Learning Smart Contracts for Business Environment

Luís Silvestre[1], Francisco Pires[2] and Jorge Bernardino[1,3][a]
[1]Polytechnic of Coimbra – ISEC, Rua Pedro Nunes, Quinta da Nora, 3030-199 Coimbra, Portugal
[2]Individual Consulting Research, Rua Arlindo Vicente, 3030-298 Coimbra, Portugal
[3]CISUC - Centre of Informatics and Systems of University of Coimbra, Pinhal de Marrocos, 3030-290 Coimbra, Portugal

Keywords: Blockchain, Smart Contracts, Business Environment.

Abstract: Currently the search for a decentralized data model in companies for its big advantage in removing the middleman has been increasing. For that reason, DLT (Distributed Ledger Technology) technologies have gained a lot of visibility in the business world, the most well-known being Blockchain and its emerging Smart Contracts. The identified problem is the lack of knowledge and skill of companies in the domain of the rising Smart Contracts. In this paper, we propose a generic model that could increase the competence of companies in this field by creating a step-by-step tutorial on how to set up the development environment of Smart Contracts.

1 INTRODUCTION

Although Blockchain has arisen for some time now, there are still many developments to be achieved with this technology and more and more are the uses of this technology reflected on the most varied topics (Maxmen, 2018; Baynham-Herd, 2017; Ahmed, 2017) since its use with Artificial Intelligence (Salah, 2019) to its advantages in IoT (Internet of Things) (Makhdoom, 2019). Like Blockchain, Smart Contracts are also revolutionizing the industry with its potential for decentralized applications. Given its recent appearance, however, there is still a very large lack of knowledge and skills in this area, especially from companies that want to enter this market (Casino, 2019).

This work aims to reduce this gap by creating a step-by-step tutorial guide on how to prepare an environment for developing Smart Contracts and giving some background. We intend answering these questions: (i) How Does Blockchain work? (ii) What are Smart Contracts, and what do they do? (iii) How to start Developing Smart Contracts? (iv) How to apply a Smart Contracts guide to Companies?

In this paper, we propose a training model following the ADDIE (Analyze, Design, Develop, Implement and Evaluate) model.

To this end, we first explore the concept of Blockchain and Smart Contracts, providing some background to what is going to be explored. After this phase of learning concepts and what will be studied, we will enter the guide itself on how to create the environment for the development of Smart Contracts. At this stage, it will be necessary to understand not only what is necessary to obtain to create the environment, but also why it is necessary.

The main contribution of this work is the creation of a step-by-step tutorial as a learning model to apply in training actions in a business environment.

The remainder of this article is structured as follows. In section 2 it is given a contextualization and a brief explanation about Blockchain and Smart Contracts. In section 3 it's described the preparation of the development environment, and in section 4 it's made an overview of the state of the art. Section 5 serves as a brief introduction of the ADDIE (Analyze, Design, Develop, Implement and Evaluate) model and its application in this work. In section 6 we describe the Learning-Transfer Evaluation Model (LTEM) and the advantages of its application in our learning model, as well as some future ideas on how the model will evolve. Finally, section 7 discusses the results and presents the main conclusions.

[a] https://orcid.org/0000-0001-9660-2011

Silvestre, L., Pires, F. and Bernardino, J.
Learning Smart Contracts for Business Environment.
DOI: 10.5220/0007959301880195
In *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications (ICETE 2019)*, pages 188-195
ISBN: 978-989-758-378-0

2 BACKGROUND

In this section, an introduction to the Blockchain domain and the Smart Contracts is given by providing a brief explanation of some important concepts, such as DLT (Distributed Ledger Technologies), Blockchain and Smart Contracts.

2.1 DLT – Distributed Ledger Technology

Distributed Ledger Technologies as the name suggests, are distributed record systems. That is, the information is not all in one place, but distributed by all system participants as shown in Figure 1.

Unlike traditional databases, distributed ledgers have no central data storage or administration functionality. In a DLT, each node processes and verifies each item, thus generating a record of each item and creating a consensus on the veracity of each item.

