Discovering on the Web (KDWEB)*, volume 1748. ceur-ws.org, 2016. [cited at p. 98]

[91] Margaret-Anne Storey, Leif Singer, Brendan Cleary, Fernando Figueira Filho, and Alexey Zagalsky. The (r) evolution of social media in software engineering. In *Proceedings of the on Future of Software Engineering*, pages 100–116. ACM, 2014. [cited at p. 36]

[92] Matt Suiche. Porosity: A decompiler for blockchain-based smart contracts bytecode. *DEF CON*, 25, 2017. [cited at p. 11]

[93] Melanie Swan. *Blockchain: Blueprint for a new economy*. " O'Reilly Media, Inc.", 2015. [cited at p. 27, 30, 51, 52]

[94] Nick Szabo. The idea of smart contracts. 1997. [cited at p. 7, 10]

[95] Davide Taibi, Valentina Lenarduzzi, and Claus Pahl. Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation. *IEEE Cloud Computing*, 4(5):22–32, 2017. [cited at p. 94]

[96] Davide Taibi, Valentina Lenarduzzi, Claus Pahl, and Andrea Janes. Microservices in agile software development: a workshop-based study into issues, advantages, and disadvantages. In *Proceedings of the XP2017 Scientific Workshops*, page 23. ACM, 2017. [cited at p. 94]

[97] H.C. Thode. *Testing For Normality*. Statistics, textbooks and monographs. Taylor & Francis, 2002. [cited at p. 36]

[98] Feng Tian. An agri-food supply chain traceability system for china based on rfid blockchain technology. In *2016 13th International Conference on Service Systems and Service Management (ICSSSM)*, pages 1–6, June 2016. [cited at p. 99]

[99] Roberto Tonelli, Giuseppe Destefanis, Michele Marchesi, and Marco Ortu. Smart contracts software metrics: a first study. *arXiv preprint arXiv:1802.01517*, 2018. [cited at p. 43, 44, 45, 46, 54]

[100] Percy Venegas. Initial coin offering (ico) risk, value and cost in blockchain trustless crypto markets. 2017. [cited at p. 14]

[101] Zhiyuan Wan, David Lo, Xin Xia, and Liang Cai. Bug characteristics in blockchain systems: a large-scale empirical study. In *Mining Software Repositories (MSR), 2017 IEEE/ACM 14th International Conference on*, pages 413–424. IEEE, 2017. [cited at p. 54]

[102] Nianxin Wang, Qingxiang Li, Huigang Liang, Taofeng Ye, and Shilun Ge. Understanding the importance of interaction between creators and backers in crowdfunding success. *Electronic Commerce Research and Applications*, 27:106–117, 2018. [cited at p. 15]

[103] Xin Wang, Libo Feng, Hui Zhang, Chan Lyu, Li Wang, and Yue You. Human resource information management model based on blockchain technology. In *Service-Oriented System Engineering (SOSE), 2017 IEEE Symposium on*, pages 168–173. IEEE, 2017. [cited at p. 89]

[104] Maximilian Wohrer and Uwe Zdun. Smart contracts: security patterns in the ethereum ecosystem and solidity. In *2018 International Workshop on Blockchain Oriented Software Engineering, IWBOSE@SANER 2018, Campobasso, Italy, March 20, 2018*, pages 2–8, 2018. [cited at p. 95]

[105] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014. [cited at p. 3, 11, 51]

[106] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1):22–32, 2014. [cited at p. 97]

[107] Yu Zhang and Jiangtao Wen. The IoT electric business model: Using blockchain technology for the internet of things. *Peer-to-Peer Networking and Applications*, page 1–12, 2016. [cited at p. 99]