This architecture represents a significant revolution in record keeping, changing how information is collected and communicated.

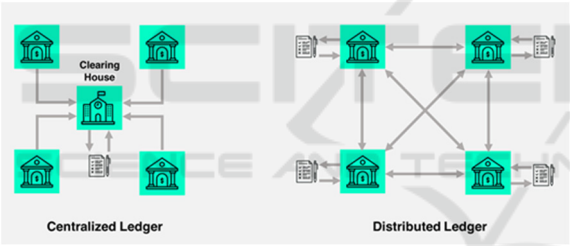

Figure 1: Centralized Ledger vs Distributed Ledger (Tradeix.com, 2019).

2.2 Blockchain

The blockchain is one of many existing DLTs and it's the most recognized of these technologies being the core of bitcoin. Blockchain works on a Peer-to-Peer network, which is a network with an architecture in which tasks are distributed among peers, in contrast to the centralized architecture where information is stored in one place. Blockchain emerged in 2008 when Satoshi Nakamoto published his article "Bitcoin: A Peer-to-Peer Electronic Cash System" (Nakamoto, 2008). As the name implies, it consists of a chain of blocks. It is in these blocks that the information is stored. Figure 2 shows a generic representation of a block.

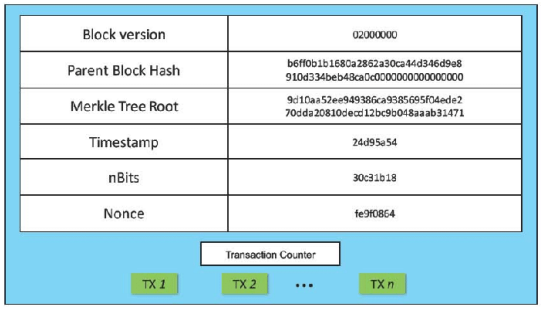

Figure 2: Blockchain's blocks structure (Zheng, 2018).

The main feature of Blockchain is immutability, which means that when a block is validated in the network, it is no longer possible to change it. The act of validating blocks is called mining and consists of finding a value for the nonce parameter, which satisfies the conditions of validation. These conditions go on to generate a valid HashCode. Hashcode is a numeric value generated by algorithms based on the data in the block, which means that if some data changes, the Hashcode changes. The mining consists of obtaining the nonce that causes the Hashcode to fulfill certain validation rules such as starting with 10 zeros. These rules change depending on the platform.

Those responsible for the mining process are called miners. A miner is any participant in the network that aims to validate a block of transactions, using the mining process.

The mining process is based on a model called proof-of-work. That is, the confidence of the validation is correct is obtained through the computational effort used to make the mining. As an incentive to make people want to mine, the act of mining is usually rewarded. Currently, there is an attempt to move from proof-of-work to proof-of-stake. This new method has the same goal of proof-of-work but varies the way consensus is achieved. Unlike the proof-of-work, in which the miners have to solve a mathematical problem and spend computational resources to validate and create a block, in this method the creator of each block is determined based on its possessions and there are no block rewards. This method causes energy consumption to decrease because the miners do not compete with each other for the rewards, which leads to an increase in the difficulty of a 51% attack. A 51% attack, as the name implies, is an attack on Blockchain networks that consists of obtaining 51% of the network participants in order to obtain consensus and consequently the validation of the blocks.

2.3 Smart Contracts

Smart Contracts are digital protocols that define a contract and use the Blockchain network for its immutability and audibility. Smart Contracts are programmable so that, in addition to validating contract conditions, they impose restrictions and penalties (Salah, 2019; Bocek, 2017). The first appearance of the term was in 1997 by Nick Szabo (Szabo, 1997).

Smart Contracts can be used for financial services or for general services, The first one means simple financial transactions, the later means that the data being changed by the Smart Contracts does not necessarily need to be of financial nature, this opens the path to what is now called DApps, Decentralized Applications. Decentralized applications are applications that run on the Blockchain network and do not need any regulatory authority. An example of a decentralized application is Bitcoin, as well as any other virtual currency platform. However, the major difference from a decentralized application with Smart Contracts is that virtual currency platforms work only with transactions, whereas a decentralized application using Smart Contracts can include much more information in its blocks than a simple transaction and other information and interactions can be programmed (Wu, 2019).

The company responsible for the great Smart Contracts revolution is Ethereum, a decentralized platform capable of executing smart contracts and decentralized applications using Blockchain technology. This platform allows users to develop and deploy Smart Contracts to the network by paying a fee of their current cryptocurrency, Ether.

3 ENVIRONMENT SETUP

In this section we describe the environment preparation process using the Windows 10 Operating System and the Google Chrome browser, listing in order the necessary software, followed by programming languages.

This preparation process consists of installing and configuring the necessary programs for the development of Smart Contracts, as well as providing an explanation for the need for these programs. The environment described will be an initial environment for testing and the first contact with the Smart Contracts and Ethereum networks. Table 1 shows a summary of all the steps to be performed, the reason and the way they are installed. Table 2 lists the programming languages involved in the development of Smart Contracts and lists their functions in this environment.

Table 1: Necessary steps and motives.

Steps	Motives	Installation
A. Nodejs e Npm	It allows you to run JavaScript applications and makes several open-source applications available as modules.	Download the installer from the official website and run it.
B. Testrpc	It simulates the ethereum blockchain network.	npm install -g testrpc
C.Web3.js	API provided by Ethereum that provides interaction with the Ethereum network.	npm install web3@0.19
D. Metamask	It allows for access and interaction with the Ethereum network in a simpler way.	Extension of Google Chrome
E. Remix	Developed and made available by Ethereum for the development of Smart Contracts.	Not necessary because there is a browser version. npm install -g remix-ide

3.1 Nodejs and Npm (Node Packet Manager)

You need to download and install node.js, which is a tool that allows you to run applications in JavaScript and supports several open source modules. That is, it works as a server for various JavaScript applications. The current nodejs installer of the official site also includes npm. This tool is used to manage the existing modules in nodejs and allows the user to obtain these modules through the Windows command line with the command "npm install <module name @ version> "(It is not mandatory to specify the version, in which case it will be obtained the latest version).

3.2 Testrpc

Testrpc, now called ganache-cli, is a tool that is provided as a nodejs module and serves to simulate the Ethereum network in a fast and flexible way.

It can be installed through npm with the command "npm install -g testrpc" and the "testrpc" command to execute it. In addition to providing the simulated network, this tool provides some default test accounts to use.

3.3 Web3.js

Web3.js is an API (Application Programming Interface) created and provided by Ethereum, which consists of a collection of already defined modules and functions that allow interaction with the Ethereum network both locally and remotely.

It is necessary to include this API in the desired project by opening the Windows command line in the project folder and executing the "npm init" command, which will create a JavaScript Object Notation (JSON) file with information about the project dependencies. Then you need to run the command "npm install web3@0.19". This command will download and install version 0.19 of web3. Optionally this API can be downloaded with the latest version with the command "npm install web3". However, the latest version is in the Beta phase, so unexpected problems can be encountered. As a result, version 0.19 is advisable.

3.4 Metamask and Lite-server

The next step is to install MetaMask, a Chrome extension that allows you to access and interact with the Ethereum network without running all Nodes, and lite-server, a fast and light web server. To install MetaMask, simply go to the extensions section of Google Chrome and look for MetaMask. After adding this, follow the installation steps. After installing MetaMask you will need to install the lite-server in the project, which will allow MetaMask to inject an instance of the web3 API. To install the lite-server in the project, it is necessary to open the command line in the project folder and execute the "npm install lite-server" command.

3.5 Remix IDE

The final step is to use the IDE offered by Ethereum for programming Smart Contracts in Solidity. This IDE is used through a browser. Simply enter the link "https://remix.ethereum.org". This IDE allows you to schedule the contracts and to launch these contracts in the Ethereum network. For this, it is necessary to access the "run" window and select the "Injected Web3" environment. The account will be automatically synchronized with the account created in MetaMask.

Table 2: Used Programming languages.

Programming Language	Purpose
Solidity	Ethereum's Smart Contracts programming language.
JavaScript	A programming language that links the web page to Smart Contracts.
Html/Css	Language that allows you to build and customize web pages.

3.6 Solidity

Solidity is the programming language for Smart Contracts, created by Ethereum. It is a high-level object-oriented language with Python, C ++ and JavaScript influences. In this language it is possible to define a Smart Contract and its properties, as well as its behavior. Figure 3 shows an example of creating a Smart Contract using the Solidity language, for this guide we will use the version 0.5.5.

```
pragma solidity ^0.5.5;

contract Version{
    uint v;
    string description;
    bytes32[] authors;

    constructor (uint version,string memory desc)public{
        v=version;
        description=desc;
    }

    function addAuthor(bytes32 author) public {
        authors.push(author);

        emit alertNewAuthor(author);

    }

    function removeAuthor(bytes32 author) public {
        uint i=0;

        for(i=0;i<authors.length;i++){
            bytes32 a = authors[i];
            if(a==author){
                authors[i--];
            }
        }
    }
    event alertNewAuthor(bytes32 author);
}
```

Figure 3: Example of a Smart Contract in Solidity.

In this example, a "version" name contract is created in which you can define the version, a description, and several authors that are saved in a vector. We can check some language properties such as variable types, String, bytes32 array and uint (unsigned int), as well as the definition of a constructor, some functions such as adding and removing an author, and events. Of these properties of Solidity referred to, the events stand out. These

events offer the possibility to mark an action, called through the keyword “emit” and defined through the keyword “event”. Events allow you to schedule so that when a certain event occurs, certain methods within the code are executed, these events can then be received in a standby function in the JavaScript code.

3.7 JavaScript

JavaScript is a programming language used for web development and it will allow us to program the interaction of the user with the Smart Contract interface.

Through JavaScript, it is possible to obtain and use the Web3.js API, which allows access to the Ethereum network. Figure 4 shows an example of JavaScript code on how to obtain an API instance.

```
if(typeof web3 !== 'undefined'){
    web3 = new Web3(web3.currentProvider);
}
else{
    web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545"));
}
```

Figure 4: JavaScript code example to obtain web3 API instance.

In this example, it is shown how to obtain an instance of the web3 API, first we check if it is already defined, if it is then we just create the instance with the current Provider, this happens because we will use MetaMask which will inject a web3 API instance, if it isn’t defined then we create a new instance passing as parameters the new provider.

3.8 HTML/CSS

HTML (HyperText Markup Language) allows you to easily create a web page and customize it with the help of CSS (Cascading Style Sheets). This language is necessary because it will be essential to provide an interface so that Smart Contracts can be tested.

In general, following all these steps will be possible to get an environment to start developing Smart Contracts for Ethereum networks.

Using Remix, the contracts created in Solidity can be deployed with MetaMask. When deploying a contract, a window with information of the transaction will be visualized (or shown), and using HTML and CSS a simple Web page can be created to test the Smart Contract. For the Web to correctly use the created Smart Contract, it will be needed to copy the Smart Contract Application Binary Interface (ABI), as well as the contract HashCode given in the remix IDE to the JavaScript code.

4 RELATED WORK

Currently, Blockchain technology has been exploited for implementation in many areas, mainly due to the great revolution caused by Smart Contracts and its ability to allow the creation of decentralized applications. Although the term Smart Contracts is not new (Szabo, 1997), its application in Blockchain opens the door to many possibilities that are already under study. Data is the most valuable asset in any organization (Santos, 2011), therefore any Blockchain implementation can be a huge improvement in some scenarios. For Smart Contracts, there is already a free online introduction course (Coursetro.com, 2019), which addresses the preparation of the environment and the introduction to the creation of Smart Contracts. In this course a video tutorial is provided, as well as a written tutorial on how to start programming the Smart Contracts, preparing the environment and then introducing the Solidity language. Despite the quality of the course, there are some missing pieces because it is outdated and also because it is meant to be just and introduction to the theme. Besides this Smart Contracts introduction course, also related to the work done in this paper, but with more focus on business fields there is also a framework that provides an easier access to smart contracts for companies by allowing the development of smart contracts logic in a controlled English, business-level rules language (Astigarraga T, 2018). This is a good solution to easily implement blockchain and Smart Contract technologies in companies however it doesn’t provide the necessary knowledge and background to understand blockchain only provides an ease of use for companies who aim to work on top of this technologies. Accordingly, the work presented in sections II and III was intended to: 1) offer an improved and updated step-by-step guide on how to prepare the environment, 2) provide the first part of introducing concepts and 3) introduce the application strand in the business environment.

5 ADDIE MODEL

For the application of this guide in a business environment, it is necessary to make it a well-structured learning model. In this context, the ADDIE model emerges, a model that allows transforming the simple guide for creating the development environment into a learning model that is apprehended by a focal point responsible for the

subsequent application to the human resources of the company.

The ADDIE model, shown in Figure 5, is an Instructional Design Model (IDM). It is a model that proposes certain rules for organizing pedagogical scenarios in order to define correct learning objectives.

As an IDM, the ADDIE model serves as a framework for the creation of training programs, and for this reason this work follows this model's guides.

The following are the essential phases of the ADDIE (Analyze, Design, Develop, Implement and Evaluate) model, as well as a brief explanation and objectives for each phase.

- Analyze: In this phase, the goals and objectives of the program are defined, and it is evaluated the current knowledge of the target audience in order to assess what the program will need to do.
- Design: At this stage, the whole program is planned in detail, from the content to the approach and tools to be used.
- Development: The development phase is where the developers create and assemble the content assets that were created in the design phase. It is also possible to review the project at this stage.
- Implementation: This phase reflects the constant modification of the program in order to achieve maximum effectiveness. It is at this stage that the program is adjusted as required.
- Evaluation: It is the final phase and consists of a final and meticulous test of the whole program, which is divided into two parts: formative and summative.

The formative assessment is done at all stages and is done during implementation with both trainers and students. The summative assessment is done after the implementation phase ends and serves to improve the program based on the results.

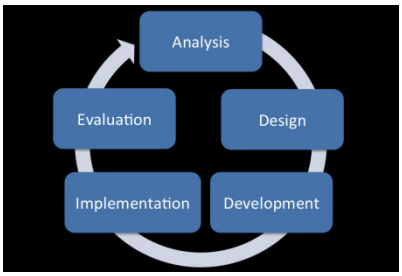

Figure 5: ADDIE model diagram (Educationaltechnology.net, 2017).

6 LTEM BASED MODEL

In the previous section, the application of a learning model, the ADDIE model, was mentioned for the transformation of this guide into a well-structured learning model. In this section, the evaluation of the learning of this model is presented. Learning models do not always work as expected, and from this perspective, there is no guarantee that the created model will play its role in perfection. To solve this, we will resort to an evaluation model, the LTEM (Learning-Transfer Evaluation Model), shown in Figure 6.

The LTEM model (Worklearning.com, 2019) is an evaluation model designed especially for organizations with the ability to obtain feedback, to improve learning processes and to validate their results. This model is divided into 8 levels, where the first step reflects an inadequate level and the last one represents the ultimate learning objective. Benefits of the latter can be applied in various learning scenarios from a classroom to e-Learning. The following is a brief explanation of each of these echelons.

6.1 Attendance

It corresponds to an evaluation method in which only confirmation of presence in a certain class or training is necessary. This method is inadequate because proof of presence does not imply proof of learning.

6.2 Activity

It is a method that consists of evaluating learning through the activity of the learner. This activity can be measured in 3 ways:

- Attention
- Interest
- Participation

This method is also unacceptable for evaluation because these 3 aspects do not reveal learning results. Attention, interest, and participation are not synonyms of learning and students who demonstrate these 3 aspects may well not have learned. Examples of this method are some online courses that require you to periodically push a button to move forward. In these methods, the demonstration of learning outcomes is lacking, because, despite the 3 aspects, students may not only have not learned, but they may have learned the wrong thing or may have focused on the wrong point.

6.3 Learner Perceptions

In this method of evaluation, students' feedback is used to evaluate learning, and this is done through student surveys. Usually in these types of inquiries questions such as the competence of the instructors, expectations of the course, among others, are asked. This is another ineffective method because the students' expectations do not match the learning outcomes.

6.4 Knowledge

Testing knowledge is another way of assessing learning and can be done right after learning or can occur after some time has passed; both forms test knowledge regarding facts and terminology.

In the first one, knowledge is tested while the information is still accessible and there is no time for the information to be forgotten. For this reason, there is no guarantee that this information is always available in the future. On the other hand, knowledge alone does not generate decision-making capacity. For these reasons, this first way to assess learning by testing knowledge quickly after being acquired is not adequate.

In the second, knowledge is tested a few days after its acquisition, which also implies the recall of knowledge and therefore is a better option in relation to the first form. However, it remains an inadequate evaluation method for the same reasons.

6.5 Decision-making Competence

More important than just knowledge, learning outcomes should reflect an understanding of knowledge, as well as the ability to make decisions and perform realistic tasks. This method consists of evaluating through the ability to make decisions. Given the importance of short and long-term memory, we can divide this assessment also in these two strands. The assessment of the ability to make short-term decisions is made on the day of learning and so it is still insufficient to be considered adequate because there is no guarantee that this capacity will be maintained over time. Long-term assessment, on the other hand, is already considered an adequate assessment for learning, offering realistic scenarios or simulations in which the student will have to make decisions.

6.6 Task Competence

At this level, both the ability to make decisions and the actions taken are evaluated. In this evaluation, the practice in a realistic environment is promoted. This is done through the SEDA (Situation, Evaluation, Decision, Action) model, which is a model that represents in a simple way what happens in the real world. Just as the decision-making ability of this evaluation can occur in the short and long term, and in the short term there is the same problem as in the previous one, there is no guarantee that this competence will be maintained in the long term. On the long term this means that a student on this level is capable of the next level, transfer of knowledge.

6.7 Transfer

At this level it is expected that there is a transfer of learning to be applied in the real world, and this can happen in two ways: Half transfer or Full Transfer. In Half Transfer, this change is assisted. That is, there is help for this transfer initiative. On the other hand, in Full Transfer this transfer occurs without significant help.

6.8 Effects of Transfer

The learning transfer result always generates impact, and it is also up to this model to assess this impact and generate a list of those affected, from team members to the organization itself or family and friends. Furthermore, it is necessary to assess whether this impact is positive or negative.

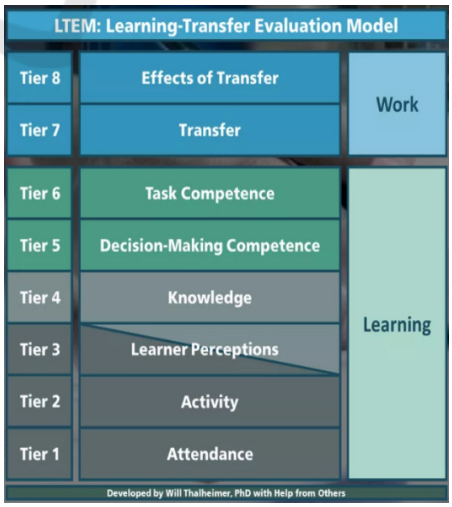

Figure 6: LTEM Model diagram (Worklearning.com, 2019).

We intend to improve this guide so that it can include more than creating and preparing the development environment, but also enabling the launch of Smart Contracts in a simulated Blockchain network. We also want to include the creation of a simulated Blockchain network, where it's expected the possibility to create and develop a complete decentralized application. More than creating these learning models, it will be essential to also apply the LTEM model to evaluate this learning offered in a business environment.

7 CONCLUSIONS AND FUTURE WORK

The gap in knowledge and skills in the business world in Blockchain and especially in the growing Smart Contracts, at a time when these concepts are increasingly important, led to an attempt to create something with the aim of filling that gap.

This article has offered in the first part, an introduction to the Blockchain and Smart Contracts domain, giving some background into its key concepts and has described a step-by-step tutorial on how to prepare the Smart Contracts development environment.

In this work, a training model was proposed with application in business environments following the norms of the ADDIE model and already considering the LTEM model for evaluation.

As future work, we intend to improve and apply this first learning model by going further and deeper within the field of Smart Contracts aiming at the creation of decentralized applications using the LTEM model for evaluation.

REFERENCES

A. Maxmen, "Ai researchers embrace bitcoin technology to share medical data", *Nature, vol. 555, pp. 293-294, Mar. 2018*.

Astigarraga T. et al. (2018) Empowering Business-Level Blockchain Users with a Rules Framework for Smart Contracts. In: Pahl C., Vukovic M., Yin J., Yu Q. (eds) Service-Oriented Computing. ICSOC 2018. *Lecture Notes in Computer Science, vol 11236. Springer, Cham*.

Coursetro.com, 'Developing Ethereum Smart Contracts for Beginners'. [Online]. Available: https://coursetro.com/courses/20/Developing-Ethereum-Smart-Contracts-for-Beginners. [Accessed 25 – Mar – 2019].

Educationaltechnology.net, 'ADDIE Model: Instructional Design', 2017. [Online]. Available: https://educationaltechnology.net/the-addie-model-instructional-design. [Accessed 25 – Mar – 2019].

Fran Casino, Thomas K. Dasaklis, Constantinos Patsakis, "A systematic literature review of blockchain-based applications: Current status, classification and open issues", *Telematics and Informatics, Volume 36, 2019, Pages 55-81*.

Imran Makhdoom, Mehran Abolhasan, Haider Abbas, Wei Ni: Blockchain's adoption in IoT: The challenges, and a way forward. *J. Network and Computer Applications 125: 251-279 (2019)*.

Kaidong Wu, "An Empirical Study of Blockchain-based Decentralized Applications.", *CoRR abs/1902.04969 (2019)*.

Khaled Salah, Muhammad Habib Ur Rehman, Nishara Nizamuddin, Ala I. Al-Fuqaha: Blockchain for AI: Review and Open Research Challenges. *IEEE Access 7: 10127-10149 (2019)*.

N. Szabo, "The idea of smart contracts," Nick Szabos Papers and Concise Tutorials, vol. 6, 1997.

R. J. Santos, J. Bernardino, and M. Vieira. "A data masking technique for data warehouses". In *Proc. of the 15th Symposium on Int. Database Engineering & Applications (IDEAS '11). ACM, 61-69., 2011*.

S. Ahmed, N. T. Broek, "Blockchain could boost food security", *Nature, vol. 550, no. 7674, pp. 43, 2017*.

Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, 2008.

T. Bocek, B. B. Rodrigues, T. Strasser, and B.Stiller, "Blockchains everywhere - A user-case of blockchains in the pharma supply-chain," in *Proc.IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM), May 2017, pp. 772-777*.

Tradeix.com, 'The difference between Blockchain & Distributed Ledger Technology'. [Online]. Available: https://tradeix.com/distributed-ledger-technology/. [Accessed 25 – Mar – 2019].

Worklearning.com, 'LTEM'. [Online]. Available: https://www.worklearning.com/ltem. [Accessed 25 – Mar – 2019].

Z. Baynham-Herd, "Enlist blockchain to boost conservation", Nature, vol. 548, pp. 523, 2017.

Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, Huaimin Wang, Blockchain challenges and opportunities: a survey, *Int. J. Web and Grid Services, Vol. 14, No. 4, 2018*.